Evaluating Techniques for Wireless Interconnected 3D Processor Arrays

Amir Mansoor Kamali Sarvestani

PhD

University of York

Department of Computer Science

September 2013

Abstract

In this thesis the viability of a wireless interconnect network for a highly parallel computer is investigated. The main theme of this thesis is to project the performance of a wireless network used to connect the processors in a parallel machine of such design. This thesis is going to investigate new design opportunities a wireless interconnect network can offer for parallel computing.

A simulation environment is designed and implemented to carry out the tests. The results have shown that if the available radio spectrum is shared effectively between building blocks of the parallel machine, there are substantial chances to achieve high processor utilisation. The results show that some factors play a major role in the performance of such a machine. The size of the machine, the size of the problem and the communication and computation capabilities of each element of the machine are among those factors. The results show these factors set a limit on the number of nodes engaged in some classes of tasks. They have shown promising potential for further expansion and evolution of our idea to new architectural opportunities, which is discussed by the end of this thesis.

To build a real machine of this type the architects would need to solve a number of challenging problems including heat dissipation, delivering electric power and Chip/board design; however, these issues are not part of this thesis and will be tackled in future.

Table of Contents

List of Figures

List of Tables

List of Algorithms

Acknowledgments

I would like to use this opportunity to thank my supervisors Professor Jim Austin and Dr. Christopher Crispin-Bailey for their restless efforts to convert me from a student to a scientist. They helped me correct my views in computer science and in scientific thinking in general. Their enthusiasm, support, constructive critique and guidance were vital in writing this thesis without which had no chance to succeed. This thesis –in particular- was so loosely connected to many research fields in its nature so that without their support and timely guidance it could end up in a mess. Besides other helps Chris has also helped me with turning the two-part network partitioning algorithm into pseudo-code used in this thesis. I would like to thank them for being good supervisors and fantastic friends to me during my PhD.

The Cybula team in the Department of Computer Science was a big source of technical and moral support for me. I would like to thank Aaron Turner who helped me through the simulation experiments running on the White Rose Grid.

I thank the members of the Advanced Computer Architecture group including Richard Hind who helped me have a better understanding about available solutions for wireless power delivery. I also thank Nathan Burles and Dai Chengliang for their helps.

I also want to thank my parents who always were my role models. They taught me most of the seriously important things I learned in my life. I thank them for supporting me during my life and particularly during my PhD time. Their presence in the last few months prior to submission was a big moral boost for me. Sadly my father was passed away by the time the final version of this thesis was published; but I am sure his sole is happy about this.

With no doubt the biggest thank goes to my beloved wife Arwin who were with me all these years. She helped me going through difficulties of a new life and returning to academia. I cannot find proper words to thank her. Without her support I had no chance to be in this position and this thesis would never be written. I had not the chance to return her favours yet but I hope I can do so sometime in the future.

To my dearest and closest in the world: my mother and my wife

Aali and Arwin

And in ever-living memory of my grandfather, Nasser and my father Rahim.

This was one of their dreams;

So I hope this sheds sparkles of light on their endless silent nights

And makes their soul happy for ever!

Declaration

This thesis is the result of my own work and includes nothing that is the outcome of work done in collaboration except where specifically indicated in the text. The work in this thesis is my own and has not been submitted for examination at this or any other institution for another award.

Chapter 2 and parts of chapters 3, 4, 5, 6 and 8 are based at the following paper:

M. Kamali, C. Crispin-Bailey, J. Austin; "**On advantages and Limitations of 3D Wireless Grids as Parallel Platforms**"; International Conference on Selected Topics in Mobile and Wireless Networking, MoWNeT'2013; August 19th-21st 2013; Montreal, QC, Canada; pp 48-55.

Also chapters 8 and 9 are based on the following paper:

M. Kamali, C. Crispin-Bailey, J. Austin; "**Evaluating 3D wireless Grids as Parallel Platforms**"; International Journal of Ad Hoc and Ubiquitous Computing, 2015; In Press.

Chapter 1: Introduction

In this thesis the viability of a wireless interconnect network for a highly parallel computer is tested with the focus on connectivity. It starts with an introduction chapter in which a general overview of the idea is presented very briefly. More detailed discussions are available in next chapters (chapter 3 in particular). Following the overview is a discussion on the motivations behind the idea. It will discuss the importance of workong on this topic, the gaps in the literature it fills, the advantages of proposed model over current parallel computers and why it is the right time to work on it,. Furthermore, the main scientific contributions of the thesis are listed which will be discussed in more details throughout the rest of the thesis. The chapter finishes with presenting an outline for the rest of the thesis.

1.1) Overview

In this thesis we want to show whether wireless interconnect networks are suitable for massively parallel computers. A wireless interconnect network has the chance to reduce a supercomputer's production costs due to eradication of extensive wiring between processing nodes. Also, wireless networks are more flexible than wireline networks because nodes can join a wireless network just by tuning to proper channel; while in a wireline network the nodes are located on process boards with fixed capacity and the capacity of each shelf and rack is also fixed. Therefore, it is anticipated that a wireless interconnect network will be more flexible and less complex than wireline networks.

The idea of building a wireless platform for parallel processing raises questions such as:

1. Is there any wireless technology at the moment that suits this purpose?
2. How high the performance of such a computer will be?
3. What are the benefits of having such a platform?
4. How Hidden Node Problem is going to be tackled?
5. How such a computer may look like?
6. How the electrical power is delivered to processing nodes?
7. How the heat generated by nodes is dissipated?

The ultimate goal is to build a parallel computer with wireless interconnects containing at least thousands of nodes that is able to compete with modern commercial parallel computers; however, this thesis does not deal with all the questions listed above. The main questions this thesis answers are:

1. Is there any wireless technology at the moment that suits for this purpose?
2. What are the benefits of having such a platform?

3. How a wireless parallel computer may look like?
4. How Hidden Node Problem is going to be tackled?
5. How high the performance of such a computer will be?

To have a picture about the available wireless choices, a broad survey on different wireless technologies has been carried out. The results of that survey are included in the thesis.

A number of implementation challenges have been researched separately by *Richard Hind* [1]. Among other things he has investigated solutions for wire-free technologies for power transmission and cooling techniques for a large 3D grid. Those issues are not covered in this thesis but this chapter will have a brief look at some of those solutions.

Another major target of this thesis is measuring the performance of a proposed wireless network connecting the processors in a parallel machine with a 3D physical topology. To do such measurements, a simulation and data visualisation tool kit is designed and developed. The thesis also tackles a number of challenges like packet collision and network partitioning.

The proposed concept 3D wireless parallel architecture is called Ball Computer (BC). Each element of a BC (a ball) is a standalone processing and communicating entity in form of an electronic board with a processor of a reasonable computational ability equipped with a series of wireless transceivers. Based on the current level of electronic technology the whole package is sought to occupy not more than a couple of square centimetres. When massed produced, the balls can be available with reasonably low prices. The whole package will be put in a plastic (or any other suitable substance) to avoid electrical contact between balls. The whole collection of the balls will be submerged in water (or any other suitable liquid). The liquid will be in charge of heat dissipation and perhaps power delivery as well. To minimise the space occupied by the nodes and also to maximise the number of neighbours per node, a 3D hexagonal topology is selected for the network. In this scheme each node is in physical contact with 12 direct neighbours. Should a 2D hexagonal topology used for the network; the number of neighbours would be 6.

Regarding the novelty of the idea, there are a number of challenges in making such a computer. This thesis focuses on connectivity issues including solving Hidden Node Problem and packet collision by introducing a multi-channel transmission scheme enhanced with a two-stage network partitioning algorithm to assign proper channels to wireless devices.

Another big question is how to deliver electrical power to nodes. Since the idea of BC is implemented in a simulation environment, the answer to the power delivery question does not directly affect the plan at its current state. At the time of implementation an effective power delivery mechanism is required. To be faithful to the original idea of an entirely wireless computer, the priority is to have a wire-free power delivery mechanism but other mechanisms may be considered should it is proven that wired solutions have significant benefit over the wireless options.

Hind [1] has listed a number of wire-free power delivery mechanisms including:

- Inductive Coupling;
- Capacitive Coupling;
- Radio Frequency;
- Optical;
- Piezoelectric;
- Dynamo;
- Chemical.

It is shown in the aforementioned source that all those approaches have some shortcomings. Also, there are chances that a hybrid solution (a combination of two or more mechanisms) may have better performance in terms of power delivery to the circuitry envisaged in this thesis (a processor and a set of fast wireless devices). *Hind* [1] particularly suggests the combination of RF and inductive (or optic) methods may be effective (each operating over different distances).

Among the methods listed above, inductive and capacitive coupling methods are efficient in distances shorter than what is envisaged in this thesis; but their performance drop significantly over longer distances. The possibility of using the cooling liquid to deliver the electrical power is another option. A properly chosen conductive liquid solution may be able to deliver electrical power to nodes and at the same time help dissipating the heat out of the nodes. There are; however, concerns about how to deal with the potentially toxic nature of the substances used in this method. The idea of storing power for transmission to other nodes is another idea that can improve any power delivery mechanism. Apart from wire-free power delivery mechanisms, direct (wired) power delivery solutions can also be considered. In this scheme nodes are placed on trays that not only fix their positions but also accommodate power lines to support nodes.

Another issue that is not part of this thesis is heat dissipation. Like standard High Performance Computing (HPC) systems, gas and/or liquid substances are candidates for

cooling the proposed platform. It is anticipated that the circuitry envisaged in the proposed architecture will generate high level of heat per volume unit. This may rule out heat dissipation strategies based on air flow. It is already mentioned in this thesis that a liquid-based heat dissipation mechanism can also be used as a means for power delivery as well. More information about different aspects of BC architecture is available in chapters 3 to 6.

A simulation test bed helps investigating the behaviour of the proposed network without dealing with any hardware implementation challenges. In real world, modern parallel computers usually use hundreds of thousands of processors. A quick survey on the network simulators already available showed that it is a challenge for all of them to accommodate networks with such a volume. For this reason designing and implementing a simulation and data visualisation tool set is part of the thesis. However, there are limitations on the number of simulated nodes in the simulation tool due to the size of memory needed for running the software. Another reason for building a new tool kit is the level of abstraction required for this research which does not match existing network simulation tools.

There has been very little research on this topic. For this reason, we have to be careful in how we compare our work with other research. Also it is tricky to evaluate the cost of such a computer. This makes it hard to have an accurate cost-benefit analysis. Since no wireless parallel machine is made yet, all the performance measurements presented in this thesis are based on simulated models. Some aspects of the simulator are based on ideal models rather than realistic ones for simplicity; therefore, the measurements are approximations of real figures. More accurate simulated models are envisaged for the next stages of this research. To have a justifiable quantitative comparison in terms of performance between this research and available massively parallel computers we need to have both an implementation of the proposed concept computer in real world and a very precise cost valuation of that platform. At this stage, what this thesis is going to investigate are the new design opportunities a wireless interconnect network can offer to parallel computer architects.

To tackle the problem with packet collision, a Multi-Radio-Multi-Channel (MRMC) scheme is devised for nodes. The radio devices should be tuned to proper frequency channels to make sure no packet collision occurs. As part of this thesis a novel two-stage network-partitioning algorithm is designed and implemented. This is a collision avoidance scheme to guarantee that, by using different channels for communication between different (yet overlapping) groups of neighbouring nodes (zones), there would

be no interference between any two packets all over the network. In the thesis it is shown that the network size plays no major role in determining the number of channels needed. In fact the network topology and the radio range (and the interference range) of nodes dictate the number of channels.

MRMC technologies have been used in different applications including mobile phone networks and wireless Internet access devices. Also, in all of these applications a network partitioning algorithm is essential to assign frequency channels to partitions of the wireless network. The differences between those algorithms are mainly in the criteria they satisfy and the topology of the network they are applied to. It is proven in the thesis that the proposed network partitioning is capable of solving the Hidden Node Problem and eradicating packet collision both in theory and practice.

To evaluate the performance of the network in the simulated environment, tasks are reduced to a degree of abstraction so that just their pattern of communication and computation are retained. By doing this, the communication to computation ratio, the rate of occurrence, the duration, the order and the pattern of the communication and computation intervals are preserved. Furthermore, any possible dependencies between communication and computation intervals are left untouched. Therefore, a level of task abstraction, called a task-model, is achieved which is a virtual traffic generator that mimics a class of real world tasks in terms of processor and channel usages patterns. When task X is turned to a task-model the computational nature of X is not preserved. As a result, the task-model is no longer dependent on the software implementation of its original task.

To have an overall understanding of the behaviour of the proposed parallel machine, its performance is studied over a range of network sizes, transfer rates and computational ability of the nodes. The results are shown and analysed in the thesis. A visualisation tool is also developed to improve our understanding of the behaviour of the proposed system through graphical description of its performance at different points.

To have a cost-benefit analysis for the proposed BC architecture, we need to have information about the cost of processors as well as the cost of data links. A brand new processor is not going to be designed for the nodes in a BC platform. The nodes will use off-the-shelf processors with known prices. But to have a cost-benefit analysis an estimation of the cost of the wireless links sought in this thesis is required. The links surveyed in this thesis are all in research stages and there is still no commercial price tag on them. Also some data rates used in simulated experiments are not available at the moment. They were chosen only to cover a range of possible data rates from the past,

the present and the future. It does not mean that they are completely unrealistic; instead, the range of data rates chosen for experiments matches the expected technologies overseen for the near future.

It is possible; however, to check the costs for existing parallel processing. Table 1 shows how the cost for HPC has decreased over time. This table just concerns the processors rather than networking (which is of particular interest in this thesis), cooling and other issues that contribute in total finished cost. A more accurate estimation on a whole parallel computer or its interconnect network is needed to have a precise figure of how much should a BC cost to be as commercially efficient as modern commercially available parallel platforms.

Date	Approximate cost per GFLOPS	Technology
1961	US$1.1 trillion, or US$1,100 per FLOPS	About 17 million IBM 1620 units costing $64,000 each
1984	US$15,000,000	Cray X-MP
1997	US$30,000	Two 16-processor Beowulf clusters with Pentium Pro CPUs
2000, Apr.	$1,000	Bunyip Beowulf cluster
2000, May	$640	KLAT2
2003, Aug.	$82	KASY0
2007, Mar.	$0.42	Ambric AM2045
2009, Sep.	$0.13	ATI Radeon R800
2009, Nov.	$0.59 (double precision); $0.14 (single precision)	AMD Radeon HD 5970 *Hemlock*

Table 1: Comparing the cost for 1GFLOPS worth of computation during last decades.[1]

In lack of such numbers what we can do is to base our comparison on prices announced for (possibly isolated cases of) parallel platforms. Such a price also includes an unknown - and possibly considerably high- profit margin. To have a very rough estimation, a BC can be valuated based on the cost announced for a number of the most powerful supercomputers in the world. Table 2 lists a few of such supercomputers as in June 2014. It should be noticed that supercomputers are not usually mass produced and therefore in many cases their construction cost are not publically known.

The table shows that at the moment 1GFLOPS of computation costs approximately $10. Based on available processors in the market it is realistic if we assume each node in a BC have 1GFLOPS of computational ability. In this case –and based on Table 2- a single node (processor plus all the wireless modules) has a price tag of $10. According to Table 1, the processors have just a very narrow share in that overall cost (something

[1] Shortened from a table presented in: http://en.wikipedia.org/wiki/FLOPS.

in range of $0.5 as announced in 2009). The remainder can be allocated to the wireless modules and other costs like cooling. This is just a case for comparison and should not be regarded as a scientific conclusion. The comparison above lacks high precision as there are many unknown issues on both sides of the comparison (i.e. the real cost of many supercomputers as well as the costs for components of a node in BC interconnect network). Therefore, it is not a scientific analysis. However, this can be a good starting point for further investigation to achieve more accurate cost-benefits analysis.

Construction Date	Rank (as in Jun. 2014)	Computer Name	Speed (PFlops)	Cost (USD million)	Approximate cost per GFLOPS ($)
Jun. 2003	1	Tianhe-2	33.86	390	11.52
Oct. 2012	2	Titan	17.59	92	5.23
2012	5	Mira	8.586	50[2]	5.82

Table 2: The cost of a number of most powerful parallel computers in the world derived from Wikipedia[3].

This thesis shows at least some of the existing on-chip wireless technologies can be used in a wireless interconnect for an HPC system; although, they are not originally designed for this purpose. The results demonstrated in this thesis show that for a given set of network attributes (in particular when the ratio of the links' transfer rate to the processors' computational ability is more than 1000) the performance is between 70% and 85% in terms of processor utility.

1.2) Motivation

The platform presented in this thesis is based on a number of technologies and algorithms some of which are separately researched and developed in other fields. Some others are inspired by existing algorithms and techniques. In this thesis we will see if these algorithms and techniques are mature enough to construct a new wireless HPC platform.

Wireless devices are available in different shapes, sizes and prices. They are used in a wide range of applications including sensor networks, mobile networks, ad-hoc networks and wearable devices. But at the same time there are applications in which wireless modules are not traditionally considered as good choices. Time critical applications, applications requiring high data rates, applications in which energy consumption should be kept very low and applications with high load of data transfer are among such applications. On the positive side, wireless devices increase the flexibility and scalability of a transmission scheme. They also reduce cost and complexity by removal of wirings.

[2] It is an estimated cost.
[3] http://en.wikipedia.org/wiki/TOP500.

It should be noticed that wireless devices have improved in terms of data rates and energy consumption in recent years. Until very recently, traditional wireless technologies have not been a good choice for time-critical applications like parallel processing.

The main motivation behind the idea of using wireless devices in parallel platforms is the new emerging wireless technologies. The recent developments in low-range high-speed wireless communication have already opened their way into some time-critical applications including inter- and intra-chip applications. Some advantages of a wireless interconnect scheme from a systems composition viewpoint are scalability, simplicity of assembly, potential for reduced power consumption, the obvious elimination of complex wiring problems and the ability to achieve economical three-dimensional physical packing of components.

The main disadvantages of radio devices are their low bandwidth compared to wireline networks, packet collision which ends up in re-transmission of the data and the energy consumption which needs even more reductions to compete with wireline solutions.

There are a number of bottlenecks in field of HPC, each of which are subject of intense research. The interconnect network has always been one of these bottlenecks. Among others are power consumption, software tools (operating systems, compilers and communication APIs) and management of high volume of data generated by these systems. Thus, the quest for novel network architectures is as lively as ever.

Traditionally wired network technology has been the only candidate for connecting processors in a parallel machine. Wireless devices have seriously suffered from a number of shortfalls, which do not let it be a favourite option for a parallel computer. On the other hand a typical wireline network is usually made of compute boards with fixed capacity for processors, shelves with fixed capacity for compute boards (and/or IO boards) and racks with fixed capacity for shelves. Adding new processing elements to such a system is not always easy and straightforward because this may need some changes in the design and instalment of the network. But wirelessly communicating processing elements can be tuned to proper channels and join a 3D wireless network. This means that wireless networks can scale easier than their wired counterparts. The reduction of the wiring complexity and expense are also among the attractive aspects of a wireless architecture; however, the deficiencies of wireless technologies have outweighed their benefits and have discouraged the architects to use them so far.

In light of new developments in wireless technologies it is now a good time to see if high speed wireless transceivers can be used to make a wireless network for HPC

applications. It is shown in different surveys that the gap between the bandwidth supported by wired and wireless technologies is narrowing during recent years. *Kürner* [2] has shown that although wireline technologies were always faster than their wireless counterparts, the difference between their data rates is constantly narrowing. There is a healthy chance to have commercially available THz-scale wireless (RF, inductive and capacitive coupling) devices in future which means the gap between wireline and wireless communication links is expected to become even narrower. Figure 1 compares the wireline and wireless technologies for local area networks during the last forty years to illustrate this point.

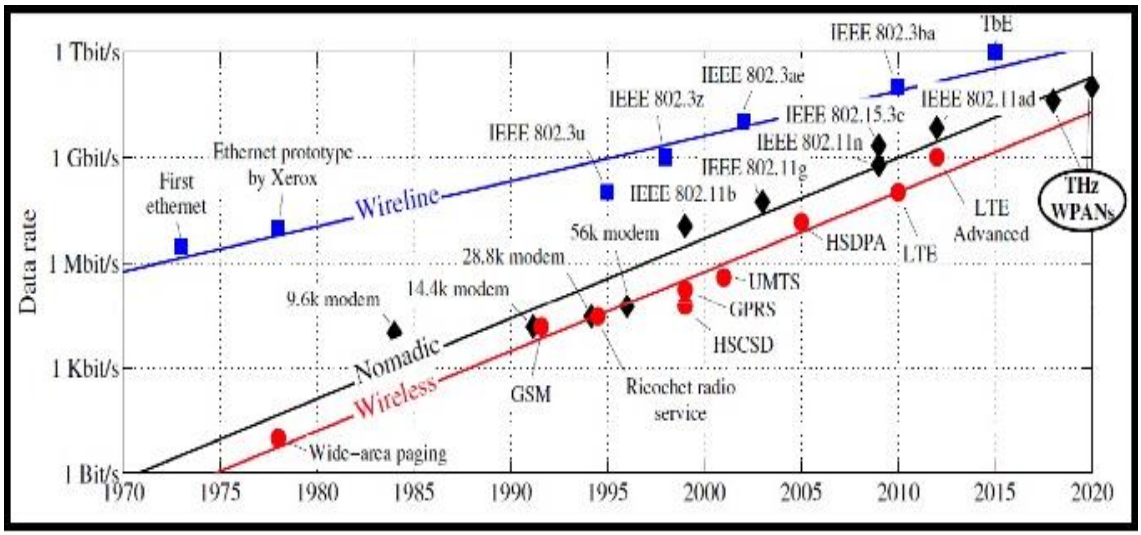


Figure 1: Evolution of data rates in wireless as projected in [2]

Improvements in signalling techniques and coding/decoding methods are among the reasons why wireline data rates are increased. Many of these improvements can be used to improve wireless technologies as well. Parallel to these improvements, researchers have tackled other restrictions exclusively concern wireless communications including power consumption. This means that there were more improvements in wireless devices than wireline technologies. This can explain why the gap between these two categories of technologies is narrower than past.

In inter- and intra-chip communication the same reduction in difference between these two types of technologies can be observed. *Moore et al* [3] has shown that there is a cap in transmission speed on wires in those applications. This gives a chance for narrowing the gap between wireline and wireless devices in terms of data rates.

Latency and bandwidth are two issues in any communication network. The bandwidth in both wired and wireless networks is now higher than ever. But the signal latency is not changed as much because to a great extent it depends on the communication media. This means that now signal latency plays a bigger role in network performance

compared to bandwidth. Sources of signal latency (AKA packet delivery time) can be hardware-based and/or software-based. Hardware latency can be expressed as:

$$Latency = Transmission\ Time + Propagation\ Delay \quad \textbf{Eq. 1}$$

These two factors should not be confused with each other. For a piece of data of size *n*, the transmission time is defined as:

$$Transmission\ Time = \frac{n}{Transfer\ Rate} \quad \textbf{Eq. 2}$$

The propagation delay is related to the medium of communication and is the time from sending a bit of data in the transmitter side until its reception in the receiver side (Eq. 3).

$$Propagation\ Delay = \frac{d}{s} \quad \textbf{Eq. 3}$$

Where *d* is the distance between the transmitter and receiver of signal, and *s* is the propagation speed. The propagation delay is not expected to be very significant regarding the high propagation speed of electromagnetic waves (300000000 m/s) and very short communication distances sought in the platform proposed in this thesis. For a 1 cm wireless link the propagation time is around 33.3 ps ($3.33 * 10^{-11}$). The transmission time in both wired and wireless networks are higher than such a number. As an example, transmission time of a packet of data of size 1 kb over a 10 Gb/s link is 10^{-7} s. As a result, the hardware latency can be estimated by transmission time. In the absence of parallel transmission of data in wireless transceivers the transmission time relates to the inverse bandwidth of the data link.

Software latency is believed to be around two degrees of magnitude higher than the hardware latency (An example is given in [4]). The coding/decoding procedure, queuing time, resource competition are among major contributors of software latency. Many of the contributors to software latency are common between wired and wireless networks and therefore do not play a major role in their comparing.

In addition to latency, the bisection bandwidth and overall throughput are other issues that are envisaged to be improved by the platform proposed in this thesis. Details about these factors can be found in coming chapters. All these have encouraged us to think if high speed low latency short range wireless modules can perform the same as wireline connections in parallel processing platforms.

MRMC devices have already been used in applications like wireless internet access networks. The same idea can be adopted over much shorter distances for on-chip wireless modules to accommodate in the platform proposed in this thesis. The number of channels and the mechanism used to manage them are different from previous applications.

A wireless HPC platform should have a solution for packet collision caused by the Hidden Node Problem. A variety of network-partitioning methods have been developed for different applications. These algorithms fall into two main categories: centralized (e.g. [5] [6] [7]) and distributed algorithms (e.g. [8] [9] [10]). These methods are used as part of a solution for packet collision in different wireless networks. This gives a chance to the author of this thesis to adopt a network partitioning algorithm that matches the characteristics of the proposed network. This is necessary to avoid excessive packet collision while all nodes in a localised group communicate on a same frequency. This is a crucial factor to build a multi-channel wireless grid for parallel computing.

Some crucial elements of a wireless HPC system have been researched and developed separately and some of them have been used in different (yet relevant) applications. This implies that there are substantial chances to borrow and adopt technologies and algorithms to make a wireless parallel computer. The cost and complexity of such a computer will be reduced due to removal of the wiring system; however, the extra cost imposed by adding the wireless modules to nodes are yet to be measured. Also, such a system scales easier as there is no restriction imposed by wiring system.

The idea of a wireless HPC platform has never been researched before. In addition to the novelty of the idea, its potential benefits over the traditional wireline interconnect networks and the fact that some crucial aspects of the plan are already researched and developed separately are the main motivations behind this thesis. Since the field is new, it can open new fields of research to researchers from different disciplines to explore its different aspects and even expand the idea to yet unknown territories. This thesis can be a start point for all those efforts.

1.3) Main Contributions

This thesis will investigate positive and negative aspects of using wireless interconnect networks for massively parallel computers. This thesis will show that the state-of-art on-chip radio devices are the best choice for such a platform. However, we will see that those devices still need to improve in transfer rate and more importantly in energy consumption.

A novel network partitioning algorithm will be proposed to solve Hidden Node Problem and eliminate packet collision. This thesis uses series of simulation experiments to show that the platform it has proposed yields excellent results (in terms of processor utility as well as link utility) for a certain range of networks. We will see that the best results are achieved when the ratio between the link’s communication speed (in bit/sec) and the

processor's computational ability (in instruction/sec) is more than 1000. Also we will see that the performance of the proposed is higher when it deals with larger data sizes.
The weaknesses of the proposed network are also investigated. The thesis shows that the main problem is in the type of routing and buffer management strategy adopted for this platform. Suggestions for improvement are made for future work.
The main scientific contributions of the thesis are listed below:

1. Introducing the concept of Ball Computer as a wireless infrastructure for HPC. Designing a simulated wireless-enabled processing module in order to be used as a member of a Ball Computer in a simulation environment.
    a. Introducing the concept of Ball Computer as a wireless HPC infrastructure.
    b. Designing a layered structure for nodes.
    c. Designing a two-layer message passing protocol for communications between layers of a node as well as different nodes in the network.
    d. Evaluating the network performance in the simulated environment.
        i. Studying how size of packets and other network parameters can be chosen to yield best performance for two major task-models.
        ii. Optimising the number of branches for a divide-and-conquer task-model in the proposed topology.
    e. Studying the behaviour of the Ball Computer under different circumstances.
        i. Running two sets of experiments with both FFT and Simple parallel task-models on a Ball Computer of sizes 1000 and 2000 nodes when other network attributes change.
        ii. Analysing the results of those sets of experiments to demonstrate the benefits and limitations of Ball Computer under different circumstances.
        iii. Investigating the role of each chosen network attributes on the behaviour of the network.
        iv. Demonstrating the significance of the role of communication to computation ratio in determining the behaviour of the network.
        v. Finding if there is a limit on the size of workload in a wireless parallel machine.
2. Solving the hidden node problem for the particular network proposed in this thesis and eliminating packet collision problem by using multiple radio devices.

   a. Designing a multi-radio-multi-channel (MRMC) scheme.
   b. Designing a two-stage network-partitioning algorithm to partition the network into zones in first stage.
   c. Introducing a systematic method (second stage of the partitioning algorithm) to assign channels to zones to solve the hidden node problem and minimise the number of channels at the same time.
3. Designing a set of task-models to study the behaviour of different tasks in simulated environment.
   a. Introducing the concept of task-models as communication/computation pattern generators.
   b. Demonstrating how a class of tasks can be converted to a common task-model.
   c. Generating two task-models with very different communication patterns inspired by real-world tasks.
   d. Design and implementation of a communication protocol for each of those task-models.
4. Studying tailor-made load balancing methods which fit the characteristics of the network.
   a. Showing how the unique topology of Ball Computer gives space for balancing workload to increase the performance of a Ball Computer.
   b. Introducing two load balancing metrics for a divide-and-conquer task.
   c. Discussion about potential benefits and limitations of each of those two load balancing metrics.
   d. Measuring the increase of performance using multi-tasking and multiple-workloads using one of introduced load balancing methods.
   e. Introducing new task-models with load balancing for a better performance.

In addition to the scientific contributions of this thesis, a set of software tools are designed and implemented which can be used and/or modified by other researchers who work on the same level of abstraction. They are listed below:

1. Discussing the potential benefits and restrictions of using wireless technologies in parallel processing.
   a. Discussion on the advantages and limitations of wireless technologies for a 3D wireless parallel computer. Showing from what aspects and how

wireless technologies can be better than their wired equivalents in this application.

b. Listing a number of available research technologies which can be candidates for wireless links in the proposed grid.

2. Implementation of a novel large wireless network simulation and visualisation tool kit.
    a. Developing a simulation tool to measure the performance of a very large wireless network.
    b. Designing and implementing ideal wireless links for such a simulated network.
    c. Giving an independent identity to channels as software modules in the simulation tool which leads to easier and more perfect control over their activities and guarantees their robust performance.
    d. Making two different versions of the simulator (with and without clusters of nodes). Testing the benefits of both approaches.
    e. Studying the geometric and task-related attributes of a wireless parallel machine.
    f. Design and implementation of a set of visualisation tools to find the network's bottlenecks.
    g. Using the visualisation results to analyse the performance of simulated network.
3. Design and implementation of a distributed simulation tool.
    a. Design and implementation of independent software agents capable of communicating with each other to run a simulated large wireless grid.
    b. Design and implementation of a communication protocol for the distributed simulation tool.
    c. Running simulation experiments to test the performance of the distributed simulator.
4. Design and implementation of a communication protocol for a multi-part packet transfer mechanism.
    a. Design and implementation of extra software layers to handle multi-part messages.
    b. Design and implementation of a communication protocol for a multi-part packet transfer mechanism.

c. Running simulation experiments to test the performance of the multi-part packet transfer mechanism.

1.4) Organisation

This thesis consists of ten chapters. Chapter 1 is the introduction chapter which gives a general idea about the subject of the thesis and the motivations behind it. It contains answers to the questions like why this topic has attracted our attention, how important this topic is and how the results from this thesis help filling some gaps in the research literature. Also the main contributions of this PhD thesis are listed.

Chapter 2 takes us through a brief background review of the relevant fields of research to be prepared for the main part of the thesis in which some of those related researches are being used.

Chapters 3 to 7 are dedicated to design and implementation of the proposed platform. In chapter 3 the main research question and the main hypothesis backing the research is presented and explained. It shows what the scope of the thesis is and defines the objectives of the thesis. It also shows what experiments are going to be run and how they are going to be measured to check the level of satisfaction.

In chapter 4 we will see what kind of communication media is going to be adopted for the proposed network and why it looks fit for this specific purpose. This does not contain many electronic details as this thesis is entirely based on a simulation of wireless networks in which not all the electrical characteristics of the elements are (and are needed) to be implemented. In this chapter it is shown what attributes of the network are important from the simulator's point of view.

Inspired by what was reviewed in previous chapters, a novel network-partitioning algorithm is presented in chapter 5 which is tailor-made for this research. The criteria against which this algorithm is designed are listed. It will be explained why those criteria are chosen and how they are different from the other network-partitioning algorithms.

In chapter 6 it is explained what is meant by "task-model" in this thesis and why it is important to use task-models instead of real tasks. It will be shown how real world tasks can be transformed to task-models to be fit for simulations. It will be shown how task-models can be used to decrease the test and analyse time for a potentially large number of tasks in real world.

Chapter 7 goes through implementation details of the simulation and data visualisation tool kit implemented in this research. Key data structures and communication protocols are presented in this chapter. It will be shown how the ideas originated from previous

chapters are translated to data structures and pieces of code. In addition to the main simulator, an off-line data visualisation tool is implemented and introduced in this chapter to present the results in an illustrative form to help analysing the results and verifying the initial hypothesis of the thesis.

The next two chapters are dedicated to results derived from the simulation tests. One of the main targets of the thesis is solving the Hidden Node Problem and elimination of packet collision. In chapter 8 results will be presented to back the claim that the packet collision is completely eliminated. Also other preliminary results are presented in this chapter. The results show how the performance of the proposed system changes when network nodes are multi-tasking. Also it is shown how using multiple independent workloads improves the overall performance of the system. Included in this chapter are results that show how successful the idea of load balancing is and how the task-models used need to be changed to let the load balancing mechanism work effectively. Another part of this chapter is dedicated to a comparison between two main task-models introduced in this thesis. As it will be discussed in details later in this thesis, the main difference between these two task-models is the degree of dependency between tasks in different nodes of the network. This chapter finishes with a set of experiments on the optimal number of child nodes per parent node in a divide-and-conquer task-model in a 2D and 3D grid.

Chapter 9 covers one of the major parts of this thesis in which the behaviour of the system is studied when its four most important parameters change. The parameters are the network size, the transfer rate of links, the computation ability of nodes and the workload (data) size. Two different task-models and two different network sizes are used in this set of experiments.

Chapter 10 is dedicated to the conclusion and future work. This chapter contains final analysis of the results presented in last two chapters. Results from previous chapters are used to assess how beneficial the idea of ball computers can be. It shows for what values for network parameters the proposed platform yields good results and for what values for those parameters the performance is not satisfactory. This chapter concludes with a look at how this thesis can be expanded. A number of new fields of research derived from this research are listed.

Chapter 2: Literature Review

A wireless platform for a parallel computer needs to borrow ideas, algorithms and technologies from a variety of fields of research. Throughout this chapter some of these fields are reviewed to choose or tune those that best fit the wireless grid proposed in this thesis. Some background on the parallel processing and interconnect networks are also included in this chapter.

2.1) Parallel Processing

From the early days of computing history researchers have been challenged by problems that need excessive computation labour. This put a constant pressure on the architects to come up with new machines that can handle larger numbers of calculations in a given time. Some of the grand challenges are plotted in Figure 2 to present their storage and computational requirements as in 1999.

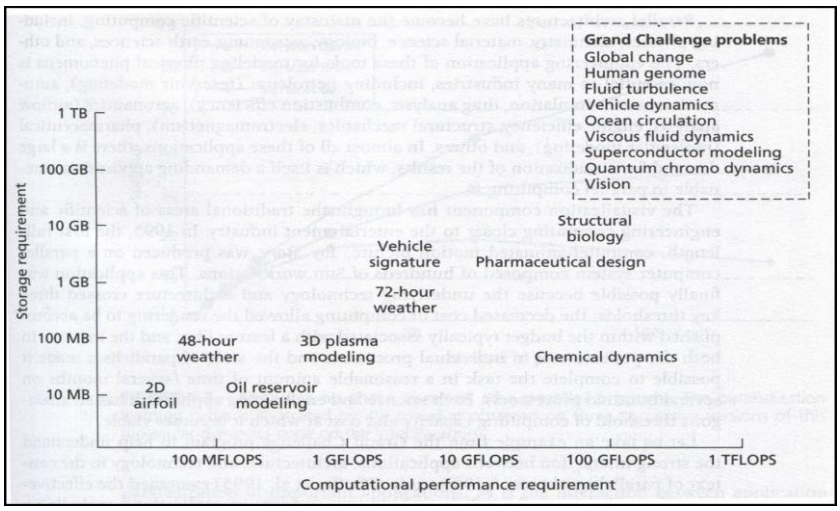


Figure 2: Grand challenges as projected in 1999 in [11]

The term "grand challenges" is first used in 1980's in US governmental documents to refer to the main challenges for the future in the field of HPC. At present the challenges are almost the same with the exception that the computational performance and storage requirements are increased now. A 2011 report published by USA's National Science Foundation [12] lists the grand challenges as:

- Advanced new materials;
- Prediction of climate change;
- Energy through fusion;
- Water sustainability;
- Condensed matter theory and Quantum chromodynamics;
- Semiconductor design and manufacturing;
- Assembling the tree of life;
- Drug design and development;
- Understanding biological systems;
- New combustion systems;

- Astronomy and cosmology;
- Hazard analysis and management;
- Human sciences and policy;
- Virtual product design;
- Cancer detection and therapy;
- CO_2 sequestration.

Some of the 2011's challenges can be found in Figure 2 (dated 1999) and others are new. The challenges in present time have more computational and storage demands compared to the last years of 20th century. This is mainly because those challenges deal with larger data sizes now.

Weather forecast, for example, is projected in Figure 2 as a 48-hour forecast which needs around 100 MFLOPS of computational capability and a 72-hour forecast with less than 10 GFLOPS; but today's challenge in this filed is forecasting for longer periods and demanding higher computational capability. As theorized by *DeBenedictis* [13] in 2005, the climate modelling needs zettaFLOPS-scale computational capacities for a time span of a couple of weeks.

A single simulation of gamma ray bursts needs around 18 million PFLOPS [12]. This means with a PFLOPS-scale parallel computer it is practically impossible to run a single simulation of that type in less than 100 days.

To simulate a high pressure turbine blade needs a total of 1000 PFLPS and 1PB of memory [14]. These are just a few examples of present challenges for current PFLOPS-scale computers[4] which push HPC community to head for Exascale computing.

To fulfil such high demands there are two basic solutions: first, using a single processor computer with very powerful processing unit; and second, using a number of processors and try to use their collective power to solve a problem that is too big to solve for each of them individually. The first approach has led to using transistors instead of vacuum bulbs; integrating transistors into ICs, increasing the density of transistors in chips and so on. The problem with this method is that even the best processors built cannot even get near what aforementioned grand challenges need in terms of computation ability.

For example the 4 most powerful supercomputers as in June 2014 [15] use Intel Xeon E5 series, AMD Opteron 6274, IBM Power BQC 16C, SPARC64 processors which are capable of executing 17.6, 48.36, 12.8, 16.0 GFLOPS per core respectively (The rest of the top 10 supercomputers use the same processors or a processor of the same family as the top 4); while, based on the aforementioned figures thousands or even millions of PFLOPS worth of computational ability is needed for handling today's grand challenges.

[4] At the moment the highest peak performance is reported by Tianhe-2 in China which is 33,862.7 TFlop/s [15].

The latter solution is the basis of what is now known as parallel processing. *Almasi and Gottlieb* [16] define a highly parallel machine as "A large collection of processing elements that can communicate and cooperate to solve large problems fast." This is a very generic definition and many questions are left unanswered, including:

1. How many processors are needed?
2. How these processors are going to "communicate and cooperate"?
3. How fast the problems can be solved?

This is the architects' duty to find answers to questions of this type to maximise the performance. These questions construct the key elements of any parallel machine.

The key point in a successful parallel execution of tasks is that the processing elements can execute instruction at the same time. A Speedup factor is introduced to have a quantitative measure of how good a parallel machine works. It is simply the performance of a task on a parallel machine divided by the performance of the same task on a sequential machine.

$$\boldsymbol{Speedup(N)} = \frac{\boldsymbol{Performance(N)}}{\boldsymbol{Performance(1)}} \qquad \textbf{Eq. 4}$$

Arguably in almost all cases raw performance can be reduced to the execution time. Therefore Eq. 4 can be rewritten as Eq. 5:

$$\boldsymbol{Speedup(N)} = \frac{\boldsymbol{Time(1)}}{\boldsymbol{Time(N)}} \qquad \textbf{Eq. 5}$$

However, no matter how good it is written, almost every piece of code has some parts that cannot be parallelised; i.e. for a programme it is impossible to run it fully in parallel. As a result, any program has its own serial and parallelizable parts. *Amdahl* [17] proved that if there are infinite processors available in a piece of code which takes *l* time units to execute and its parallelisable part *p* time units, the best parallel performance would be like Eq. 6:

$$\boldsymbol{Speedup} = \frac{\boldsymbol{l}}{\boldsymbol{l-p}} \qquad \textbf{Eq. 6}$$

Eq. 7 states that if there are just N processors involved, the maximum speedup is:

$$\boldsymbol{Speedup(N)} = \frac{\boldsymbol{l}}{\boldsymbol{l-p+\frac{p}{N}}} \qquad \textbf{Eq. 7}$$

Whether this theoretical peak performance is achievable or not depends on how efficient the connection between the processors is. The interconnection between the processors can provide facilities for the processors to communicate with each other without which the whole idea of parallel processing will simply fail. In practice the speedup is even more limited by other factors. Figure 3 shows how an increase in the number of processors does not necessarily improve the performance. In this figure the horizontal axis represent the number of processors. Parameter "*r*" in this figure is the ratio of computation to communication times.

According to Figure 3, from a certain point increasing the number of nodes may decrease the performance because of the excessive packets sent over the network. This makes it harder for processors to find a free communication channel and as a result the performance drops.

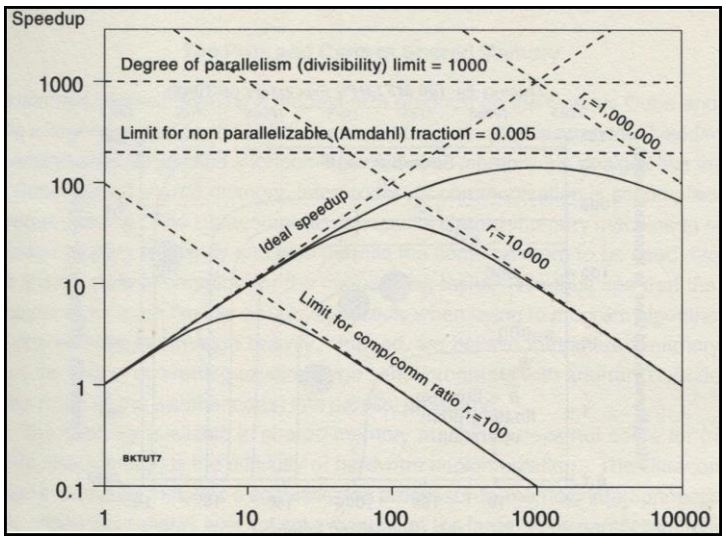


Figure 3: Different factors limiting the speedup factor presented in [16] (x axis represent the number of processors)

The above figure is based on data from 1994 but the relationship between the speedup and the number of processors has remained almost unchanged despite the claims that Amdahl's model may not hold any more. This does not mean that there is no limitation on speedup anymore; instead as, for instance, *Hill and Marty* [18] explain, because of new developments including introduction of multicore processors the form of threshold suggested by Amdahl's law is now changed. Like Figure 3, in new parallel computers an increase in number of processors means more communication which is potentially taking place over longer distances and more congested long routes. This increases the chance for more delays during communication. Also, the comm/comp ratio still plays a role in the speedup factor. The finer grain the parallelism (corresponding with lower comp/comm ratio) the communication and synchronization overhead between tasks takes longer than the computation. On the other hand for coarse grain parallelism (corresponding with higher comp/comm ratio) there is a chance to reach higher speedups by saving time due to less number of task synchronisations. Also, Data derived from real-world measurements confirms that the assumptions suggested by Figure 3 still hold ([19] [20]).

Gustafson and *Barsis* had a rather different analysis on parallel computing which is led to what is known in literature as Gustafson's law. The law is introduced in their article [21] published in 1988 which is different form Amdahl's law as the former assumes that the data size is not fixed; while the latter deals with fixed data sizes. Gustafson's law

states that in a process with a given non-parallelizable fraction any speedup ratio can be achieved by choosing the right data size. Amdahl's law, on the other hand, deals with a fixed data size that means with a known non-parallelizable fraction of a process the speedup cannot exceed a certain limit. Eq. 8 formulates Gustafson's law:

$$S(P) = P - \alpha \cdot (P - 1) \qquad \textbf{Eq. 8}$$

Where P is the number of processors, S is the speedup and α is the non-parallelizable fraction of the process.

While Amdahl's law explains why adding too many processes to perform a task (with both parallelizable and non-parallelizable fractions) in parallel may not lead to a substantial performance improvement; Gustafson's law tries to find how the size of the data processed with a parallel system in a given time can be increased by increasing the number of processors. Gustafson's law does not apply to tasks that do not fundamentally deal with large data. Such tasks are not good parallel tasks as far as it is concerned with Gustafson's law. The term "Scalable Parallelism" is used for tasks for which Gustafson's law holds.

2.2) Network Topologies

Network interconnection has always been a hot topic in parallel processing. A wide range of topologies and technologies are used to let processors communicate as effectively as possible.

Figure 4 illustrates three classes of interconnect networks. In part *(a)* a crossbar switch is used to connect processors and other elements. The advantage of this technology is that the bandwidth available for each node does not decrease when new elements are added as it uses a different set of connection lines. One negative point is the extra cost needed for the switch itself. Another problem is its expansion cost. To add a new element the switch may need to be replaced with a bigger one if no spare ports are available on the old switch. An increase in the size of a switch may leave some switch ports unused which means they have paid for a hardware which is partially of no use.

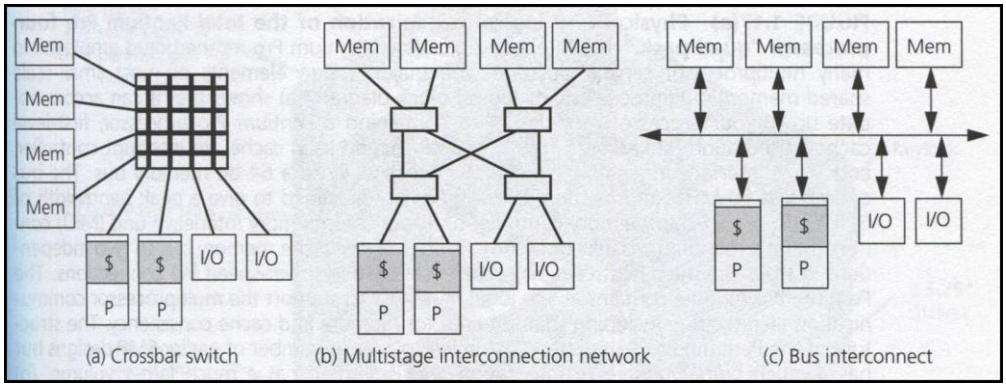


Figure 4: Network interconnect topologies presented by [11]

In contrast to crossbar switches, bus technology (Figure 4.*c*), has the apparent advantages of simplicity, scalability and low cost (since there is no need for any extra network control device). However, the bandwidth available for each element drops as the network scales up. This is because all elements use the same physical communication medium. The length of the bus is also another restricting factor. Any type of cable has a length limit over which the signal send over it is not guaranteed to be decoded correctly.

In part *(b)* a multistage switch network is used to solve the problem crossbar switches had with expansion and at the same time do not decrease the bandwidth when adding elements. Butterfly and Dragonfly networks are examples of modern multistage interconnect networks. The number of switches needed in these topologies still depends on the number of processing elements; but the number of switches relate to logarithm of the number of nodes that is significantly better than crossbar switches that have a linear relation between switch size and the number of nodes.

In a crossbar network no message from node *a* to *b* can never block another simultaneous message from node *s* to *t* as these two messages use completely separate paths. From this sense a crossbar network is *non-blocking*. An issue with this type of network is that if the same situation occurs in some multistage switches, the correct delivery of those messages cannot be guaranteed. This means that some multistage switch networks are *blocking* networks. Whether a multistage network is *non-blocking* depends on the type of the network, the number of switches and their capacity (*radix*) of among other factors.

Rank	Computer	Site	Made by	Network
1	Tianhe-2	National Super Computer Center in Guangzhou, China	NUDT	TH Express-2 (Fat Tree)
2	Titan	DOE/SC/Oak Ridge National Laboratory, USA	Cray	Cray Gemini
3	Sequoia	DOE/NNSA/LLNL, USA	IBM	Blue Gene/Q
4	K computer	RIKEN Advanced Institute for Computational Science (AICS) – Japan	Fujitsu	Tufo (6D Torus)
5	Mira	DOE/SC/Argonne National Laboratory, USA	IBM	Blue Gene/Q
6	Piz Daint	Swiss National Supercomputing Centre (CSCS), Switzerland	Cray	Cray Aries
7	Stampede	Texas Advanced Computing Center/Univ. of Texas, USA	Dell	Infiniband FDR
8	JUQUEEN	Forschungszentrum Juelich (FZJ), Germany	IBM	Blue Gene/Q
9	Vulcan	DOE/NNSA/LLNL, USA	IBM	Blue Gene/Q
10	-	Government, USA	Cray	Cray Aries

Table 3: 10 most powerful supercomputers in the world derived from [15]

Apart from these three categories there are some other topologies with no switches in which processing nodes are connected directly to each other. That is why they are given the general name of *Direct Networks*. In fact bus topology is also an example of direct networks but there are many more topologies which fall into this category. Trees, fat trees, Meshes and tori are among other direct network topologies.

The most powerful supercomputers at the present time are also divided when it comes to using switches. Table 3 lists the 10 most powerful supercomputers as in June 2014 according to TOP500 website. Here we review some of widely used network architectures some of which use switches and some others are direct networks. But before that we will review a number of metrics for network performance measurements.

2.2.1) Network Properties and Performance Metrics

There are a number of metrics to measure the performance of an interconnect network. We have already had some short discussions about latency and bandwidth. Other network properties and performance metrics are introduced in this section.

2.2.1.1) Diameter

In a network the diameter is the maximum number of hops in the shortest path from any given two nodes. In a bus topology, for instance, the network diameter is 1. Also, in a fully connected network (a network in which every node is connected to any other nodes via a dedicated direct link) the diameter is 1 as well. The shortest path between two nodes sometimes is referred to as *minimal path*. In some occasions non-minimal paths may be chosen for a data transaction, for instance, because of minimal paths being blocked by other transactions.

In light of developments in packet routing methods the importance of number of hops in an interconnect network is reduced. In older methods (e.g. *store-and-forward*) the number of hops is one of main factors in determining the overall latency (hardware latency plus software latency). But newer methods like *cut-through* switching and *wormhole* routing are capable of hiding most of the hop cost; therefore, the number of hops is less important when these methods are used [4]. The aforementioned categories of routing algorithms will be briefly reviewed later in section 2.3.

2.2.2.2) Bisection Bandwidth

Bisection bandwidth (sometimes known as Bisection width) is the bandwidth across smallest cut that divides network into two equal halves. This metric is important particularly for algorithms in which all processors need to communicate with all others. If the network is not symmetric along its different axes, the bisection bandwidth is

measured across the narrowest part of the network. The bisection bandwidth of a bus network is the same as the bandwidth of a single link; while the same metric for a ring topology is twice the bandwidth of a single link.

2.2.2.3) Throughput

The aggregate rate of successful message delivery over all links in a network is known as the overall throughput of that network. The throughput is usually measured in bits per second. Packet per second and packet per time slot are also used. The throughput of a network is different from the aggregate bandwidth as throughput is only about real data transmitted over data links rather than packet overheads added by communication protocols. In theory, the maximum throughput is equal to available bandwidth; but in real world bandwidth is just an upper margin for throughput, and throughput cannot reach that limit for reasons including transaction overheads.

2.2.2) Fat Tree and Hypertree

First introduced by Leiserson [22], fat trees have the standard tree topology with one exception which is the links are not all of the same *width*. Here the word *width* is used to refer to the transfer capacity of a link. The links closer to the root of the tree are *fatter* than those further from the root. In other words, the root of a fat tree is connected to its direct children with high bandwidth links and the bandwidth decreases as we traverse down towards the leaves.

Based on the assumption that the root of the tree and nodes close to it face higher amount of traffic, faster links are dedicated to those nodes while leaves of the tree (or nodes close to them) do not experience heavy traffic over their links. A fat link can be either a single high bandwidth link or more than one low bandwidth link (Figure 5).

Tianhe-2, the world's fastest supercomputer (June 2014) has a network called *TH Express-2* which is a fat tree with 13 switches, each with 576 ports at the top level [23].

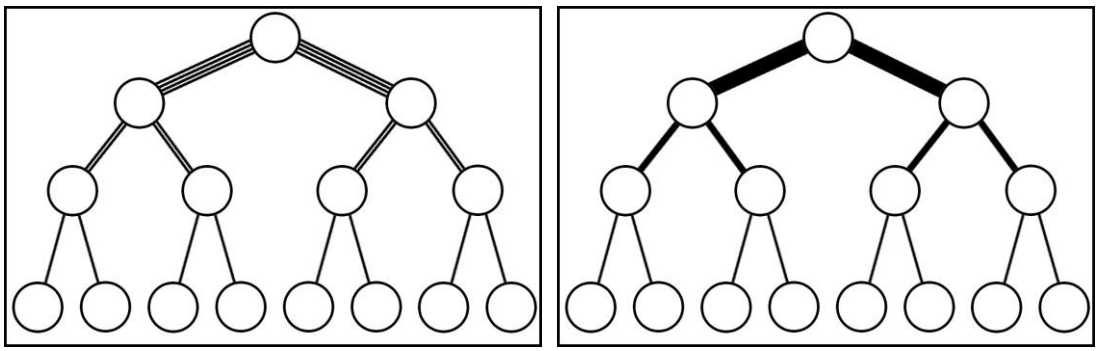


Figure 5: Two implementations of fat trees. The widths of the edges represent the link bandwidth.

The idea of fat trees has also been expanded to something called Hypertrees. A hypertree is a combination of two topologies: binary trees and fat trees both of which

can be visualised as 2D structures; however, in contrast with both of those topologies, a hypertree is a 3D topology. A hypertree can be visualised as a tree (binary or k-ary) of depth x while at the same time it can be regarded as a number of upside down binary trees of the same depth (Figure 6). Because of this dual nature of hypertrees, they are not trees as we know in graph theory. This is because in this topology a node can have more than one parent nodes.

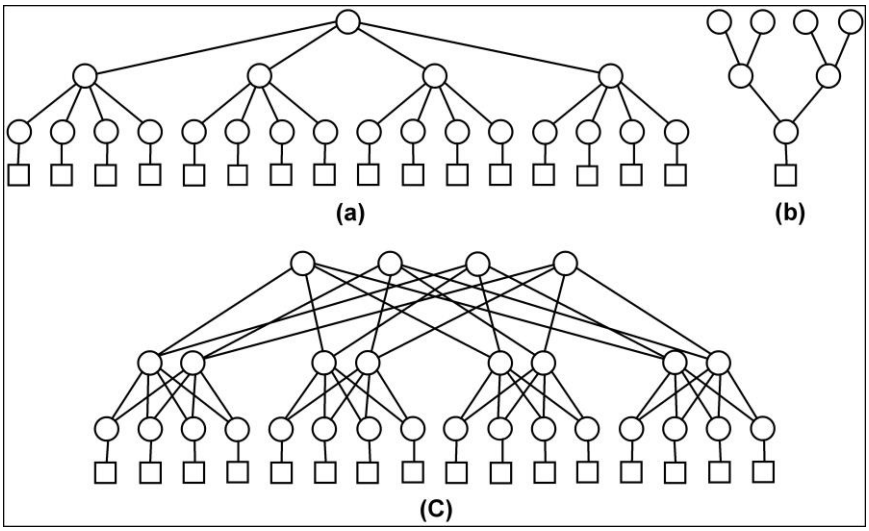


Figure 6: (a) A k-ary tree; (b) An upside down binary tree and (c) a hypertree as a combination of (a) and (b)[5]

Some applications well suit fat tree and hypertree networks. The divide-and-conquer nature of FFT, for instance, matches the hierarchical structure of fat trees. This makes these networks different from networks like mesh or torus in which the network has a more uniform structure.

2.2.3) Butterfly

Butterfly topology is one of the most well-known multistage interconnect networks. In a multistage network nodes are connected via one or more layers of switches. The number of switches and the number of inputs of each switch (AKA the switch radix) depend on design criteria. Kim et al. [24] has shown that the radix of the switches for a massively parallel computer with multistage interconnect network is beyond any technology available in present or foreseen for near future if data is about to be switched to its destination in only one hop (Figure 7).

Therefore it is believed that a single stage switch network is impractical for modern day large supercomputers. A layered switch network saves the network installation cost since a high radix switch is more expensive than a number of smaller radix switches with the same number of aggregate inputs. On the other hand, the wiring costs increase

[5] This figure is a reconstruction from an Internet page: http://blog.yam.com/snese1007/article/5659839.

hen the number of switches increases. The bandwidth of each input/output line and its effect on the number of lines is also an architecture issue. Using lower radix switches means that each line can have a higher bandwidth (given a fixed bandwidth for the whole switch); while higher radix switches need to be implemented using lower bandwidth cables. The trend in state-of-art HPC systems is to use higher radix switches with lower bandwidth lines (larger number of thinner inputs to switches) [25].

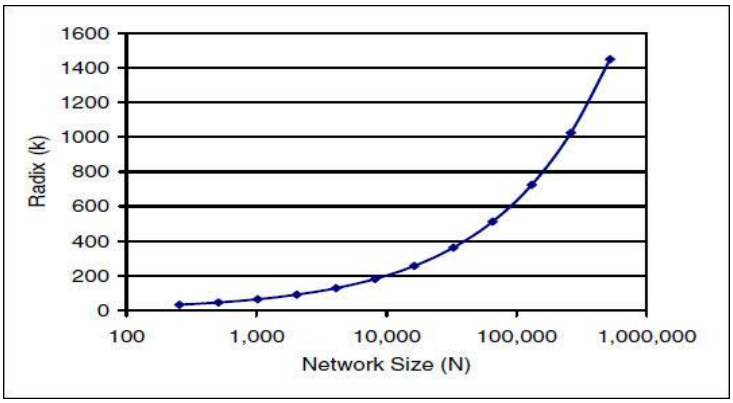


Figure 7: "Radix (k) of the routers required to scale the network (N) if only one global hop is required for each packet" [24].

The simplest building block of a Butterfly network is a 2*2 switch (Figure 8.a) which operates in two modes: pass-through and cross-over. An Omega network (Figure 8.b) is a three layer network made of those switches. In this network the IDs assigned to nodes help switches decided how to operate. At each stage if the corresponding bits of sender and receiver nodes IDs are the same the data passes through otherwise it crosses over the switch. Switches of higher radix can be implemented by combining proper number of 2*2 switches. As Figure 8 shows, the number of stages needed relates to $\log_n(N)$, where *n* is the switch radix and *N* is the number of nodes. In Figure 8, n=2 and N=8. An increase in the number of nodes affects the total number of switches in two ways: first: more switches are needed in any stage; second: more number of stages may be needed.

Knowing that the network cost directly relates with the number of switches, the relationship between the network installation cost and the number of nodes can be expressed by Eq. 9:

$$\boldsymbol{Cost(N) = Cost(1) * m = Cost(1) * \frac{N}{n} * log_n N = O(N * log_n N)} \qquad \textbf{Eq. 9}$$

Where *N* is the number of nodes, *m* is the number of switches and *n* is their radix.

The Butterfly topology is derived from such a network. A *k*-ary *n*-fly Butterfly network consists of *n* layers of switches each of which having *k* switches of radix *k*. The overall capacity of this network is k^2. The idea of Butterfly topology has then been extended to Flattened Butterfly topology [26]. Figure 9 shows two Butterfly networks (a, c) turned to Flattened Butterfly (b, d) by replacing all layers of switch with a single switch.

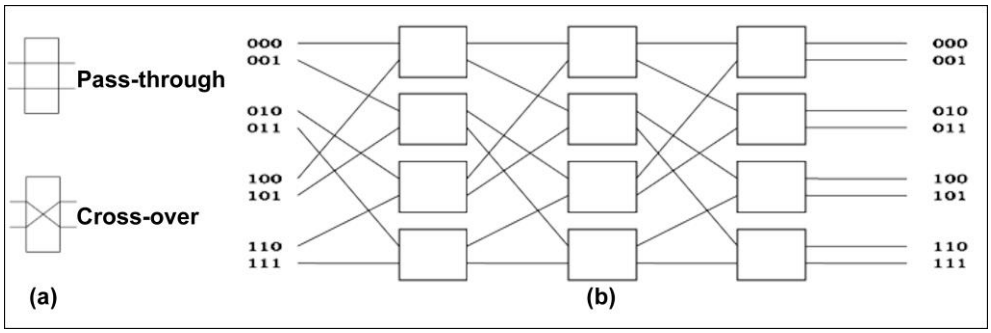


Figure 8: (a) Two modes of a 2*2 switch; and (b) an omega network made of 2*2 switches[6]

All unidirectional connections between different layers of switches is now replaced by bidirectional (or two unidirectional) connections (red lines in Figure 9). This simplifies the network and also decreases the number of switches needed for the network and therefore decreases the cost.

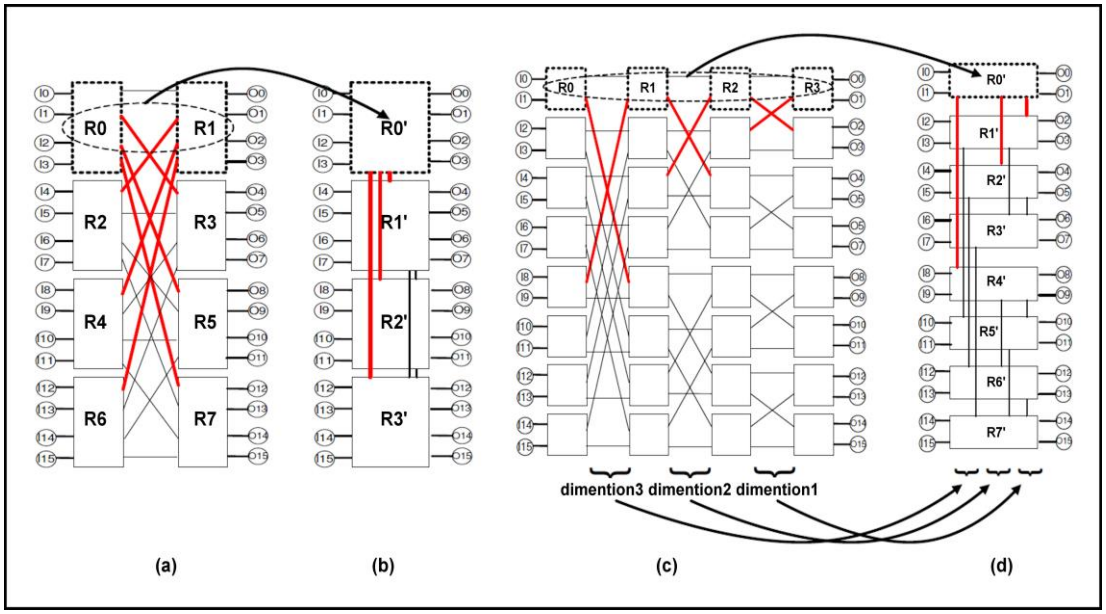


Figure 9: Butterfly topologies (a, c) and Flattened Butterfly topologies (b, d) as projected in [26]

2.2.4) Dragonfly

Dragonfly is the second multistage interconnect network discussed in this thesis. This is a topology mainly designed and developed by Cray and Stanford University [24]. This topology is inspired by both Butterfly topology and direct networks. It has a layered structure. In the low layer processing nodes are connected via switches to make "*groups*" of nodes (Figure 10.a).

This can be a Flattened Butterfly network, a fully connected network or any other topology the architects find suitable for the specific purpose of a network. In the top layer Dragonfly is a direct network which connects groups as a fully connected (all-to-all) network via optical or electrical connections (Figure 10.b). A Dragonfly topology is

[6] The images are taken from [290].

described by the group size, the number of groups and the number of links connecting each group to all of the other groups. Figure 11 shows a sample implementation of Dragonfly topology.

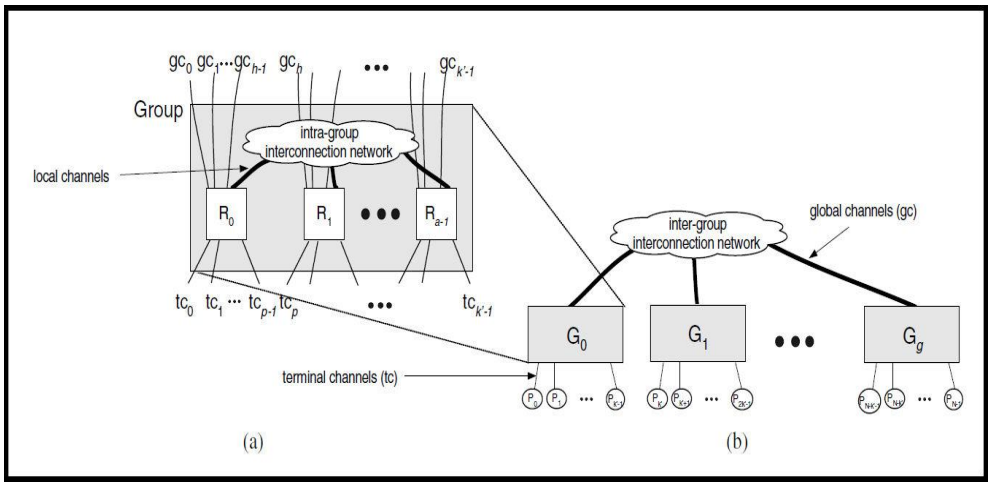


Figure 10: (a) Diagram of a group and (b) block diagram of a Dragonfly topology made of many groups [24].

Cray XC30 (Aries) technology is an implementation of Dragonfly topology. A Cray XC30 group consists of two cabinets (6 chassis). The number of XC30 groups depends on the maximum allowed length of optical or electrical cables used for inter-group connections (Figure 12).

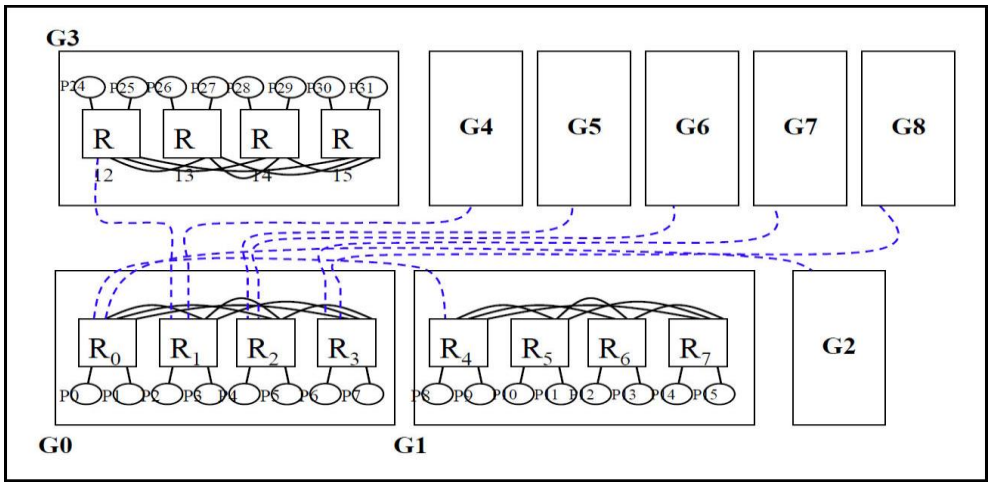


Figure 11: A sample implementation of Dragonfly topology as presented in [25]

The maximum group size depends on the maximum length of electrical links which in turn is determined by links' data rates. The actual length of copper links which connect chassis is 3m or less [27].

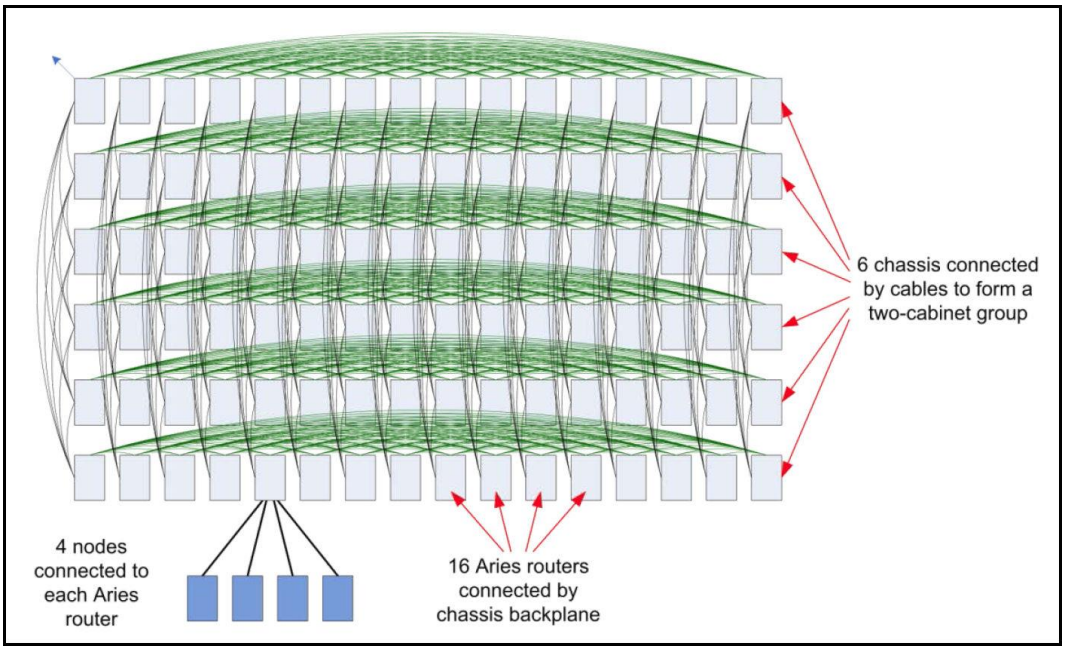


Figure 12: Structure of a Cray XC system's electrical group as projected in [27].

2.2.5) Mesh, Torus and Hypercube

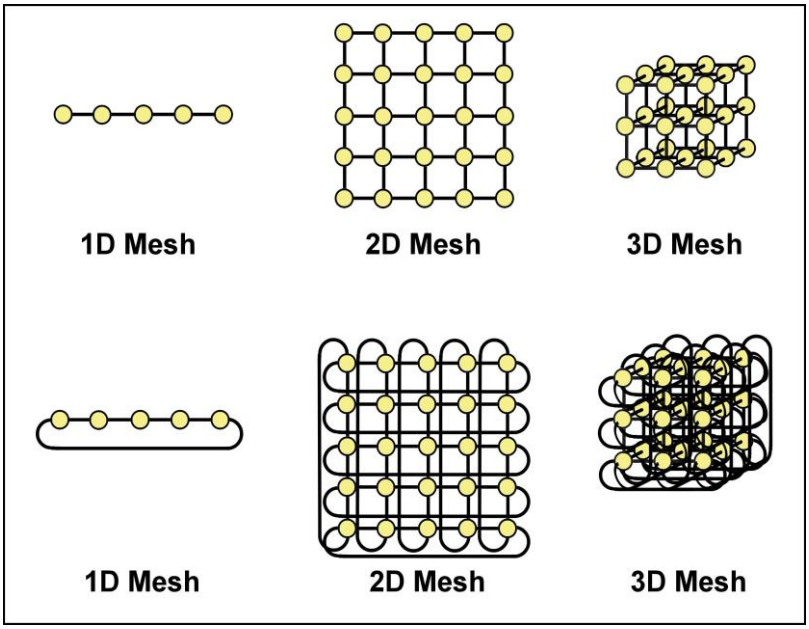


Figure 13: Mesh and torus topologies in 1D, 2D and 3D

In contrast with Butterfly and Dragonfly topologies, mesh, torus and hypercube are all instances of direct network topologies. All three of topologies can be implemented in n dimensions ($n \geq 1$). A 1D Mesh topology is basically a string of nodes connecting to each other in a linear way. A 1D torus is the same as 1D mesh with the exception that the first and last nodes are connected to each other via a direct link. A 1D torus is in fact a ring topology. A 2D mesh and a 2D torus are 2D expansions of the 1D mesh and torus. Higher order mesh and torus can also been made in the same way (Figure 13).

A hypercube is an expansion of the idea of cubic topology in which all neighbours of a node are orthogonal with it. Having orthogonal neighbours is an advantage for this

topology because a network of this type has a low diameter while it accommodates a considerably large number of nodes (i.e. nodes can be reached by any other nodes in relatively low number of hops). Figure 14 shows cubes and hypercubes of up to 5D. Higher dimension hypercubes are implemented with the same fashion. Cray's Gemini [28] is an example of using 3D mesh networks. IBM's Blue Gene/Q is based on the idea of using a 5D hypercube as interconnect network.

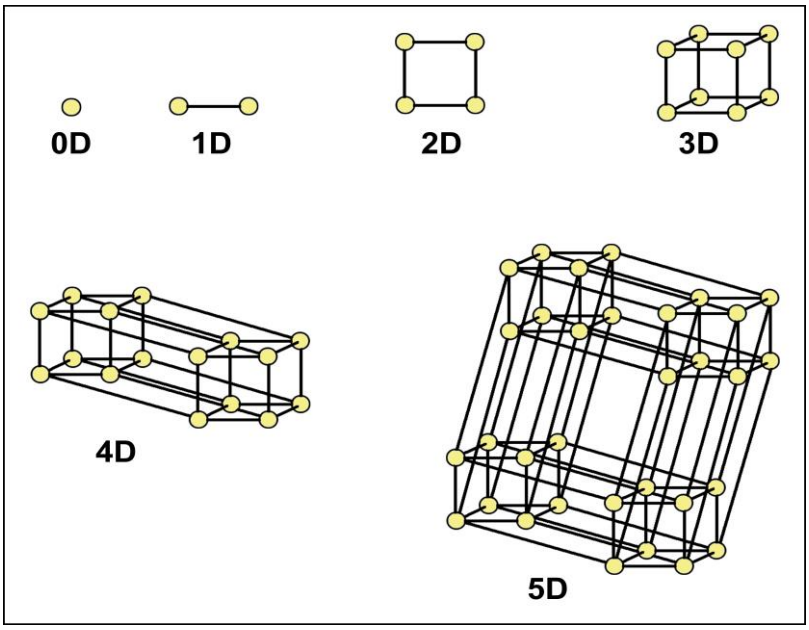


Figure 14: Cube and hypercube topologies of dimension 0 to 5

2.2.5) Infiniband

IBA Architecture (IBA) is a standard designed and promoted by key players in electronics industry emerged from merging of two previously proposed standards. Supported by Intel, Microsoft and Sun *Next Generation I/O* was merged with *Future I/O (ngio)*, which was developed by Compaq, IBM and HP, in 1999 to make the first drafts of IBA.

The association behind this standard is called the Infiniband[SM] Trade Association (IBTA) co-chaired by IBM and Intel. Other major contributors are Dell, Compaq, HP, Microsoft and Sun. Sponsor companies include 3Com, Cisco Systems, Fujitsu-Siemens, Hitachi, Adaptec, Lucent Technologies, NEC and Nortel Networks. According to [29] IBA is introduced in order to standardise reliability, availability, performance and scalability on different levels to solve some issues bus-oriented approaches have. IBA standardises servers' I/O models as well as inter-server transactions.

IBA is not a fixed architecture for one or more purposes; instead, it gives the architects a range of options on switches, links, endnodes and subnet managers to find the best design for their networks. IBA suggests a network of switches as a backbone for the

interconnect network. Subnet managers are in charge of establishing and managing subsets of the whole network. Switches facilitate finding proper path for messages from sender and receiver endnodes.

In addition to its original usage (in server systems) it is now a favourite architecture for HPC platforms and data centres. Major data base software developers like Oracle has invested a lot in using IBA's potentials and it has become the most powerful architecture for enterprise data centres.

IBA uses serial data links that can be used in one of the following data rates:

- Single data rate (SDR),
- Double data rate (DDR),
- Quad data rate (QDR),
- Fourteen data rate (FDR)
- Enhanced data rate (EDR).

Infiniband 2010 roadmap [30] has announced that two more data rates are going to be added: high data rate (HDR) which is due in 2014 and next data rate (NDR) which is not yet due for a specific time. The data rate of these two is unknown at the moment. IBA gives the option to vendors to aggregate 2, 4, 8 and 12 links to make them more powerful. Figure 15 illustrates the present and future data rates IBA supports in different modes.

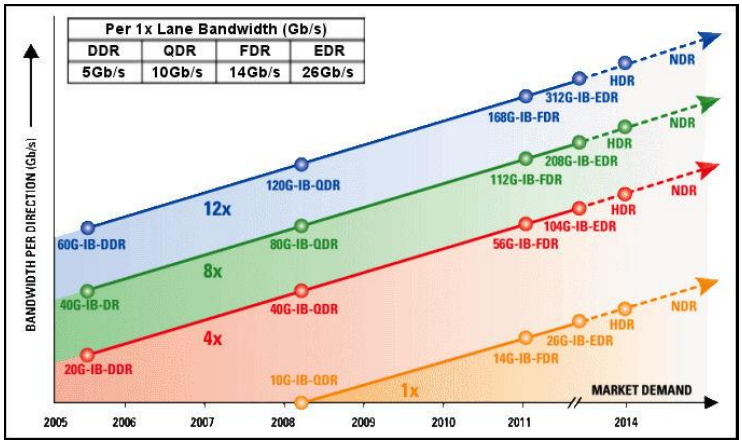


Figure 15: IBA's current and future data rates adopted from 2010 roadmap [30]

There are three types of devices involved in an IBA network: endnodes, switches and subnet manager. The first two are passive entities while subnet managers are active. A network needs at least one subnet manager but it can accommodate more than one subnet manager. In this case the subnet manager that has initialised the network is called the master subnet manager. To cover all types of communication needed in different applications, IBA supports a number of communication service types listed below:

- Reliable Connection (RC)

- (Unreliable) Datagram (UD)
- Unreliable Connection (UC)
- Reliable Datagram (RD)
- Raw IPv6 Datagram and Raw Ethertype Datagram (optional)

Among the services listed above reliable datagram is of importance because there are some analogy between that type of service and the communication mechanism adopted for this thesis. In a network of multicore processors up to thousands of connections between different processes on different cores on different processors may be needed. To handle such a complex situation what many commercial databases do is to dedicate a process on each processor for communication purposes. Such a process gathers all the messages from local cores and sends them to target nodes. It also receives and processes all incoming messages to decide what process on local node should receive the message. Even such a solution can increase the complexity of the system especially when the number of nodes increases. What IBA suggests is to use RD services. In this type of communication each node has a set of software entities called End-to-End (EE) context. All messages to and from a remote node go through that remote node's dedicated EE. It reduces the complexity of the system and scales well with the number of processes and the number of nodes.

IBA-based networks facilitate memory access through three software entities: memory region, memory windows and protection domains. In most cases each system process has its own protection domain. It is also possible for a group of processes that are related through a shared memory to have a single protection domain for all the members of the group. Inside a protection domain it is possible to have one or more memory regions. Memory regions are in charge of mapping between physical and virtual addresses. A registration process is needed to create a new region which will generate a key called the L_Key which is necessary for memory accesses to the same region. The key is used to find the actual map as well as checking the access rights. A different key called the R_Key is needed for accesses to remote locations (e.g. to different endnodes). Memory windows are the smallest of the three with byte granularity and are used to restrict an individual operation to a specific set of data.

To regulate the relations between hosts and their clients in a network of multiple hosts a partitioning mechanism is introduced in IBA. It uses a separate set of keys and key tables to ensure that remote hosts cannot access local devices unless such a transaction is done through local host.

2.2.6) Blue Gene/Q

Blue Gene is a platform made by IBM for massively parallel computing. Three generations of this system are introduced. Blue Gene/Q is the latest version. The system architecture of Blue Gene/Q (which is the latest version of Blue Gene at the moment) is shown in Figure 16. What is particularly of interest is the internal structure of the processor network of Blue Gene. Some other parts of the system including the functional LAN can be implemented by Infiniband standards. Site LAN and functional LAN can be merged or they can be separated. Since Blue Gene is a diskless system, it needs a separate file system that is connected to the main racks through the functional LAN.

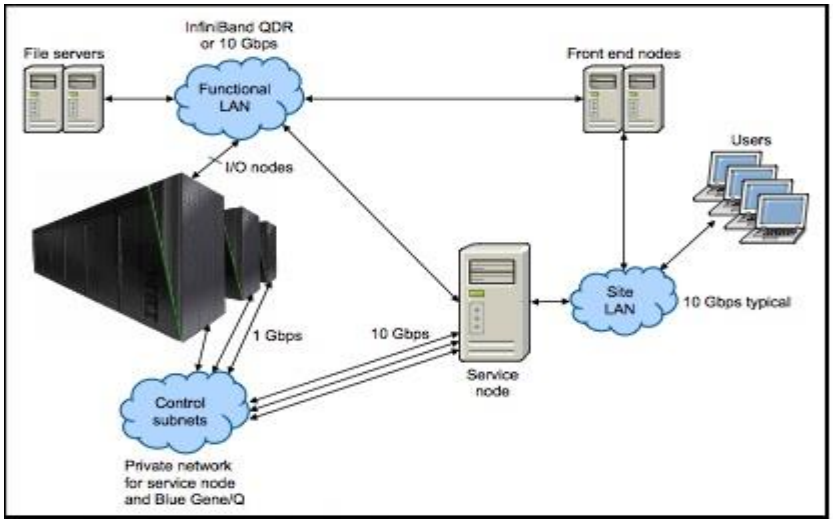


Figure 16: IBM Blue Gene/Q system architecture as illustrated in [31]

Figure 17 shows how multi-core processors come together to form a rack of processors and finally build the whole Blue Gene system. The processors usually used are 16 cores IBM PowerPC® processors. The nodes in a node board form a 5D (of size 2*2*2*2*2) torus to have the fastest access time between nodes. The signalling rate between these nodes is 4Gbps. Midplanes are also 5D tori (of size 4*4*4*4*2). This means that each midplane accommodates 512 nodes. A rack can have one or two midplanes and the whole Blue Gene/Q system can scale up to 512 racks. The network size of a full scale Blue Gene/Q system can be 32*32*16*16*2. Knowing that a torus has a round formation (i.e. the nodes on two ends on each dimension are directly connected) the maximum number of hops between nodes in that scale is 16+16+8+8+1=49.

The connection between nodes on a node board is through optical fibres. Electrical, Ethernet and Infiniband links are also used on different parts of the system. The data rates of the links are in Gigabit per Second range (1Gbps, 10Gbps and 40Gbps as dictated by design criteria). I/O drawers are made of the same processors but each

drawer has 8 processors and is just used for I/O purposes. They can be placed on racks (one, two or four drawers per rack); alternatively a separate I/O rack can be used.

The cooling system of I/O compute cards is based on air cooling; while, water is used for cooling the compute cards. According to an IBM document [31] 91% of cooling system is based on water and the rest is by air. The power consumption for each rack is 80kW (typical) and 100kW (maximum). The theoretical performance of a rack is 209.7 TFlops. A sustained 170 TFlops performance is reported [31]. It can be concluded form the above numbers that the typical power efficiency of a Blue Gene/Q rack is 0.47 kW/TFlops.

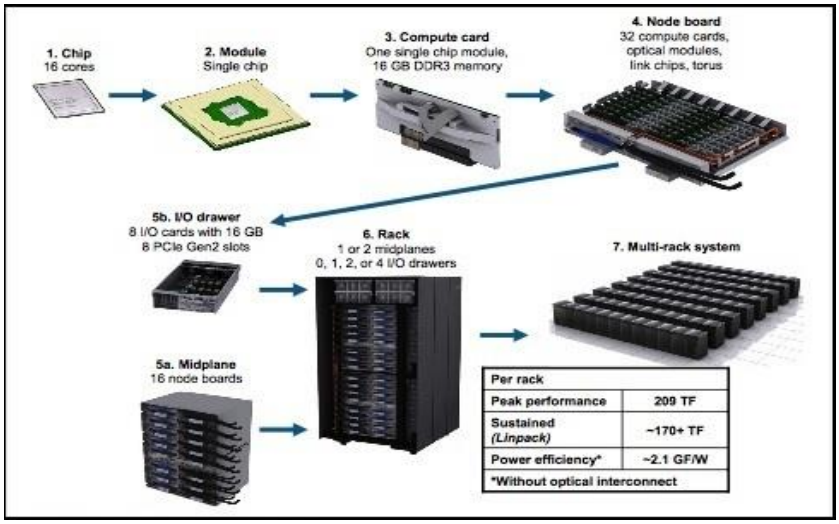


Figure 17: Blue Gene/Q hardware overview as illustrated in [31]

2.2.7) Network Topology Comparison

After reviewing a number of interconnect network topologies we are able to compare them based on a number of network properties and performance metrics. Table 4 is a comparison between some of the most popular topologies based on the Degree of nodes (the number of connections a node has to other nodes), the network diameter (minimal path) and the dissection bandwidth of the network. It is also assumed that all links have the same bandwidth of *x*. In fat tree topology the nodes are connected through switches; therefore instead of degree of nodes the radix of switches is included in Table 4.

Another point is that Dragonfly topology is not included in Table 4 because its metrics depends on the design decisions and varies from one supercomputer to another. However, the minimum number of hops between two nodes (including inter-group connections) via a minimal path is 5. Not all the transactions are guaranteed to take place via minimal paths. According to *Alverson et al.* [27] although in most cases (except for the largest systems) non-minimal transactions have six or seven hops, the absolute maximum number of hops in non-minimal paths is 10. The bisection

bandwidth is defined by the inter-group connections which are typically high-speed and expensive optical links (compared to cheaper electrical intra-group connections).

Network Topology	Nodes	Degree	Diameter	Bisection Bandwidth (*x)
Bus	K	K-1	1	1
Ring	K	K-1	1	2
Crossbar	K^2+2K	4	2(K+1)	K
1D mesh	K	2	K-1	1
2D mesh	K^2	4	2(K-1)	K
3D mesh	K^3	6	3(K-1)	K^2
nD mesh	K^n	2n	n(K-1)	K^{n-1}
1D torus	K	2	K/2	2
2D torus	K^2	4	K	2K
3D torus	K^3	6	3K/2	$2K^2$
nD torus	K^n	2n	nK/2	$2K^{n-1}$
kD hypercube	2^K	K	K	2^{K-1}
Fat tree (n-ary)	n^K	n	$2\log_n^K$	K/2
Butterfly	$(K+1)2^K$	4	2K	2^K

Table 4: Comparing some of the most popular HPC interconnect network topologies (mainly based on data from [32])

There are other performance metrics like latency and energy consumption that depend not only on the network topology but also on the design and implementation of the supercomputer. Different implementations of a single topology may differ in latency and energy consumption. For example, the length of the electric/optic cables plays a big role in determining those performance metrics.

Some other metrics are also affected by software (e.g. message passing API and routing algorithm) and operating system installed on a supercomputer. Network throughput is usually measured using the volume of *real data* transferred through a network; therefore, the amount of overhead attached to real data is important in determining the network throughput. Such overhead depends on I/O protocol, routing algorithm and underlying operating system among other factors.

For these reasons, the aforementioned network performance metrics should be compared on a system-by-system basis rather than a topology-by-topology basis. Here are a number of comparisons and measurements on different topologies.

The hardware latency in different supercomputers of different sizes and topologies are mostly in microseconds range. According to *Chen et al.* [33] a 16*16*16*12*2 Blue Gene/Q computer operates with a hardware latency of 6.5 µs. The latency for a dragonfly network in a situation in which there is low traffic (quiet network) is estimated 2 µs or less [27]. On an older HPC platform like IBM SP system the hardware

latency is reported to vary from 0.5 µs to 1.5 µs; while the user-level software latency is around 36 µs [4]. *Kim et al.* [26] have studied the effect of routing algorithm on latency in a Flattened Butterfly topology. They have also measured how choice of topology affects the latency. Also included in their paper are studies on how changes in number and radix of switches can affect the hardware latency in Flattened Butterfly topology. As reported by Cray [28], the end-point latency for a Gemini network is 1.0µs or less for a remote put, 1.5µs or less for a small MPI message (all measured for a quiet network).

The optical and electrical cables used in Dragonfly technology are tested to measure their energy consumption. Results are reported by *Kim et al.* [24]. Those results are shortened and shown in Table 5. The table compares two types of optical cables used for long inter-group connections and one shorter electrical cable used for intra-group connections. In the same document the effect of routing algorithm on network latency is studied.

Cables	**Distance**	**Data Rate**	**Power**	**E/bit**
Intel Connects Cable	<100m	20Gb/s	1.2W	60pJ
Luxtera Blazar	<300m	42Gb/s	2.2W	55pJ
conventional electrical cable	<10m	10Gb/s	20mW	2pJ

Table 5: The data rate and power consumption of a number of cables used in Dragonfly topology

Dally [25] estimates a 10pJ energy consumption for a floating point operation in yet-to-build Exascale computers. To compare this with the data communication energy cost, he says a single word transaction between two nodes in a cabinet is equivalent to 32 FLOPS in terms of energy consumption. For the same transaction between cabinets his estimation is as energy consuming as 256 FLOPS.

2.3) Routing Algorithms

In previous section it is briefly stated that the packet routing algorithm (AKA packet switching algorithm) plays a role in the performance of an HPC system. Here we will briefly review three important categories of routing algorithms used for interconnect networks of supercomputers. The difference between these methods is in the way they handle multi-hop communications.

2.3.1) Store-And-Forward

Store-and-forward algorithms are based on the idea of relaying a message only after receiving all of it in an intermediate node (i.e. a node that is not neither the sender nor the receiver of the packet). It is important for intermediate nodes using this category of

algorithms to check the integrity of the packets before transferring them to the destination (or another intermediate) node and for this reason they cannot start the relay process before having the entire packet. The advantage of such algorithms is the simplicity of implementation compared to other two categories.

Store-and forward is a good idea especially for networks with intermittent connectivity. Examples of such networks are highly mobile networks and networks of nodes scattered over open rural areas in which there is a lower control over the connectivity of nodes. This category of routing algorithms offers maximum error checking by sacrificing forwarding speed.

2.3.2) Cut-Through

The main objective of a cut-through routing algorithm is to start relaying a packet in an intermediate node before receiving all of its contents in order to increase the forwarding speed. Such an idea saves some transmission time as there will be some overlap between the reception of the packet from the transmitter and sending it to the receiver. This means that intermediate nodes may keep transmitting corrupt packets because they can only decide if a packet is corrupt after receiving the entire packet (although it can eventually detect the error by checking the packet CRC). In real world the relay of packet starts after reading the first 14 bytes of incoming packet [34]. For links with data rates of 10Mbps and 100Mbps it makes a 25µs and 7.5µs of latency respectively. A restriction this idea has is that in some circumstances it cannot be used because of the nature of the network. To implement this idea it is important to have a guaranteed continuation of stream of data in both receiver and transmitter links. Also the transmission speed on both sides should be almost the same.

A partial solution to this problem is introduced by a modification of this method called *fragment free*. In this method, the intermediate node starts relaying the packet after receiving the first 64 bytes of the packet. This is historically based on the fact that the minimum length of packets in IEEE 802.3 protocols is 64 bytes. Fragment free methods do not sent packets with less than 64 bytes (AKA *runts*) assuming that they are typically produced as a result of packet collision [34]. This method is expected to be a bit slower than pure cut-through algorithms when there is no collision (around 60 µs of latency according to [34]). Compared to cut-through, fragment free algorithms are capable of detecting more errors. The problem both pure cut-through and fragment free algorithms have is that it is assumed that the packet corruption occurs (or is detected) in the transmitter side. This is not always the case with wireless connections. This issue should

be taken into consideration if such algorithms are chosen for a platform like the one proposed in this thesis.

Another approach would be having a compromise between pure store-and-forward and pure cut-through in a way that the node operates in cut-through mode when it is possible; otherwise it retreats to store-and-forward. In practice, nodes change their switching mode if the number of runts and CRC errors cross a threshold. The Table 6 summarises how the change in switching strategy happens:

Switching mode	Detects	Then adaptive mode changes switching mode to
Cut-through	High number of CRC errors	Store-and-forward
	High number of runts	Fragment-free
Fragment-free	High number of CRC errors	Store-and-forward
	Low number of runts	Cut-through
Store-and-forward	Low number of CRC errors	Fragment-free
	Low number of CRC errors and Runts	Cut-through

Table 6: Adaptive switching strategy based on Intel documents [34]

2.3.3) Wormhole

This category of routing algorithms extends the idea of cut-through routing by introducing the concept of *virtual channels*. It is based on the idea of splitting a packet into smaller pieces called flits (**fl**ow control digi**ts**). The first flit of a packet is called the header flit contains the data needed for routing (e.g. the destination node's ID) with or without part of real data. The header flit can be followed by none, one or more than one data flits.

The difference between a flit and a packet is that two packets with the same source and destination may go through different paths and do not receive in the same order as they have been sent. But two flits of a packet always pass through the same path and in the exact order as they have been sent. Also, a packet has everything needed for routing; but flits do not have such additional data (except for the header flit). In fact it is the virtual channel mechanism that handles the routing tasks for header-less flits. If every flit had to carry the packet header it would have imposed a significant traffic overhead and consequently it would be very hard to compensate for such a loss in transmission time.

It can be said that one of the main differences between cut-through and wormhole routing is that in cut-through routing the buffers allocated in packet-level while in wormhole routing the buffers are allocated in flit-level. This saves considerable buffer size over the network.

A header flit needs to be allocated proper network resources in the intermediate node before being sent to it. The channel used for data transaction and a space on flit buffer are among those resources. These resources are adopted for other flits as well. That is why non-header flits do not need to carry the routing information with them. The process of resource allocation repeats as the header flit moves from one node to another. However, there is a common problem with both cut-through and wormholing methods and that is they do not let another packets cannot cut in until the whole packet is sent. But the good news is, in wormhole methods the physical and virtual channels are separated and a physical channel can be associated with several virtual channels. This means that different packets can use different virtual channels on the same physical channel at the same time.

Wormhole routing (switching) is the algorithm of choice for many modern supercomputers including Cray's 3D torus Gemini, Dragonfly-based Cray's Aries and Flattened Butterfly technologies. It helps reducing the overall software latency and increasing the throughput.

2.4) Parallel programming Models

Regardless of the network topology choice, the model of parallel programming can be classified into a number of classes. Two of the most popular among these models are *shared memory and message passing*. The main difference between these two is in their global image of the system (and memory in particular) [35].

Shared memory models (Figure 18) look at the whole set of processors, memory modules and other elements as a single machine. The key point is that there is only one memory address space [11]. All memory modules are subsets of this global space. Shared memory multi-processor machines can be compared with multi-threading in single-processor systems. The only difference is now we are dealing with real parallelism while in multi-threading a single processor uses time sharing techniques to give the users the impression of parallel multi-tasking.

Like multi-thread systems, in the shared memory, each processor has its fair share of private and public memory space but all public and private memories are parts of a global memory space. Since there is a global and single memory space, all the memory accesses are handles by simple read/write instructions. There are two major shared memory schemes regarding the memory access time [36]. The memory access time can be uniform or non-uniform based on the system design. Uniform memory access times can be seen in Symmetric Memory Multiprocessors (SMP) [36] (Figure 19).

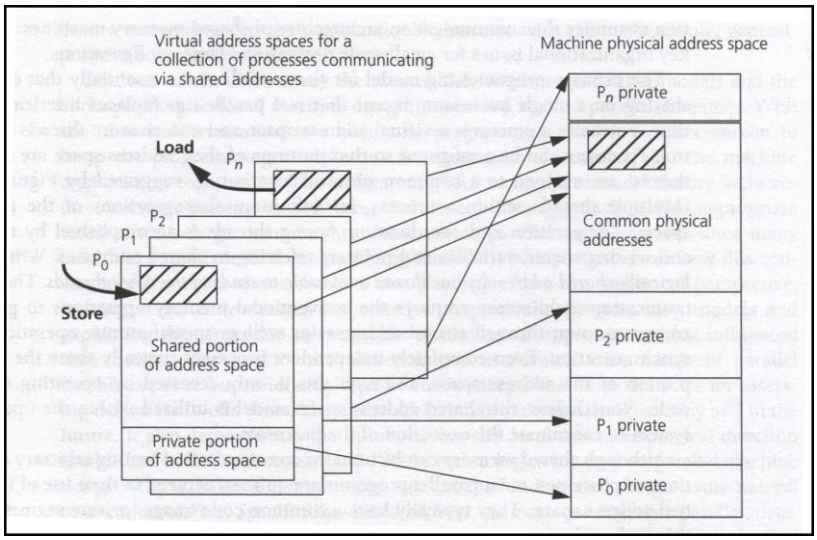


Figure 18: Shared Memory model and its similarity to multi-threading in single-processor systems; from [11]

This is in fact a natural generalisation of bus technology in parallel machines. All processors and memory modules are connected to a single bus; therefore, there is a unique memory access time for all the processors.

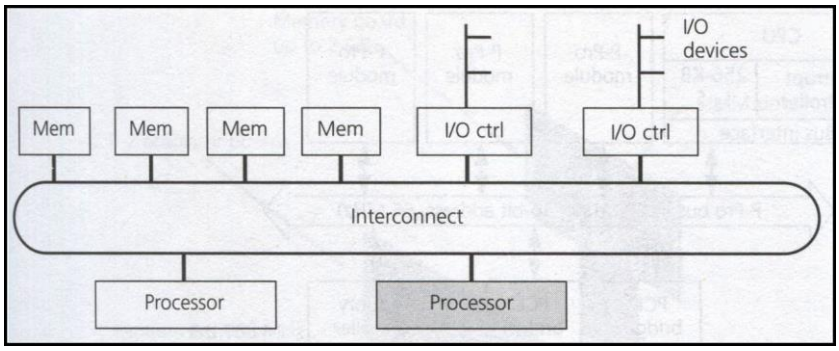


Figure 19: An SMP architecture as shown in [11]

In many applications processors prefer to keep some of their most frequently used data as close to them as possible while other processors may not necessarily so eager to have fast access to those pieces of data. Such an imbalance between memory access times leads us to another major group of shared memory methods called Non-Uniform Memory Access (NUMA) methods [36] (Figure 20). Both these methods are popular between computer architects.

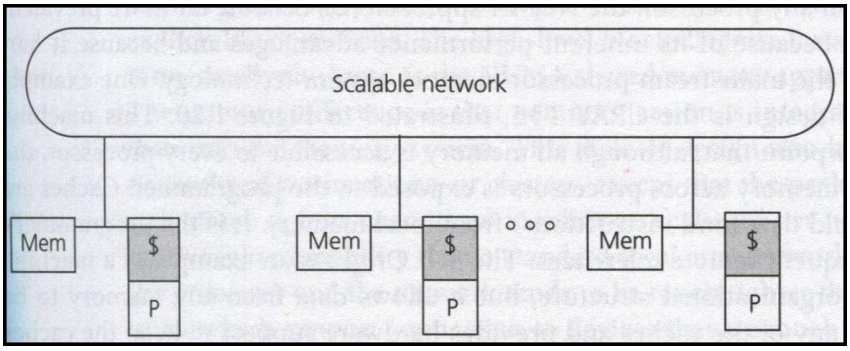


Figure 20: A NUMA architecture as shown in [11]

In contrast to shared memory approaches, in message passing method (Figure 21) there is no global memory address space; instead, each processor is in charge of its own local memory space. The processors can communicate with each other to access other processors' private memory locations. In this method read/write instructions are replaced by send/receive instructions when it comes to remote memory location access.

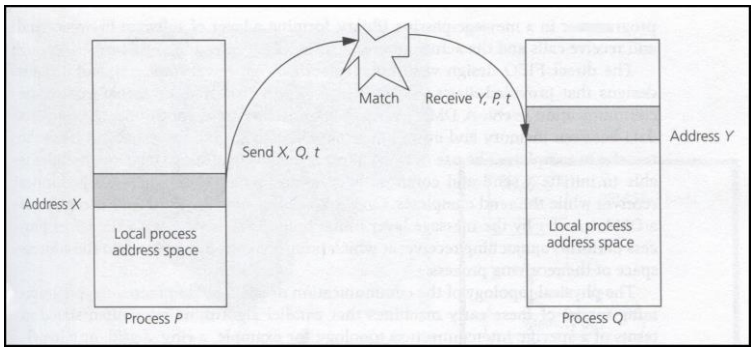


Figure 21: A message passing communication method derived from [11]

2.5) Wireless Communication

There is a range of commercially available radio devices already used in different applications. At the current stage, this thesis is only based on a simulated form of BC; therefore, the following review of available radio technologies is just for having a picture of the current level of technology rather than choosing between them for this thesis. Apart from off-the-shelf radios, researchers are constantly increasing the baud rate of wireless transceivers. Reviewing the state-of-art radio technologies still in their early stages is another part of the background reading in this thesis.

2.5.1) IEEE Wireless Protocols

IEEE 802 [37] is a family of protocols dealing with local area and metropolitan area networks. Among its 23 sub-protocols it is the IEEE 802.15 family that deals with wireless personal area networks (WPANs). IEEE 802.15 [38] consists of 7 task groups covering different aspects of wireless communications. Among these 7, two task groups are of particular interest namely IEEE 802.15.1 (Bluetooth) and IEEE 802.15.4 (ZigBee). These are protocols for short and medium range radio communications. Both protocols are widely used in domestic and industrial applications; however the baud rates they are designed for, are not high enough to be considered in the field of HPC. Figure 22 compares the ranges and data rates of IEEE 802 family members [39].

None of the protocols are specifically designed for very high data rate communications; however, they are good for low-energy communications in noisy environments. ZigBee [40] is a low-cost, low-power, wireless mesh network standard built based on IEEE

802.15.4 and is widely used in a range of applications including home entertainment and control, wireless sensor networks, industrial control, medical data collection and building automation.

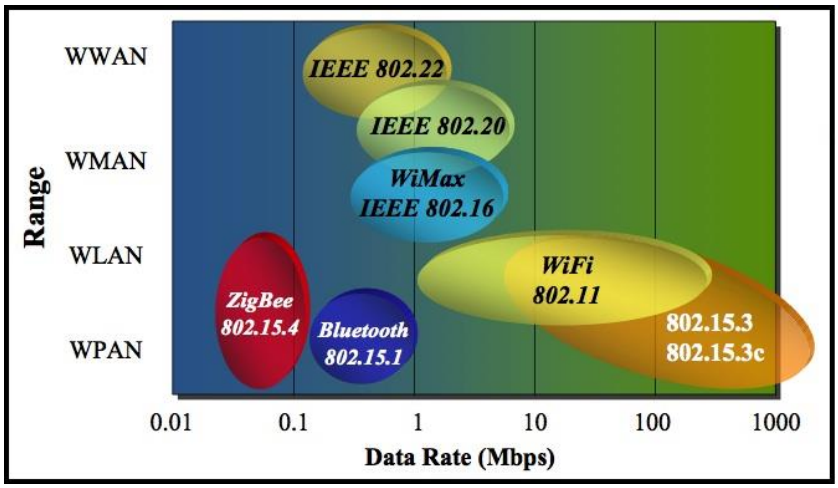


Figure 22: Comparing IEEE 802 protocols as projected by [39]

What all these applications have in common is that they do not need high data rates. Also in a normal situation for a large majority of its life time a ZigBee device is inactive (sometimes referred to as sleep mode) and they just send/receive data when they receive a signal (mostly from a local sensor). This way, they can keep the power consumption to a very low level for most of their time. As far as it is concerned with our research, ZigBee's biggest drawback is its data rate that is no more than 250 Kbits/Sec. Bluetooth [41] is another protocol built based on IEEE 802.15.1 protocol for low power short range communication and has become the dominant communication protocol for personal area network applications especially in home and entertainment, mobile phone interfaces and wireless connections between PCs and other devices. Its maximum data rate differs between its different versions and varies from 1Mbit/s to 24Mbit/s. Later on this report it will be shown why bandwidths supported by both ZigBee and Bluetooth are too low to be used in this research.

The highest data rates within IEEE 802 family belong to IEEE 802.15.3 and 802.11 (commonly known as Wi-Fi). IEEE 802.15.3's data rate varies between its different versions. Its 2003 version supports up to 55Mbit/s while its more recent versions including that of 2009 supports up to 3Gbit/s. IEEE 802.11 is designed for wireless local area networks. The communication range it is designed for does not match with what this thesis is dealing with.

The data rates supported by Wi-Fi differ between its numerous versions. Different characteristics of different versions of IEEE 802.11 protocol is summarised in Table 7.

The interesting point in this table is that the latest versions of 802.11 protocol supports Multiple-Input Multiple-Output technologies, which - as shown later in the thesis - has some similarities with the network presented in this thesis.

Version	Release	Freq. (GHz)	Bandwidth (MHz)	Data rate per stream (Mbit/s)	Allowable MIMO streams	Modulation	Approximate indoor range		Approximate outdoor range	
							(m)	(Ft.)	(m)	(Ft.)
—	Jun 1997	2.4	20	1, 2	1	DSSS, FHSS	20	66	100	330
a	Sep 1999	5	20	6, 9, 12, 18, 24, 36, 48, 54	1	OFDM	35	115	120	390
		3.7					—	—	5,000	16,000
b	Sep 1999	2.4	20	1, 2, 5.5, 11	1	DSSS	35	115	140	460
g	Jun 2003	2.4	20	6, 9, 12, 18, 24, 36, 48, 54	1	OFDM, DSSS	38	125	140	460
n	Oct 2009	2.4/5	20	7.2, 14.4, 21.7, 28.9, 43.3, 57.8, 65, 72.2	4	OFDM	70	230	250	820
			40	15, 30, 45, 60, 90, 120, 135, 150			70	230	250	820
ac (Draft)	Nov. 2011	5	20	up to 87.6	8					
			40	up to 200						
			80	up to 433.3						
			160	up to 866.7						

Table 7: 802.11 Network Standards derived from [42]

Regarding the size of a parallel computer, it is not expected that long distances of communication is needed; therefore, protocols like IEEE 802.16, 802.20 and 802.22 are not going to be considered.

The maximum number of nodes in a protocol is another concern. This is particularly important since the proposed system is supposed to accept large number of nodes (something in the range of hundreds of thousand or even a million nodes). This is a common deficiency of all IEEE 802 protocols.

As a conclusion all IEEE 802 family members support the communication range needed but they have a number of limitations. The first limitation is on the data rates which are mostly below 1 Gb/s and are not high enough for HPC systems. Another problem with most of the wireless modules of this family is their physical size. Although some ZigBee devices are made in very small sizes but the majority of IEEE 802 wireless modules occupy in large spaces particularly because they need large antenna. Also, the network sizes supported by this family of protocols do not look very promising. For these reasons it is necessary to look for other wireless communication technologies for the application sought in this thesis.

2.5.2) Inter-chip 3D Network of processors

In addition to parallel processing applications, the idea of arranging processors in a 3D grid has been investigated for inter-chip applications. *Akasaka*'s article in 1986 [43]

emphasises on the restrictions of a 2D grid of modules in a chip and encourages the researchers to look for 3D solutions for future ICs. The use of wireless connection makes it even easier to think about stacking chips on top of each other.

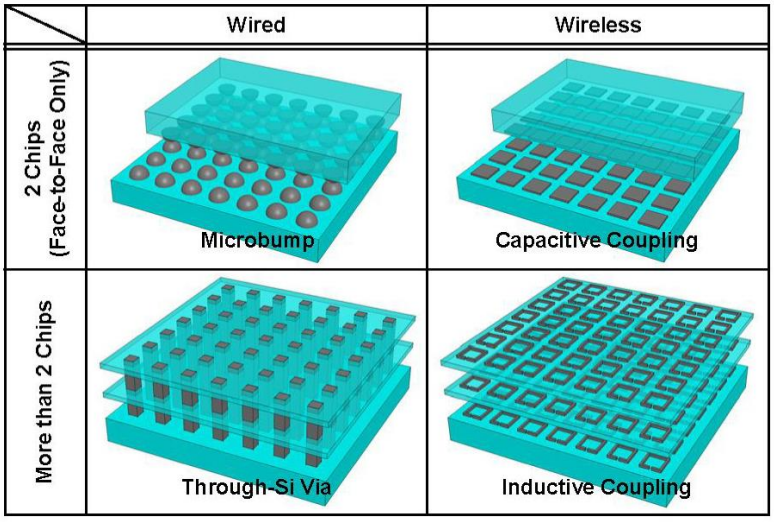


Figure 23: Different solutions for 3D stack of processor interfaces as projected in [44]

Basically there are four strategies for 3D assembly of chip stacks (shown in Figure 23). The first two, which are not wireless in fact, are microbumps and Through-silicon-via (TSV). The former introduces very tiny microscopic metal bumps on the surface of two chips that facilitate the transfer of data between them. In the latter solution a metallic rod that goes through the two (or more) adjacent chips are used to connect the data lines of the chips.

The other two solutions are capacitive and inductive coupling techniques. In these methods, the energy (in electric or magnetic form) saved by capacitors or inductors is used to transfer the electronic signal between two (or more) chips that otherwise have no other data connection. These two techniques are discussed in more details on section 2.5.3 when reviewing On-chip Short-Range High-Speed wireless communication technologies.

The restriction of both microbump and capacitive coupling is that both of them can connect no more than two chips. Moreover, the capacitive coupling can support only very short distances of 1 to 2 μm. On the other hand, although inductive coupling supports longer distances, it was not originally suitable for parallel communications because of its large power consumption. There are, however, some successful attempts (including *Sasaki et al.* [45]) to reduce the power consumption. Therefore inductive coupling is the most favourable technique in this field to date. As stated by *Miura et al.* [46] in January 2007: "The state-of-the-art inductive-coupling transceiver achieves the second highest bandwidth with the second lowest power and the smallest layout area".

Figure 24 compares some wireline and wireless inter-chip communication technologies in terms of performance and cost. Wireless approaches have a number of advantages over using wirelines or even over microbumps and TSV.

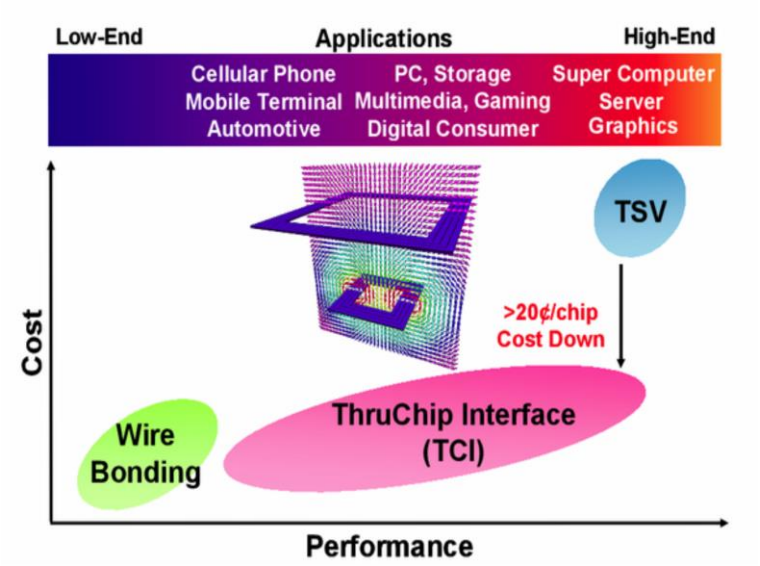


Figure 24: Cost/Performance Comparison between Wire Bonding, TCI and TSV as depicted by [47]

Miura et al. [46] list five advantages of wireless approaches over using wirelines, microbumps and TSV:

1) Since the coupling elements can be created at the same time as the other parts of the chip (as opposed to wireline methods that needs an additional mechanical process) the cost and time of the process is saved.

2) A micrometre distance between face-to-face chips can be achieved because of the elimination of an extra mechanical process in coupling techniques, which makes it easier to reduce the size of the finished product.

3) The non-contact wireless connection does not need an electrostatic discharge (ESD) protection module. This leads to even more cost and area reduction as well as higher communication speed.

4) Due to chips being detached from each other the reliability of the whole package is increased. It is also easier to test chips. According to *Salzman and Knight* [48] this can solve the known-good-die (KGD) problem[7].

5) Particularly for inductive coupling it is possible to save even more layout area by placing the transceiver circuits under the metal inductor.

When facing high demand for high-performance and high scaling one may consider use TSV for power delivery for chip stacks. There are, however, some concerns over the heat dissipation of such a scheme. The number of elements in such a plan is restricted.

[7] As defined in [287]:" The issue of not being able to test bare silicon is referred to as KGD (known good die) problem".

2.5.3) On-chip Short-Range High-Speed wireless communication

Short-range high-speed and low-power wireless communication has been a hot topic in VLSI design. This is mainly because the gap between the bandwidth inside chips and the bandwidth between chips is increasing. One of the solutions to fill this gap is the idea of replacing wirelines with wireless connections between modules in a chip (or to connect two chips). *Matsuzawa* investigates possible solutions for an RF System-on-Chip (RF-SoC) IC based on main design criteria. In his work [49] he concludes that CMOS and SiGe can be used to embed RF technology but the keys to massive use of RF in inter-chip applications are "continuous cost reduction, ease of function and specification change" and also it is important to resolve process portability issues. Basically there are three categories of short-range wireless communication:

- Capacitive coupling;
- Inductive coupling;
- Radio waves.

On 2008 *Moore et al.* [3] did a survey on chip-to-chip communication technologies. They concluded that compared to other technologies inductive coupling is a better choice for that purpose. According to them the radio devices will struggle crossing 1Gbps line unless they resort to MIMO. Although recent developments have proven that conclusion was not 100% correct, it seems that their prediction over the restriction over power consumption of radio devices is legitimate. Their analysis over *the crossover point* is of importance.

The crossover point as they define it is a critical distance between sender and receiver in which the nature of communication switches from *reactive* to *radiation*. A reactive communication is based on data transfer using electrical fields (e.g. inductive and capacitive coupling); while radiation communication is based on transfer of data over electromagnetic (radio) waves. They stated that the crossover point increases with transmission frequency of a radio device.

For a 1GHz frequency the crossover point is 2mm and it decreases to 0.5mm for 10GHz signals. This explains why it is hard to find a good inductive or capacitive coupling technology for cm-range transactions when the data rate is tens of Giga bits per second or more. We will see more evidence on this issue later in this section. Appendix A contains a list of different research projects on all the aforementioned technologies during the last decade or so. Some of those are as follow:

2.5.3.1) Capacitive Coupling

Capacitive coupling (also called AC-coupling) is based on the electrical field created by capacitors to transfer power. It is particularly useful when DC-isolation is an important design issue; but it has also been used for data transfer purposes recently.

Capacitive coupling has lower power consumption compared to other wireless techniques. On 2007 *Fazzi et al.* [50] reported a capacitive channel that consumes 0.08pJ/bit and 0.12pJ/bit of energy in Mono- and bi-directional modes respectively. The main problem with capacitive coupling is that it cannot support long distance communications. Almost all the recent research on this field is dealing with distances not longer than 10 μm.

Due to small facing pads of capacitors it is very crucial to align the pads correctly. Capacitive coupling is very sensitive to misalignment of pads. Compared to intra-chip applications, it is easier to meet the alignment criteria for inter-chip applications. This is another shortfall of this method for millimetre- or centimetre-range communications.

It is possible to use multiple parallel capacitors to send parallel data in order to increase the data rate. The small area each channel occupies on chip has made it possible to accommodate multiple parallel channels on a relatively small chip area. It may in some cases be possible to utilise overlying top-metal layers to reduce area utilisation.

As an example *Drost et al.* [51] used 16 parallel channels each with 1.35Gbps data rate. They managed to achieve 21.6Gbps data rate. *Aung et al.* [52] with two channel capacitors with 6Gbps and 0.015pJ/bit, *Canegallo et al.* [53] with 128 parallel capacitors with 32Gbps and 1.12mW power, *Gu et al.* [54] who reported a capacitive link with 11Gbps and 0.39pJ/bit and *Kim et al.* [55] with a 15Gbps link (0.47pJ/bit) are other examples of parallel high data rates with very low power consumption. Even higher data rates are reported including *Hopkins et al.* [56] who reported 230Gbps and 260Gbps parallel links (144 links) with 3pJ/bit energy measured for the latter case. Some capacitive coupling model analysis can be found at *Lee et al.* [57].

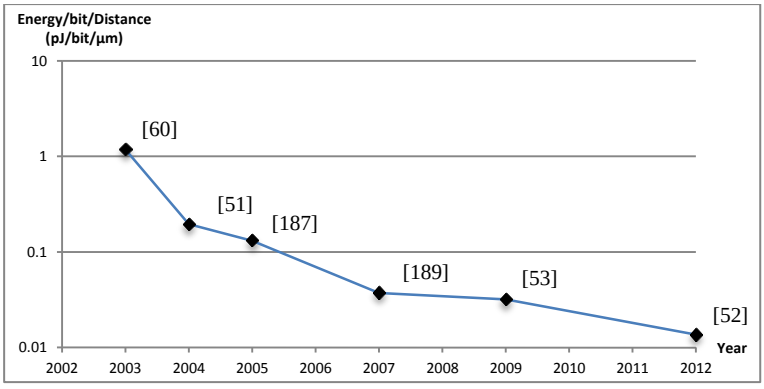


Figure 25: An approximation of the energy for sending a bit over one μm using capacitive coupling

It is not easy to compare the energy consumption of different research projects. This is because the research differs with taking different design choices (e.g. the distances supported by them are not always the same). Therefore, such a comparison in energy consumption is hard to be very accurate. Figure 25 is not very accurate; but it shows the overall trend of reduction of energy consumption per distance unit over the last decade. It means data can be sent over short distances with less energy year after year.

2.5.3.2) Inductive Coupling

The magnetic fields made by micro coils embedded on chips are the basis for data transfer using inductive coupling. Inductive coupling is popular in inter-chip applications especially for making 3D stacks of chips.

Compared to capacitors, inductors can transmit data over longer distances. Distances up to a few hundred micrometres can be easily achieved. Over such a distance it is possible to achieve data rates in range of Gigabits per second. There are some reports on using inductive coupling in cm-range (less than 10cm) mostly in wearable Body Area Networks (BAN) applications (*Lee et al.* [58]). Such a distance is achieved with the expense of sharp drop in data rate to tens of Megabit per second. Other applications include high-speed wireless proximity communication (e.g. *Matsubara et al.* [59]).

A series of projects led by *Kuroda* (including [60] [61] [62] [63] [46] [64] [65] [66]) has demonstrated the possibility of such a communication scheme for a range of bandwidths, distances and power targets. The team has reported up to 12.4Gb/s/channel bandwidth [67] (and 30Gbps in a simulation environment [68]) and energy consumption as low as 0.02pJ/bit [69]. The idea of using parallel links is also tested and brought successful results including 1024 parallel 1Gbps lines (1Tbps in aggregate) [46] in 2006 and 1024 parallel 8Gbps lines (8Tbps in aggregate) [70] in 2010. Researchers like *Han* and *Wentzloff* have presented the numerical analysis of the advantage of resonant inductive coupling over standard inductive coupling in wireless power transfer [71]. Analytical models [72] and lab experiments [73] reinforce that idea as well.

There were also some researches on transmitting clock signal using inductive coupling. *Kumar et al.* [74] present an inductive coupling scheme to transfer a 1GHz clock signal over a 15μm with a power dissipation of 1.55mW and 0.75mW. *Miura et al.* [75] introduce a twofold strategy to reduce the crosstalk effect in inter-chip inductive coupling communication that involves finding a critical distance between adjacent parallel inductors[8] and use of time interleaving (TDMA) technique. *Miura et al.* [69]

[8] Their measurements show that the critical distance varies when the number of parallel channels changes.

projected an improved version of their wireless clock and data transfer in which no time interleaving was used and still managed to transfer a 3.3GHz clock signal. When it comes to the communication distance it is the inductive method that can transmit data to longer distances compared to capacitive coupling. It is shown in [76] and [67] that data can be transmitted on high rates (2.4Gbps/channel, 12Gbps and 12.5Gbps respectively) over much longer distances (3.6mm, 0.5mm and 1mm respectively).

Like capacitive coupling, it is possible to send parallel data using a number of parallel inductors and increase the aggregate data rate. The alignment problem still exists in inductive coupling but it is not as crucial as capacitive coupling. Single links of up to 8 GB/s are reported by *Miura et al.* [70] in 2010. They use 1024 parallel links to form an 8Tb/s communication scheme. This is way beyond the current links used by commercially available best supercomputers in the world which is in range of tens of gigabit per second (node-to-node) but the distance over which this type of communication is possible is less than a hundred of micrometres.

Here, the energy consumption is reasonably low for on-chip applications. Although the transmission distance is typically longer than capacitive coupling, though still not long enough for the application envisaged in this thesis. On the other hand, cm-range inductive coupling devices have two major problems; first: they do not support high data rates; and second: they consume too much energy.

Figure 26 shows how the energy consumption per bit per micrometre has been improved over the last decade. The discussion about the lack of accuracy in normalisation of results in capacitive coupling is also valid for inductive coupling. However, Figure 26 is good for an approximation of normalised energy consumption of inductive coupling and shows its shard drop over time.

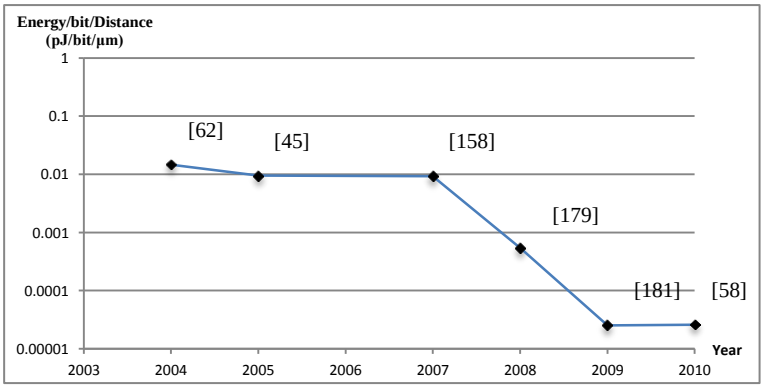


Figure 26: An approximation of the energy for sending a bit over one µm using inductive coupling

The area occupied by inductive coupling devices is particularly small because they do not need an electrostatic discharge (ESD) protection module. This means a potential for these devices to save some space (and cost).

2.5.3.3) Radio Waves

Radio waves have also been tested for longer distances (less than a meter) for communication between devices and chips. The big problem with radio technology is its high power consumption compared with other methods. The low data rate is also another issue.

On 2006 *Reynolds et al.* [77] demonstrated a 2Gbps on-chip radio. On 2007 *Daly et al.* [78] presented their 1.5Gbps on-chip radio. As both these projects were designed to work for long distances (2.5m and 10m respectively) there is no surprise that they consume large volume of energy (1.3W and 0.87W respectively). On 2008 *Laskin et al.* [79] successfully transmit 4 Gb of data per second over a 1.15m-link with the cost of 1.5W. This shows an improvement but the power is still a big issue. On 2009 *Chen et al.* [80] made a 6Gbps radio with 117mW power consumption for a link of up to 4cm. On the same year *Tomkins et al.* [81] presented an on-chip radio with 6Gbps bandwidth over 2m that consumes 232mW to 374mW of power. The constant push for lower power consumption and higher data rates in recent years brought much better results. On 2010 *Deb et al.* [82] managed to transfer 16Gbps signal over a 2cm-link that only needed 90mW of electrical power.

Data rates of around 10 Gb/s have been reported several times in recent years by different researchers. Some attempt to increase the data rate using different methods. As an example, on 2011 a team from Sony in collaboration with CalTech (*Fukuda et al.* [83]) demonstrated how using plastic stripes as waveguides can improve the performance of an on-chip radio. Their 25Gbps on-chip radio over a 12cm-link consumes 140mW of power. Recently on 2012 the same team (*Tanaka et al.* [84]) hit the record of 26Gbps over the same distance that consumes only 137mW over the same distance.

Figure 27 summarises the consumption of energy for transmission of a bit over a millimetre over the last decade. Like Figure 25 and Figure 26, this figure cannot be 100% accurate but the general trend of sharp reduction in energy can be detected. Among other factors, the bit error rate target is different in different projects.

Ishigaki et al. [85] have announced a 1TeraHz transceiver using resonant tunnelling diodes (RTD). Using a 542 GHz signal with a cut-off frequency of 1.1 GHz they reported a successful transmission of an ASK signal of up to 3.25Gbps with reasonable bit error rate over a 1cm distance. They anticipated higher data rates by using higher cut-off frequencies. Projects like this can create opportunities for developing high-speed wireless communication for a wide range of distances. A wide range of chip-area is

reported for different projects to accommodate the radio devices embedded on chip. In some cases like *Foulon et al.* [86] an area of 0.05 mm^2 is enough for their radio module while in other cases like *Miyashita et al.* [87] 56 mm^2 is needed for that purpose.

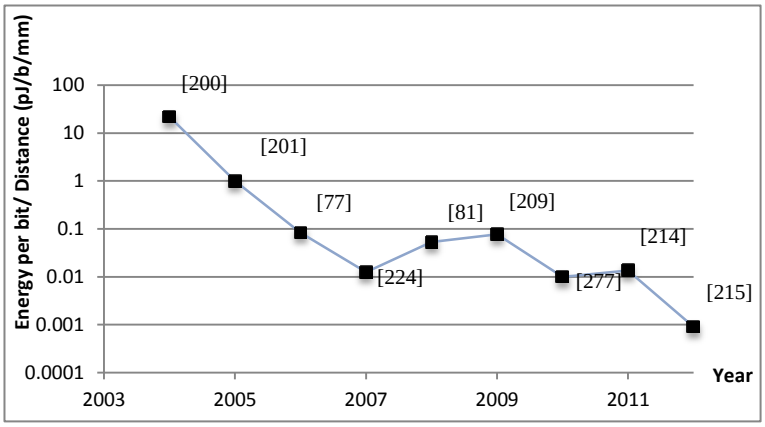


Figure 27: An approximation of the energy for sending a bit over one mm using radio waves

In addition to pure radio communication solutions some hybrid methods are tested. *Deb et al.* [82] showed how a mixture of wired and low power mm-wave wireless inter-chip links can significantly improve the performance of a NoC system. Another example is what *Chang et al.* [88] have done in which a number of wireless shortcuts are used in a multi-processor integrated circuit to reduce the total power consumption.

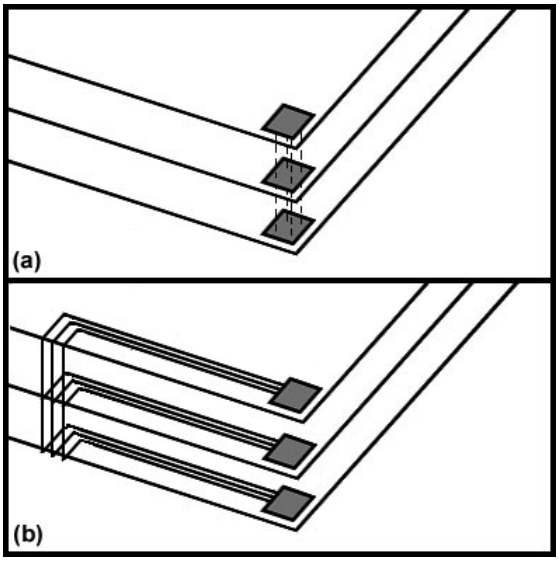


Figure 28: Comparing the distance of data links in (a) wireless and (b) wired networks.

In comparison between wireless and wireline links it should be noticed that a wireless link is a straight line between the sender and receiver of data while this is not necessarily always the case in a wired network. Wired networks are made of CPU boards, shelves and racks. The processors in a CPU board can be connected via a fairly short set of wires. But to connect boards, shelves and racks a considerably long data link is needed. Figure 28 shows how a wireline link can be much longer than its wireless equivalent. This means that the power consumption in a wireline link should not be

compared with that of a wireless link of the same length. To identify the exact equivalent length of a wireline link one should have detailed design information of both the wired and wireless networks.

2.5.4) Multi-Channel Communication

It was back in 1941 when a famous actress of that time called *Hedy Lamarr* inspired by pieces made by her composer neighbour, *George Antheil*, accidentally came up with the idea of switching between communication channels during a transmission time of a signal[9]. Their invention [89] was the first single-radio multi-channel (SRMC) radio device. The idea is also known as frequency hopping and is widely used in spread spectrum communication.

The main difference between their works and an MRMC device is that in MRMC multiple radio modules are used while in SRMC devices there was only one radio module switching between channels. Around 30 years later *Kaye and George* [90] introduced something that later was recognised as the first Multiple-input and multiple-output (MIMO) module. (MIMO); however, is not restricted to the radio wave medium and includes other communication media like fibre optics. In 1974 *Brundenburg and Wyner* [91] derived a formula to describe the performance of a multi-channel system in presence of Gaussian noise. *Van Etten*'s linear multichannel transmitter/receiver [92] and *Van Etten* and *de Jong*'s optimisation [93] are also among the early works on this field. *Golden et al.* introduce the first lab prototype of MIMO systems [94] in Bell Labs in 1998.

Wireless Internet access is one of the main applications of MRMC today. Moreover, a slightly different form of multi-channel communication is used in mobile phone technologies. Mobile handsets switch between channels when they join a new base station. From this point of view they can be categorised as a single-radio multi-channel device. However, most of today's mobile phones, tablets, laptops and desktop computers can all be regarded as MRMC devices as most of them can use Bluetooth, Wi-Fi and other radio technologies at the same time.

2.6) Packet Collision in Wireline and Wireless Networks

In all types of network (wireline or wireless) packet collision is a major problem which causes long transmission time due to back-off times' overhead and possibly

[9] At that time they anticipated their idea being used in radio-guided torpedoes to switch between frequencies to make it harder to detect, decode or jam. However, their idea did not attract proper attention until 1962 by USA military when their original patent was expired.

retransmission of packets after acknowledgment packet's timer expired. Any node that wants to send data on a shared communication media first listens to the link in order to detect a silent link and then starts the transmission process. This is called *Carrier sensing*, which is the basis for many communication strategies on shared transmission media. Carrier Sense Multiple Access (CSMA) method of multiplexing is introduced by *Kleinrock and Tobagi* [95] in 1975. In this method, nodes do not start the transmission when the shared link is already busy with another transmission; therefore, the collision just happens when two or more nodes detect a silent link and initiate their transmission process all exactly at the same time (in practice this may happen in a certain time period around the start of transmission rather than the exact start time of transmission). In this case an arbitration mechanism is needed to decide which of those nodes have the permission or the priority to use the link. Like other types of resource contention problems there are two major strategies to tackle packet collision: collision detection and collision avoidance.

Tobagi and Hunt [96] added collision detection feature to the original CSMA (CSMA/CD) in which transmitters stop transmission over a link as soon as they detect a collision. Then they wait for a period of time of a randomly selected length and then start again. This cannot guarantee that collision will not happen for a second time. In theory, the transmission time is not deterministic and it can be even infinitely long. But in practice, the chance for more consecutive collisions is very low.

In case of simultaneous attempt of two or more nodes to gain control over the communication media, what usually happens is that one of the nodes is permitted to continue with sending data and other nodes wait for some time and try again. As mentioned before, by using this technique, the correct transmission of all packets is guaranteed but there is no deterministic value for the transmission time. The key point is that these algorithms cannot avoid the situation. In time critical applications (including parallel processing) a high number of collision incidents increases the overall task time and decreases the efficiency of the network. Another key point is that in CSMA/CD collision is detectable on the transmitter side while in a wireless network not all collisions can be detected on transmission side (Hidden node problem [97]).

On the other hand, CSMA with collision avoidance (CSMA/CA) techniques act to prevent collision in first place (compare with CSMA/CD that tries to solve the packet collision problem). IEEE 802.11 [98] is a family of protocols that defines CSMA/CA among some other Media Access Control (MAC) and Physical (PHY) layer protocols for Wireless Local Area Networks (WLAN). Sometimes CSMA/CA focuses on

separating potential transmitters over time or transmission media. One of the collision avoidance techniques is based on dedicated timeslots to each node to send their data. This means no other node initiates a transaction during that timeslot. This makes it impossible to have interference between two competing nodes to access the shared link. In practice there should be a guard band after each timeslot to solve the problem with off-sync nodes. During these periods no node initiates a transaction. Solving the problem with off-sync nodes comes with a drop in effective bandwidth of the link.

Another technique is to use a pair of signalling packets called Ready-to-Send (RTS) and Clear-t-Send (CTS). When a node wants to send a packet to another node it sends an RTS packet before the main data packet. If the receiver is ready to receive the packet it sends back a CTS packet which in the transmitter side is interpreted as a signal to start the main data transaction. This type of communication is particularly useful in wireless networks in which existence of hidden nodes is a problem. Hidden nodes are nodes that cannot be detected by one node or a group of nodes. When a node (e.g. node A in Figure 29) detects a silent channel, the channel can be used for a data transaction. But, the problem is the silence of that channel is guaranteed just inside the transmitter's collision domain (i.e. interference range); at the same time, there can be other nodes out of this domain (like node D in Figure 29) that are still capable of interfering with the signal in the receiver side[10]. This is one of the main differences between wireless and wireline networks. In wireline networks packet collision can be detected in transmitter's side; whereas, in wireless networks it is the receiver that can detect the collision. The difference is because in a shared wireline link all node members have the same collision domain; but in wireless networks the collision domains of different nodes are not necessarily the same.

The RTS/CTS technique gives both the transmitter and receiver the chance to inform all nodes in their collision range about the data transaction. This is the basis for Multiple Access with Collision Avoidance (MACA) methods for wireline networks. In view to solve the hidden node problem in wireless networks a variation of MACA algorithm is introduced called MACAW (MACA for Wireless) [99] in which each data packet should be followed with an acknowledgment packet from the receiver to transmitter.

Another collision avoidance solution is called Carrier Sense Multiple Access with Collision Avoidance and Resolution using Priorities (CSMA/CARP) which is also

[10] There is a similar, yet slightly different, problem called Exposed Node Problem [291], in which it is the transmitter of a message that is receiving a signal from other node while the receiver of its own message is out of the interference range of the second signal.

defined in IEEE 802.11. In this method a modified version of RTS/CTS mechanism is used. After sending the RTS packet the transmitter does not start transmitting the main data right after receiving the CTS signal. Instead, it waits for a period of time called inter-frame space (IFS). The value of IFS varies with the type of packet. This is basically a prioritising mechanism. High priority packets have smaller IFS while low priority packets have bigger IFS.

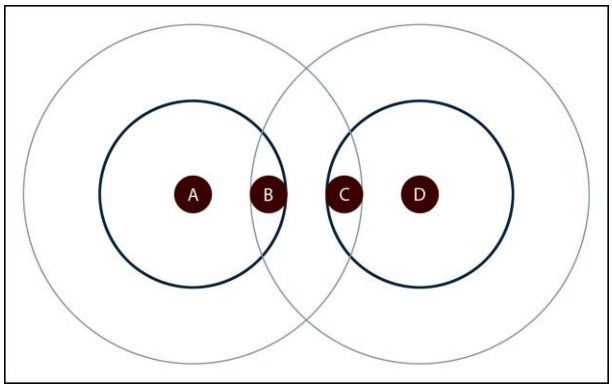


Figure 29: Hidden node problem in wireless networks. Black circles represent Radio range of nodes A and D. Grey circles represent interference ranges of A and D.

In wireless networks RTS/CTS mechanism can be used to avoid packet collision but it is the timeslot scheme that is proven to be a nearly perfect solution for packet collision in an environment in which packet collision can just be detected on the receiver side.

2.7) Network-partitioning

In this thesis the term "network partitioning" is used to refer to methods and algorithms used to divide a wireless network into subsets to let them operate on a given radio frequency. It can be compared with graph colouring problem (or a variation of that). As part of this thesis we are dealing with this concept and so it is necessary to know about what is already done on this field by other researchers. Network partitioning is a matter of interest in two separate fields: first: mobile phone networks and second: wireless Internet access networks and wireless mesh networks. In a majority of applications in both of these fields a network-partitioning algorithm has two stages. On the first stage a given network is split into overlapping or non-overlapping subnets and on the second stage a radio channel is assigned to each of these subnets. The latter stage is usually known as channel assignment.

In a mobile phone network the objective of a network partitioning and channel assignment algorithm is to optimally use the available radio spectrum while guaranteeing the network connectivity and safety of calls for mobile users when they cross the partition borders. A wide range of algorithms are designed and implemented for this purpose. [100] is a good survey on channel assignment and resource allocation

algorithms for mobile phone networks. What mobile handsets need to do in a properly partitioned network is to detect different signals received from different land antennas and decide which one is best to choose to communicate on. No two adjacent partitions should have the same channel. The size of the partitions and the bandwidth allocated to each of them are (either statically or dynamically) determined by the network-partitioning algorithm.

Network-partitioning and channel assignment have also been investigated in Internet access networks and wireless mesh networks. Most of the algorithms presented for these applications are dynamic algorithms as the characteristics of the network and the traffic measures are subject to change during the time. For these reasons bandwidth allocated to each branch of the network needs to be reviewed to minimise the latency and maximise the spectrum usage. Network-partitioning algorithms can be centralised or distributed. *Weisheng et al.* [101] have had a good review on different network-partitioning methods in wireless mesh networks. Although the network proposed in this thesis is very different form wireless mesh networks for internet access, there are still some key design issues in common between mesh networks (as listed by *Weisheng et al.* [101]) and the network proposed in this thesis including:

- Interference: To reduce the interference the variety of the channels used should be increased. The problem is due to some restrictions made by either governmental regulations or technical issues meaning the number of available channels is usually not sufficient for eliminating the interference. Another problem is the distance between access nodes and user nodes that is referred to in [101] as node deployment. In a mesh network node deployment can be considerably high; while in our proposed network that number is very low. Because of these two facts we think it is possible to eliminate the packet collision altogether.
- Connectivity: This is a concept borrowed from graph theory and is dealing with the availability of paths from any given node in a network to any other given node in that network. In our proposed network connectivity also has a close relation with path redundancy which is the availability of several paths from any given node to any other given node. It has been one of the objectives of our network-partitioning method to increase these two factors by having as much overlapping partitions as possible.
- Stability: The stability of a mesh network may normally be in danger from two sources: ripple effect and channel oscillation. As reported by *Weisheng et al.*

[101] the ripple effect is first described by *Rainwala and Chiueh* [8] and refers to consecutive channel switching caused by channel dependency among nodes leading to failure of packet delivery. A ripple effect typically occurs when a node switches to a new channel over one of its radio devices (say to avoid heavy load on the previous channel) and this makes the node(s) on the other end of the link(s) to switch to that new channel. This change can propagate throughout a large part of the network and cause problems. Channel oscillation is caused by repeating channel switching due to algorithm not converging. According to *Weisheng et al.* [101] this may happen when there is a dynamic metric for channel assignment. A decision to make a switch to a new channel based on such a metric can affect that metric which in turns may push the nodes to return to the old channel very quickly. In some cases this switch-back is so fast that do not let the nodes handle their communication duties properly. As an example, two nodes may detect an under-utilized channel; then they may simultaneously switch to that channel. As a result, the new channel may now face too much load and consequently both nodes decide to switch back to their older channels. This may keep repeating if the nodes are *too vigilant* in responding to traffic metrics. None of these two can happen in the network proposed in this thesis because both are related with dynamic channel assignment.

- Throughput/latency: Like all wireless networks, throughput and latency are major issues for both mesh networks and our network. The difference is, in HPC interconnect networks there is a more urgent need for the least possible latency. In dynamic channel allocation techniques the bandwidth available to links can be increased to respond to temporal increase in traffic load. Such measures are not part of the present thesis.
- Fault tolerance: The main fault tolerance tool in our algorithm is collision avoidance but in many mesh networks a collision detection and recovery approach is adopted.
- Algorithm time: Since in mesh networks the channel assignment algorithm should be run on regular intervals it is important for the algorithm to be executed on a reasonable time otherwise the performance of the network is sacrificed to achieve precision. The wireless network presented in this thesis is static and therefore the channel assignment algorithm does not need to be run more than once. This can be done in advance and for this reason the algorithm execution time is not a major concern.

Some graph partitioning methods are inspired by biological systems like the research *Hernández* and *Blum* have done on the calling behaviour of Japanese tree frogs [102] to make a distributed graph colouring algorithm.

To the best knowledge of the author of this thesis no 3D wireless grid for parallel processing is yet implemented. As a result no network-partitioning method for this purpose is designed yet. Such an algorithm on an MRMC network should satisfy the following criteria:

- No signal interference occurs between any two data communication intervals;
- Have the most path redundancy and partition overlapping;
- Have the most channel reusability.

2.8) Deadlock Detection/Avoidance/Recovery

The simulated network developed and tested for this thesis can be regarded as a set of resources and consumers. From this point of view the wireless links can be treated as resources while the processors are the consumers who may compete for taking control over those resources to transmit their packets. Like any resource-consumer system there are chances for ending up in a situation which is known as deadlock. The deadlock situation is investigated for decades and different definitions for it can be seen in different sources and may vary based on the context of the sources.

One of these is the definition of deadlock in operation systems introduced by *Silberschatz et al.* [103] and describes deadlock as a situation which occurs when a process or thread enters a waiting state because a resource requested is being held by another waiting process, which in turn is waiting for another resource. If a process is unable to change its state indefinitely because the resources requested by it are being used by another waiting process, then the system is said to be in a deadlock. Deadlock is very common in systems which are multithreaded, multiprocessor, parallel or distributed. Deadlock can also occur in databases[11].

In the BC platform introduced in this thesis deadlock can occur when two (or more) processing nodes have packets to send to the other node and at the same time the input and output queues of both nodes are full and therefore no transmission is possible. In this situation a free space in input queue of node number one is a resource that the node number two needs while node number one also needs a free space in input queue of node number two. In certain situations both input and output queues in both nodes can

[11] For further material on deadlock in databases refer to: https://docs.oracle.com/cd/ E17277_02/ html/TransactionGettingStarted/blocking_deadlocks.html

be full and therefore no progress is possible. This situation, how it may happen and how it has been tackled in this thesis will be discussed later in this manuscript; but in this section some background on classic ways to avoid and recover from a deadlock will be reviewed.

2.8.1) Deadlock Conditions

According to *Silberschatz et al.* [103] simultaneous occurrence of all the following four conditions are necessary for a deadlock. These conditions were first described by Edward G. Coffman, Jr. in 1971 [104] and therefore are known as Coffman conditions:

1. **Mutual Exclusion:** At least one resource must be held in a non-shareable mode.
2. **Resource Holding (Hole and Wait):** The process that gained the control over a resource and wants to gain control over other resource(s) which are being hold by other processes.
3. **No Preemption:** The possession of the resource cannot be ended by anyone other than the process itself (e.g. after finishing its task).
4. **Circular Wait:** Given a set of waiting processes $P= \{P_1, P_2, \dots , P_N\}$ P_1 is waiting for a resource held by P_2, P_2 is waiting for a resource held by P_3 and so on until P_N is waiting for a resource held by P_1 [105].

Prevention of any of the above conditions is enough for stopping the deadlock.

2.8.2) Deadlock Detection

Some systems simply ignore the possibility of a deadlock situation. This approach is only suitable for systems in which deadlock occurs very rarely so that its damage can be tolerated. In this approach deadlocks can occur, it will be detected and then the system tries to correct it. This may include tracing the processes involved in the incident, rolling those processes back and restarting them. The resource scheduler is the reference for detecting the deadlock and the processes involved in it.

There are two options when it comes to correct a deadlock: first: terminating/restarting one or more processes involved in it; second: forcibly freeing one or more resources held by processes involved in the incident.

2.8.3) Deadlock Prevention

As mentioned before to prevent a deadlock situation, the system should make sure that at least one of the Coffman's conditions will not happen. This is the basis for all deadlock prevention techniques.

It is not always possible to prevent mutual exclusion because some resources cannot be spooled and shared by their nature. For other resources that can be shared between

processes it is the duty of the resource allocator to make sure that the resources are properly allocated and no resource can exclusively take control over any resources.

If it is not possible to avoid mutual exclusion, another option is to prevent resource holding situation. This can be done by urging processes to ask for taking control of all their resources at the same tie so that in case of failure in holding any of those resources is equal to failing in holding all of them. Despite its simplicity, this approach is both inefficient and (sometimes) hard to implement. It can be inefficient because it holds all the resources from the start of the process some of which may be needed only in a short period of time. This is also hard to implement because in some occasions the resources needed for a process may be determined by the dynamism of the process and can bot be predict beforehand.

To prevent the third condition, the operating system should have the authority to take back a resource from a process but in many cases it can be difficult or impossible. Preemption in many cases ends up in rolling the process back otherwise the results of the process could be inaccurate or inconsistent. Also, rolling back adds extra time and resource costs. As a result there are situations that preemtion do not resolve the deadlock situation in an efficient way.

Disabling interrupts during critical sections is one of the solutions to prevent circular wait. If the nature of the process and resources permits, creating a hierarchy for resources is also a solution for preventing this condition.

2.8.4) Deadlock Avoidance

If the nature of the processes is known to the resource allocator (e.g. operating system), then there can be chances to avoid a deadlock situation beforehand. This means that the resource allocator should have an information about: resources currently available; how many resources a process needs; what type of resources it needs; what resources it has already held; when resources are needed and for how long they are going to be held.

Given that at the start time the system is in a safe state (i.e. free of deadlock); using the aforementioned information the resource allocator can decide if allocating a resource to a process can put it in an unsafe state (i.e. on its way to a deadlock). A deadlock will not occur if the resource allocator makes sure that no request for accessing to a resource is granted unless the system remains in a safe state as a result of that action. This should be noticed that the an unsafe state does not mean that the deadlock has already occurred but it means that deadlock can happen; therefore, the system can be in an unsafe state but does not end up in deadlock (e.g. by releasing some resources to prevent circular wait).

2.9) Network Simulation tools

This thesis needs a network simulator to run experiments to study the behaviour of the concept of BC platform. This needs a survey on a number of most popular network simulators. In this section we will review some of these network simulators. A longer list of network simulation tools can be found in [106].

2.9.1) NS Family

Ns (stands for Network Simulator) is a family of discrete event network simulators including three well known simulators: ns-1, ns-2 and ns-3. They are open source tools which are made mainly for teaching and research use [107]. Regarding the scale of work needed to develop such a simulation tool which covers different networks, implementation of the tools is spread over a large number of developers.

The first version which was originally called the LBNL Network Simulator and later was known as ns-1 was based on an older simulator called REAL developed by S. Keshav. It had a C++ core and was developed in Lawrence Berkeley National Laboratory (NLBL) between 1995 and 1997 by Steve McCanne, Sally Floyd, and Kevin Fall among others. Sun microsystems, UC Berkeley and Carnegie Mellon University are among long-running contributors [108]. Ns-2 was released in 1996-97 followed by ns-3 in June 2008.

In its current version, ns-3 has several so called modules each of which contains models for one or more devices and protocols in real-world. It is reported that a large majority of ns-3 users are using it to simulate wireless networks [107]; therefore, wireless networks such as Wi-Fi and WiMAX are supported by ns-3. A simulator written in C++ can use ns-3 as a library which can be statically or dynamically linked to it [109]. Almost all ns-2 APIs are now exported to Python as well.

There are some steps for creating a simulation using ns-3 [107]:

1. Topology definition: Creation of basic facilities and their interrelationships;
2. Model development: Other components of a model is added (e.g. IPv4, Point-to-point devices and links, applications);
3. Node and link configuration: Setting default values of the model;
4. Execution: Generating events and logging data requested by the user;
5. Performance analysis: Draw conclusions by analysing the time-stamped event traces of data;
6. Graphical visualisation: tools are provided to graphically represent the simulation results.

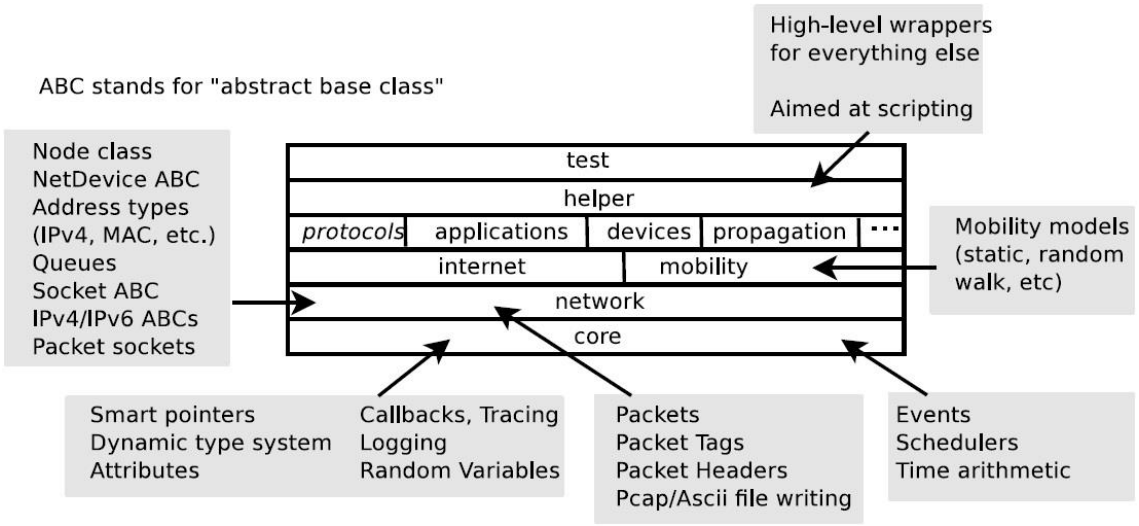


Figure 30: Software organization of ns-3 as plotted in [109].

Figure 30 shows the internal software organisation of ns-3. As described in [109], the core of the simulator contains the components common across all protocol, hardware and environmental models. Packets are implemented in network section. These two sections are independent of any specific network and device models. The higher sections of ns-3 are application specific and may vary depending on the simulated network. The propagation section can be of interest in this thesis as it defines how the signal propagates and what the loss pattern is.

2.9.2) OPNET

OPNET[12] was a software provider specialist in performance management for computer network founded in 1986 and was acquired by Riverbed in October 2012. Its object oriented discrete event network simulation toolkit is known by either its old name, OPNET, or its new name, Riverbed Modeller. More than 400 protocols and technologies are covered by this tool kit including VoIP, TCP, OSPFv3, MPLS, and IPv6 [110]. It has an open interface for integrating external object files, libraries, and other simulators [111]. Simulating 3D networks is also possible with OPNET. The OPNET modelling covers details about different aspects of a network including network protocols, resources, algorithms, applications, and queuing policies. Data visualisation tools are also part of this tool kit.

2.9.3) NetSim

NetSim[13] is developed by TETCOS[14] and is a popular stochastic discrete event network simulation tool that covers many technologies including Wireless Sensor Networks,

[12] http://www.riverbed.com/products/performance-management-control/opnet.html?redirect=opnet
[13] http://www.tetcos.com/netsim_gen.html
[14] http://www.tetcos.com/index.html

Wireless LAN, Wi Max, TCP and IP [112]. It is mainly written in C and Java and its first release was in June 2002. Its libraries code are open for user modification. Different levels of abstraction are available when it comes to provide performance metrics. The abstraction level can be network, sub-network, node and a detailed packet trace [112]. To have an interface between the users' code and libraries and kernel NetSim provides an in-built development environment. Also facilities for debugging and tracing the code on different levels are provided.

2.10) Some Related Works

To the best knowledge of the author of this manuscript the idea of using wireless links for parallel computers is something new; therefore, there is almost no previous work on this particular topic. On the other hand the idea of incorporating large number of processors for different applications is widely researched previously. There is not enough room in this thesis to introduce all those projects; instead, it is decided to choose just a few of them which have some similarities with the proposed plan. Some of the projects presented here are not directly related to parallel processing; they are chosen just for the sake of diversity.

2.10.1) SpiNNaker

SpiNNaker [113] [114] is a computer architecture to simulate the human brain. It uses ARM processor in a massively parallel computing platform. A six-layer thalamocortical model developed by *Eugene Izhikevich* [115] is the basis of SpiNNaker's architecture. Instead of modelling conventional artificial neural networks, SpiNNaker models real spiking neurons. The architecture is sought to achieve modelling of a billion neurons using a million SpiNNaker chips.

SpiNNaker's connection network [116] has a wireline 2D mesh topology with six neighbours per node. Nodes have external 6-ports in the SpiNNaker chip which allow for a three-dimensional (3D) torus arrangement. Based on the SpiNNaker documentations their network is not as good as a 3D torus in terms of bisection bandwidth and distance; but their network has advantage over 3D torus in terms of simplicity of manufacture and deploy, extra link redundancy due to diagonal links (Southwest and northeast in addition to usual 2D east, west, north and south) and ease of implementation of their own routing algorithm on that topology.

Their communication medium, the number of neighbours and the topology of the network is different from the proposed platform in this thesis. Also, SpiNNaker's prime usage is in simulating the operation of part of a human brain which is not the same as

the target of this thesis. But what is common between SpiNNaker architecture and the network proposed in this thesis is the idea of using large number of processors in a massively parallel platform with large number of connections between nodes.
Part of the SpiNNaker project concerns designing new processors and boards for their parallel platform. Designing processors is out of the scope of the current thesis; but the idea of incorporating large number of processors to shape a 3D grid for parallel processing is common between the two ideas.

2.10.2) Amorphous computing

In 1996 a project was started in MIT to use large number of processing elements that are identical, capable of local communication, limited in terms of computational capability and prone to faulty functioning used as a massively parallel system. The number of processors depends on the application and they may also be equipped with actuators so that, at least in theory, they can be in physical contact with each other and even form geometric shapes [117]. Inspired by biological systems, the basic idea was to see if a large number of not-very-reliable and not-very-powerful processing elements can collectively show very strong computational power and higher level of intelligence. They called their effort the study of *amorphous computing* [118].
The nodes in MIT's proposed network are distributed randomly and densely on a surface (a 2D plane in some simulations); therefore, there is no information about a node's neighbour in advance. Their self-organising network is implemented in two ways [119]: a group-based hierarchical network and a self-exploring emergent network. In the first method nodes come together and make groups of nodes with a leader for each node which is in charge of inter-group communications. The groups can have overlaps. In second method each node explores its environment with sending a *search message* to find a given destination. All neighbours respond to this message by either relaying it or sending back a *path setup message* in case they are the final destination. Nodes react to the first search message of the same origin to help the original node have the shortest path. In other words, a search message works in a *breath-first search* manner. On reception of the path setup message, the original node adds an entry to its routing table which is limited in the number of entries. The positive point about this method is that it can react to presence of barriers and blockage of links. This method is more applicable to the platform presented in this thesis, but since the BC network is static and all information about the neighbours is available to all nodes then there is no need to none of these methods in a BC.

Their idea was to have a self-organising, self-healing network that performs tasks through collaboration between large number of elements, each of which has limited resources and little global knowledge [120]. They have made both simulated and small real-world pilot networks [121]. Part of their work is to make the nodes the ability to learn how to communicate between and learn from each other. The concept of *Bootstrapping Communications* was introduced by Beal [122] in order to "bootstrap a simple communication system from observations of a shared environment". This is part of their plan to have a robust network in which each element may fail to operate occasionally.

The idea of using large number of processors with limited communication abilities is shared between the platform proposed in this thesis and amorphous computing. Amorphous computing; however, investigates the cases in which faulty elements can be tolerated by the group. The performance of the group is proven not hugely infected by the isolated failure of nodes. This is one of the main themes in their research. The Ball Computer (BC) concept (at least at its current stage) is not aiming for adding fault tolerance capabilities. Amorphous computing has a heterogeneous network in which some nodes (faulty nodes) are not performing as well as the majority of nodes (healthy nodes). On the other hand the current version of the BC platform is a homogeneous network in which all nodes are equally good at performing their tasks. The idea of emerging higher level of intelligence out of a large collection of rather modest processors is not also an objective in the BC project. The BC is sought as a pure parallel computer while the MIT's project is sought more as a platform to study the collective behaviour of agents. In amorphous computing the platform is going to look a bit like an alive entity that uses its numerous simple elements to achieve high level of capabilities.

Redundancy of processors is a key factor in amorphous computing while in the BC platform all the nodes are tested to carry out their own tasks. It should be noticed that the redundancy in communication channels plays a big role in the BC platform.

Another difference is that the nodes in MIT's plan are sought to be very simple and limited in memory size while in a BC although it is tried to keep the cost as low as possible, the simplicity of nodes is not a vital factor. Also, in MIT's plan the processing nodes do not have any information about the topology of the network and their own coordination before the start of the network. Parts of these data may be explored by the nodes during the run time if needed. But in the BC the network is a static wireless network in which nodes know everything about their coordination and the topology of the network beforehand.

2.10.3) Multicore Processors

The early computers have been made with just one processing unit. The idea of having more than one processor once only belonged to high performance computers. In light of reduction in manufacture cost, more and more commercially available computers are now equipped with at least two processors. Having more than one processing core is something usual even in mobile phones, tablets, laptops and desktop computers today. Many companies have produced different multicore processors for general or specific purposes. Most multicore processors for domestic usages have two or four processors. Eight and sixteen cores (in smaller quantities) are also available even for personal computers. Processors with larger number of cores are rarely used for home or office uses.

The most famous multicore processors of the date may be Intel's *Core* processor family including Core i3, Core i5 and Core i7 processor series with 2 to 4 cores (Used in many desktop and laptop computers), Intel's *Xeon* processor family with up to 15 cores (Used in many HPC platforms including CRAY XC30[15]) and IBM's *PowerPC* processor series with up to 18 cores (Used in Blue Gene/Q HPC platform[16]). Higher number of cores are embedded in a processor by other manufacturers. Tilera Corporation[17] is just one of those manufacturers that has produced a series of scalable multicore processor chips including TILE64, TILEPro64, TILEPro36, TILE-Gx72, TILE-Gx36, TILE-Gx16 and TILE-Gx9. It is chosen just as an example (not necessarily the best or the first one in this field) to see the similarities and differences of multicore processor platforms with the platform proposed in the present thesis. Tilera is chosen only to reflect the diversity of products and producers.

Figure 31 shows the block diagram of a TILEPro64 processor. A 2D array of 64 cores (tiles) are embedded in each TILEPro64 processor with Tilera's *iMesh* in-chip network [123]. Each tile has a complete, full-featured processors with L1 and L2 caches and a switch to connect the tile to the rest of the network (Figure 31-right). The interesting point about these tiles is that a core is able to run a full operating system on its own or multiple cores can be used to run a symmetrical multi-processing operating system like SMP Linux.

In this manuscript we just focus on the mesh interconnect network connecting the tiles inside the Tilera processors and do not go through any details of other building blocks

[15] http://www.cray.com/Products/Computing/XC/
[16] http://www-03.ibm.com/ibm/history/ibm100/us/en/icons/bluegene/
[17] http://www.tilera.com

of those processors. According to Tile processor architecture overview for the TilePro series [124] there are 6 separate and independent networks connecting tiles in a TilePro64 processor (this number is 5 in Tile64).

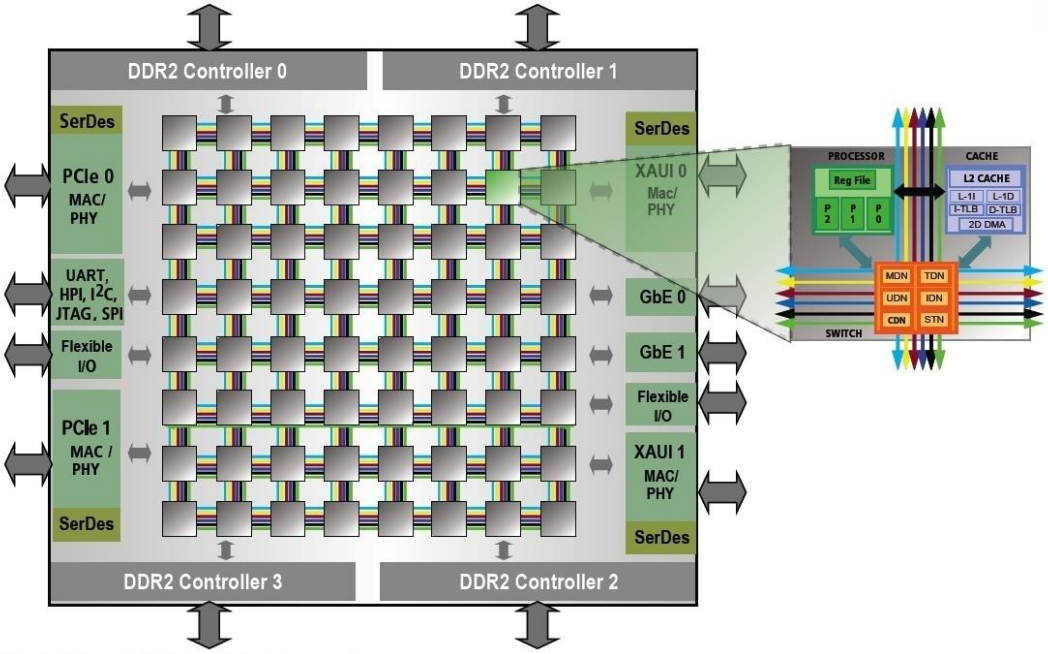


Figure 31: Block diagram of the TILEPro64 Processor as projected in [123]

One of the networks is a static network and the other five are dynamic networks. Of these six networks only one of them are visible to users and the others are used for the processor's internal operation. The networks are:

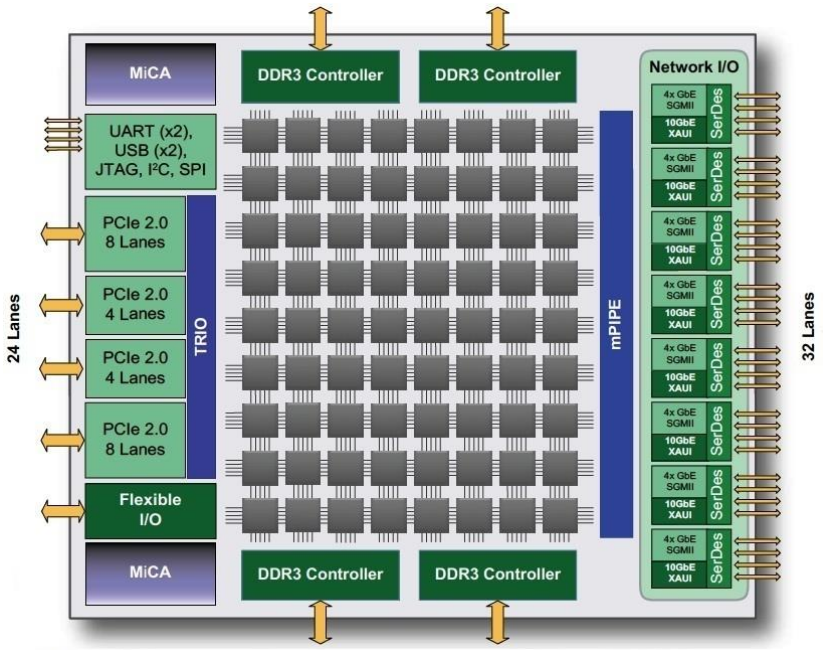


Figure 32: Tile-Gx8072 processor block diagram as projected in [125]

- STN: The Static Network switches scalar data (not in form of packets) between tiles with very low latency.

- UDN: The User Dynamic Network that is the only network accessible by the user (usually through C library routines).
- IDN: The I/O Dynamic Network that is used for transfers between tiles and I/O devices and between I/O devices and memory.
- MDN: The Memory Dynamic Network is used for memory data transfer between tiles themselves, and between tiles and external memory. Only the Cache Engine has a direct hardware connection to the MDN.
- CDN: Coherence Dynamic Network is used to carry cache-coherence invalidate messages. This network is not implemented in Tile64 processors.
- TDN: Tile Dynamic Network's usage similar to the MDN and supports memory data transfer between tiles. Only the Cache Engine has a direct hardware connection to the TDN.

Newer processors like Tile-Gx8072 are made with larger number of tiles and with similar (yet slightly different) network (Figure 32).

What any scalable multicore processor has in common with the BC is the parallelism achieved by using large number of processors (e.g. 64 cores for a TILE64, TilePro64) located very close to each other. The differences between these two platforms are:

- A BC is a collection of independent nodes each having their own processor, memory and I/O. The memory address space of each node either exclusively belongs to the node. In a multicore processor there can be a variety of memory modules with different access mechanism from different cores. Parts of the memory are dedicated to nodes. Others parts of memory can be shared between cores and are connected through an on-chip interconnect network.
- The connection network in a multicore processor is usually a wired network; however, there have been some experiments with wireless inter-chip networks for multicore ICs.
- Some multicore processors implement several parallel networks some of which designated to processor's internal signal and others are used for data transfer (e.g. Tilera's 6 independent networks). Separation between signal and data is not part of the current stage of the BC platform.
- The data link distances in a multicore IC are much smaller than those envisaged in a BC. This means that the same technology may not be used for a BC.
- In most of the cases the multicore networks are 2D networks that reduce the number of neighbours for each core compared to nodes in a 3D hexagonal network like BC.

2.10.4) Net-X

Being developed by the Distributed Algorithms and Wireless Networking (DAWN) Group at University of Illinois at Urbana-Champaign (UIUC) Net-X [126] is a test bed for experiments with multi-channel wireless mesh networks. Earlier in this chapter it is discussed that for a wireless mesh network, routing is a major issue. Routing for the Net-X platform also seems to be a topic for research. Some of their publications concern methods of routing designed for their network (e.g. [127], [128], [129]). They also did some work on reducing adjacent channel interference through channel assignment [130] [131]. Their channel assignment algorithm is based on gathering information from one- and two-hop neighbours of each node and the number of such neighbours that use a given channel *i*. The minimum, maximum and mean value of each channel's data is then passed to their local balancing and channel assignment algorithm to decide about the channel of the links of a node. The same procedure repeats periodically (in some of their experiments the algorithm repeats every 5 seconds).

Since it uses multi-channel wireless devices it has some similarities with the BC platform but it should be noticed that the number of channels in Net-X is lower than those in BC. The application and the network being set up for that application are different from what sought in the current thesis. The distances it supports (it is sought for application inside a house or working place) and the physical size of the nodes are also different from a BC. The data rates Net-X deals with are in range of tens of Mb/s [130]; while the data rates sought in this thesis are in range of up to tens of Gb/s. The applications sought for this platform are not as time-critical as those of a parallel computer; therefore, the latency and packet collision are not issues of such an importance compared to the BC platform.

2.11) Summary

The literature reviewed in this chapter covers components that can be building blocks of the wireless 3D grid proposed in this thesis.

Different network topologies for HPC systems have been reviewed. Direct and indirect interconnect networks have been used for this purpose. Multistage switch networks offer lowest number of hops but our survey showed the cables, particularly used by Dragonfly and Flattened Butterfly networks, are rather long. Direct networks including multi-dimensional tori used in Cray's Gemini and IBM's Blue Gene/Q save the cost for switches but they suffer from larger maximum number of hops which increases as the network expands.

The short review on routing algorithms has been presented in this chapter which shows that there are chances for direct networks to compensate one of their deficiencies over switch-based networks that is their larger maximum number of hops. This is particularly possible by incorporating cut-through and wormhole routing with virtual channels.

When it comes to choosing the wireless technology for physical links our survey showed that there are three possibilities: inductive coupling, inductive coupling and on-chip radio. All these technologies have their own deficiencies. While inductive and capacitive coupling technologies can offer data rates of hundreds of Gigabits per second (or even Terabits per second) with low energy consumption, they cannot extend their communication over cm-range. On the other hand, on-chip radio modules can deliver data over cm-range but at the moment the data rates such devices support are up to tens of Gigabits per second which in a multi-channel platform like the one proposed in this thesis is not quite high enough. On-chip radio's bigger issue is with energy consumption that at the moment is unable to match the equivalent wireline platforms. Radio modules reviewed in this chapter consume around five Pico Joules per bit in the best case while energy consumption of one Pico Joules or less is common between wireline networks over cm-range distances. This should be taken into consideration that electrical/optical links in supercomputers are not in cm-range. It has been shown in this chapter that in a dragonfly network, like Cray's Aries, the energy needed for sending a bit of data over a minimal path can be in range of 60 to 70 of Pico Joules. This is roughly equal to ten to twelve on-chip radio hops (based on on-chip radio technologies surveyed in this thesis). Those technologies are not designed and developed for applications sought in this thesis. This implies that it is yet to be determined how far the current technologies can be stretched to fulfil the criteria imposed by the Ball Computer wireless platform.

The review presented in this chapter on network partitioning and channel assignment algorithms showed that the Hidden Node Problem and packet collision can be solved using a multi-radio-multi-channel platform. However, the criteria current algorithms try to satisfy are different from what we need in this thesis. This means that a new algorithm should be designed to match the particular needs of the platform proposed in this thesis.

Now we are ready to discuss our plan for a wireless 3D wireless massively parallel computer. To test the idea, a pilot concept machine is presented later in this thesis called the Ball Computer (BC); but before that, in the next chapter (hypothesis and Rationale) the main research question of this thesis is presented and we will see what this thesis is particularly wants to achieve.

Chapter 3: Hypothesis and Rationale

Based on the literature reviewed in previous chapter it is the time to ask the main research question and define the boundaries of this thesis. Later in this chapter the platform briefly proposed in introduction chapter is discussed in more details. Next three chapters, also, contain detailed information about that platform.

3.1) Research Question

This thesis is part of a bigger plan to test the viability of a massively parallel computer with wireless interconnect network. The question of viability raises a number of other questions such as:

- Is a wireless network capable of serving a parallel processing system?
- Is there any advantage for a wireless network over a wired one in a parallel system?
- What are the minimum technical requirements of a wireless network to work properly in a parallel system?
- Can available wireless technologies produce satisfactory results?
- If available wireless devices cannot deliver satisfactory results, what are the major problems with them?
- How an improvement in wireless technology can facilitate use of wireless devices in such a system?
- How restricting a wireless network in a parallel system is?
- What characteristics such a system may have?
- What range of applications may execute with acceptable performance on such a system?
- How large a parallel system of this type can be?
- How costly such a system is?

From the technological point of view there are a number of challenges in building a wireless HPC system. Each of the following issues should be tackled with proper attention:

- Connectivity issues including latency incurred by hidden node problem and packet collision;
- Buffer flow control strategy;
- The power delivery strategy (wired/wireless);
- Heat dissipation.

Answering all these questions is beyond the scope of this thesis. This thesis restricts itself to only some of aforementioned challenges. Network connectivity is the focal point in this thesis; therefore, the main research question of this thesis is:

Is there an effective solution for connectivity in a massively parallel computer with wireless interconnect network?

We want to know if (and how) replacing a wireline interconnect network with a wireless network in an HPC system affects the performance of that system. In particular this thesis investigates solutions for reducing communication latency via eradicating packet collision problem and implementing a suitable buffer flow control strategy. Regarding the other challenges that this thesis does not cover, this thesis cannot be regarded as a blue print for a wireless parallel computer. There are still serious issues that need tackling before having a prototype of a wireless parallel computer of any type. What this thesis is going to answer is:

- What hardware and software technologies are needed to build a wireless parallel machine?
- How such a computer may look like?
- Can packet collision be eliminated by proposing a multi-channel multi-radio platform enhanced with effective network partitioning and channel assignment algorithms?
- What routing strategy is suitable for it?
- How the performance of such a computer may be?

By the end of this thesis we will find answers to the question if the current state-of-art wireless technologies suit the proposed platform in terms of the area they occupy, their transfer rates and energy they consume. These should particularly be tested against a dense and multi-channel platform like the one proposed in this thesis. Also it will be shown that:

- The proposed network partitioning and channel assignment algorithm is capable of solving hidden node problem and eradicating packet collision.
- The save-and-forward algorithm implemented in this thesis yields good performances for a given network attributes; although its performance is not good enough for other values of network attributes.
- A wormhole switching (routing) mechanism with virtual channels is needed to have a communication scheme which offers low latency over a wide variety of network attributes.

3.2) Ball Computer

Section 1.1 has briefly introduced our wireless architecture that –to the best knowledge of the author of this thesis – has not been used for HPC purposes yet. The name *Ball Computer* (BC) is chosen for this concept machine. A BC is basically a collection of

independent processing elements in a 3D hexagonal wireless grid which operates as a massively parallel computer. Each node in the proposed interconnect network consists of a processor and a series of wireless transceivers to give it the ability to share data with other nodes. The type of the wireless module will be determined later in this chapter. Based on the current level of electronic technology, it is anticipated that such a collection of circuitry (a processor and a series of wireless transceivers) can be implanted on electronic boards of size $1cm^2$ or $2cm^2$; however, the actual process of making such a circuit is beyond the scope of this thesis. It is anticipated that when mass-produced, the price per node is reasonable. An exact cost analysis for the proposed network is not available at this stage because the network is only implemented in simulation tools rather than real world.

The whole electronic board is envisaged to be placed inside a plastic (or any suitable dielectric material) sphere to prevent it from direct and uncontrolled electrical charges from outside. These so called *Balls* are then packed to form a 3D wireless interconnect network which in this thesis is referred to as *Ball Computer (BC)*. Each *ball* is envisaged to have a size of 1cm to 2cm in diameter. Deciding about power delivery mechanism is beyond the scope of this thesis, but to have a general idea about this issue, there can be wireless, wireline or hybrid solutions for the issue of power delivery. Some of the available options and their benefits and limitations have been already discussed in chapter 1. Another very important issue is heat dissipation which, again, is out of the scope of this thesis. This issue is also briefly discussed in chapter 1. The fact that these two issues are not discussed in this thesis does not undermine the important role of these two issues in shaping a 3D wireless interconnect network of any type in real world.

The number of wireless modules per node depends mainly on the topology chosen for the network and the relative position of the nodes. In the next chapter more discussions on selecting the network topology are included. It will show that a Face-Centred Cubic (FCC)[18] topology is chosen for this grid to let a maximum number of nodes to be packed into a given space. Each node in this topology has 12 neighbours.

This thesis is solely concentrated on simulation analyses; however, even in next stages in which a prototype of the proposed platform is to be built, the design of processing elements is not a priority. Instead, a processor should be chosen from commercially available processor that satisfies some criteria mainly concerning the processing ability, the energy consumption and heat generation. At the current stage, this thesis can only help having an overall idea about what these criteria are.

[18] FCC topology will be explained in chapter four.

A candidate wireless technology should be comparable to wireline networks used in parallel computers. This comparison is on a number of characteristics. To have a chance to replace conventional connection networks, a wireless technology should prove that it is better than (or at least comparable to) wireline technology on the following items:

- Data communication rate;
- Communication distance;
- Power consumption;
- Occupied area;
- Locality of communication.

An on-chip wireless communication mechanism can be based on capacitive coupling, inductive coupling or radio waves. Each of these is strong on some of the items listed above. The target of this manuscript is to show that there is either an existing wireless technology which is better than wireline on the items listed above, or the wireless technology is heading to the direction of making such a technology in near future. For this reason the author of this thesis believes that it is the right time to start thinking about considering wireless technology in parallel platforms.

3.3) Data Communication Rate

One of the strengths of reactive methods (i.e. capacitive and inductive coupling) is the high data rates they support. As far as it is concerned with data rates, there are some promising results reported by several researchers and it is anticipated that in the short or middle term wireless devices can be compared to their wireline counterparts in terms of data rate. Figure 33 shows some reported results on all the main three categories of wireless mechanisms. It shows that radio waves are more capable of supporting longer distances but it comes with the cost of lower data rates.

Coupling techniques have the chance to use parallel data lines to increase the data rate. In fact there is no theoretical limitation on the number of parallel links except for the device size as there is a threshold on how close two consecutive parallel links can be.

The idea of parallel links is very popular in coupling methods and gives an upper hand to them especially to inductive coupling techniques. This is the main reason why there are inductive coupling mechanisms with data rates of up to 8Tb/s. Parallel transfer of hundreds (e.g. [63] and [132]) or even one thousand parallel links (e.g. [46] and [70]) have already being reported.

For radio waves there is no such chance to have parallel independent links, as parallel links interfere with each other. For this reason, radio devices should stick to serial communication of data that stops the radio data rates reaching the Terabits or even

hundreds of Gigabits range. The best performance in terms of bit rate the author of this thesis came up with is around 26 Gb/s [84] in 2012.

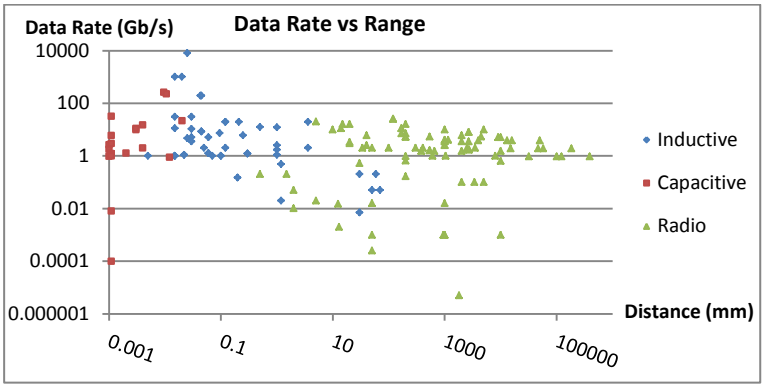


Figure 33: Comparing wireless technologies in terms of data communication rate

Another point is the frequency of radio waves used for present on-chip radios. At the moment 60 GHz waves are used by most of researchers in this field. There are some attempts to test higher frequencies like 120GHz (Including earlier works of *Kosugi et al.* [133] in 2004, *Deferm* and *Reynaert* [134]in 2010 and *Fojimoto et al.* [135] in 2010 which is followed by *Katayama et al.* [136] in 2012). Some other impressive results like *Fukuda et al.* [83] in 2011 which is developed further by *Tanaka et al.* [84] in 2012 use a full-duplex multi-carrier scheme which uses 57GHz and 80GHz frequencies. In light of new developments in the terahertz on-chip radios (e.g. *Ishigaki et al.* [85] and *Hu et al.* [137]) it is highly anticipated that much higher data rates can be achieved by using suitable modulation/demodulation methods.

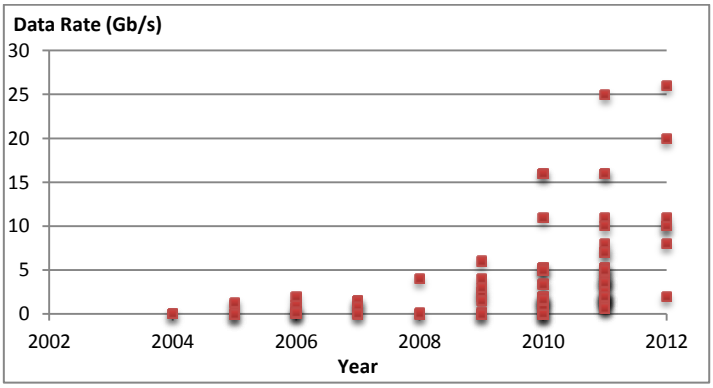


Figure 34: Improvement of data rates in radio devices during last decade

The increase in data rates in on-chip radio devices during last decade can be observed in Figure 34. The data rates plotted in this figure is high enough for a single-channel on-chip radio device but if the bandwidth is going to be shared by more than one channel. Terahertz radio modules can help increasing the data rate of digital signals (e.g. at hundreds of gigabits per second or more) in future when Terahertz on-chip radio devices are mature enough.

3.4) Communication Distance

The major problem with coupling methods (both capacitive and inductive) is the very short communication distance they can support. The only exceptions are inductors used in wireless BANs which operate in cm-range distances; however, their data rates are considerably lower than 1 Gb/s and therefore cannot be considered in this thesis.

The other deficiency of coupling methods is their high sensitivity to alignment of coupled elements (capacitors or inductors). The fact that both of these schemes are mainly developed for inter-chip communication applications explains why none of them push for neither supporting longer distances nor more robustness against misalignment of coupled elements. Long-range reactive methods consume high amounts of energy and therefore are not favourite options for the platform proposed in this thesis.

Wireless communication is mostly based on transferring data in a straight line from transmitter antenna to the receiver antenna; while, in most of wireline schemes a straight line from transmitter to receiver is not available and wires run through longer distances and may pass different circuits to deliver data to the receiver. This should be considered when finding a wireline equivalent for the wireless scheme proposed in this thesis. It is not possible for the author of this thesis to find an exact length of wire that is equivalent to a 1cm wireless link because of differences in architecture of both categories of technology. In a multi-processor board the number of processors are fixed; therefore, for a processor on a board a transaction with processors on other boards should go through board connectors and then data busses. In many parallel platforms separate I/O boards (possibly on separate racks) are used to send a packet from a processor board to another. This makes the actual average length of wires between a transmitter and a receiver much longer than 1cm (as envisaged in the proposed platform).

According to Figure 33 no capacitive coupling technology is suitable for the distance envisaged in this thesis (i.e. 1-2cm). A few inductive coupling transceivers satisfy the criteria. A much larger number of embedded radio devices are suitable for such a link in terms of communication distance. This means that there are many suitable options when it comes to the length of link; however, as it will be seen later in this chapter many of those candidates do not meet other design requirements.

3.5) Power Consumption

One of the strengths of reactive methods (i.e. capacitive and inductive coupling) is the low electrical power they consume. The consumed power for radio devices in cm-range is higher compared to reactive technologies. This is because of first: the nature of

electromagnetic waves; and second: radio devices are usually used over longer distances compared to the other two methods.

Different design choices have been made in different papers reviewed by this thesis. The main differences are in:

- Communication distance: Theoretically the power consumption is related to the square of the length of the link. It should be noticed that not all the power in the transmitter side is consumed in the amplifier and antenna; therefore, the transmitter's total power consumption does not scale with the length of the link.
- Target Bit Error Rate (BER): A wide range of BER is reported in different projects (See Appendix A, Table 46, Table 47, Table 49). The BER of inductive coupling technologies surveyed in this thesis vary between 10^{-10} and 10^{-16} (excluding wearable devices that have BERs in range of 10^{-3}). Surveyed capacitive coupling devices have almost the same nature as inductive coupling in terms of BER as their BER range between 10^{-10} and 10^{-15}. But BERs reported for on-chip radio devices surveyed in this thesis vary between 10^{-3} and 10^{-12}.

  Among the factors that may affect BER are data rate, transmitter's power and even the chosen type of modulation and coding method. Very weak signals are hard to be detected and correctly decoded in the receiver side while too strong signals may contribute in cross-talk interference. Both these situations may lead to high BERs. Since the reviewed technologies vary in BER, it will be very inaccurate and misleading if the power consumed by those projects are compared with each other.

The current thesis has set a fixed communication distance between the transmitter and the receiver (1-2 cm). To compare the results reported by other researchers those results should be normalised to the envisaged distance. In the absence of an accurate normalisation mechanism (as discussed in previous paragraphs) approximation of normalised values are used. This approximation is based on the theoretical square distance rule.

Comparing wireline and wireless is even harder because the actual length of wire in a wireline equivalent of the proposed wireless scheme is not 1-2 cm. Figure 28 shows that the data in a wireline network does not always choose a straight line between the transmitter and receiver; but in a wireless link the signal chooses the shortest distance which is a straight line. The average link distance in a wireline network with the same design targets of the proposed wireless network is higher than 1-2 cm; but an accurate

estimation of such a wire length needs detailed technical information of the equivalent wireline network which is not available at the moment.

In 2004 Cho et al. [138] compared wireline and fibre optics connections for short distances. Their comparison on power consumption is shown in Table 8. As this table shows when the distance is short, wirelines are better in terms of energy consumption and for longer distances fibre optics outperform wirelines on this matter. Regarding the date of the paper, the results cannot necessarily reflect the current state for wireline and fibre optic connections but it helps having a taste of the range of energy electrical and optical links in sub-100 cm links consume.

Technology	Data Rate (Gb/s)	Distance (mm)	Energy per bit (pJ/bit)
Wireline	4	100	0.75
	4	1000	2.5
	4	1000	8.75
	6	100	0.75
	6	800	2.5
	6	800	7.5
Fibre Optics	4	100	2.5
	4	1000	6.25
	6	100	2.67
	6	1000	5.83

Table 8: Comparing wirelines and fibre optics in term of energy per bit as projected by [138]

In 2007 *Koo et al.* [139] compared fibre optic and copper communication lines over short ranges[19]. They tested these two media of communication using different transistor technologies and measured the energy consumption, latency and bandwidth density. The results (Table 9) demonstrate a reduction in reported energy consumptions

Another comparison is made by *Stucchi et al.* [140] in 2013. Table 10 summarises their comparison between two types of wireline and two types of fibre optic devices.

Media	Technology (nm)	Data Rate (Gb/s)	Distance (mm)	Energy (pJ/bit)
Copper	22	10	10	≈1
	32	10	10	≈1.25
	45	10	10	≈1.7
	65	10	10	≈2.35
Fibre Optic	22	10	10	≈0.4
	32	10	10	≈0.49
	45	10	10	≈0.51
	65	10	10	≈0.75

Table 9: Comparison of energy consumption in copper and optic technologies by *Koo et al.* [139]

[19] Their paper also includes carbon nanotubes as well which is out of the scope of this thesis.

Technology	Data Rate (Gb/s)	Distance (mm)	Energy (pJ/bit)
Wireline (RC)	1.55/μm	10	0.58
Wireline (TL)	50 (8/μm)	10	0.052
Fibre Optic, off-chip laser, 45nm	50 (25/μm)	10	0.075
Fibre Optic, on-chip laser, 45nm	50 (25/μm)	10	0.018
Wireline (TL)	50 (8/μm)	20	≈0.14
Fibre Optic, off-chip laser, 22 nm	50 (25/μm)	20	≈0.075
Fibre Optic, on-chip laser, 22 nm	50 (25/μm)	20	≈0.02

Table 10: An energy consumption comparison between copper and fibre optic communication media over 1-2 cm, reported by Stucchi et al. [140].

Some other results can be seen in Table 11 which concern distances less than one metre. Table 8, Table 9, Table 10 and Table 11 belong to different technologies and they might significantly differ in design criteria but those tables show a constant decrease in energy consumption for optical and electrical communication devices over cm-range among different platforms. These tables show that a sub-pJ/bit energy consumption for a cm-range wireless link is not far from reality particularly when using fibre optic cables.

Ref.	Date	Technology	Data Rate (Gb/s)	Distance (mm)	Energy (pJ/bit)
[141]	2010	Wireline	470	5.08	1.4
[142]	2011	Wireline	40	200	11.43
[143]	2011	Wireline	6.4	200	4.1
[144]	2011	Wireline[20]	22	100	≈0.35
[144]	2011	Wireline[21]	≈33	100	0.5

Table 11: A number of short-range wireline links operating under 100 cm

We have already seen in section 2.2.7 (Table 5) that how much energy a typical modern commercially available HPC system (Cray's Aries) supplies to send a bit of data across a minimal/non-minimal path. We know that a minimal path in this technology includes one long range (optical) and up to four short (electrical) links. Also we know that, in practice, not all the paths in an Aries platform are minimal. Non-minimal paths have 10 hops (as an absolute maximum) and 7 hops (as an average number of hops). In all these cases there is one long optical link involved in the transaction process. This means that a packet exchange over Cray's Aries can consume between 60 to 70 pJ/bit depending on the number of hops.

This thesis has surveyed more than 200 papers published during last decade on short-range wireless communications in search for potential candidates for replacing wireline networks in HPC systems. A thorough list of those technologies is included in Appendix A. Here those technologies are compared based on three major factors:

[20] This minimises energy per bit.
[21] This maximises data-rate to energy-per-bit ratio.

energy consumption, communication distance and data rate. The results belong to three major categories of wireless technologies: Inductive coupling, capacitive coupling and on-chip radio devices. Three thresholds are introduced as technology selection criteria:

- 1cm is the minimum distance needed for a link sought in the BC platform.
- A data rate of 1 Gb/s is the minimum requirement for a successful candidate; as modern-day supercomputers extensively use cables with data rates of Giga bits or tens of Giga bits per second.
- Later in this thesis it will be shown that BC platform proposed in this thesis needs up to 24 hops to send a piece of data from one node to another (given its number of nodes is the same as an operational Cray Aries)[22]. This means that each hop in the proposed topology should consume around 3 pJ/bit to have the same energy demand as a modern commercially available supercomputer. An energy threshold of 10 pJ/bit is chosen to choose any wireless technology with an energy consumption of the same degree of magnitude (3 pJ/bit).

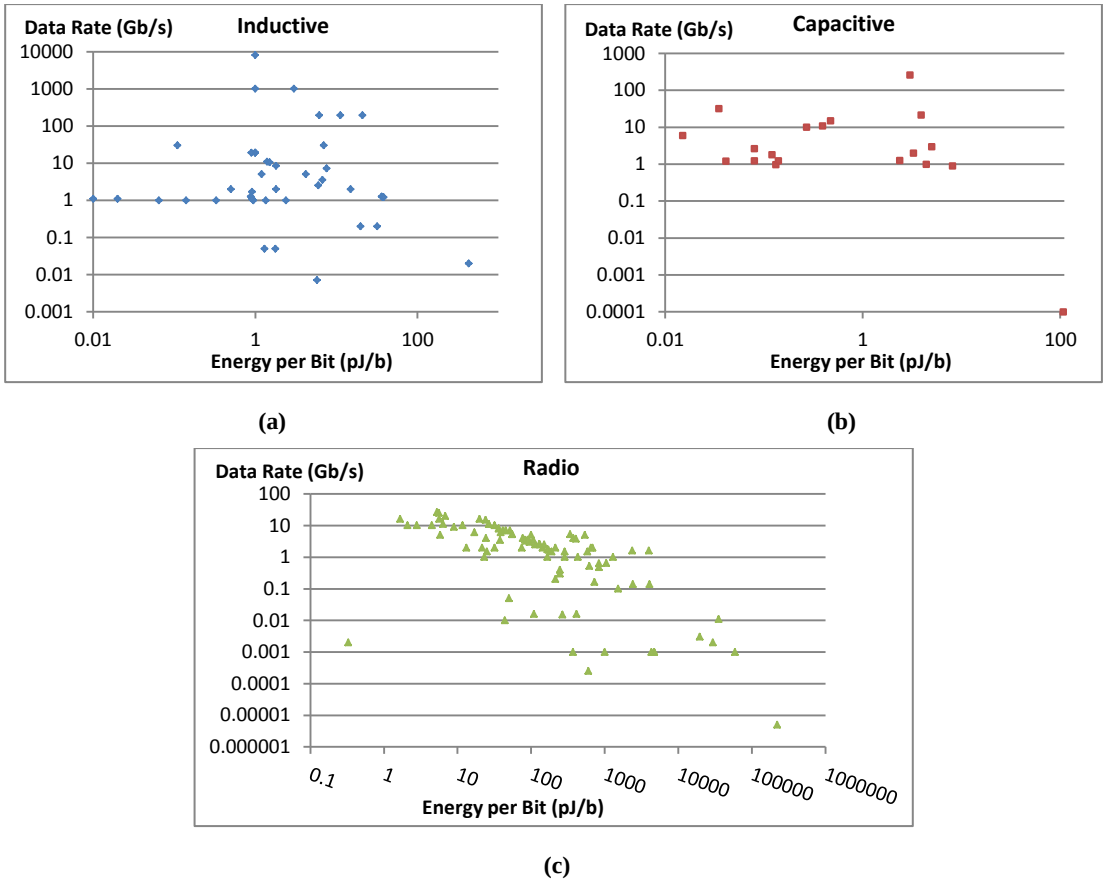


Figure 35: Comparing data rate vs. energy per bit for (a) capacitive coupling, (b) inductive coupling and (c) radio wave technologies

The results are shown in Figure 35, Figure 36, Figure 37 and Figure 38. Figure 35 plots data rate against energy consumption per bit for three major categories of wireless

[22] The Cray Aries platform used for this comparison is Switzerland's Piz Daint with 14498 nodes (115,984 cores). A 3D BC grid of size 25*25*24 has almost the same number of nodes (14500).

technologies. To make the differences between these three even more visible, in next three figures surveyed technologies are plotted in three different ways. In Figure 36 current wireless technologies are compared based on the data rate and energy criteria for the proposed network. Energy consumption is plotted against communication distance in Figure 37; and Figure 38 plots data rate versus communication distance.

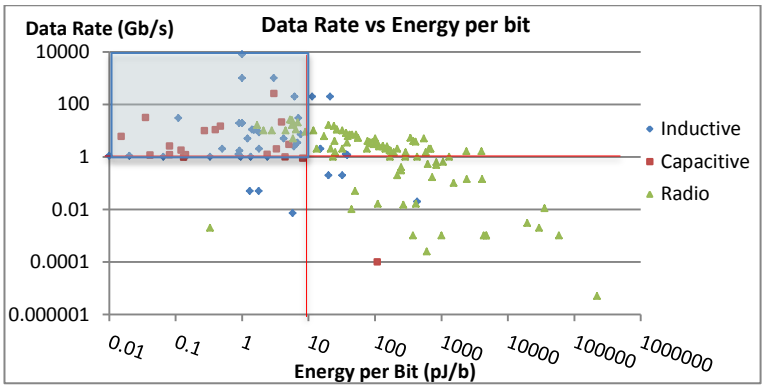


Figure 36: Comparing wireless technologies in terms of power consumption and data rate

These comparisons show that the power consumption of solutions based on radio waves are considerably higher than both two coupling methods. The best reported radio solution over 10s of millimeter is in range of 2 pJ/bit (which is a great achievement on its own); while both coupling schemes deal with power consumptions of 1pJ/bit or lower (not forgetting that they deal with distances much shorter than those of radio applications). Some technologies based on both coupling techniques have yield energy consumption in range of 0.01s pJ/bit that is even better than most wireline solutions.

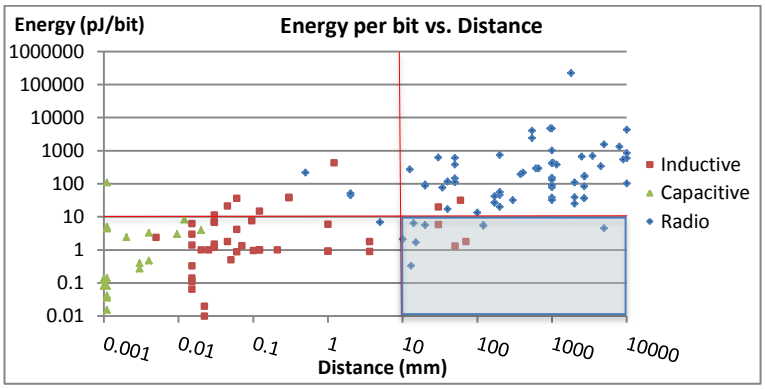


Figure 37: Comparing wireless technologies in terms of power consumption and communication distance

The distinct difference in the energy consumption between the three methods is visible particularly in Figure 36. While most of reactive methods easily satisfy the energy consumption criteria, most of radio devices struggle satisfying this criteria. The deficiency of coupling methods are shown in Figure 37 and Figure 38 in which none (or very few) of them can mutually satisfy the energy consumption and data rate criteria or energy consumption and distance criteria. Energy consumption is known to be of the most importance. Reducing the energy consumed by both processing and communicating modules is the main objective in any parallel processing platform [145].

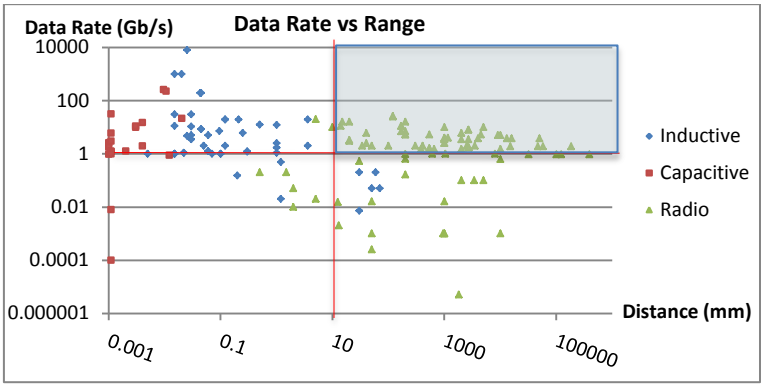


Figure 38: Comparing wireless technologies in terms of data rate and communication distance

Figure 36, Figure 37 and Figure 38 also show that the tightest constraint is in energy consumption versus communication distance (shown in Figure 37). According to this figure neither capacitive coupling technologies nor inductive coupling modules can satisfy both these criteria at the same time. Even among on-chip radio modules there are very few technologies that can deliver data to the envisaged distance (i.e. 1-2cm) with less than the energy threshold set in this thesis (i.e. 10 pJ/bit).

But as Figure 39 -derived from surveyed papers (Appendix A) - shows this problem is not as severe as it was one decade ago. It is mentioned before that it is hard to normalise the network parameters of different projects because of their difference in design details but Figure 39 still can be used as an estimated measure of the energy consumed to send one bit over a 1mm link. This leaves us with the hope that following this trend we will see on-chip radios in future with much lower power consumption rate which makes them easier to be incorporated in the network proposed in this thesis.

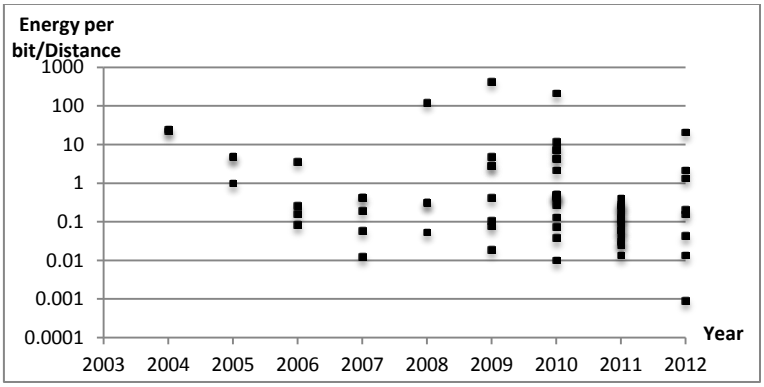


Figure 39: Mutual improvement in range and energy per bit for radio devices during last decade

In comparison, reactive technologies have the best record in energy consumption (around 10 fJ/bit) but such a low energies do not let them send data over more than tens of micrometres. The energy consumption of radio devices is significantly higher but it is decreasing year by year (see Figure 27). During the same period the communication distance supported by capacitive and inductive coupling have not been improved enough to match the design necessities of this thesis.

Ref.	Data Rate (Gb/s)	Distance (mm)	Power (mw)	Energy per bit (pJ/bit)
Deb.et al. [82]	16	20	90	5.63
Kawasaki et al. [146]	11	14	70	6.4
Yu et al. [147]	16	15	26.7	1.67
Fukuda et al. [83]	25	120	140	5.6
Foulon et al. [86]	10	10	21	2.1
Tanaka et al. [84]	26	120	137	5.27
Tanaka et al. [84]	20	5	137	5.85

Table 12: Candidate wireless technologies suitable for a 3D wireless grid

If the historic trend of reduction in energy consumption of on-chip radio continues there is a solid chance to have an on-chip radio option for the envisaged wireless network with reasonably low energy consumption in near future while no strong push for extending the range of communication for fast inductive or capacitive coupling technologies over cm-range distances can be detected. For this reason, to deliver the data over distances envisaged in this thesis, the best available option is radio waves, and this is despite the fact that they still need to improve their electrical power consumption. We are cautiously optimistic to see millimetre-wave on-chip radio communication technology is approaching the sub-Pico-Joules-per-bit scale.

According to above figures and based on the papers surveyed in this thesis, there are a number of candidates that satisfy all three aforementioned criteria (Table 12).

3.6) Occupied Area

It is not possible to define an exact area each wireless module should occupy. Such an exact measure is derived during the design process. Designing a real node for a BC is out of scope of this thesis. A review of literature on all three major wireless technologies show that both capacitive coupling and inductive coupling do not usually occupy more than a square millimetre of chip area. Even hundreds or thousands of parallel capacitors or inductors can be easily fit into less than a square centimetre. Therefore, both of these two categories of wireless technologies satisfy the area criteria.

On-chip radio modules usually need more area compared to the first two wireless categories. This is particularly because of the embedded antenna needed in this type of technology. However, mm^2-range modules are quite common in this field. Figure 40- derived from surveyed papers (Appendix A) - shows how the density of communication rate over an area unit improves over the last decade or so. There is a substantial chance for future research projects to follow this trend of making smaller modules with faster communication rates available. For all these reasons the author of this thesis sees no real constraint over area occupied by wireless modules.

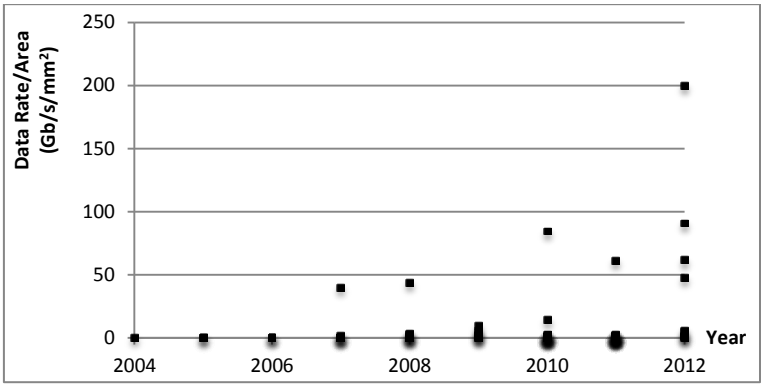


Figure 40: Improvement in data rate and occupied area in radio devices during last decade

3.7) Locality of Communication

Many of the current commercially available parallel computers are based on or inspired by Blue Gene/Q platform. The 5D structure of Blue Gene gives it a huge flexibility and a low number of hops between its nodes. It should be also noticed that the topology of Blue Gene is a round torus. This means that the nodes on the two ends of the network are directly connected. This halves the maximum number of hops. The BC architecture; instead, has a simple 3D structure. This is good in terms of system simplicity but the number of hops between nodes increases faster than Blue Gene as the number of nodes increases. Table 13 compares the maximum number of hops in Blue Gene platform and BC. The structure of a BC - as shown in detail in next chapter – gives the nodes the chance to communicate diagonally. For this reason in a 3D network of type N*N*N the maximum number of hops is N.

Blue Gene/Q Dimension					Number of Nodes	Ball Computer Dimension (Range=d)	Ball Computer Dimension (Range=2*d)	Maximum Hops			
								Blue Gene/Q	Wireless FCC(r=d)	Wireless FCC (r=2*d)	Flat Torus Blue Gene
2	**2**	**2**	**2**	**2**	**32**	**3.17**	**1.59**	**5**	**4**	**2**	10
4	**4**	**4**	**4**	**2**	**512**	**8**	**4**	**9**	**8**	**4**	18
16	**16**	**4**	**4**	**2**	**8192**	**20.16**	**10.08**	**21**	**21**	**11**	42
16	**16**	**16**	**16**	**2**	**131072**	**50.80**	**25.40**	**33**	**51**	**26**	66
32	**32**	**16**	**16**	**2**	**524288**	**80.63**	**40.32**	**49**	**81**	**41**	98
32	32	32	32	2	2097152	128	64	65	128	64	130
64	64	32	32	2	8388608	203.19	101.59	97	204	102	194
64	64	64	64	2	33554432	322.54	161.27	129	323	162	258

Table 13: Comparing the maximum number of hops in Blue Gene platform and Ball Computer

Here two wireless schemes are tested and are compared to Blue Gene/Q's 5 dimensional structure. In the first hypothetical scheme each node is just in contact with its direct neighbours; while in second scheme the radio range of each node covers all direct neighbours as well as all direct neighbours' neighbours. As shown in Table 13 and Figure 41, the first architecture proposed in this thesis is better than Blue Gene in terms

of the number of hops for network sizes of 8K nodes or less. The second scheme is better than Blue Gene/Q in terms of number of hops even for very large networks. It should be noticed that the maximum number of nodes in a Blue Gene in real world never exceeds 32*32*16*16*2 nodes; therefore, the last three rows in Table 13 are extrapolations of existing Blue Gene/Q platform and just show hypothetical situations.

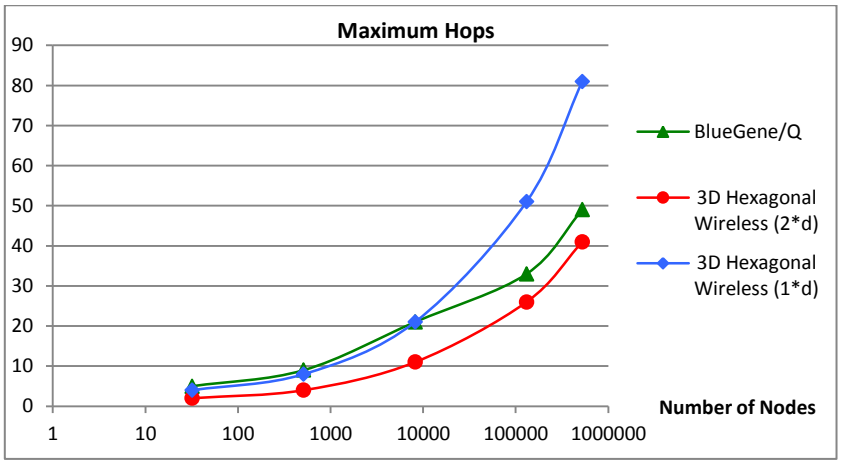


Figure 41: Number of hops in Blue Gene/Q and Ball Computer

The number of hops is not the only factor to determine the communication time between nodes. The latency in a one-hop communication is another important factor. The communication in a platform like Blue Gene goes through I/O boards which locate on different shelves that can locate in a separate rack. According to Blue Gene documentations a direct neighbour one-way latency is 0.3 μs [31]. The corresponding delay time for a BC is not measured yet as it is yet to be built.

Based on the literature reviewed in this thesis the author of this manuscript believes that despite some concerns over the power consumption and number of hops, it is the right time to start thinking about using wireless devices in parallel platforms.

3.8) Buffer Management, Routing, Deadlock Avoidance

Among the buffer management methods reviewed in section 2.3 the store and forward method is chosen for BC platform. The writer of this manuscript is aware of the potential delays this method may impose but the main reason for this decision is that compared to other methods a store-and-forward mechanism is simpler to implement. In the literature review chapter it has been noted that cut-through and wormhole routing methods are not suitable for links with intermittent connectivity. At this stage it is believed that the links sought for the BC platform are fairly robust and reliable so there is no need to worry from this point of view; having said that the reliability of the physical radio links can be verified only when a prototype of the BC platform is built in real world.

By making this decision we expect having considerable delay times in data transactions incurred by long times packets spend in I/O buffers in intermediate nodes. Another drawback of this decision is that the total transfer time for a packet will be directly related to the number of hops. Regarding the fact that the BC network is a direct network the number of hops in a packet transaction can be considerably high which means that the total transaction time can be hugely affected by that. More efficient buffer management methods will be considered for next stages of this research after making sure about persistence of link connectivity.

In lack of switches or routers in the BC platform packets will be routed using global and local information stored inside nodes. A simple routing algorithm is adopted for the BC platform which sets a fixed route between any two nodes. The route is determined by the coordination of the sender and receiver of the packet. The coordination of nodes in BC network is fixed; therefore, the selected route is fixed as well. The algorithm runs on all nodes on a path from the source and destination of a transaction including intermediate nodes and starts with calculating the distance between the current node and the final destination of the packet on all three dimensions. Then the packet is sent to a neighbour alongside the axis which had the largest distance calculated. This method is chosen mainly for the sake of simplicity of implementation and is not guaranteed to be flexible enough if there are some nodes or links in the path that are missing or out of order.

This can be risky to have one and only one route because in case of a failure in even one of the intermediate nodes no alternative route is considered and no packet can be delivered. This may be less importance in the current stage of the research in which only a simulated BC platform is tested in which it is assumed that all nodes are up and running throughout the whole simulation time. This routing algorithm is used just to explore the routing method possibilities and will be changed or modified for next stages.

A deadlock avoidance technique is sought for the BC platform to avoid deadlock situation when two or more neighbouring nodes are dealing with high rate of input and output packets. In this thesis this situation is referred to as a communication hot spot. There will be a communication hot spot when both input and output queues of a node (let’s call it node A) are full with packets to/from another node (node B) while the input and output queues of node B is also full with packets to/from node A. In this situation node A cannot find a free space in input queue of node B to send a packet and free one slot in its own output queue. The same thing happens in B as well and no progress is

possible. This particularly can happen when more than one workload are loaded to the BC network.

A deadlock prevention technique is used for this thesis which is based on shifting some of the traffic load off the nodes that face a temporal high traffic load. Referring to situation discussed in previous paragraph, this means that when nodes A and B detect that there are many packets in their I/O buffers to/from each other, they signal other nodes to either stop sending them other packets or send packets to them in a lower rate. By doing this those two nodes have the chance to focus on sending/receiving packets left in their I/O buffers. The normal transmission of packets from other nodes can be resumed when the heavy load period is spent. This may not guarantee that no deadlock occurs but it can be shown that it can effectively delay a deadlock situation. More details about this deadlock prevention method can be found in chapter 7 of this thesis.

3.9) Summary

This chapter starts with introducing the main research question of this thesis which is:

Is there an effective solution for connectivity in a massively parallel computer with wireless interconnect network?

It is also discussed why we need to narrow down the scope of the thesis to connectivity issues in a wireless parallel platform. Other major issues are also briefly discussed but left for future works. A number of more detailed questions around the connectivity issue are also presented in this thesis.

A simulated network is needed to test the idea of having a 3D wireless parallel computer. This platform is called *Ball Computer* (BC) in this thesis. Following a brief description of the concept of a BC in first chapter of this thesis, more detailed discussions is included in the current chapter. By the end of this chapter, one will have a clear picture of what a BC is and how it is supposed to operate.

This chapter shows what is our expectations from a BC platform in terms of data communication rate, communication distance, power consumption, physical occupied area and maximum number of hops.

The evaluations presented in this chapter shoes that the maximum number of hops in BC platform is comparable to most popular parallel platforms in networks of small and medium sizes. In large networks (e.g. a majority of most powerful supercomputers in the world today) the maximum number of hops of the 3D BC platform is higher than modern direct networks like CRAY Aries's 5D network. This issue can be solved in future expansions to the idea of BC by incorporating more efficient buffer management methods (e.g. worm hole switching/routing).

This chapter also shows that there are two other important factors namely energy consumption and data rates in which wireless devices are yet to be improved to be able to match current wireline technologies. It is shown that the wireless-wireline gap is wider in terms of power consumption.

Through chapters 4, 5 and 6 more detailed discussions are presented over some important aspects of BC architecture. These are network topology, network partitioning and task modelling respectively.

Chapter 4: Communication Media and Network Topology

4.1) Communication Media

The analysis in the previous chapter indicates that radio devices are the best choice for a 3D wireless grid. The radio range and the area radio modules occupy match the criteria of this thesis. There are still some concerns over their data rate and power consumption. As it is shown in chapter 3, there are already a few on-chip radio technologies that can meet the energy requirements envisaged in this thesis. More suitable on-chip radio technologies (in terms of energy demands) for the proposed platform will be available in near future if the historic trend of reduction in power consumption continues.

As mentioned before, heat dissipation and power delivery are two other big challenges which are out of the scope of this thesis. In real world these two factors can play a role in choosing communication technology. As an example, heat dissipation requirements may set a minimum distance between nodes. This can have a direct effect on the type of wireless technology. None of these two challenges; however, has any direct effect on a simulated environment like the one used in this thesis.

The other option for communication media would be inductive coupling. The positive point about such a technology is its low energy consumption and high data rates. The big problems with inductive coupling are its short data links it supports and its high sensitivity to alignment of coupled elements. The short data link problem is already solved (e.g. in wireless BAN applications) but it comes with huge cost of sharp reduction in data rate. In case of a breakthrough in improving the length of the link while preserving the high data rate; inductive coupling can be a very favourite candidate for a 3D wireless grid for parallel applications.

Regarding all the benefits and deficiencies of the two wireless categories mentioned above, it is decided to use radio modules in the BC platform. Each node of the grid in this network has a number of radio modules which can operate independently. The number of modules depends on the dimension of the grid and the ratio of radio interference range to the node's diameter.

4.2) Radio Modules per Node

A 3D network is preferred to a 2D network in order to have the maximum connectivity between nodes. In a 2D network the maximum number of neighbours for a node is 6; while the same number for a 3D network is 12. Also to accommodate a large number of

nodes in a 2D network a relatively large area is needed; while the same number of nodes can be packed in a relatively smaller cube.

Another important factor is the radio range of modules. The larger the radio range the larger the number of neighbours for a node which consequently leads to higher connectivity and lower average number of hops. But this comes with the cost of lower bandwidth available for each link given a fixed overall bandwidth available to a node.

The communication range of a radio module is the range in which the radio signal is strong enough to be received and decoded in the receiver side. The interference range of a radio device is the range in which the radio signal can interfere with other signals but it is not necessarily strong enough to be decoded in the receiver. In many radio simulations as well as analytical papers the interference range of a radio device is approximated as twice the radio communication range. In this thesis the interference range is approximated as twice the communication range. The communication range plays a major role in determining the number of neighbours of a node as shown in Table 14; while the interference range will affect the channel assignment process (discussed in next chapter). The channel assignment algorithm determines the total number of frequency channels needed for the whole network. The larger the interference range the larger the number of channels and consequently the narrower each channel will be.

This shows that an increase in the radio range has a positive effect on the performance of the network as well as a negative impact. The positive effect is the larger number of neighbours and higher degree of connectivity. The negative impact is the higher number of channels needed for the whole network which means more restriction on the bandwidth on each channel.

Table 14 shows how the number of neighbours increases with the radio range. It will be shown in next chapter that when there are just 12 neighbours the number of channels needed for each node is eight. When the number of neighbours is higher than 12, the number of channels needed is equal to the number of neighbours.

Radio Communication Range	Maximum number of neighbours for a node
1*d	12
2*d	54
3*d	168
4*d	356

Table 14: The increase in the number of neighbours with increase in radio range (“d” is the diameter of the node)

When it comes to decide about the nodes' radio communication range, it has been decided to keep the radio range as short as possible (i.e. "1*d" as shown in Table 14). In this case the number of neighbours for each node will be 12 and the number of radio modules for each node is 8. Larger communication ranges are practically impossible because of the sharp increase in the number of radio modules needed for each node.

4.3) Hexagonal topology

A 3D hexagonal topology is adopted for the wireless grid to maximise the number of nodes in a volume unit. We know from literature that there are two 3D topologies with maximum density:

1. Hexagonal Close-Packed (HCP)
2. Face-Centred Cubic (FCC; also known as Cubic Close-Packed)

The space occupied by spheres in both these lattices is the highest among different lattices. Both these topologies are based upon surfaces of balls in a triangular tiling. FCC topology is chosen for this the BC platform. It is known from Gauss [148] that the packing density of the spheres (balls or nodes in this thesis) in such an arrangement is given by Eq. 10 which shows how much of a given space can be occupied by equal-size spheres in aforementioned topologies[23]:

$$\boldsymbol{Density = \frac{\pi}{3*\sqrt{2}} \approx 0.74048} \qquad \textbf{Eq. 10}$$

There is little difference between the two lattices that is discussed in [149]. In Figure 42 letters A, B and C resemble different layers. There are only two distinct layers in HCP matrix; while, the number of distinct layers in FCC is three. These two topologies can be converted to each other with a shift in FFC's layer C. The shift is shown in Figure 42 where a shift in white circle with letter C to the dashed circle converts layer C to layer A and consequently converts FCC to HCP. The shift is by "d" – which is the diameter of the nodes – and should be applied along Y-axis.

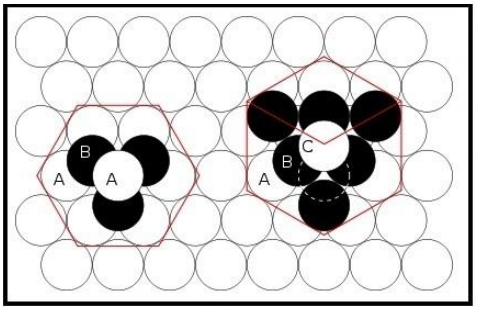


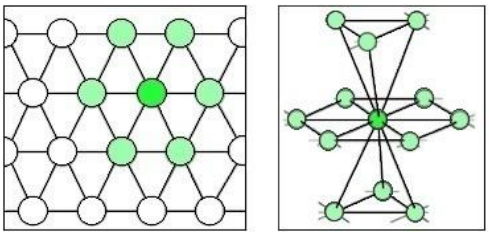
Figure 42: HCP (left) and FCC (right) lattices as projected in [149]

The reason why FCC is preferred to HCP originates from the zoning mechanism which is discussed in next chapter. The number of zones in both HCP and FCC topologies are

[23] Density is also called Atomic Packing Factor (APF). The term is borrowed from crystallography.

the same; but the zones are symmetric and balanced in FCC while in HCP the zones are not evenly distributed over all neighbours. In FCC each neighbour is a member of two zones.

Figure 43 compares the grid in 2D and 3D[24]. It has been stated earlier in this thesis that the radio communication range of nodes just covers their adjacent neighbours. It means that each node is in direct contact with 6 nodes in 2D and 12 nodes in 3D.



(a) **(b)**

Figure 43: The topology of a ball computer in (a) 2D and (b) 3D

The number of links between nodes is decided to be at its maximum. It is possible for a real-world prototype to reduce the number of links to reduce the implementation cost as well as the overall number of channels needed. This; however, comes with the expense of a reduction in network connectivity and possibly an increase in average number of hops between nodes. The actual number of links in a real prototype is a matter of trade-off between these factors and cannot be determined at this stage.

What is important at the moment is to make sure that connectivity of network is maximised by choosing a hexagonal topology (rather than cubic) for a 3D grid (rather than a 2D grid). If the actual area occupied by real radio modules is small enough, there would be another option which is increasing the number of radio modules from 8 to 12 for each node. In this case each radio is dedicated to one and only one neighbour. This may increase the connectivity of the network and may lead to faster data communication and lower wait time for data packets.

As a quantitative measure for the degree of connectivity in interconnect networks the connectivity matrix is used by a number of scholars including *Fountain* [150]. The basic connectivity matrix is a square matrix of size *n*n* where *n* is the number of processing elements. Each element *(i,j)* of this matrix is equal to 1 if there is a direct path between processing elements *i* and *j*. That element is zero otherwise. Connectivity matrices of higher degrees can also be made based on the basic matrix. For example in a second degree connectivity matrix the matrix element *(i,j)* is equal to 1 if processing elements *i* and *j* are connected in less than two hops; otherwise that element is zero. These matrices

[24]In real world the size of nodes are bigger than what is shown in Figure 43. Nodes in real world will be in physical contact with their neighbours as shown in Figure 42 and Figure 46.

helps studying how effective a topology is when it comes to sending a message in least possible hops. Connectivity matrices can be illustrated as a checked board in which each small square *(i,j)* is painted white if the corresponding element in the matrix is zero; otherwise that square is painted black.

Figure 44 shows illustrations of connectivity matrices of degrees 1 to 5 for a 5D torus which is the basis for connectivity network of a modern Blue Gene/Q parallel computer. The figure shows that the maximum number of hops between two nodes in such a network is five.

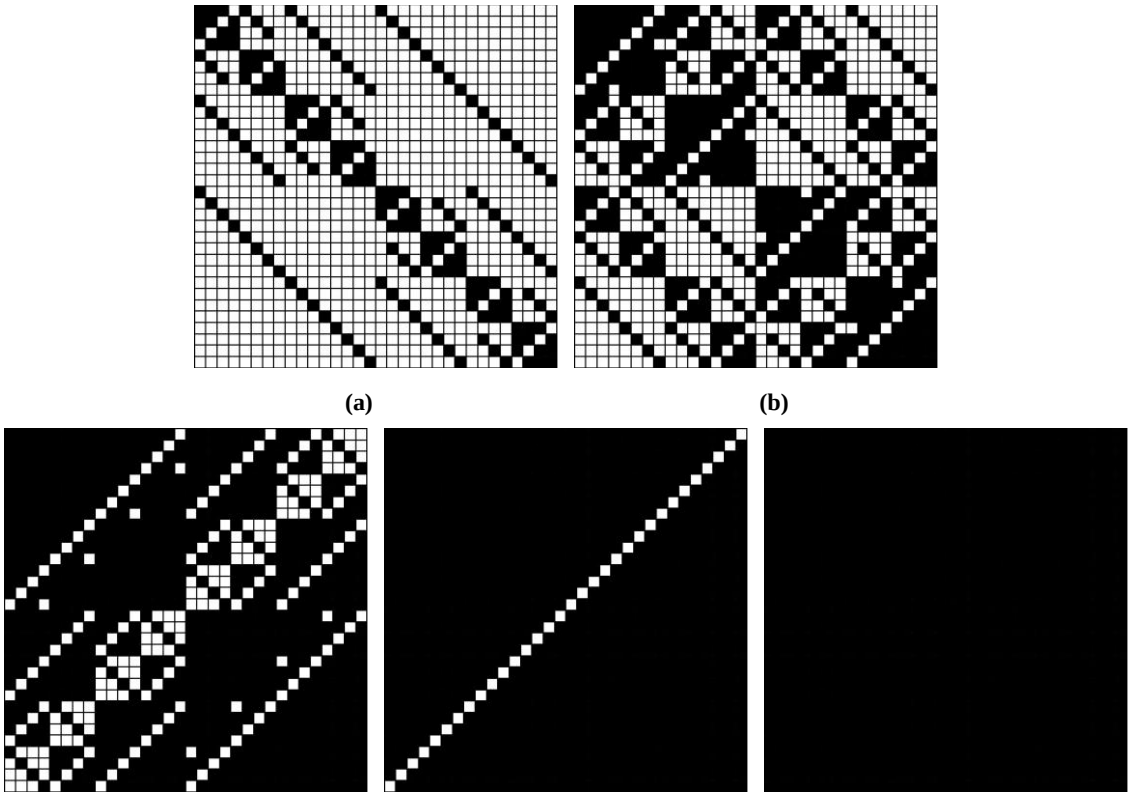


Figure 44: Connectivity matrices of a 5D grid (2*2*2*2*2). Path degree varies from 1 to 5 in (a) to (d) respectively

The same set of matrices is reproduced for a 3D hexagonal grid. The results are shown in Figure 45. It shows that the theoretical maximum number of hops for such a network is four. There are just two exceptions to this rule which is due to the connectivity limitation for the nodes in the corners and edges.

Figure 44 and Figure 45 show that for small and medium size networks a 3D hexagonal network can send messages with less number of hops thanks to larger degree of connectivity compared to a 5D network. Since each node has 12 neighbours, it has many options to choose from and the message will be delivered in small number of hops. But for large scale network the orthogonally connected nodes in a 5D network show their strength and help delivering messages in smaller number of hops compared to a 3D hexagonal network. It should be noticed that although in a hexagonal network

each node has many neighbours but they are not orthogonal to each other and this hinders the performance of such a topology in large networks. These results reinforce the results presented in Table 13 and Figure 41.

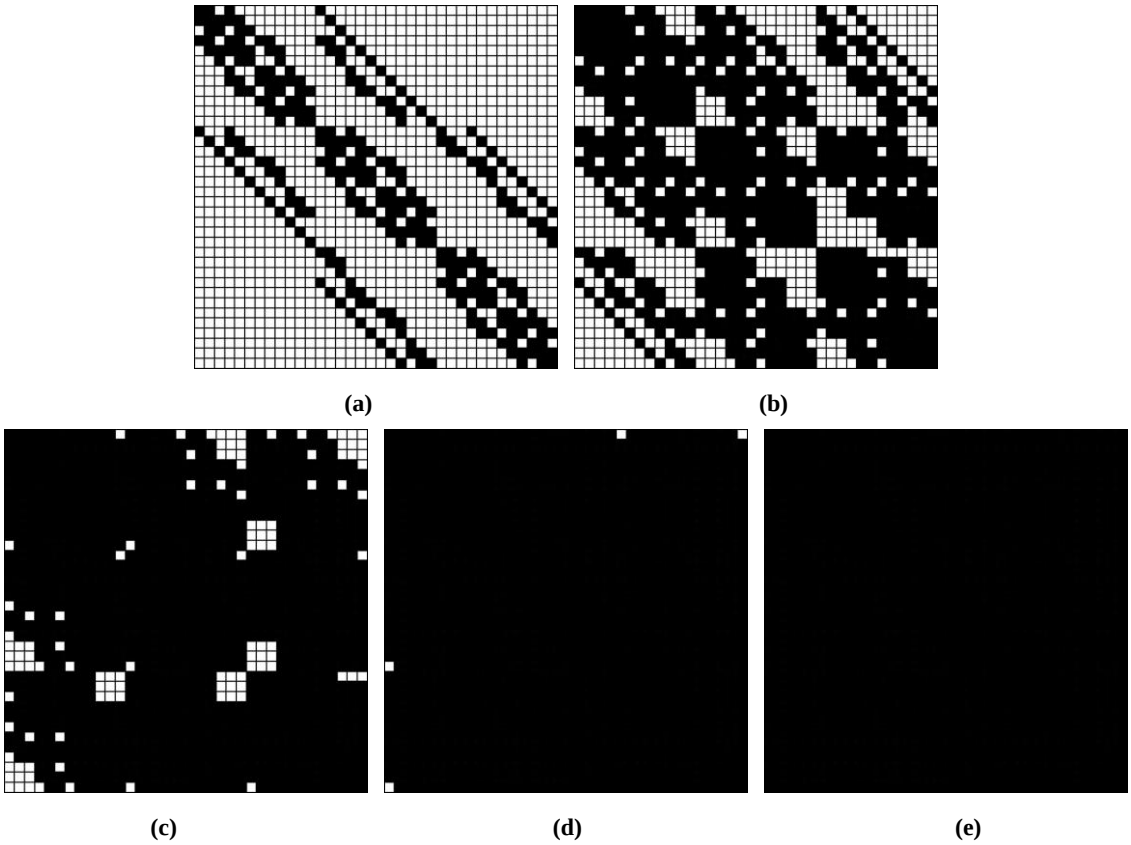


Figure 45: Connectivity matrices of a 3D hexagonal grid (3*3*4). Path degree varies from 1 to 5 in (a) to (d) respectively

4.4) Performance Metrics

The main performance metrics in this thesis are the execution time of tasks, processor utility, the network utility, the aggregate transaction time of all nodes and the aggregate link wait time which shows how much time nodes were waiting to access the control of a channel for packet transactions. The state and performance of a network can be evaluated using other metrics as well. Some of the most important of such metrics are degree, diameter, bisection bandwidth and throughput of the proposed platform.

As discussed in chapter 2, the degree of a node in a network is the number of nodes in direct connection with that node. Regarding the 3D hexagonal structure of a BC platform the degree of a node (other than nodes on edges and corners) is 12 (See Table 14 and Figure 43). The diameter of a network is the maximum number of hops between any two given nodes given the path is a minimal path. In a 3D BC of size $N=n*n*n$ the diameters will be n. This is because of existence of vertical links in nodes from different layers of the network.

In a 3D BC of size $N=n*n*n$ each node in the network has up to 12 links with its neighbours among which 6 links are shared with neighbours on its right hand side and the other 6 with neighbours on its left. When such a network is sectioned into two equal pieces there are $n*n$ nodes on the border on each side. Therefore the bisection bandwidth of a 3D BC of size $N=n*n*n$ would be $6n^2$.

These values can be compared to Table 4 which lists metrics of some of the most popular topologies already used in HPC systems. It can be seen that the BC platform's metrics can match (or be better than) those topologies' metrics.

Chapter 5: Network-partitioning

5.1) Packet Collision Problem

The main motivation for incorporating a network-partitioning algorithm in the platform proposed in this thesis is to solve the packet collision problem which is known to be the main source of delay in data transmission in wireless networks.

Some of the most popular strategies to tackle packet collision are discussed in section 2.6. The RTS/CTS mechanism effectively lowers the packet collision incidents but it comes with a drop in effective bandwidth because of the extra packets needed for each data packet. RTS, CTS and data acknowledgment are the extra packets needed to securely send a packet from the transmitter to the receiver. It should be taken into consideration that the packet collision is not guaranteed to be totally eliminated in this scheme; although the number of collision incidents is expected to be reduced dramatically.

Another option is the timeslot mechanism which can guarantee total elimination of packet collision. But the bandwidth for each node is reduced because of the fact that the overall bandwidth available to all nodes is divided between them.

The strategy adopted for the platform proposed in this thesis has a minor analogy with timeslots but it operates on the frequency domain rather than time domain. In other words, in timeslot scheme different time slots are dedicated to different nodes; whereas, in this thesis different channels are dedicated to different nodes to communicate with certain nodes. RTS/CTS mechanism is not used because of its overhead (extra packets per data packet) and the fact that it cannot reduce the collision incidents to zero. The current mechanism for avoiding packet collision is based on dividing available spectrum between nodes but it can be easily converted to timeslot scheme. From this point of view all the material in next three sections can be applied to the time domain as well to make an effective timeslot mechanism. A multi-channel approach is adopted for the wireless network used in the BC platform and by choosing right number of radio devices and right channels for each radio in each node the packet collision is guaranteed to be completely eliminated.

5.2) Network-partitioning Criteria

In previous section a multi-channel network was introduced as the backbone for the 3D wireless grid proposed in this thesis. Like all other multi-channel wireless networks (e.g. wireless internet access points and mobile phone networks), this network needs a

network-partitioning algorithm to divide the network into (overlapping or none overlapping) regions inside which a unique channel is used for communication between nodes. Network-partitioning algorithms differ mainly in the criteria they are going to satisfy. This has a direct effect on how the algorithm looks like and how it works. Two-stage algorithms are very popular in network-partitioning. A two-stage network-partitioning algorithm is designed and implemented against the following criteria:

- Eliminating packet collision;
- Maximising the network connectivity;
- Minimising the number of channels.

Referencing back to the design issues listed in section 2.7 it can be seen that some of those issues are included in the criteria while some others are not main issues in this thesis. Interference, Connectivity and fault tolerance are among the important issues in this algorithm. Stability is not an important issue in designing this algorithm. Dynamic measures to increase the throughput of the network are not included. It will be shown that the algorithm has a polynomial execution time but this should be noted that because of the static nature of the network there is no need to run the algorithm more than once. This can be done before running the main simulator. The results can be stored and used in next experiments as far as the network size and dimension are not changed. For this reason the algorithm time is not a main design-time concern for the network-partitioning algorithm. In addition to the algorithm's criteria the topology of the network is also very different from other applications like wireless Internet access devices and cell phone networks. The number of nodes, their 3D formation and their closed packed arrangement are some of this network's unique features. These unique features make the network-partitioning algorithm designed for this network different form other algorithms in this field. The next two sections discuss the two stages of the network-partitioning algorithm in more details. The first sub-algorithm partitions the network and the second one assigns proper channels to the partitions.

5.3) Zoning; Proposed Algorithm

The zoning algorithm divides the network into overlapping sections (which are called *zones* in this thesis) in a way that:

- Each node in a zone can hear from all other nodes in that zone;
- The zone is in its maximal size; i.e. all nodes that satisfy the first criteria have joined the zone.

This means that a zone is a sub section of the communication range of its members (see Figure 46). This figure shows that a zone is in fact the intersection of communication

range of three neighbours in a 2D hexagonal grid. In this figure *r* is the radius of the radio range and *z* is the radius of the zone. The same idea can be generalised to 3D networks as well. A zone in a 2D environment is a rounded triangle whereas in 3D it looks like a pyramid. The zone members are the corners of these triangles and pyramids. The idea is that all zone members can use a unique channel dedicated to that zone. The nodes near but outside the zone are banned from using that channel for any of their data transactions. The green and red areas in the bottom right image in Figure 46 show these two groups of nodes. In this figure it is assumed that the radio range just covers the node's direct neighbours. The radio range plays a role in defining the inclusion area (i.e. the green area in Figure 46 which is the zone itself) while it is the interference range of the node which determines how big the exclusion area (i.e. the red area in Figure 46) is. Like many other simulations, in this simulation it is assumed that the interference range of a radio device is approximately twice as long as its radio range.

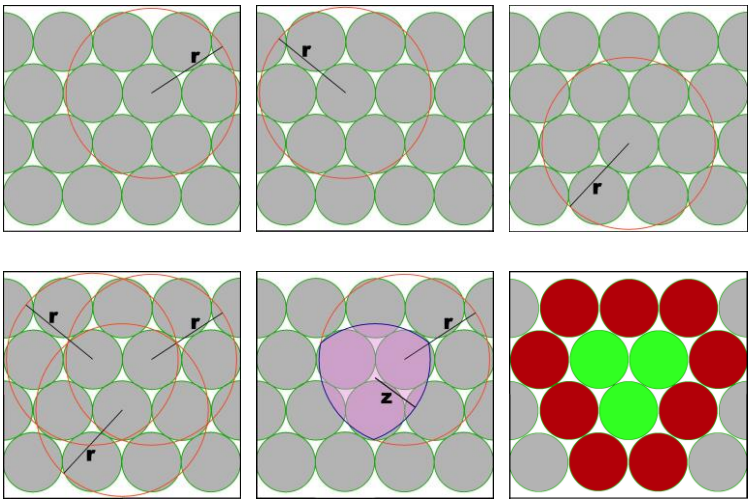
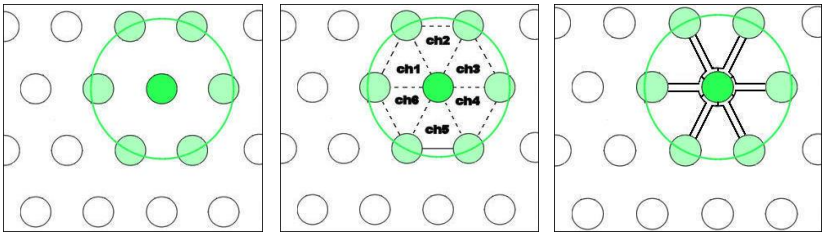


Figure 46: The process of forming a zone. The channel inclusion and exclusion areas are defined.

The zoning algorithm is designed so that each zone accepts as many member nodes as possible provided a new member can hear from all old members of the zone. This means that the zoning process stops only when no other nodes can join the zone. A node in the network can be a member of more than one zone. These zones are overlapping with each other. As a result, a node can communicate with some of its neighbours over more than one channel. In a 2D network there can be up to 6 zones per node (Figure 47.b). This figure also shows that the central node can use two channels to communicate with either of its neighbours. This provides redundant paths for all nodes and increases the connectivity and robustness of the network. In case of a failure in a link between two nodes there are alternative routes and the connection is not necessarily lost. This also helps in case of heavy traffic over a link. In this case alternative paths can redirect the traffic and help balancing the traffic load. This degree of flexibility comes with the

price of an increase in the number of zones and consequently an increase in the overall number of channels needed to set the network up.



(a) **(b)** **(c)**

Figure 47: (a) and (b) A central node and its zones. (c) Compared the wireless connections with their wireline equivalent.

Figure 47.c compares the proposed wireless network with its wireline equivalent. The equivalent wireline network can also be seen in Figure 48. Each zone is replaced by a small star wireline network which has just three members. Redundant independent star networks are the equivalent of overlapping zones in wireless version.

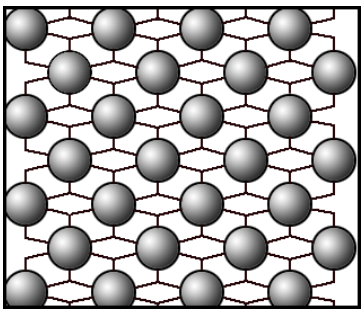


Figure 48: A wireline equivalent of the proposed wireless network in 2D

A greedy algorithm is designed and developed for zoning process. Its pseudo code is shown in Algorithm 1. The algorithm needs to know about the topology of the network. One standard way to represent a graph is via its adjacency lists. An adjacency list[25] is a set of unordered lists in which *i*th member of the set (*i*th list) contains all neighbours of node *i*. The radio range of the nodes determines their neighbours. The adjacency list in proposed zoning algorithm is called *Neighbours* and is passed to the algorithm as an input. The output of the algorithm is sought to be a collection of zones called *ZonesFound* here (each zone is represented by a list of member nodes). The only thing the main function of the algorithm (called *zoning* here) does is calling a recursive function. This recursive function (called *recursiveFindNodes*) is the function that actually handles the zoning process. This sub-algorithm is called once for each of the members of the network in the main function (line 6 in Algorithm 1).

Like any greedy algorithm two sets are involved in *recursiveFindNodes* function: *Candidates* and *Answer*. The *Answer* set contains a list of nodes that are chosen for a zone. In line 6, when the *recursiveFindNodes* sub-algorithm is called for the first time

[25] As defined in: http://en.wikipedia.org/wiki/Adjacency_list

for any node -which is referenced to as *central node* in this algorithm- the *Answer* set includes the *central node* only. If a zone is found by the recursive function, then it is added to the list of zones found so far (line 18). At any stage of the algorithm, the *Candidates* list contains all the nodes that have a chance for being added to the *Answer* list. At the top stage (line 6) this list contains all the neighbours of the *central node*. When the *candidates* set is empty it means that an answer (a zone) is found and no other node can be added to it (line 17); otherwise the same recursive function is called with the current *Answer* and *Candidates* lists (line 21) until the *Candidates* list is empty or all its members are checked.

```
1: // Global variables:
2: array-of-sets  Neighbours []        // Neighbour[i] is the adjacency list for the network's ith node.
3: set-of-sets ZonesFound              // An initially empty set of sets. At the end, it has all the zones.


4: void Zoning () {
5:          For i=0 to sizeOf (Neighbours) do // Neighbours has a list for any node in thee network.
6:                    recursiveFindNodes ({i} , Neighbours[i])
7: }


8: void recursiveFindNodes (Answer, Candidates) {
9:          // Inputs:
10:         //          Answer: an integer set representing one answer list (a zone)
11:         //          Candidates: a set representing the candidates list (initially all neighbours of node i)
12:         int-set TempAnswer          // A temp integer set representing an incomplete answer (zone)
13:         int-set TempCandidates      // A temp integer set representing current state of candidates
14:         for i=0 to sizeOf(Candidates) do {
15:                   TempAnswer ← Answer ∪ Candidates[i]
16:                   TempCandidates ← Candidates ∩ Neighbours[Candidates[i]]
17:                   if (TempCandidates == ∅) then {
18:                             ZonesFound ← ZonesFound  ∪ TempAnswer
18.a:                           for j=0 to sizeOf(TempAnswer) do {
18.b:                                     int nodeMember= TempAnswer[j]
18.c:                                     int zoneNo= sizeOf(Nodes[nodeMember].Z)
18.d:                                     Nodes[nodeMember].Z[zoneNo].M= TempAnswer
18.e:                           }
19:                   }
20:                   else
21:                             recursiveFindNodes (TempAnswer, TempCandidates)
22:         }
23: }
```

Algorithm 1: Zoning sub-algorithm

Lines 18.a to 18.e are not part of the algorithm itself but they are added to facilitate the execution of the second part of network partitioning process (channel assignment sub-algorithm). What these lines do is updating the *Nodes* data structure which is to be used in channel assignment algorithm (Algorithm 3). Those lines add a new entry to the *Z* list in *Nodes* and store the newly found zone in its *M* field. More information on *Nodes* data structure and its fields can be found later in this chapter.

The algorithm works correctly as long as it can guarantee that in any stage of the algorithm the *Candidates* list only contains all nodes that can be added to the *Answer* list. This means that each node in *Candidates* list is a neighbour to **all** the members of *Answer* list. Line 16 gives the aforementioned guarantee as in any run of the recursive function the candidates list is refined by performing a union operation with neighbours of the newly selected node for *Answer* list. Therefore, by performing the union operation in line 16, the new member of the *Answer* list is removed from the *Candidates* list (like any other greedy algorithms) followed by all members of the *Candidates* list which are not a neighbour of the new member of the *Answer* list.

Compared to a standard greedy algorithm, the only twist in the algorithm is that all the members of the candidates list that are not neighbours of the new member of *Answer* set also leave the candidates list. To have overlapping zones it is important to let all different combinations of nodes be tested (refer to the loop in lines 14 and 15). The *"if"* statement in line 17 prevents premature termination in process of finding a zone. This way, it is guaranteed that zones will be in their maximal size and no other node can be added to any zone.

The time complexity of the algorithm can be studied using *big-O* asymptotic notation. The notation *O(.)* is defined as follows:

f(x) = O(g(x)) if and only if there are real numbers *M* and x_0 such that f(x) ≤ *M**g(x) for all x ≥ x_0[26]. This notation is usually used to describe the behaviour of a function when its variable is very large (or infinite). We use this notation to study how the execution time of the proposed zoning algorithm increases as the number of nodes (its input) increases towards very large numbers.

Execution of Algorithm 1 has three phases. The first phase which is not explicitly mentioned is the preparation of the inputs (*Neighbours* data structure). The second part is the Zoning function. This function just calls a recursive function once for each node which brings the algorithm to its third phase (*recursiveFindNodes* function). The execution times of these phases in terms of big-O are:

- To construct the *Neighbours* data structure, the geometric coordination of each node should be adjusted which takes *O(n)* time units where *n* is the number of nodes in the network. Consequently, to define neighbours of each node, the location of a node should be checked with those of all other nodes via two nested loops each of length *n* (complexity of order $O(n^2)$). Therefore the whole preparation phase is of order $O(n)+O(n^2)$. When the number of inputs increases

[26] The definition is based on: http://en.wikipedia.org/wiki/Big_O_notation

towards infinity, the $O(n^2)$ part dominates; therefore this phase has an execution time of order $O(n^2)$.

- The *Zoning* function is called once per each node and therefore it has an order of $O(n)$ multiplied by whatever the execution order of the function *recursiveFindNodes* (third phase) is.
- The function *recursiveFindNodes* has a main loop (line 14) which repeats x times where x is the size of *Candidates* list. x has a finite value and does not scale with the number of nodes. There is also a second (inner) loop in this function (lines 18.a to 18.e) which is added to assist the execution of the channel assignment algorithm. This loop also executes for a limited number of times and is not related to the number of nodes. Parts of the algorithm that have a constant execution time do not take part in the big-O analysis of the algorithm. The only part of the main loop which its execution time depends on the number of nodes is the union operation (line 18). The union operation is needed because the algorithm as it is, may generate multiple copies of the same zones when treating different members of them as central nodes (on different executions of line 6). Depending on the implementation of set operations, the execution time of this operation varies. When a set is implemented with a heap, the execution time for a union operation can be of order $O(\log_2(n))$.

As a result the aggregate operation time of both functions (*recursiveFindNodes* and *Zoning*) is of order $O(n*\log_2(n))$. Therefore, the overall time complexity of Algorithm 1 is $O(n^2)+O(n*\log_2(n))= O(n^2)$.

The execution of the aforementioned functions can be improved by making two changes in Algorithm 1. In the enhanced version of the algorithm (Algorithm 2) the main loop in *recursiveFindNodes* function is altered so that it does not process members of *Candidates* list with indexes lower than the current central node. This is based on the fact that any zone with such members of *Candidates* list is definitely found before when that member of *Candidates* list was the central node.

This means that there will be no duplicate zones in *ZonesFound* data structure. As a result, a newly found zone can simply join the *ZonesFound* without a need for union operation. This gives a constant operation time ($O(1)$) to the recursive function and consequently the *Zoning* function is of $O(n)$ order. But the overall execution time of the algorithm is still dominated by the preparation process which is of order $O(n^2)$.

Figure 49 shows a 3D neighbourhood in which 8 pyramid-shape zones exist around the central node. Each neighbour shares two zones with the central node. It can be seen that

the top most layer and the bottom most layer are not exactly the same. This means that – like all FCC lattices- there are three distinct layers in this topology.

```
1: // Global variables:
2: array-of-sets  Neighbours []      // Neighbour[i] is the adjacency list for the network's ith node.
3: set-of-sets ZonesFound            // An initially empty set of sets. At the end, it has all the zones.


4: void Zoning () {
5:          For i=0 to sizeOf (Neighbours) do // Neighbours has a list for any node in thee network.
6:                    recursiveFindNodes (i, {i} , Neighbours[i])
7: }


8: void recursiveFindNodes (CentralNode, Answer, Candidates) {
9:          // Inputs:
10:         //          Answer: an integer set representing one answer list (a zone)
11:         //          Candidates: a set representing the candidates list (initially all neighbours of node i)
12:         int-set TempAnswer          // A temp integer set representing an incomplete answer (zone)
13:         int-set TempCandidates      // A temp integer set representing current state of candidates
14:         for i=0 to sizeOf(Candidates) do {
15:                   if (CentralNode < Candidates[i]) {
16:                             TempAnswer ← Answer ∪ Candidates[i]
17:                             TempCandidates ← Candidates ∩ Neighbours[Candidates[i]]
18:                             if (TempCandidates == ∅) then {
19:                                       ZonesFound ← ZonesFound  + TempAnswer
19.a:                                     for j=0 to sizeOf(TempAnswer) do {
19.b:                                               int nodeMember= TempAnswer[j]
19.c:                                               int zoneNo= sizeOf(Nodes[nodeMember].Z)
19.d:                                               Nodes[nodeMember].Z[zoneNo].M= TempAnswer
19.e:                                     }
20:                             }
21:                             else
22:                                       recursiveFindNodes (CentralNode, TempAnswer, TempCandidates)
23:                   }
24:         }
25: }
```

Algorithm 2: An enhanced version of Zoning sub-algorithm

For many nodes in the network (except those locating on edges or corners of the network) there are eight zones per node. The algorithm imposes no limit on the number of zones and tries to find as many zones as it can. This may not be always desirable. If – due to technical considerations- the number of zones needs to be reduced, a modification in Algorithm 1 is needed to find just a limited number of zones (i.e. less than eight zones in 3D) for each node. The number of zones is the same for a node regardless of its position in the network. This lets the network have a monotonic connectivity. The algorithm can be modified by letting the position of a node in the network play a role in generating different number of zones for it. This may be particularly useful if some points in the network are proven to be communication hot spots and need higher degree of connectivity whereas other parts of the network can cope with lower connectivity. In this case, more zones are needed for a hot spot; and therefore the algorithm can respond to this demand by lowering the number of zone

members and increasing the number of zones. At the current version of the algorithm there were no need for such modifications and it is simply assumed that all the nodes have the same communication demands.

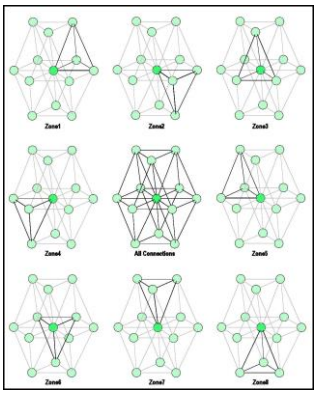


Figure 49: A 3D grid of wireless devices and the pyramid partitions around a central node

5.4) Channel Assignment-Proposed Algorithm

The channel assignment algorithm used in this thesis can be regarded as a variation of map colouring algorithm. Mathematicians in 19th and 20th century tried to find an answer to the question of how many colours are needed to colour a map (consisting of contiguous regions) so that any two adjacent regions have different colours[27]. In 1890 P.J. Heawood [151] showed that any map can be painted with only five colours. Also it can be shown that three colours are not enough for maps in which a region is surrounded by an odd number of other regions that touch each other in a cycle. In 1976 K. Appel and W. Haken proved that four colours are enough for colouring any sort of maps (their work was published in 1977 and 1989 in [152], [153], [154] and [155]).

The algorithm proposed in this thesis differs from the original map colouring problem in the sense that in the current algorithm two zones cannot have the same channel (compare with colours in map colouring) if there are less than two zones located between them.

In this sub-algorithm the zones found in previous stage (zoning) are used as an input to assign proper frequency channels to them. The word "*proper*" in this context means that the channels should be chosen in a way that in none of close neighbourhoods the same channel is used. The key parameter in determining what zones are "*close*" to each other is the interference range of nodes. Two zones can use the same frequency channel if and

[27] A more formal definition of the problem is presented in 2008 by Georges Gonthier [293] which states: "A planar map is a set of pairwise disjoint subsets of the plane, called regions. A simple map is one whose regions are connected open sets. Two regions of a map are adjacent if their respective closures have a common point that is not a corner of the map. A point is a corner of a map if and only if it belongs to the closures of at least three regions. Theorem: The regions of any simple planar map can be coloured with only four colours, in such a way that any two adjacent regions have different colours."

only if none of the members of one of the zones falls inside the interference range of none of the members of the other zone.

It should be mentioned that in the platform proposed in this thesis all nodes are assumed to have both the same communication range and the same interference range. This is not always the case in real-world applications. The radio range differs from a radio module to another. Among other reasons this can happen because of the difference in the level of electrical power applied to the devices and physical obstacles between the transmitter and receiver which cause different patterns of reflection, absorption and diffraction. However, in a controlled area and a dense network like the platform proposed in this thesis there is a legitimate chance to approximate all the radio ranges to one common distance. Algorithm 3 shows the pseudo code of the channel assignment algorithm used in this thesis. In the original version of the map colouring problem no neighbouring regions can have the same colour. What makes the channel assignment different from the original map colouring problem is that in channel assignment problem the shortest distance between two regions with the same colour should be greater than or equal to the interference range of nodes.

The input to the proposed channel assignment algorithm is a data structure representing the nodes of the network. Each member in the *Nodes* structure contains a list of its zones (*Z*), a list of nodes locating in its interference list (*I*) and a list of channels that cannot be assigned to any zone of that node (*E*). Each member of the *Z* structure consists of a channel assigned to that zone (*F*) and a list of nodes belonging to this zone (*M*). The list of forbidden channels for a zone (called *E*, also known as exclusion list) is initially empty and new channels are added to it as the algorithm assigns channels to other zones. The interference list (*I*) is needed because the coordination of nodes are not included in *Nodes* structure. The information stored in list E and list *M* (in *Z*) could have been extracted from other members of the *Nodes* array; but it comes with the expense of increase in the algorithm's execution time. This means that there are some redundant data stored in the *Nodes* data structure. Although this increase the size of space needed by that array; but, in light of significant reduction in algorithm's run time, it is decided to include those redundant data.

The goal of the proposed algorithm is to assign values to the frequency channel of each zone (represented by *Nodes[i].Z[j].F*) so that the assigned frequency satisfies the criteria proposed in section 5.2.

When a channel is going to be assigned to a zone, the smallest channel number which is not a member of the exclusion list (*E*) is selected (lines 13 to 19 in Algorithm 3). As soon as a channel is assigned to a zone (line 26), that channel is also added to exclusion

list of all zones in the interference range of all the zone members (lines 27 to 29). This guarantees that there would be no interference between none of the zones over the network. When a channel is assigned to a zone all the zones and zone members affected by that assignment are updated as well (lines 24 to 29). This saves execution time when checking other members of that zone (refer to *if* statement in line 25).

```
1: void ChannelAssignment (Nodes) {
2:       // Inputs: Nodes: A set of nodes, where each member N consists of:
3:       //              Z: A set of zones. Each zone consists of:
4:       //                      F: An assigned frequency channel initially set to 'null'
5:       //                      M: A set of nodes belonging to the same zone membership
6:       //              I: An interference list including nodes in the signal interference range
7:       //              E: An initially-empty set of frequencies forbidden for this zone
8:       int Frequency ← 1
9:       Boolean match ← FALSE
10:      for i=0 to sizeOf(Nodes) do
11:              for j=0 to sizeOf(Nodes[i].Z) do {
12:                      if (Nodes[i].Z[j].F != null) then continue
13:                      Frequency ← 1
14:                      match ← FALSE;
15:                      repeat
16:                              if (Frequency ∈ Nodes[i].Z[j].E) then
17:                                      Frequency++
18:                              else
19:                                      match ← true
20:                      until (match)
21:                      for k=0 to sizeOf(Nodes[i].Z[j].M)  do {
22:                              int zoneMem ← Nodes[i].Z[j].M[k]
23:                              Nodes[zoneMem].E ← Nodes[zoneMem].E ∪ Frequency
24:                              for m=0 to sizeOf(Nodes[zoneMem].Z) do
25:                                      if (Nodes[i].Z[j].M == Nodes[zoneMem].Z[m].M) then
26:                                              Nodes[zoneMem].Z[m].F ← Frequency
27:                              for m=0 to sizeOf(Nodes[zoneMem].I) do {
28:                                      int interfer= Nodes[zoneMem].I[m]
29:                                      Nodes[interfer].E ← Nodes[interfer].E ∪ Frequency
30:                              }
31:                      }
32:              }
33: }
```

Algorithm 3: Channel assignment sub-algorithm

In many channel assignment algorithms in other applications it is tried to include a fairness criterion. This is mainly for balance in traffic of channels to minimise things like inter-channel crosstalk. Algorithm 3; however, does not include a fairness criterion mainly because it is designed for a simulated network and the algorithm should be as simple as possible. When the algorithm is going to be applied to a real-world network, the fairness criteria should be included.

Figure 50 shows the results of the Algorithm 3 (with added fairness criterion) applied to a 2D grid. Each distinct colour represents a distinct frequency channel. The repetitive pattern of colours in these figures indicates that the total number of channels both in 2D and in 3D grids are fixed numbers and are irrelevant of the grid size.

The execution time of the algorithm is shorter than other channel assignment algorithms because in many applications the algorithm should be executed in specific intervals but in the platform proposed in this thesis the topology of the network is fixed and there is no need to run the algorithm more than once. Many channel assignment algorithms use heuristic measures to approximate the optimised answer otherwise their execution time would increase exponentially with the network size. Although the proposed algorithm can be executed only once before running the simulator for several times, it is still good to know how the execution time of Algorithm 3 increases with the network size.

The preparation of the *Nodes* algorithm can take no time if the modified version of the zoning algorithm (Algorithm 2) is used in which the *Nodes* is loaded with information generated by that algorithm. The main body of the proposed algorithm consists of two nested *for* loops. The outer loop repeats *n* times; where *n* is the size of the network. The inner loop runs for *z* times; where *z* is the number of zones a node has. This is a fixed number and it is shown in Figure 49 that in a 3D grid *z* is less than or equal to 8. This means that the complexity of the main body of the proposed algorithm is of order *O(n)*O(1)*Complexity of the contents of the inner loop*.

The *repeat-until* loop (lines 15 to 20) has a limited upper limit on its repetition because of the repetitive nature of the channels (see Figure 50). Therefore this loop has a time complexity of order *O(1)*. There are also three other *for* loops in lines 21, 24 and 27. The latter two locate inside the former. All these loops have a constant execution time regarding their fixed repeat times. This means that the whole last three loops (lines 21 to 29) have an execution time of order *O(1)*. As a conclusion, the channel assignment algorithm has a linear execution time of order *O(n)*O(1) *O(1)=O(n)*.

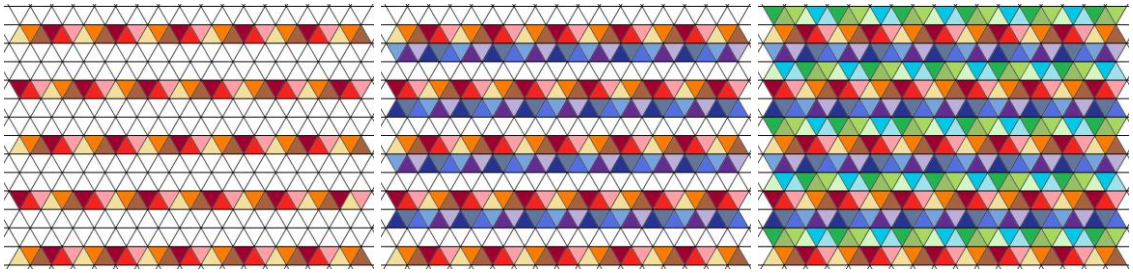


Figure 50: Representing channel assignment as a variation of map coloring problem

There is some room for improving the algorithm particularly in two ways. First is assigning orthogonal channels to neighbouring zones to decrease the interference. Second is choosing the channels so that all channels are assigned to an almost equal number of zones (to satisfy fairness criterion). In the present version of the algorithm a free channel (which is not a member of the exclusion list) with the smallest channel identifier is selected for a zone for the sake of simplicity of implementation. This means that there are chances that smaller channel IDs are assigned to larger number of zones.

This may cause problems in certain situations. It is not studied how this may affect the performance of the network and what the extent of this effect would be. This can be a subject of research in future work.

5.5) Simulated Signal Interference Mechanism

The signal interference mechanism adopted in the current version of simulator is using the geometric location of the nodes. The mechanism is as follows:

Each node has two radio attributes: Radio range and interference range. The radio range of a sender node is the distance in which the signal can be successfully decoded. The interference range is a larger area in which the signal can be detected but it is too weak to be decoded. This means that although the signal cannot be accepted as valid data but it is still strong enough to interfere with another signal which otherwise may had a chance to be decoded correctly. The second signal, in this case, cannot be decoded because of high noise added to the channel because of the first (weak) signal. The interference range is twice the radio range in most of widely used radio simulators. It is the case with the current simulator as well.

Apart from signal interference there are other sources of noise for radio signals in real world. As this is the first version of the simulation tool and for the sake of simplicity in implementation those sources of noise are not modelled in the current version of the simulator. They will be added to the signal propagation model in future versions of the tool kit. An independent channel object is implemented in the simulator programme to handle the whole signal propagation and check for any packet collisions.

5.6) Summary

Network-partitioning and channel assignment algorithms have been already used in applications like mobile phone networks and wireless internet access. From this point of view the twin algorithms (network-partitioning and channel assignment) introduced in this thesis are not 100% new concepts. What makes these algorithm novel and different from other similar algorithms is mainly the criteria against which these two algorithms are designed. In particular the elimination of packet collision is the main objective of the twin algorithms which is not sought in other algorithms of this type. Eradication of packet collision is achieved by solving the Hidden Node Problem which is a common problem in many wireless networks.

The price paid for solving that problem is that the communication range of a node over a given frequency channel is reduced to something which is called a *zone* in this thesis. Figure 47 shows that a zone is only a fraction of the communication range of a node.

This means that not all the neighbours of a node are included in a zone and consequently that node needs more than a zone to communicate with all of its neighbours.

In this chapter a systematic method is introduced to partition a network into zones so that the elimination of hidden node problem can be guaranteed. The algorithm has an execution time of order $O(N^2)$ where N is the number of nodes in the network. Also a channel assignment algorithm is introduced which guarantees no signal intervention between any two zones happens and at the same time keeps the number of frequency channels as low as possible. This algorithm also has an execution time of order $O(N^2)$. In fact, the network-partitioning algorithm makes sure there is no signal intervention inside a zone while the channel assignment algorithm removes any chance for intervention between zones. Proofs for correctness of twin algorithms have been included in this chapter. Results in chapters 8 and 9 demonstrate their correct execution in practice.

Chapter 6: Task-modelling

6.1) Tasks vs. Task-models

Like other parallel platforms the proposed grid should run a set of workloads to evaluate its performance. LINPACK is one of the widely accepted toolkits for measuring the performance of a computer. Its parallel version is also available to measure the actual maximum performance of supercomputers (as opposed to their peak maximum performance which in most of the times cannot actually be achieved). Other benchmarks are also introduced including NASA's parallel benchmark (NAS Parallel Benchmarks – NPB) which is a set of benchmark tasks applicable to highly parallel machines to determine how efficient these machines are.

The parallel platform proposed in this thesis is implemented in a simulated environment; also this thesis is not about studying different implementations of different tasks in the real-world. What this thesis needs is a task abstracted from implementation details. Furthermore, the natures of the tasks are also not of great importance in this thesis. What this thesis is eager to find out is how the proposed BC architecture responds when it comes to handling network traffic in situations that looks like real situations. This similarity is in terms of the pattern of computation and communication periods real-world tasks have. In other words, we want to see how nodes react when they mimic the computation and communication behaviour of real tasks.

To do this, it is decided to introduce a concept called *task-model* which as its name suggests models a task in a way that its pattern of communication and computation intervals are preserved while neither its internal operation nor the way it is implemented are of interest; and therefore those features are omitted from the model.

A task-model is essentially an artificial traffic generator which mimics the pattern of packet generation of one or a set of real-world tasks. A task-model does not run a real task and therefore does not do any real computation. The packets a task-model generates do not contain real and meaningful data. The only things that are real in a task-model are the computation and communication intervals. This is the only thing a designer needs to extract for a task-model from a real-world task. All of these mean that a task-model is a higher level of abstraction for modelling a real task in a way that only its communication behaviour is preserved. This level of abstraction is vital for this thesis particularly because the platform is going to be built only in a simulation environment.

Merge Sort algorithm[28] is used as an example to demonstrate the task-model generation process:

What Merge Sort algorithm does is basically dividing an input array of numbers into two sub-arrays, sorting each sub-array separately and then merging sorted sub-arrays into one sorted output array. Since each sub-array would be an input to another run of a sort algorithm of any type; the Merge Sort can be used recursively to split the two sub-groups into even smaller sub-arrays.

Merge Sort has the ability of being implemented as a parallel algorithm in the sense that each sub-array can be sorted in parallel on different processors. It is the balance between communication speed and the computational ability of processors that determines when processors should stop splitting the data for further parallel execution and sort the input data locally.

Like any other task in real-world, the only aspect of Merge Sort algorithm which is of interest in this thesis is the pattern of communication and computation periods each processor has. This means that while the contents of the input and output data is not important; it is just the size of data which is of concern in this thesis. Also the time it takes to send and receive the data and also the time needed to sort the input array is important.

This means that from this thesis's point of view a Merge Sort task starts with a period of time in which the task splits the input data to two pieces. This typically does not depend on the size of data and therefore has an execution time of order *O(1)*. Then the task starts looking for candidates among its neighbour processors that accept sorting each sub-array remotely. This is also of order *O(1)*. Then the data will be sent to those processors. The time needed for this step is of order *O(N)* and depends on the data size (*N*) and the speed of the communication link. On the receiver side, the arrival of a data packet starts a new sort algorithm. In case Merge Sort is the algorithm of choice in the new node, the aforementioned process is repeated until the size of data is small enough to make it possible for a local processor to run the sort algorithm locally. In case of a local execution of Merge Sort on a processor the execution time is of order $O(N*log_2N)$. Returning the results (of size *N*) to the parent node is also of order *O(N)*. On arrival of all result packets, a node merges the results which takes a time of order *O(N)*.

To construct a task-model for Merge Sort, an internal flag is used to model the status of the processor. This flag can have one of the following states:

[28] According to Donald E. Knuth [292] Merge Sort is invented by John Von Neumann in 1945.

- Communication: Indicates the node is sending/receiving a packet to/from another node.
- Computation: Indicates that he processor is busy. There is no real computation.
- Waiting: Indicates that the node has sent sub-arrays to other nodes and waits for results.
- Idle: Indicates that the node is not involved in any tasks.

There is no need to develop a code for sort algorithm. Also no real data is needed. A task-model does not produce real results but what it does is:

- It finds right number of nodes that among other things means it occupies the links for a right amount of time cycles for transactions needed for this search. This step takes an unknown number of time cycles which depends on the availability of channels and number of idle nodes in the neighbourhood.
- It sends data packets to selected nodes with dummy contents but with right packet length, to the right receiver, over the right link and over the right time interval (*N* time cycles for Merge Sort).
- It switches to Computation mode (but not really executes a task) for a right amount of time cycles ($N*log_2N$ time cycles for Merge Sort).
- It sends the data back to the original node over the right link and over the right time interval (*N* time cycles for Merge Sort) but again the content of the result packet is not real.
- All other computation (e.g. initialisation and merging results) and communication intervals mimic the real task (in Merge Sort it takes *N* time cycles to merge the results).

An accurate task-model should accurately follow the computation and communication activities of a task without using real data and real code. What follows are a number of task-models designed and implemented for the simulated platform. Frequency of communications plays a major role in a task-model. The frequency of communication has also a secondary effect on tasks and that is when the number of transaction between nodes increases the dependency between tasks on different nodes increases. This dependency in some tasks is very high and this means that finishing a task in a node depends on finishing the task on many other nodes. This explains why the main difference between the first two task-models below is the degree of dependencies between tasks on different nodes. Back to grand challenges plotted on Figure 2 some challenges can be spotted with low communication dependency including weather

forecasting. Some other tasks like computational fluid dynamics (CFD) usually need many communications between processors.

6.2) A Simple Parallel Task-model (SPTM)

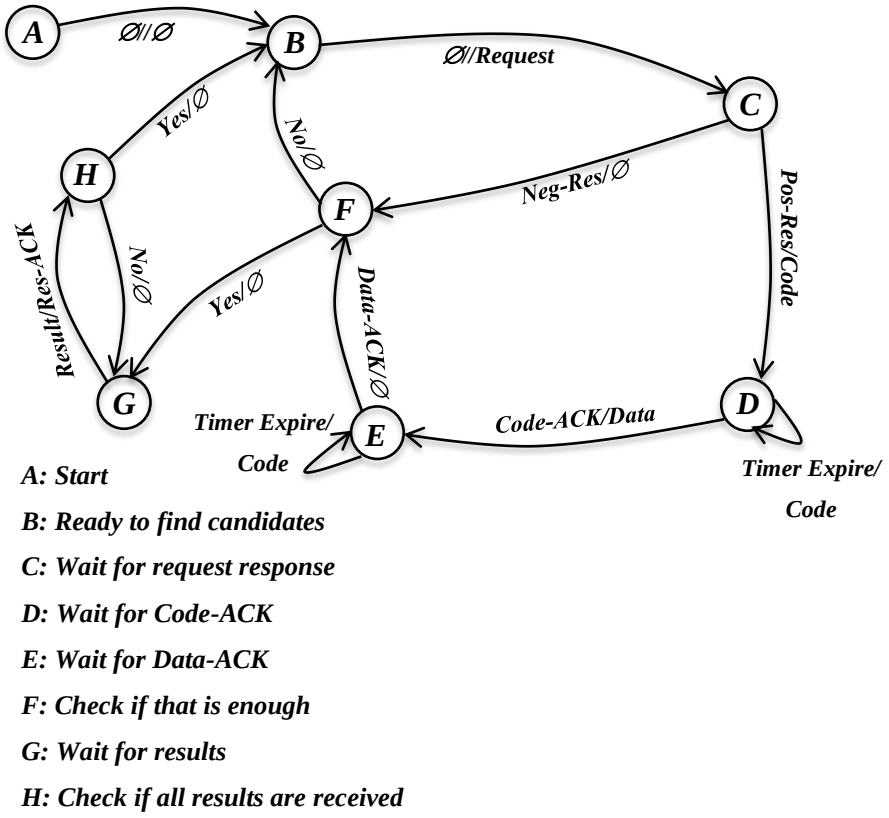


Figure 51: State diagram of SPTM in parent node

The first task-model introduced in this thesis is a task with loose and casual connections with other tasks. Each task is in contact with some of its neighbours through sending and receiving data packets and from this point of view the execution of a task depends on the execution of tasks on some of its neighbours. But the scope of task dependency never exceeds a node's direct neighbours. This can be compared to a number of workloads in real world. Weather forecasting and finite element method (FEM) analysis are among those tasks[29]. In all these tasks an object is divided into small partitions and each node in the parallel platform is in charge of applying the task to one of these partitions. The result of the computations on each partition can be exchanged with its neighbouring partitions for further computations. The point is, the exchange of results is restricted to the immediate neighbouring partitions. This restricts the degree of dependency between processing nodes. The number of synchronisation points (communications) is not too high in this category of tasks.

[29] This is a very general categorisation; and this thesis does not go through detailed parallel programming techniques used for applications like weather forecasting or FEM.

The task-model introduced in this section randomly selects some of its neighbours. Then it sends a data packet to them and waits for results. By receiving all results it executes a merging process and its operation is finished. On the receiver side, the process time of the packet is proportionate to its length. In other words, when a packet of size ***M*** is received the processor needs ***M*** instructions to process the input packet and prepare the results. As mentioned before, the actual transaction time and process time depends also on the data rate of the link and the instruction per second of the processor respectively. The actual transfer time and the actual compute time in this example are ***M/x*** and ***M/t*** respectively where ***x*** is the link's transfer rate and ***t*** is the number of instruction per time cycle.

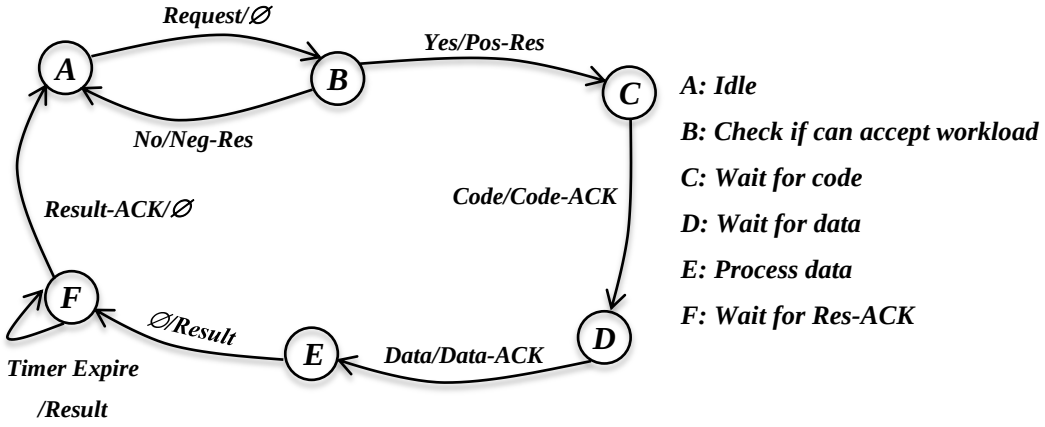


Figure 52: State diagram of SPTM in child node

Nodes can accept their neighbours' packets while they are waiting for the results from other nodes and even when they are already finished with their own tasks. Our test results show that the linear execution time has no particular effect on the behaviour of the network. Other execution times like logarithmic or square execution time can also be adapted. For the sake of simplicity, a linear execution time (with regards to data size) is chosen for implementation.

Packet Type	Description	Parameters passed
nodeStatusRequest	The source node asks for accessing over the target node	Task-model identifier, Task identifier
nodeStatusResponse	The target node accepts the source node's access request	Index if free task, Number of busy tasks
transferData	The source node sends data to the target node	Data transfer time
transferDataACK	The target node acknowledges the data	"0"
transferDataNACK	The target node do not acknowledge the data	"0"
transferResults	The target node sends the results back to the source node	Results transfer time
transferResultsACK	The source node acknowledges the results	"0"
transferResultsNACK	The source node do not acknowledge the results	"0"

Table 15: List of packets in SPTM protocol.

Figure 51 and Figure 52 are two state diagrams that show how the internal state of the task-model reacts to external events (mostly as packets) and internal events (like

start/end of data processing). Nodes can play both parent and child roles simultaneously; i.e. the reception and processing of a data packet can be handled separately from sending a node's own packet to other nodes.

A communication protocol is introduced for SPTM based on the state diagrams in Figure 51 and Figure 52. Table 15 lists the packet types used to implement SPTM.

6.3) A Highly Dependent Task-model

At the other extreme of node dependency are tasks that are highly dependent on a vast number of other tasks. Many workloads of this type can be found among mathematical transformations adopted for parallel execution. Different parallel sort algorithms are also included in this category. Fast Fourier Transform (FFT), Merge sort and Bucket sort are just some of these workloads. Many scientific and industrial parallel applications have this type of workloads. A divide-and-conquer strategy is the basis of most of these tasks. Here the main idea is to split the workload into (equal or non-equal) pieces and recruit other processors to execute the task with these smaller chunks of data. The receiver processors can keep doing the same thing to split the data or task into even smaller bits depending on the granularity of data and code. This means that the execution of a task on a node not only can depend on the execution of the task on its direct neighbours but also it may depend on the execution of the task on many other processors; therefore, the scope of task dependency can easily exceed the small neighbourhoods and even can be over the whole parallel platform.

FFT is an example of such tasks and is chosen to implement the task-model for the simulated platform proposed in this thesis. Theoretically an FFT workload can be split into any number of parts. In a ***radix-n*** FFT the original workload is split into ***n*** pieces. Different radix-n (and even variable-radix) FFTs with different values for ***n*** are implemented and adopted for parallel execution. ***Radix-2*** is the most popular one and ***radix-4*** is also used especially for 2D data like matrices.

The version of FFT used to construct the task-model in this thesis is a variable-radix FFT with a limit of 4 for the number of pieces of data on each stage. The FFT task-model (FFTTM) operation is as follows:

> A task starts with making a decision on whether to run the FFT entirely locally or try to do it in parallel. If the size of data is bigger than a given threshold the node starts finding up to 4 neighbours to do the FFT in parallel by sending polling packets to its neighbours. This stops when 4 idle neighbours are found or all the neighbours are contacted. The task will be executed locally if no idle neighbours are found. Another approach would be waiting for a bit and try again

but the first strategy is adopted in this thesis. It is assumed that the code does not exist on the receiver side; therefore, the code should be first sent to them. Then each piece of data would be sent to the receiver neighbours. On receiving the code and data the receivers take the same steps.

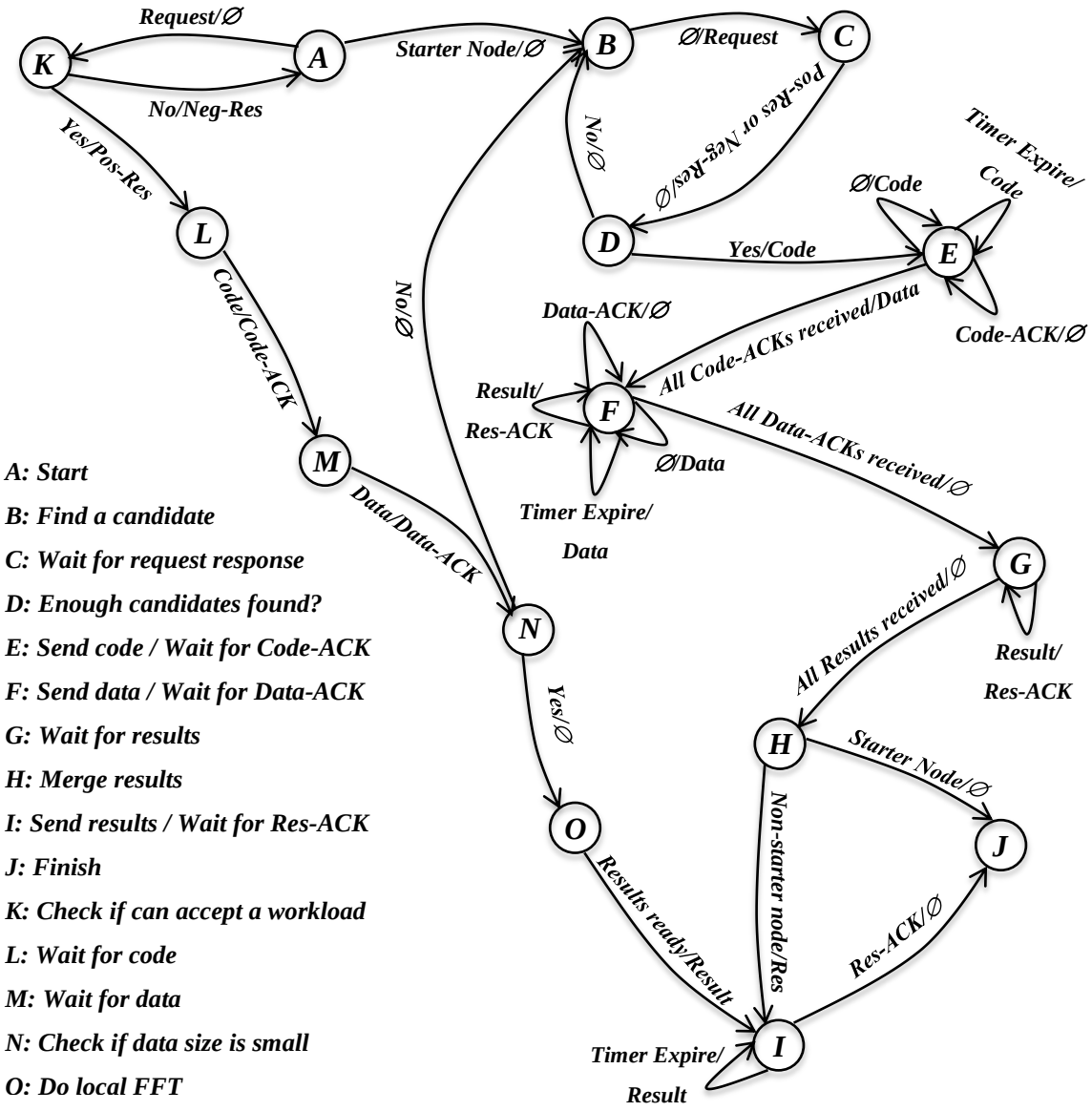


Figure 53: State diagram of FFTTM

If the data size is small enough (lower than the given threshold) the FFT code is executed locally. To remain faithful to the FFT tradition, the local execution time is ***n*log(n)*** where ***n*** is the size of data. The results then can be sent back to the parent node.

On receiving all the results, a parent node need to merge them to construct its own results to send it back to its own parent node. The merge time depends on the implementation of the algorithm. The merge time is zero in this task-model implying that it is an on-place FFT.

An amendment to this task-model would be letting waiting nodes accept other workloads to save time and increase the utilisation of the network. This means that nodes that have already split their workload between their neighbours and are waiting for results can accept other workloads during their wait time. Different variations of this task-model are tested to accommodate multi-tasking and multiple workloads.

Figure 53 shows the internal operations of an FFTTM in form of a state diagram. The architect of the task-model can decide on the maximum number of neighbours a node recruits for parallel FFT. In this thesis different numbers (e.g. 4 and 6) are tested. A discussion is included in chapter 8 on an optimum value for this parameter.

A communication protocol is introduced to translate the state diagram in Figure 53 into command and response packets. Table 16 lists the packet types used to implement FFTTM. The results show that there is always a limit on the performance of this task-model which is because of the imbalance between the pieces of workload sent to selected neighbours. For this reason its structure needs to be changed thoroughly. The following section discusses this new task-model.

Packet Type	Description	Parameters passed
nodeStatusResquest	The parent node asks for accessing the child node	Task-model identifier, Task identifier
nodeStatusResponse	The child responds to the parent's access request	Index if the free task, Number of busy tasks
transferTask	The parent sends task code to the child	Task transfer time
transferTaskACK	The child acknowledges the task code	"0"
transferTaskNACK	The child do not acknowledge the task code	"0"
transferData	The parent sends data to the child	Data transfer time
transferDataACK	The child acknowledges the data	"0"
transferDataNACK	The child do not acknowledge the data	"0"
transferResults	The child sends the results back to the parent	Results transfer time
transferResultsACK	The parent acknowledges the results	"0"
transferResultsNACK	The parent do not acknowledge the results	"0"

Table 16: List of packets in FFTTM protocol.

6.4) Modified FFTTM

The problem with the previous FFTTM is that it sends equal amount of data to all its neighbours. This is because it assumes that its selected neighbours have equal chances in finding idle neighbours. This is not always a valid assumption. The results show that in almost all experiments there is a big difference between the neighbours found by different nodes. This is mainly because of the topology of the network. This means that the dependency tree in many cases is highly imbalanced. The number of nodes and the depth of the sub-trees vary from one sub-tree to another. In such a situation sending an equal size of data to these highly unequal sub-trees end up in huge execution time on smaller sub-trees and consequently reduces the performance of the system.

This shows that although assigning equal workload for all neighbours is a very simple and easy to implement strategy but it can make a significant imbalance in workload in many cases. There are a number of solutions to tackle this problem. A lower imbalance in workload can be achieved when the workload is assigned to each sub-trees based on the number of nodes in each sub-tree. It means that the larger the number of members of a sub-tree the larger the size of the data assigned to that sub-tree. This solution is not as accurate as the second (following) solution but it is much easier to measure and the balancing factor (number of nodes in sub-trees) is easier to work with. This solution is chosen for the simulated platform proposed in this thesis to keep the system as simple as possible. The second solution can be tested in the future.

The second solution is to take the number of link traverses in each sub-tree into account to assign workload to each sub-tree. The total number of links to traverse a sub-tree can be a clue to estimate the total transfer time of data (and result) packets. There are always a nondeterministic wait time for packets in I/O queues due to link blockages that cannot be presented in a closed form formula but it is definitely related to the number of traversed links. Therefore, it can be said that the traversed links affects the total transfer time but there is not a direct linear relation between these two. The simulator's approach is to assign an index to each sub-tree of the dependency tree so that this index can represent how ready the sub-tree is for accepting workloads. Eq. 11 shows just two of the possible sub-tree indexes but other type of indexes can be used.

$$I_a = Subtree\_Index_a = \begin{cases} \dfrac{Nodes_a}{Link\ Traverses_a} \\ \dfrac{\sum_{i=0}^{n} Branches_i}{n} \end{cases}$$

Eq. 11: Two load balancing sub-tree indexes. n is the number of nodes in a sub-tree starting with node a.

It should be noticed that the number of link traverses is not the same as the number of links in the sub-tree (see Figure 54). The total number of hops packets need to take from the root of a tree (a sub-tree) to all the other nodes can be used as a measure of how costly the communication can be. For simplicity it is assumed that the cost of all the hops (which correspond with the hop time) are the same. This is not always true because in some tasks (including FFT) there are some communication hotspots in which there are higher number of communications which makes it very probable to make the I/O queues of some of the nodes becoming full. This increase the number of transaction delays and consequently increases the average hop time. For tasks like FFT the

communication hot spots may create near the starting nodes of the task because all the data should be transmitted from and returned to those nodes.

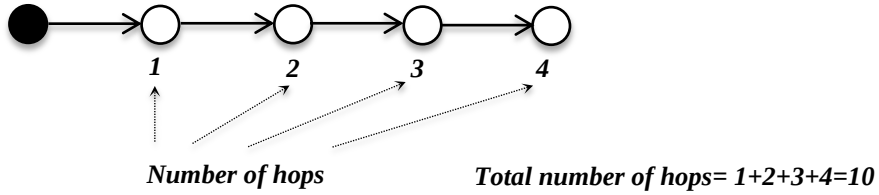


Figure 54: Total number of hops needed to contact all nodes in a tree from the root (black) node can be used as a tree index.

Figure 55 shows how the three aforementioned approaches work. In part (a) the workload is not balanced and instead it is divided into equal pieces between two sub-trees. In part (b) the workload is shared between two sub-trees based on the number of nodes belonging to each sub-tree. In part (c) the workload is sent to two sub-trees based on the index assigned to each sub-tree.

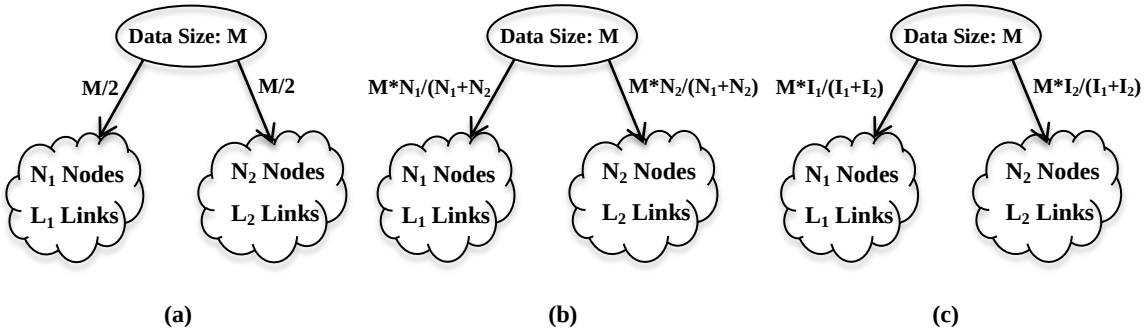


Figure 55: Three approaches for assigning workload to sub-trees. (a) Equal workload; (b) Workload balancing based on the number of nodes; (c) Workload balancing based on a sub-tree index

Figure 56 shows that the number of nodes in two trees can be the same while the number of links passed by packets are different. The depth and the width of a tree affect the number of link traversals from the root node to all other nodes. In part (a) all non-root (white) nodes are directly connected to the root (black) node; therefore, to send a packet from the root to each of the nodes we need just 4 transactions. But in part (b) the number of nodes and even the number of links are the same as case (a) but since the tree is deeper than case (b) then there needs to be 10 transactions to communicate between the root node and other nodes.

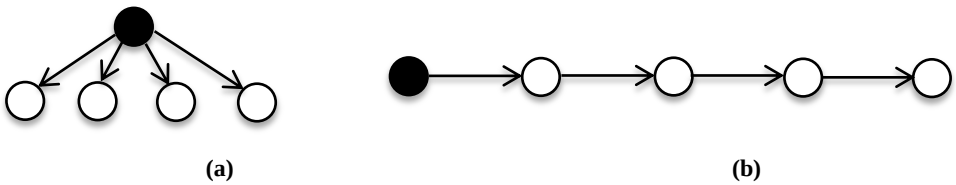


Figure 56: Comparing the number of link traversals when the number of nodes and the number of links are equal in two cases: (a) when the width of the tree is large and its depth is small and (b) when the width of the tree is small and its depth is large

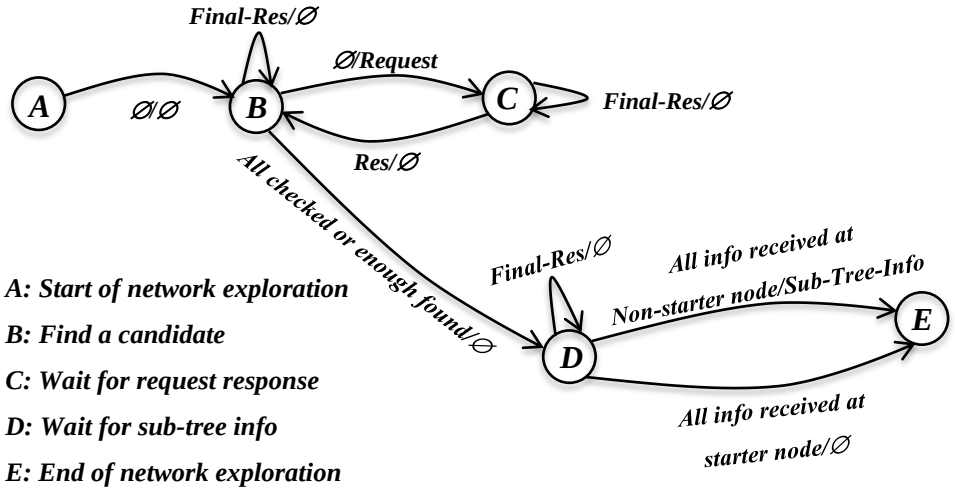


Figure 57: State diagram of the network exploration of modified version of FFTTM

This difference can particularly be interesting because most of the transactions in part (a) can be handled in parallel in the proposed architecture while in the same platform all the transactions in part (b) should be serial. There is a considerably big difference in the overall transaction time between cases (a) and (b). An effective load-balancing strategy can use both the number of link traverses and the number of nodes of a sub-tree.

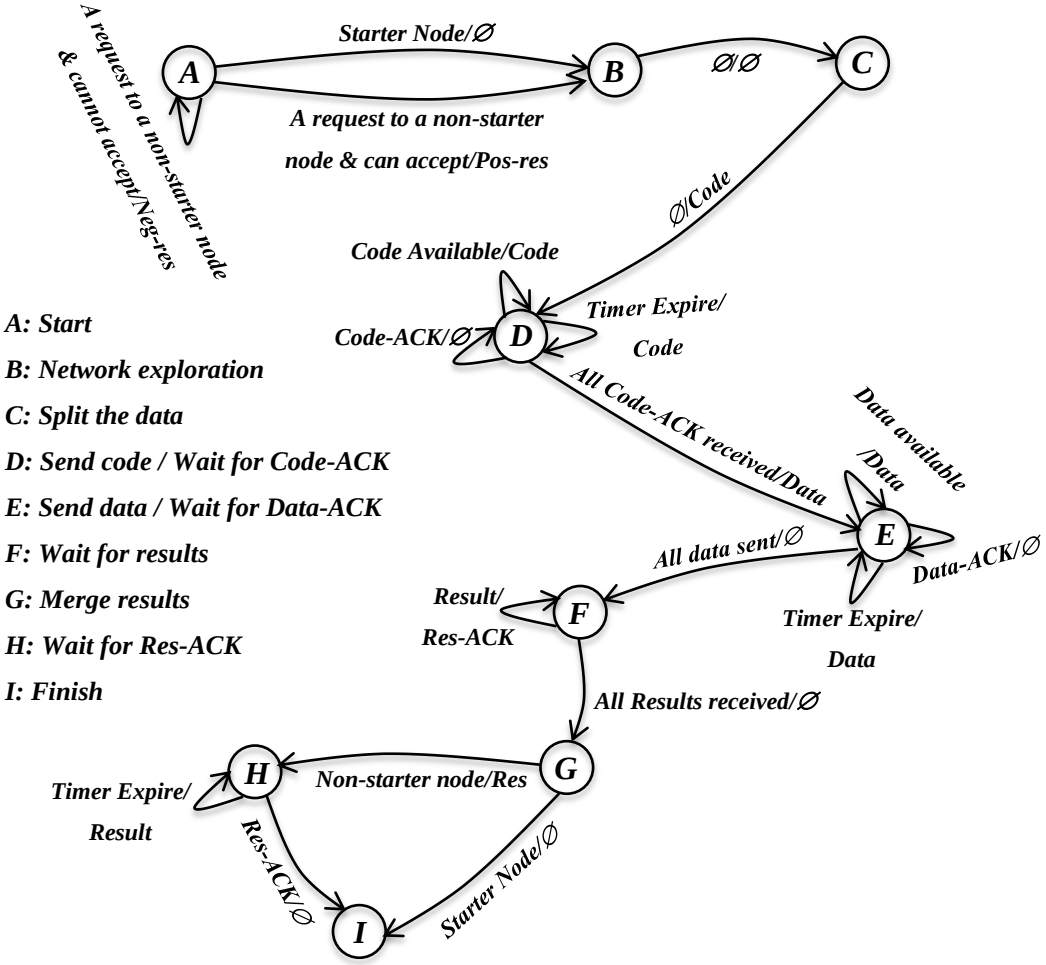


Figure 58: State diagram of the modified version of FFTTM

The state diagram of the network exploration sub-task is shown in Figure 57 and also represented by state "***B***" in Figure 58. The sub-tree information generated during this process is vital in assigning workload to each node. Figure 58 shows how the FFTTM with load balancing support works. This figure can represent both of the workload balancing strategies.

Regardless of the sub-tree index chosen for implementing the load balancing mechanism, the same state diagram applies to FFTTM with load balancing. The only difference is in the implementation of state "***C***" which concerns the calculation of the sub-tree indexes. There is another difference in two sub-tree indexes introduced in this thesis which is the number and size of the packets sent from the starter node to its selected neighbours. If the number of nodes in a sub-tree is used as the sub-tree index then the original data will be divided into larger number of smaller packets each sent to a node. The number of packets is equal to the overall number of nodes in the dependency tree. If the average number of branches or the ratio of the nodes to the hops is used as the sub-tree index then the original data is divided into smaller number of larger packets. The number of packets in these two cases are equal to the number of selected neighbours of the starter node.

Table 17 summarises the communication protocol designed for FFTTM with load balancing. It is in fact a modification of the original FFTTM's communication protocol presented in Table 16.

Packet Type	**Description**	**Parameters passed**
nodeStatusResquest	The parent node asks for accessing over the child node	Task-model identifier, Task identifier
nodeStatusResponse	The child responds to the parent s access request	Index if the free task, Number of busy tasks
nodeSelection	The parent announces that the child is selected	"1" for selection, "0" for rejection
nodeStatusFinalResponse	The child sends its sub-tree data back to the parent	A string representing the sub-tree data
transferTask	The parent sends task code to the child	Task transfer time
transferTaskACK	The child acknowledges the task code	"0"
transferTaskNACK	The child do not acknowledge the task code	"0"
transferData	The parent sends data to the child	Data transfer time
transferDataACK	The child acknowledges the data	"0"
transferDataNACK	The child do not acknowledge the data	"0"
transferResults	The child sends the results back to the parent	Results transfer time
accessDenied	The child responses to a late or unauthorised contact	"0"
transferResultsACK	The parent acknowledges the results	"0"
transferResultsNACK	The parent do not acknowledge the results	"0"

Table 17: List of packet types in FFTTM with load balancing protocol.

6.5) Summary

The level of analysis used in this thesis dictates that there is no need to be involved in the details of tasks running on a simulated BC platform. What do have real impact on the execution time of the whole tasks are the pattern of communications and computations, the timing of the communications and the length of packets (rather than their contents). For these reasons, it is decided to abstract tasks from their technical and implementation details to construct something which is called task-model in this thesis.

A task-model is an artificial traffic generator which mimics the high-level behaviour of a group of tasks in real world. It is shown in this chapter that one important factor in a task-model is the degree of dependencies it creates between nodes in a network. Two task-models are designed and used in this thesis. They are called Simple Parallel Task-Model (SPTM) and FFT Task-Model (FFTTM). This chapter has shown how they are extracted out of real-world tasks. Also, it is shown that SPTM creates a very local net of dependency between direct neighbours only; while FFTTM tends to create an expansive tree of dependency between many nodes (and possibly all the nodes) in the network. A modified version of FFTTM is also introduced in this chapter to support load balancing. Results presented in chapters 8 and 9 are all based on these three task-models.

Chapter 7: Simulation and Visualisation Tools

In this chapter the simulation and data visualisation tools designed and implemented specifically for this thesis are discussed in detail. Other publically available network simulation tools do not have the suitable abstraction level; therefore, these tools are designed and implemented from scratch.

The tool kit can be used for the next stages of this research as well as other researchers who work on the same level of abstraction from a physical environment. The code will be open and accessible for future changes and extensions.

The main aim of the simulation tool is to measure the performance of the network expressed in terms of speedup factor and processor utility factor as defined by Eq. 12 and Eq. 13. The physical and electrical details of the network are not implemented in this simulation. This does not mean that these details are not important; however, on the specific level of analysis in which the execution time and the high level I/O behaviour of the nodes are measured for this thesis those details do not play a major role and; therefore, it is decided to not include them in the simulation.

The visualisation tools are designed and implemented to give a graphical explanation of the behaviour of the network during the simulation time. The state of nodes, the dependencies between them and the state of their I/O queues are among factors that these tools deal with.

The time unit in these tools is equal to one iteration of the simulator which models a clock cycle of the processing nodes. All other timings including the transmission mechanism are measured based on this quantum of time. In next chapters when the experiment results are presented all the time axes are expressed in simulation iterations (which is equal to simulated clock cycle).

7.1) Simulation Environment

Making a real 3D wireless platform is left for the next stages of this research; therefore, the idea of ball computing is evaluated in the current stage using a simulation tool. The simulator is running on a level of abstraction so that it can avoid unnecessary hardware design details. This thesis analyses the efficiency of the proposed system from the execution time point of view. This does not mean that the hardware design details are trivial; but they need a separate analysis that is out of the scope of this thesis.

The simulator should be able to simulate the I/O operation and to mimic the signal propagation for all radio devices on all nodes. The signal propagation in the real world

is affected by a number of factors including free space loss, refraction, diffraction, reflection, aperture-medium coupling loss, and absorption. The way these factors contribute in path loss is too low-level to be modelled in detail for this thesis. With a certain level of approximation signal propagation can be related to the inverse square of the communication distance[30]. It should be mentioned that in a very precise analysis of signal propagation even the geometry of environment, the propagation medium even the minute change in position of the receiver and transmitter antenna can occasionally play a role in signal propagation pattern.

7.1.1) Choice of Simulation tool

There are a number of well-accepted network simulators already available to researchers including *Network Simulator* (*ns-1, ns-2* and *ns-3*), *OPNET* and *NetSim*. The degree of details involved in such popular simulators is too much for the analysis presented in our research. To run our models on such simulators many details should be specified which are out of the concept of this thesis. The level of abstraction this thesis is dealing with plays a key role in choosing the simulation. To the best knowledge of the author of this thesis none of the well-known network simulation softwares operates exactly at the desired level of abstraction. Many network simulation tools model the physics of signal propagation while it is not the main issue in this thesis. As a result, such a level of precision does not add a great deal of value to the type of analysis made in this thesis and at the same time the computational overhead imposed by such a level of modelling consumes lots of computational power.

Because of the abstraction of our ideal simulator from electrical details of signal propagation mechanism and many other physical layer issues, it is decided to design and develop a new network simulator for this thesis.

Another reason for that decision was the size of the network this thesis was going to deal with. Our goal is to use as many wireless nodes in the simulator as possible. The number of nodes could be easily reach tens even hundreds of thousands. This cast a serious doubt over using current simulation tools in terms of their scalability. There are some data structures and pieces of code attached to each node as a software object. This makes it vital to rip off any unnecessary bits of data from the node objects to make room for more simulated nodes especially when the simulation is running on PCs with limited memory resources. In order to be as general purpose as possible, simulators

[30] As stated in many engineering sources including H.P. Westman [294]: in free space, all electromagnetic waves obey the inverse-square law which states that the power density of an electromagnetic wave is proportional to the inverse of the square of the distance from a point source.

developed by others deal with many details, many of which are trivial and unnecessary in this thesis. This means that they occupy potentially large amount of memory with no particular contribution in the analysis made in this thesis.

7.1.2) Discrete Time Simulator vs. Discrete Event Simulator[31]

For the first version of this simulation tool a discrete time structure (DTS) is chosen. In this structure a quantum of time is introduced for the whole simulated system. The objects in the simulated environment can exist, operate and communicate on this discrete time scale. The state of the system is updated on each step of time. To resolve the problem with simultaneous changes in a state of an object by more than one entity at the same time step some of the attributes of simulated objects should be buffered to let the simulator resolve any ambiguity about the new values for those attributes at the end of each time cycle. The attributes that deal with the internal operation of an object are not prone to be changed by other objects. On the other hand there are some attributes that can be manipulated by other objects. These are the attributes that need to be buffered in order to avoid any inaccurate operation of the simulator.

The shortest meaningful period of time (quantum of time) in this simulator is the time in which a floating-point operation is executed. The real clock cycle of the processors are not directly involved in the design of simulator. Different CPUs in real world can perform a floating-point operation in different number of clock cycles. In many new CPUs more than one floating-point operation can be executed in a single clock cycle. This simulator is not going to deal with the internal structure of processors used in each node of the network and for this reason from an external point of view what is important in a CPU in a parallel scenario is its number of floating-point operations it can execute per second rather than its real clock cycle which is something related to its internal operation. Since this simulator simulates the network in a rather high level of abstraction, the simulator can avoid dealing with the real clock cycles.

The fact that the quantum of time in this simulator is the floating-point operation time does not mean that nothing can happen in a time period shorter than that. Depending on the data rate of the links it is possible to transmit more than one bit of data in a quantum of time. This can cause a little bit of error in measuring transfer times as some I/O operations may finish in the middle of a quantum of time. Although this may add a bit to the system error but this error is neither scalable nor accumulative. Even if a bit

[31] More detailed information about these two types of simulations can be found in [295], [296], [297], [298] and [299]. Shorter introductions can be found at [300] and [301].

transfer time was taken as the quantum of time the problem was not guaranteed to be solved.

A more familiar structure for a simulator is the discrete event structure (DES) in which it is the events that are the source of progress in time rather than a global time cycle. In other words the simulator can progress in time with time steps of variable size. If the simulator has nothing to do for a period of time there is no point to pass this time in several time cycles; instead it can jump right to the next important point in time scale (which is the next event in the simulated network). In the second version of the simulation tool the simulator is modified to look a bit like a DES structure. However, it cannot be categorised as a real DES because in a DES in each step only the events fired at that time are handled. The reason why more familiar DES structure is not fully adopted in this thesis is that for a major part of the simulation time there are many events fired on each time cycle. This means that for a major period of the simulator's run time the simulator jumps only one cycle on time scale and the idea of having a list of events sorted by their firing time is redundant. It is true that for some parts of the simulation time (particularly for its final stages) there are fewer simulated network nodes still active and this means that the time hops can now be longer to save execution time but the problem with shortage of memory space has dictated to save the precious system memory by not implementing a rather large data structure for handling events. This comes with the cost of wasting time on looking for events on inactive objects. The time lost in this way can be considerable but in a balance between simulation time and the system memory usage it is decided to modify the simulator in a way that it borrows some features from DES architecture while it keeps some other DTS features.

7.1.2.1) A Discrete Time Simulator

Figure 59 shows how the first version of the simulator works. During each iteration of the simulator (which represents a quantum of simulated time) all nodes have their chance to do:

- Respond to input packets
- Handle internal events e.g. timer expiration of packets waiting for acknowledgment.
- Execute a bit of the task-model allocated to the node.

All the above activities can be handled in parallel since it is assumed that the nodes in the network have separate processors for I/O activities and therefore the main processor's activities will not be interrupted by I/O operations.

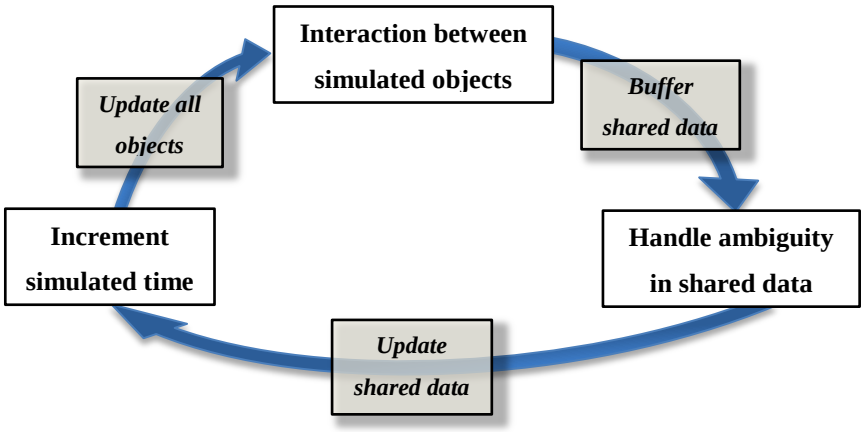


Figure 59: A discrete time architecture for the first version of the simulator

Like all discrete time/event simulations there can be some data structures accessed by more than an object (nodes, in this application) at the same quantum of time. In this case different objects may want to update the value of those accessed data differently. This means that data shared between objects need an arbitration process at the end of each iteration of the simulation to verify what the final value of the shared data is. For this reason those pieces of data should be buffered on temporary locations to let the arbitration process decide on the final value.

Most of the data structures used in this simulator are totally internal to the network nodes and therefore no other nodes can access and manipulate them directly. The only shard data structures are the packets that are in the middle of transmission process (on-air packets) which are shared between the transmitter, the receiver and all the other nodes that are in the transmitter's interference range. This piece of data remains shared until the transmission is finished or terminated. In case of an attempt by a second entity (node) to update the data structure representing an on-air packet the result would be:

1. If the second node has accessed the communication channel in the middle of the first node's transmission process, this means that a packet collision has occurred and as a result none of the packets can be received and decoded at the corresponding receivers correctly. In this case all the nodes in the interference range of both transmitter nodes should be informed about the packet collision, the transmission channel should return to idle mode, both transmitters should stop transmitting and both should back-off for a random time and try later. Resolving this situation does not need to be postponed to the end of simulations iteration and therefore no buffering mechanism is needed.
2. If the second node has started its transmission at the same time as the first node, this means that there is a race between two nodes to access a shared

communication channel. One way to resolve this situation is to rate all nodes with a priority index. This index shows that which node wins the competition to access a channel in case of a simultaneous attempt.

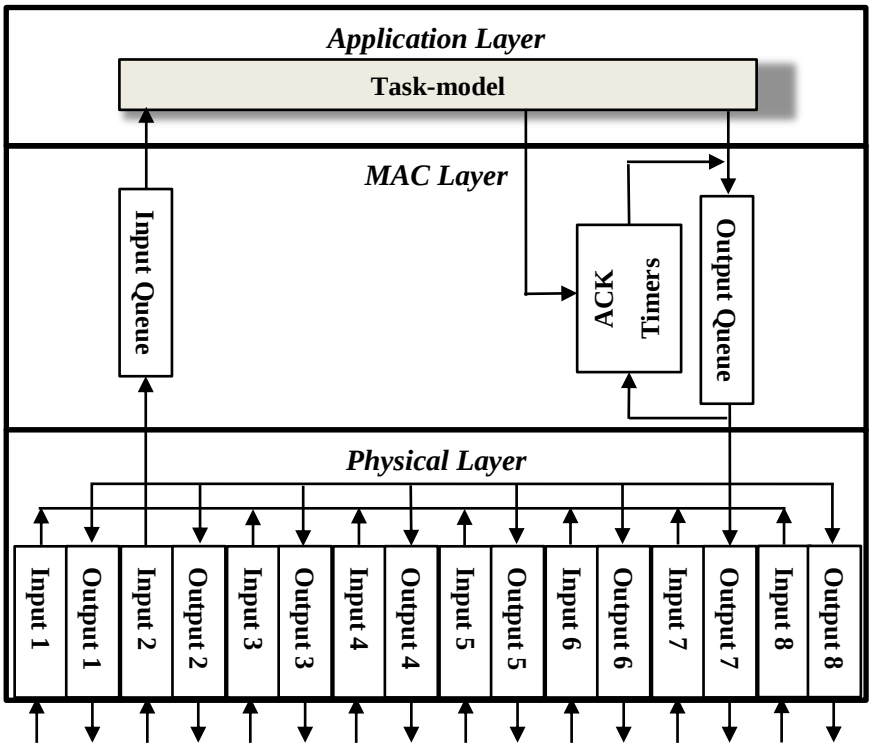


Figure 60: The block diagram of the internal structure of a simulated node

Base on the above material there is no need to have a buffer for shared data in this simulation because in case of a multiple access to a shared data the priority of nodes decide what node can succeed and the other node backs off and tries later. The winner node does not even detect the simultaneous attempt of the other node and no additional time is imposed to its transaction process. In none of the above cases no temporary buffer is needed.

The internal block diagram of the simulated nodes can be seen in Figure 60. The physical layer consists of eight wireless tranceivers which can operate independently. Also, the main processor is just dealing with the task-model in the application layer and does not intervene with lower layers' operation. The ACK packets are task-specific packets and stopping the ACK timers should be authorised from the application layer.

If the input queue is full with packets waiting for being processed by the task-model there will be a risk of loosing new in-comming packets. At this version of the simulator, nodes never stop other nodes transmitting their packets and if the receiver's input packet is full the new packet would be rejected without the transmitter node being informed. This shows the importance of having a mechanism of having the ability to stop other

nodes sending packts when a node's is not ready to receive them; e.g. by updating other nodes about the situation of a node's I/O buffers.

7.1.2.2) Ideas from Discrete Event Architecture

The incremental nature of time in a DTS structure can make the simulator run slow especially at the start and end of the simulation time. The simulation time can be shortened if the simulator can jump to a point in time in which the next event is going to take place. There is no use in incrimentally increasing the current time when no activity is anticipated.

Figure 61 shows how a DES architecture works. Compared to DTS, a data structure should be added to the simulator to store the data about the events scheduled for future. This structure should keep the track of the object the event is related to and the firing time of the event among other information about the event. The idea of buffering shared data should also be implemented in a DES architecture as the concept of time has a descrete nature in this architecture.

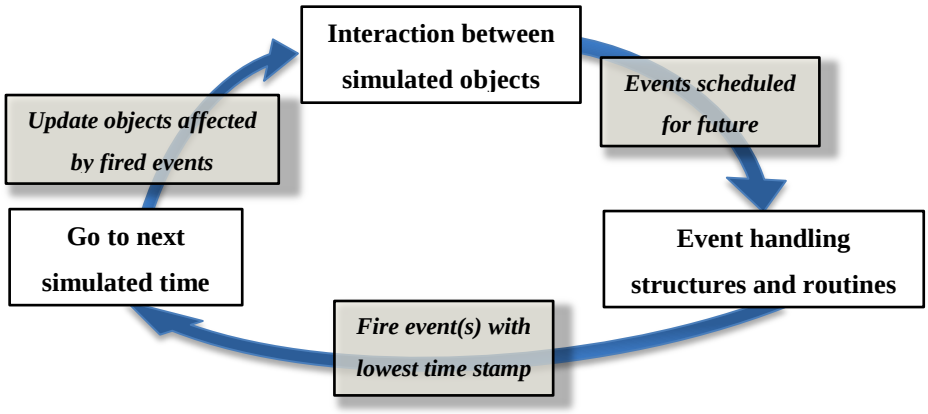


Figure 61: An equivalent of the simulator with discrete event architecture

The idea of jumping to next important point in time rather than simply take equal quantum steps looks very promising and can improve the execution time of the simulation tool. On the other hand, the necessity of storing events for future firings urges software designer and developers to dedicate an unknown and potentially large size of memory space for a new event-storing data structure.

7.1.2.3) The Second architecture for a Discrete Simulator

The second architecture proposed for the simulation tool has in fact borrowed the idea of taking variable time steps from DES architecture while like a DTS architecture does not save any data about the events that are going to be fired in the future. To take time steps of variable sizes it is needed to save the lowest firing time of all events at any time point of the simulation.

Unlike a pure DES architecture, the simulator saves neither the events nor all the firing times. This is because it can occupy a potentially large memory space. Also for a large part of the simulation time the simulator is under heavy work load (especially when the size of the simulated network is large) and this means that it has something to do at each quantum of time and the time step rarely is bigger than one. For this reason, the large data structure for storing future events has practically no use.

It should be mentioned that at the early and final stages of the simulation the number of active nodes are low and therefore the number of events scheduled for future can drop dramatically. This means that the nodes that need to be checked for updates caused by this low number of events is also low in number. In this case, running methods from all the nodes to seek for any possible new activity wastes some time. This is the real sacrifice we make to keep the memory usage of the simulator as low as possible.

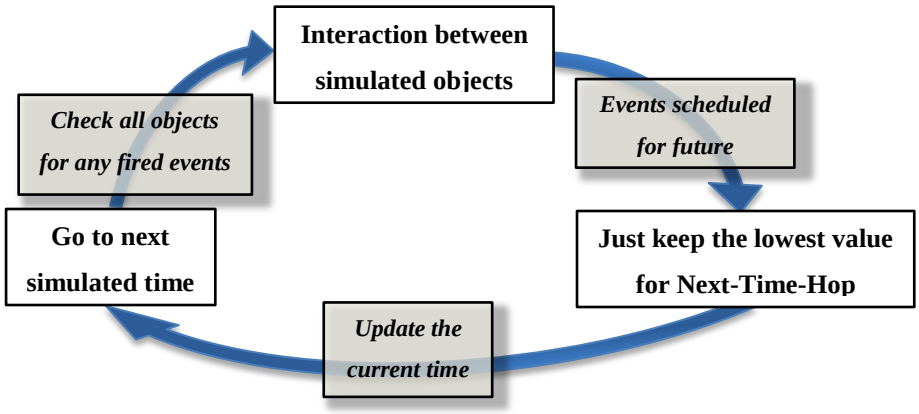


Figure 62: The second version of the simulator which is a combination of discrete event and discrete time architectures

Figure 62 shows this new architecture which is in some ways half way between DTS and DES. As this figure shows the only thing that is saved from an event is its firing time. Even the name or the object bound to it is not saved. The author of this thesis believes that this is a good -yet not necessarily the best- balance between the execution time and memory usage of the simulator.

7.1.2.4) Multi-Tasking Network Nodes

As early experiments show, the performance of the proposed parallel platform is affected by a drop in the number of active processing elements caused by an increase in the number of nodes waiting for some results from other nodes. It can be seen particularly in tasks like FFT in which a bigger workload is split into pieces and sent to other processors to be processed in parallel. In this case the original processor does nothing but waiting for results and is not an active node consequently. The idea of

multi-tasking comes into reality to use such a processing potential left unused in waiting processors.

In a processing node enhanced by multi-tasking a new middleware should be introduced to facilitate the activity of the processor, I/O devices and several tasks running on the node. This software entity is called the *task-manager* in this thesis. A task manager also decides which task has access to node's resources and can be executed. The task execution time and the order of the tasks are among things that this new object is in charge of. All these extra activities are added to the traditional computational duties of a processor. Therefore, the luxury of multi-tasking comes with the cost of time and processing overhead imposed by multi-tasking duties. The task (context) switching process is also time consuming. The switching time can be reduced when multithreading; but the tasks in this simulated network are sought to be single threaded for the sake of simplicity. Storing the old task's data and restoring the new task's data and re-activating the task take a bit of time which may translate into a decrease in the performance. In a balance of costs and benefits of a multi-tasking system it is decided to include this feature to the proposed system. The results presented in the next part of this thesis prove that this was a right decision and an apparent improvement in the performance can be seen.

In a multi-tasking model the tasks are no longer in direct contact with the MAC layer objects, e.g. I/O queues. The newly introduced task manager object is the mediator between the application layer objects and those of lower layers. Each task, however, may have their own I/O queues to react to ripples in response speed of packet processing. The task manager controls the main I/O queues and decides which packet should be redirected to what task.

A block diagram of a multi-tasking node can be found in Figure 63. The new task manager object and the array of tasks are among new objects in this architecture. Each task has its own I/O queues which are separate from the main I/O queues implemented in MAC layer. What the task scheduler does is not just giving equal processor time to each task. It can also prioritise tasks and assign processor time to them based on that priority. Those tasks that are just waiting for an external event (e.g. receive of code, data, results, ACK etc.) can yield the control to other active tasks to improve the performance. This is also part of the task scheduler's duties. The task switching period is modelled in this simulator by a pre-specified period of waiting in which no tasks are active (remember that the tasks are just models of tasks and therefore there is no real task switching). In a real task switching the old task's data and status should be saved

on memory for future and the new task's status and data should be loaded to memory and the control should be transferred to that task. None of these are really done in task switching in this simulation tool.

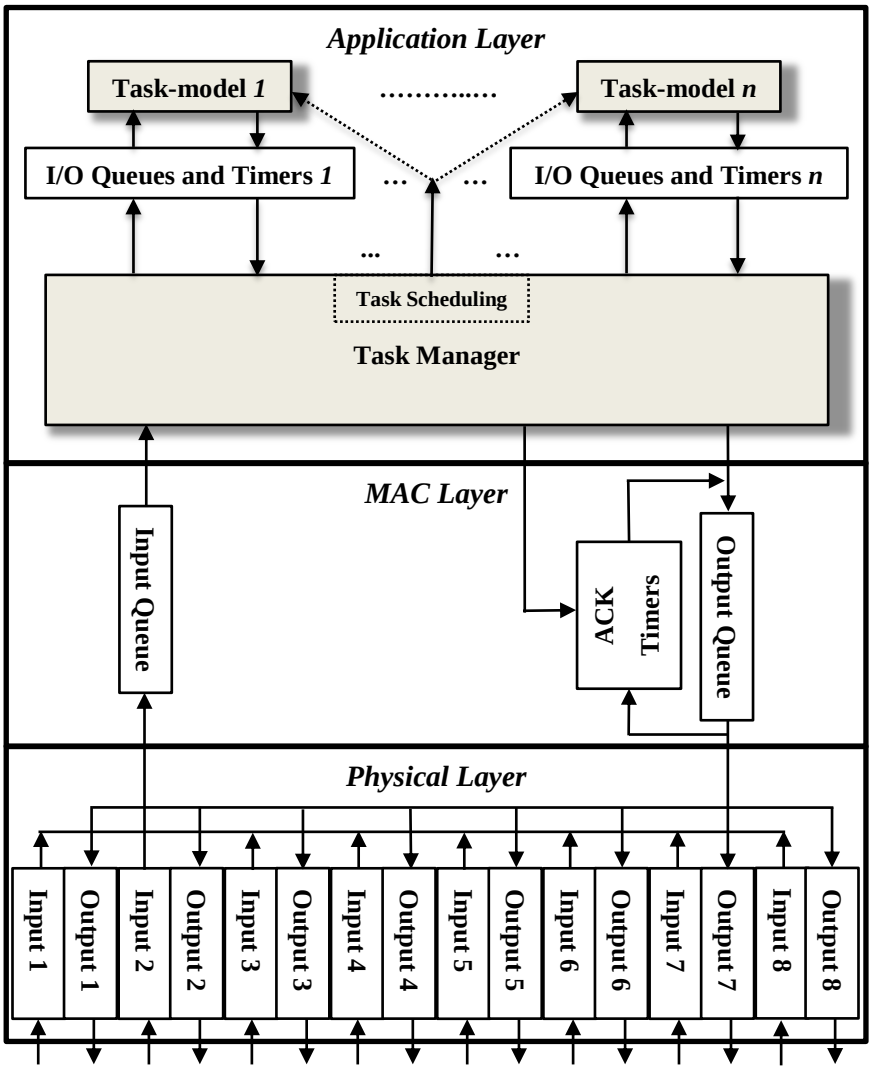


Figure 63: The block diagram of the internal structure of a simulated node with multi-tasking

The question about the maximum number of tasks per node is experimentally answered and the results are presented in the results part of this thesis. As it is shown in the results part, the main effect of multi-tasking is to decrease the number of waiting nodes by recruiting them for other tasks which consequently increases the performance of the network.

Algorithm 3 shows more details about how the task switching mechanism works. In addition to pass control to tasks that are already finished, it tries to maximise time efficiency as much as it can by not passing control to waiting tasks, tasks with no input and output packets to process and tasks with application layer input/output queues full of packets. The algorithm particularly does not pass control to a task that has some packet in its input queue but cannot process it because its output packet is full.

```
1: Inputs:
2: tasks: An array representing the type of tasks for each node.
3: activeTask: An integer showing the current active task.
4: numberOfActiveTasks: An integer showing the number of active tasks.
5: lastTimeHop: An integer representing the last time hop
6: forceSchedule: A Boolean showing if the task scheduling is called unexpectedly

7: Global Variables:
8: taskRemainingTime: An integer showing the remaining time of a task in a time slot
9: duringTaskSwitchingOverhead: A Boolean showing that two tasks are switching
10: taskTimeSlot: An integer defining the time dedicated to each task
11: taskSwitchTime: An integer defining the time needed for switching between tasks
12: maxTasks: An integer showing the number of tasks per each node

13: Output: newTask: An integer representing the new task activated

14: taskRemainingTime -= lastTimeHop
15: newTask= activeTask
16: Boolean nextWaitingTaskFound= false
17: if (forceSchedule || (taskRemainingTime <= 0))
18:         if (!forceSchedule && (numberOfActiveTasks < 2) && (activeTask > 0))
19:                 return (newTask)
20:         int candidateTask= activeTask
21:         Boolean nextTaskFound= false
22:         if (duringTaskSwitchingOverhead)
23:                 taskRemainingTime= taskTimeSlot
24:                 duringTaskSwitchingOverhead= false
25:         else
26:                 for all i in available tasks
27:                         if (nextWaitingTaskFound  && tasks[candidateTask] != null)
28:                                 W-C-T= candidateTask
29:                                 nextWaitingTaskFound= true
30:                         if (outACKReady(W-C-T) || inACKReady(W-C-T) || (inReady(W-C-T)
                            && (NumberOfOutReady(W-C-T) >= NumberOfOutNeeded(W-C-T))))
31:                                 nextTaskFound= true
32:                                 break
33:                         if (isReady(W-C-T) && (NumberOfOutReady(W-C-T) >=
                            NumberOfOutNeeded(W-C-T))))
34:                                 nextTaskFound= true
35:                                 break
36:                         candidateTask= (candidateTask +1) % maxTasks
37:                 if (!nextTaskFound)
38:                         if (nextWaitingTaskFound)
39:                                 activeTask= W-C-T+1
40:                                 taskRemainingTime= taskSwitchTime
41:                                 duringTaskSwitchingOverhead= true
42:                 else
43:                         activeTask= candidateTask+1
44:                         taskRemainingTime= taskSwitchTime
45:                         duringTaskSwitchingOverhead= true
46: return (newTask)
```

Algorithm 4: Task scheduling algorithm for a multi-tasking node.

The same task switching algorithm can be used for any type of task-models. The algorithm does not care about the task type and does not use task-specific details during its decision time. For this reason the proposed task switching algorithm is task independent.

7.1.2.5) Priority Output Queues

Some experiments have shown that one of the problems task-models like FFT have is the existence of communication hot spots. Unlike an application like weather forecasting in which a node is in touch with just a restricted number of neighbouring nodes, in a task like FFT there is a rather big chunk of data on a node which is going to be distributed over a subset of the network for parallel execution of the task. This means that the closer a node is to the source of the data, the higher the chance to have a heavy load of incoming and outgoing packets. This is what is described in this thesis as a communication hot spot. Part of the traffic load belongs to the acknowledgements sent to signal the safe receipt of a data packet to the original transmitter. These packets are particularly important not only because they occupy locations in the input queue but also because unprocessed ACK packets mean that there are some data packets in the output queue that are sent but still are in the queue waiting for the corresponding ACK packets. This may lead to a deadlock situation in which both input and output queues are full on both sides of a communication link and there is no actual progress potentially for ever. Deadlock is already reviewed in sections 2.8 of this thesis. Also section 3.8 briefly introduces a deadlock avoidance algorithm designed for this thesis. Later in this chapter (section 7.1.7) it is shown how the deadlock avoidance algorithm introduced in section 3.8 is implemented in this thesis.

This shows that it is better to give the ACK packets a higher priority over data packets. The experiments show a significant improvement in the performance of the system when the output queues are replaced by priority queues. One way to implement a two-level priority queue is to have two separate and parallel queues one for data packets and the other for ACK queues.

7.1.3) Network Partitioning and Channel Assignment

The algorithms introduced in chapter 5 for network partitioning are implemented in the simulation environment. Since the network is static (i.e. the position of the nodes do not change) there is no need to have a dynamic network partitioning algorithm. It is not even necessary to include the algorithm in the main simulation software. When the size of the network, its topology, the size of the nodes, their radio ranges, and the signal interference scheme are unchanged there is no need to run the network partitioning algorithm every time the simulation runs since it definitely produces the same results all the time. Therefore, the network partitioning algorithm can be both static and off-line.

A separate Java software is implemented based on the algorithms proposed in chapter 5. The results are saved in a number of text files. These files are read at the beginning of

the main simulation and the whole network is built based on their contents. The files can be reused in following runs to save time. A new execution of the algorithm is needed only if at least one of the aforementioned parameters is changed.

The structure of the data saved by the network partitioning for the main simulation can be found in appendix B at the end of this thesis. Appendix B also contains a detailed description of the structure of different packets used in different parts of the simulation.

7.1.4) Independent Channel Objects

To handle the signal propagation and radio interference in a more efficient way an independent channel object is implemented. In addition to simulating the behaviour of the radio channel, the radio devices of all nodes (which were previously implemented as part of the node data structure) are now moved to this new channel object. This makes it much easier to handle signal propagation and interference in all different circumstances.

It is the geometrical coordination of nodes as well as the strength of the signal which determine if two or more signals on a channel interfere with each other. For this reason, the channel object has a record of the coordination of all the nodes working on that particular frequency.

In the current version of the simulation a packet is labelled as corrupt (as a result of a packet collision) if and only if it is in the receiver list of a packet and at the same time in the interference list of another packet. In this case the receiver node cannot correctly decode the first packet. If that node is the intended receiver of the packet the packet transfer is failed. For the sender of the packet to be able to detect the packet failure, the sender should be in the interference list of the second packet as well. In this case the first sender tries again after a randomly selected wait time; otherwise, the sender of the first packet assumes that the packet is sent correctly. This is in fact the hidden node problem discussed earlier in this thesis.

The channel object will detect any instances of such problem. As it is shown in the results chapters, the network-partitioning algorithm introduced in this thesis has successfully solved the hidden node problem by eliminating the packet collision. As a result of including the channel object the structure of the node object is also changed. The node object shown by Figure 63 is changed and now looks like Figure 64

The components of the physical layer (previously implemented in node object) are now migrated to the newly created channel object. An array of channel objects are created each of which is assigned to one of the frequencies. Each input/output pair in the physical layer is a member of its own channel object; therefore, the physical layer does

not exist anymore. The channel object is shown in Figure 65 which exchanges data with node objects.

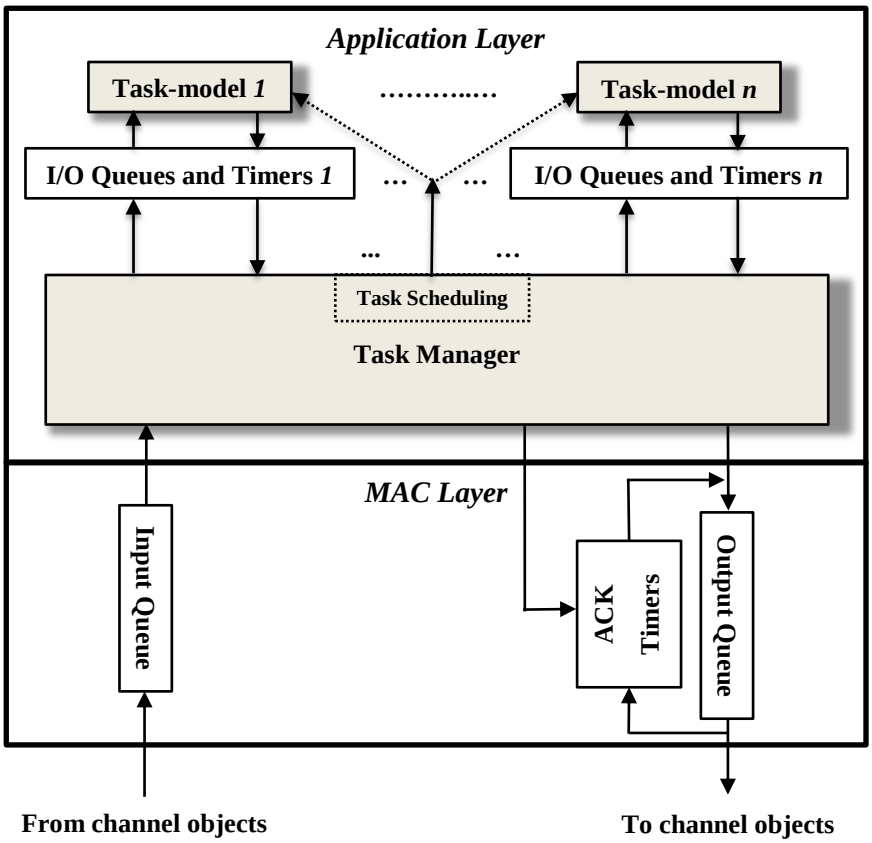


Figure 64: The structure of the node object after introduction of channel object

The dotted lines show a possible lead for extending the current version of the signal propagation mechanism in which other channels may interfere with the signals on the current channel because of the cross-channel interference effect. This effect is not implemented in the current version of the simulator yet. The internal block diagram of a member node of a channel object can be seen in Figure 66.

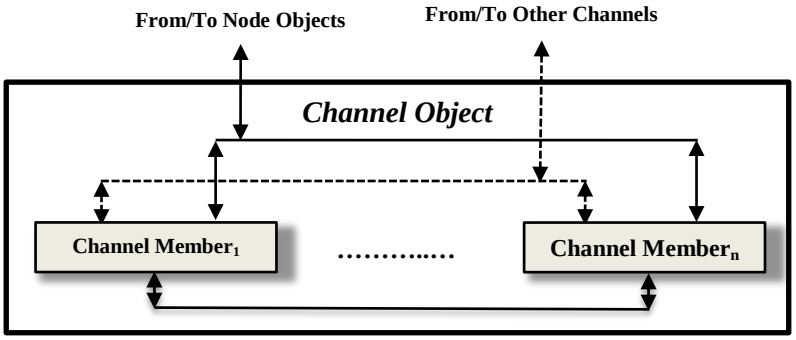


Figure 65: The structure of the channel object which includes part of physical layer of all its member nodes

Like Figure 65, the dotted line represents the cross-channel interference effect that is not implemented in the current version of the simulator yet. Most parts of the channel member objects are in fact the physical layer of the member nodes which have been

migrated to this object for the purpose of simplicity of implementation and more centralised and better control over the signal propagation scheme.

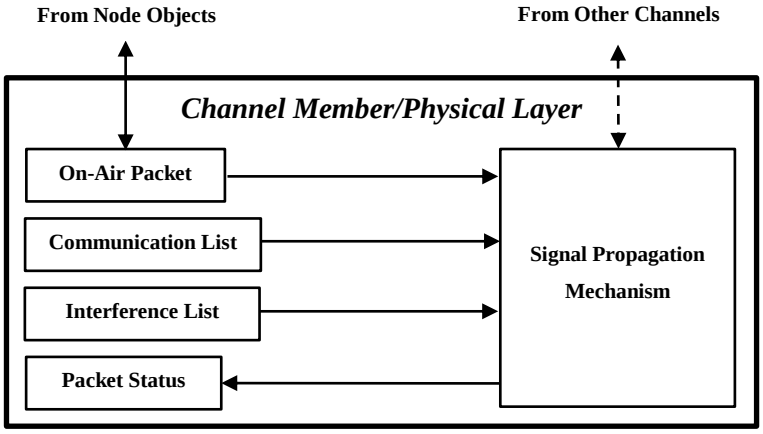


Figure 66: The internal block diagram of a channel member node

7.1.5) Decentralised Simulation platform

The current version of the simulator consists of a single large Java program that handles all the simulation duties. Nodes, tasks, and the network are all modelled and simulated in this program. There are chances that separating these parts from each other and arranging them in independent programs reduce the complexity of the simulator and make it easier to test different nodes and/or tasks while leaving the rest of the simulation unaffected.

Another potential benefit of a decentralised simulation is the possibility of running the components of the simulator in parallel and reduce the demand for resources for each of the components. Therefore, the simulator can be run easier on a grid platform. In such a distributed architecture the components of the simulator use message-passing techniques to interact between each other.

A decentralised architecture for the simulator is designed and developed to test if it can yield better performances compared to the older centralised architecture. Node and channel objects are separated from the main network simulator. In the new architecture these two objects have evolved to two Java programs. A communication protocol is implemented to facilitate the message-passing mechanism between the three major programs. This protocol is independent from the communication protocols each task uses. The first protocol is implemented in MAC layer while the latter is implemented in application layer. Each of these two are transparent from the other's point of view.

Figure 67 shows how the proposed distributed and decentralised simulator looks like. It is hard to manage large number of small independent node objects. Therefore, for simplicity of implementation a number of nodes are aggregated to an object called cluster.

Detailed information about the distributed version of the simulator including the state diagrams of the three main objects in this architecture and the communication protocol and packet types can be found at Appendix B of this manuscript.

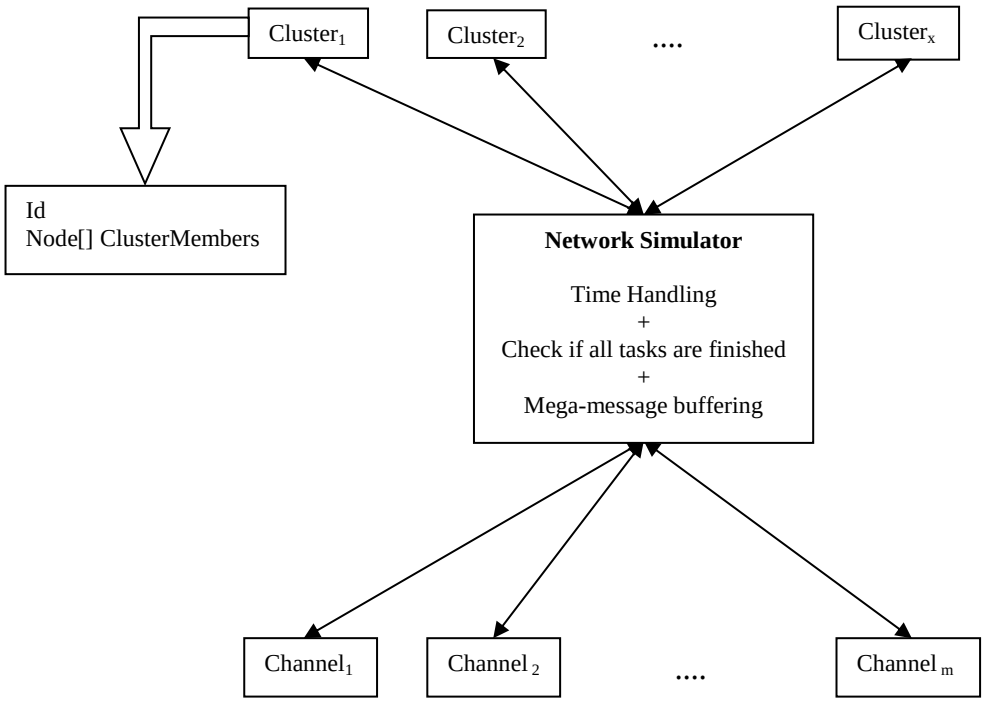


Figure 67: Block diagram of proposed distributed simulator

The distributed platform introduced in this section of the thesis is implemented but it did not yield expected performance. The delay imposed by the message-passing mechanism is the main reason for the poor performance. Also, a distributed platform needs additional software data structures that increases the complexity of the code and therefore makes it harder to maintain. For these reasons we decided to switch back to the original centralised architecture in which all the objects are implemented in a single large Java program.

7.1.6) Multi-part packet delivery

As part of the network optimisation we needed to answer this question: Does transmitting large data packets cause long waits for other nodes using the same channel? If the amount of wait imposed to the other nodes is significant then one may put a limit on the packet size. In case of sending larger bulk of data, the data should be sent in a number of packets. The receiver should merge those packets to reconstruct the original bulk of data on the receiver side. This needs a new split/merge mechanism in MAC layer. The whole split/merge mechanism should be transparent to application layer. Also the physical layer is too low-level to deal with such an issue.

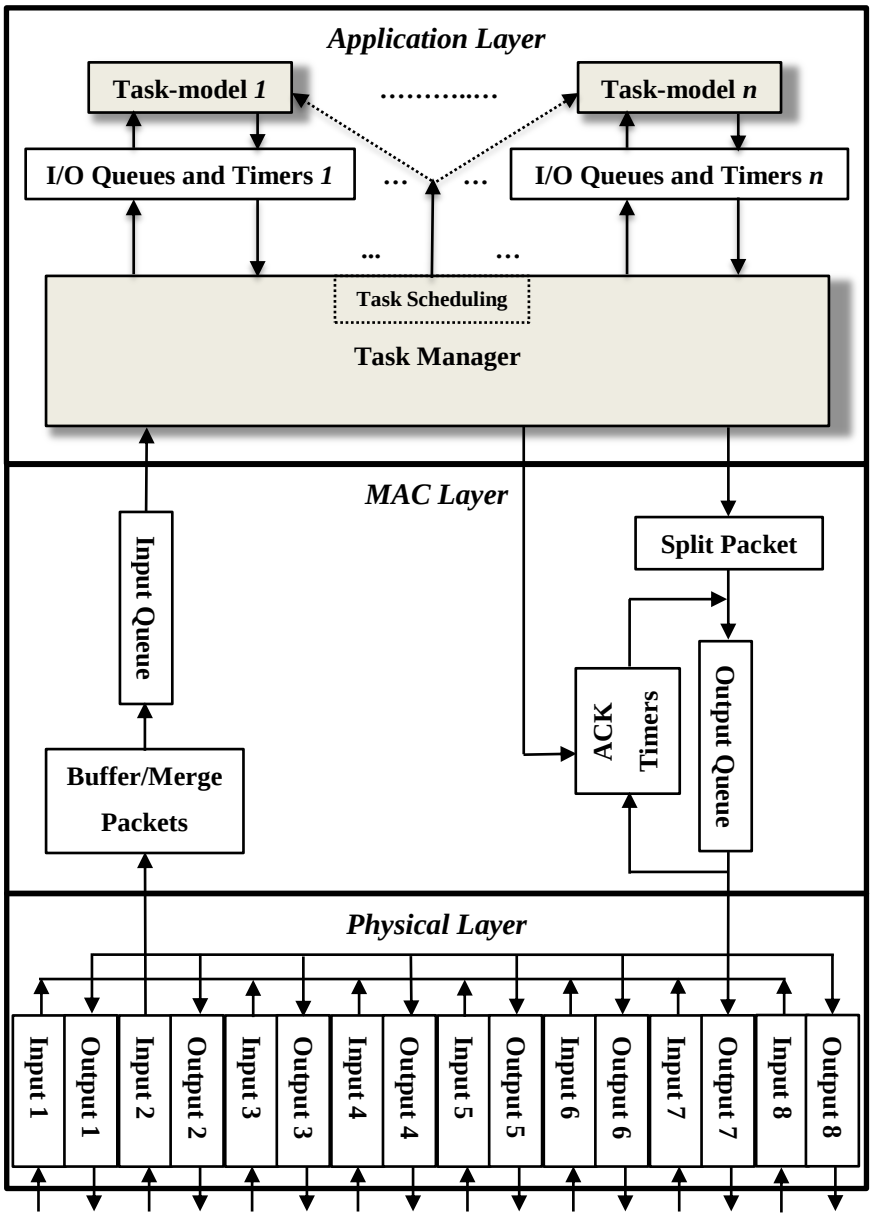


Figure 68: Block diagram of a network node with split/merge mechanism

Figure 68 shows the new node architecture with split/merge mechanism. The "Split Packet" block splits large output packets, generates a number of smaller packets and puts them into the output queue as independent packets. An input packet in the MAC layer first is checked in the "Merge Packets" block to determine if they are part of a bigger bulk of data. Small independent packets pass through this block towards MAC layer input queue with no changes.

If a packet is part of a larger data the "Merge Packet" block buffers them and waits for other pieces of the original bulk of data. When all the pieces of data are received the "Merge Packets" block merges them and sends the big packet to the MAC layer input Packet. To distinguish between different parts of a big data a new field is added to the header of packets. This field shows what part of data the packet is.

The multi-part packet delivery mechanism is implemented in the simulator but the experimental results show there is no significant improvement in the performance of the network after introducing this mechanism. Regarding the time and software overhead imposed by this mechanism and also because of very little (if any) improvement in performance, it is decided not to include this mechanism in the final version of the simulator. The results shown in next chapters are derived from a simulator without multi-part packet support.

7.1.7) Buffer Management Strategy and Deadlock Avoidance

To choose a buffer management and routing method for the BC platform the top priority was simplicity of implementation. As the first stage of research in using wireless links in HPC, we were only exploring some options for routing algorithm rather than looking for the most efficient and robust method. Another reason for choosing this method over others is that SPTM which is one of the task-models introduced in this thesis has no multi-hop transactions at all (except an exception which will be discussed shortly); therefore, it does not any routing mechanism altogether. The other task-model (FFTTM) treats multi-hop transactions as several single-hop transactions and again does not need a complex routing algorithm. This means that the staring node in an FFTTM task-model just knows what its direct children in the dependency tree are. The starting node only knows about the number of packets sent to each of its children but it does not specify which packet should be received by which leaf in the dependency tree. It is the duty of its children to decide to what node the packets should be sent.

The only time the routing method will be used is when all direct channels between two nodes are busy and as a result the source node decides to send the packet indirectly via a mutual neighbour of both the source and the destination of the packet. The routing algorithm is tuned so that this will happen only once in a packet's life time.

As discussed in sections 2.3 and 3.8, the algorithm which is the simplest to implement but not necessarily the most efficient one is store-and-forward. In this method an intermediate node (a node that is neither the original source nor the final destination of a packet) no packet will be relayed to its next destination until it is fully received and then it will be added to the node's output queue for its next hop of its journey. It is expected that using this method of routing substantially increases the amount of queuing time (the time a packet spends in different I/O queues). This is enforced by the simulation experiments presented in chapter 9.

For next stages of this research a wormhole routing algorithm with virtual channels are sought to be incorporated. By doing this it is expected to decrease the overhead imposed by long queuing times the BC platform faces at its present form.

In an interconnect network like BC platform deadlock situation can occur in areas that is called communication hot spots in this thesis. In this manuscript a communication hot spot is a subset of the network with a (persistent or temporary) high rate of packets both inwards and outwards. As an example, the simulation experiments show that in cases that there are multiple workloads running at the same time it is possible to end up in a situation in which a number of nodes are sending many packets to a node (let us call it node A) all of which are going to be relayed to another node (node B). This can be particularly the case with FFTTM with a rigid tree-like structure of dependencies between nodes. In existence of multiple workloads sometimes there are also a large flux of packets from some nodes to node B which need to be relayed to node A. In this case, these two independent streams of packets (one from node A to node B and the other one on the opposite direction) can occupy the whole input and output queues of both nodes A and B.

In this communication hot spot node A wants to read a packet from its input queue to free a slot for new inward packets but it fails because it should be forwarded to node A's output queue (which is also full) to be relayed to node B. This is also exactly the case in node B. In this situation the processes on both nodes are looking for resources (places in each other's input queues) which cannot be freed by the other process.

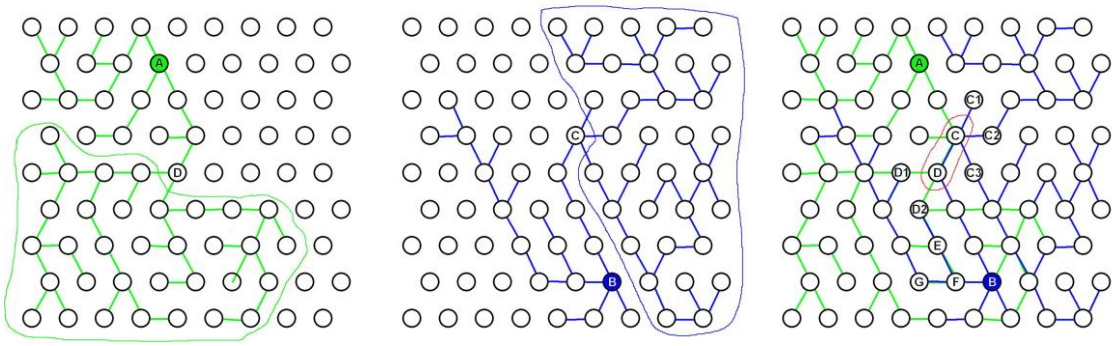
To tackle the deadlock situations a deadlock avoidance algorithm is used in this simulation tool. The core idea of this algorithm is to avoid input and output queues of nodes (especially in communication hot spots) become full at the same time[32].

To start with, each node dynamically indexes its neighbours based on the number of packets in the input queue and output queue from/to its neighbours. This index (known as hot spot index inn this thesis) assigns a number to each of the neighbours which resembles how much a node is in communication with any of its neighbours (both in receiving and sending mode). The higher the hot spot index for a neighbour, the higher the communication dependency between the node and corresponding neighbour and

[32] When one of the input or output queues is full it implies nothing more than a heavy load of packets in corresponding direction. This may put a bit of pressure on communications but it needs only a bit of time to return that queue to normal situations by sending packets stored in output queue or processing/relaying the packets in input queue. This situation is not a deadlock. To have a deadlock both queues on at least two nodes should be full.

consequently a higher chance for having a communication hot spot with substantial risks to end up in a deadlock situation between the node and that particular neighbour.
In case a node detects an increase in a neighbour's hot spot index, it signals other nodes to send less packets to it to let the node sort out the packets to/from that hot spot. To let the situation relax other nodes should send either no packet or few packets to those two nodes in the hot spot.
This is implemented by introducing some network management command packets sent over ACK input/output queues. By receiving such a packet the receiver refuses to send some packets to the hot spot members. This can be either a complete ban on sending those packets or send only a fraction of packets to the hot spot. In case of choosing the second approach, one option is to use a random function to decide whether to send or not to send the packet to the hot spot.
The actual deadlock avoidance method implemented for the simulation tool uses the aforementioned network management commands with a random function to decide if the packet should be sent. As the hot spot index increases that random function tunes itself to increase the restrictions over sending new packets to the hot spot.
This can help decreasing the chance for having a deadlock situation but it can not 100% guarantee that deadlock will never happen. An extra feature is added to the deadlock avoidance method as the last solution based on which if a hot spot exists and nodes inside it are in deadlock situation, they discard all packets in their both queues and signal other nodes about end of communication hot spot. This will add an overhead caused by retransmission of timer expired packets which is not desirable. In practice this incident has not happened in the simulation experiments.
Figure 69 helps explaining how two separate FFTTM workloads (in node A and node B) can contribute in a deadlock situation. Like any FFTTM workload nodes A and B first create their own dependency tree which is shown in Figure 69.(a) and (b) in green and blue respectively. Nodes C and D on those dependency trees play important roles. The green and blue boundaries shown in parts (a) and (b) defines relatively large sub-trees headed by nodes D and C respectively. When those two workloads are running at the same time the link between nodes C and D will be very busy and therefore those two node's input/output queues are at risk of getting full with packets from the other node. The risk is high regarding the size of both sub-trees which implies that many packets need to pass the link between nodes C and D. This means that a hot spot is very probable to create containing nodes C and D as shown with red borders in Figure 69.(c). This situation is far less probable in links between nodes E-F and F-G. Although like

link C-D, links E-F and F-G are used by both workloads but deadlock is not expected because of the small sub-trees attached to nodes E, F and G.



(a) (b) (c)

Figure 69: How two separate FFTTM workloads can contribute in forming a deadlock

When nodes C and D detect a growth of number of packets to/from each other in their input/output queues, they will signal their other neighbours (C signals C1, C2 and C3; D signals D1 and D2) to ask them either to slow down the rate of packets to C and D or to stop sending packets to C and D. This lets nodes C and D empty their I/O queues by transmitting packets left in their queues and avoid deadlock situation consequently.

7.2) Visualisation tools

The visualisation tools are implemented to project the behaviour of the network during the simulation time. Two separate tools are developed, each of which projects different attribute of the network and its performance.

7.2.1) Process Visualisation

The first tool of the visualisation tool set is called process visualizer. The software animates how nodes in the network are involved in a workload. This gives a measure of the utilisation of the network during the simulation time. The performance of the network (as expressed by Eq. 12 and Eq. 13 introduced later in this thesis) is also projected in this tool. The state of nodes and the dependency relations between them are graphically presented. This was initially used to debug the simulator but now after having stable simulation software the tool can be used for its envisaged purpose which is graphical representation of the behaviour of the network.

The data needed for the visualizer is stored by the simulator to a number of log files. The data contains information about the location of the nodes, the state of the nodes, the task dependencies between them and the packets transmitted between them. More details about the contents of the log files can be found in appendix D.

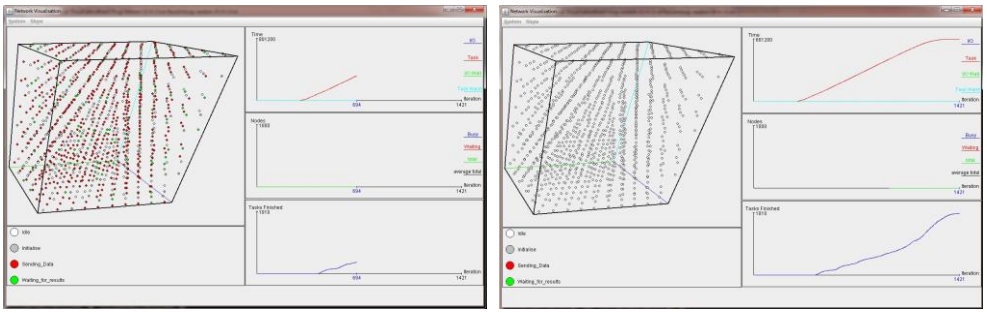


Figure 70: Two snapshots of the process visualizer tool for an SPTM on a 3D network of size 10*10*10 nodes.

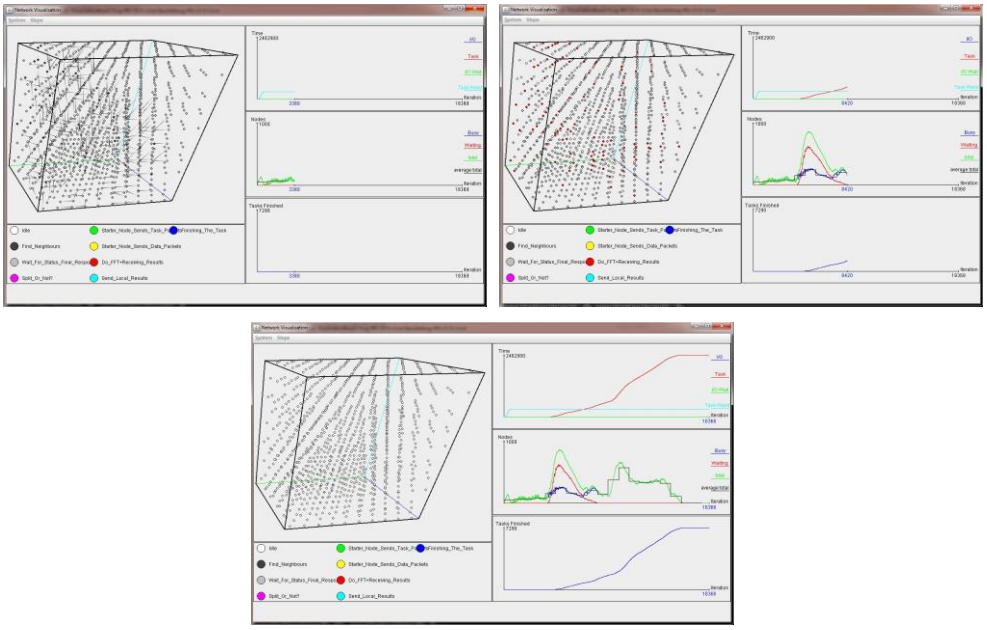


Figure 71: Three snapshot of the process visualizer tool for an FFTTM on a 3D network of size 10*10*10 nodes.

Those pieces of data are used to show how exactly a given task-model behaves in a given network. Bottlenecks in the network can be detected through such log files. The process visualizer also uses this logged data to plot the utilisation of the network during the simulation time. The process visualizer is an animation resembling the performance of the simulator. Therefore, its performance cannot be shown fully in this thesis but a number of snapshots of the process visualizer can be seen in Figure 70 and Figure 71.

Two snapshots of the visualizer can be seen in Figure 70 which shows two points of running an SPTM on a 3D network of 1000 nodes. The figure below shows that on this occasion the tasks on different nodes were executing with an almost constant rate until the very final stage of the simulation. In contrast to SPTM an FFTTM has very dramatic changes during its execution time.

It is mainly because of the high degree of task dependency between nodes and its effect on creating traffic hot spots. Figure 71 shows how the number of nodes involved rises and falls during a typical experiment with FFTTM. This number is affected by the number of tasks per node as well as the number of workloads imposed to the network among other factors.

7.2.2) Results Visualisation

A second data visualizer tool is developed to graphically show the overall value of a set of network parameters. These are particularly of interest when the bottlenecks of the network are under scrutiny. At the current version of the visualizer there are six network measures that are plotted by the visualizer. They are: The time when channels are busy and waiting, the time when tasks are busy and waiting, the overall execution time of nodes and the task finish time of nodes. Figure 72 is a snapshot of the results visualizer tool showing how nodes are busy with their internal tasks as well as I/O transactions. Eight FFTTMs are used as initial workloads each starting on different times. The size of the network is 10*10*10. Multi-tasking nodes support up to eight tasks at the same time.

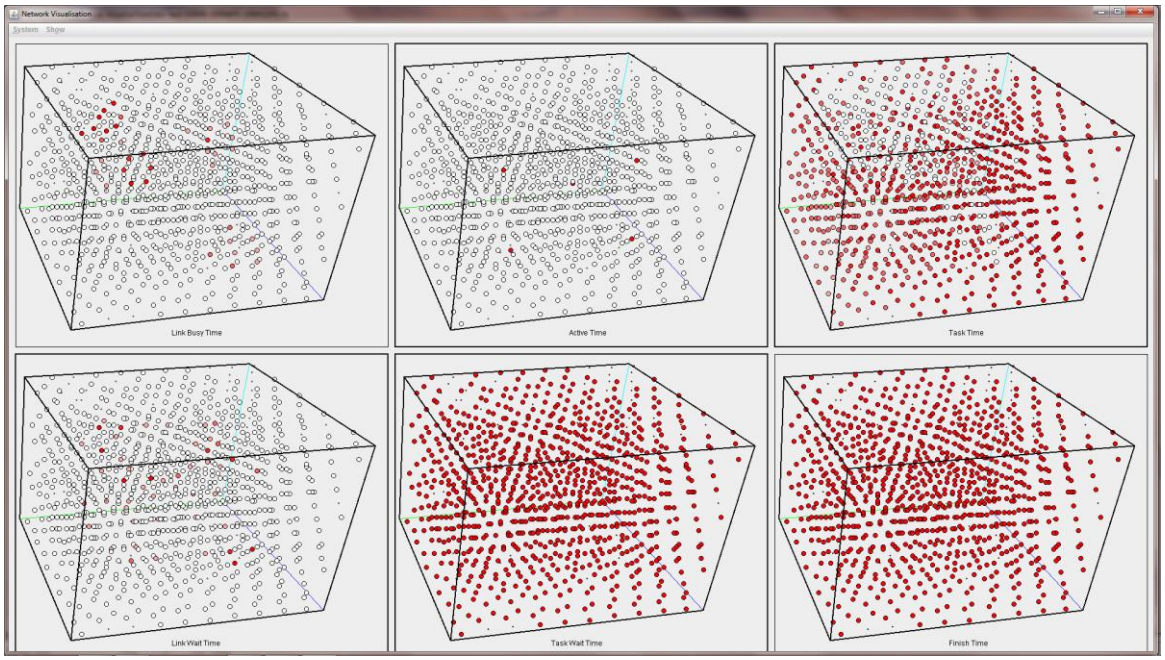


Figure 72: A snapshot of the results visualizer for a 3D 10*10*10 network

Chapter 8: Primary Network Optimisation

8.1) Performance Metrics

This chapter mainly focuses on tuning some network parameters to maximise the performance of two main task-models introduced in this thesis. The results presented in this chapter are produced by the discrete event simulator developed particularly for this thesis. In section 8.1 a simulated 2D network was used. All results reported in other sections (as well as chapter 9) belong to different 3D wireless networks.

The performance of the simulated BC platform is mainly measured based on two factors: the processor utility and the link utility. It was shown in chapter 7 that the iterations of the simulator resemble the clock cycle of processors. The total number of iterations a node is busy with computation, communication or waiting for some internal or external events estimate the computation, communication and wait time of the node in real-world. The overall compute time is calculated as the sum of compute times of all nodes in the network. The overall communication time, the overall wait time and other important times are calculated the same way.

The speedup factor (Eq. 12) in a parallel task is the ratio of the overall compute time to the execution time of that task. In a simulated environment the speedup factor is estimated by the overall compute time divided by the simulation time both expressed in simulation iterations. Speedup factor can be regarded as the average number of nodes that are busy with computation throughout the entire simulation.

The processor utility factor depends on the speedup factor and the number of nodes involved in the task. Eq. 13 formulates the processor utility factor which is expressed in percentages in this thesis. Other definitions for these two factors may be introduced in other sources.

$$\boldsymbol{Speedup} = \frac{\boldsymbol{total\ compute\ time}}{\boldsymbol{execution\ time}} \qquad \textbf{Eq. 12}$$

$$\boldsymbol{Processor\ Utility} = \frac{\boldsymbol{Speedup}}{\boldsymbol{n}} \qquad \textbf{Eq. 13}$$

Where *n* is the number of nodes involved in the simulated task (not necessarily the total number of nodes).

These two metrics show how fast a task is finished and how efficiently a task is using the processors. Besides these metrics other metrics are needed to measure how efficiently the links are used throughout a simulation. Three metrics of this type are introduced in this thesis:

- Overall link busy time (OLBT);
- Average link utility (ALU);

- Overall link wait time (OLWT).

OLBT is defined in this thesis as the aggregate link busy times of all links involved in a task (Eq. 14). The busy time of a link is the total number of iterations of the simulation in which the link is busy with data transmission.

$$\boldsymbol{OLBT = \sum_{n=1}^{N} LBT_n} \qquad \textbf{Eq. 14}$$

Where *N* is the total number of links involved in the task and LBT_n is the link busy time of link *n*. This metric is related but not equal to network throughput. Throughput is about the rate of successful message delivery usually expressed in bit/sec, packet/sec or packet/timeslot; while the OLBT only estimates the time spent on communication over all links involved in a task.

The other metric used in this thesis is called Average Link Utility (often expressed in percentage) and is used to show how much of a link's time during a simulation is spent on transmitting data rather than being idle waiting for a node's internal or external event. Eq. 15 is the formal definition of ALU:

$$\boldsymbol{ALU = \frac{OLBT}{N * Simulation\ Time}} \qquad \textbf{Eq. 15}$$

Where *N* is the number of links involved in a task and *Simulation Time* is measured as the number of simulation iterations (not in seconds). By dividing *OLBT* by *N* we will have an average link busy time; and by dividing this value by *Simulation Time* we will have an idea about what portion of a link's time is dedicated to data transmission.

The last metric is OLWT which is defined same as OLBT. Overall link wait time (Eq. 16) is the aggregate link wait time of all links involved in a task. The wait time of a link (assumed to be in working condition) is the total number of simulation iterations in which a node is trying to take control of that link but denied regardless of its reason.

$$\boldsymbol{OLWT = \sum_{n=1}^{N} LWT_n} \qquad \textbf{Eq. 16}$$

Where *N* is the total number of links involved and LWT_n is the link wait time of link *n*.

Speedup and processor utility are used to analyse simulation experiments in both chapters 8 and 9. Also in chapter 9 OLBT, ALU and OLWT metrics will be used to measure the performance of the links as well.

8.2) Hidden Node Problem and Packet Collision

The two-stage network-partitioning algorithm introduced in this thesis is envisaged to solve the Hidden Node Problem and eliminate the packet collision in the network. To test this, a 2D network is simulated. To test the efficiency of the network-partitioning

algorithm an ideal propagation method is implemented in which radio signals can be detected and decoded only in their communication range; however, they can interfere with other signals beyond their radio range. This range is called “interference range” in this thesis and is twice the size of the radio range. If a receiver receives some noise from a node while receiving a signal from the transmitter, the signal is assumed corrupt and the packet would be rejected as corrupt packet. The number of packet collisions is counted in three cases. A 2D version of a BC with 10*10 nodes with SPTM was used in an experiment to see if network-partitioning algorithm can eliminate packet collision. The results are shown in Table 18. When all the nodes use one channel there is high number of collisions. When just two channels are used the number of collision incidents decreases but it is only when the full channel assignment algorithm is applied that no traces of packet collision is detected.

Number of frequencies	Number of packet collisions
1	1731342
2	665271
6	0

Table 18: Packet collision in a 2D hexagonal array with different frequencies per node.

The network partitioning algorithm introduced in chapter 5 shows perfect results for 3D networks as well. During all the experiments reported in this thesis not even a single instance of packet collision occurred which shows that the proposed algorithm works correctly for both 2D and 3D networks.

8.3) Cubic Networks vs. Spherical Networks

A 3D hexagonal topology is chosen for the BC network. In this section of the thesis it is investigated if the same topology behaves differently if it fills a cubic or a spherical space. It should be noticed that in both cases the topology of the network remains unchanged (3D hexagonal FCC).

In fact the only difference between those two cases is the state of the network in its edges and corners. In an infinite 3D hexagonal grid all nodes have exactly 12 neighbours. But it is not the case with 3D hexagonal topologies with finite sizes. In such a network the number of neighbours is less when a node is located on the edges or in the corners. This translates to lower connectivity for those nodes. Consequently, there is a chance that a network with lower average connectivity is more susceptible to have poorer performances compared to networks with larger average connectivity. It has already been observed in Figure 45 that although in theory a hexagonal network of size 3*3*4 needs no more than 4 hops to link a node to any other nodes, since the nodes on the corners are suffering from low connectivity the actual maximum number of hops for such a network is 5. For this reason, the same topology is used to fill two different

containers to study the effect of low connectivity on the edges. A spherical shape may increase the number of neighbours for nodes on edges and corners as the edges and corners in a (semi-) spherical container are smoother compared to a cubic container.

A series of experiments has been conducted to test the effect of container on the average number of neighbours per node and consequently on the performance of the network. In these experiments FFTTM is applied to two shapes for a 3D hexagonal networks of almost same number of nodes: a cubic container for 800 nodes and a (semi-) spherical container for 783 nodes. The number of nodes are not exactly equal to keep the cubic and spherical shapes as accurate and symmetric as possible. In both cases the nodes can execute 32 million instructions per second. The data rate of links is 10 Gb/s which is the same in both cases. The size of data packets varies from 10 Bytes (real data) to 20 KBytes (real data). Such values may not be usual in real-world applications but they are included to have a wider understanding of the behaviour of the BC platform.

A range of intervals between packets are tested to understand how important the interval between packets is. In all tests the processor utility of the network (in form of percentage as defined by Eq. 13) is measured. The results for the cubic container are listed in Table 19 and graphically shown in Figure 73.

Time Interval \ Packet Size	10	50	100	500	1000	2000	5000	10000	20000
1	27.42	76.29	86.37	90.39	90.08	89.07	86.98	88.26	91.45
5	16.78	66.96	81.97	90.49	90.60	89.09	88.50	91.68	88.35
10	15.36	62.72	83.21	89.95	87.74	89.44	90.98	90.43	91.32
20	12.80	54.34	76.44	89.50	91.52	89.92	89.65	89.57	90.62
50	8.40	36.28	62.31	88.05	87.57	89.29	91.68	91.09	89.88
100	5.28	24.17	41.22	80.49	84.70	89.99	89.07	91.65	91.16
200	3.01	14.37	29.00	70.63	82.04	87.61	90.65	91.36	88.73
500	1.32	6.70	12.82	49.67	64.84	81.63	88.39	90.76	88.39
1000	0.73	3.66	6.97	29.42	53.63	66.24	84.79	88.44	91.16
2000	0.37	1.83	3.60	16.12	30.15	53.25	73.43	82.83	89.31

Table 19: Processor Utility (%) of a cubic ball computer consisting of 800 nodes.

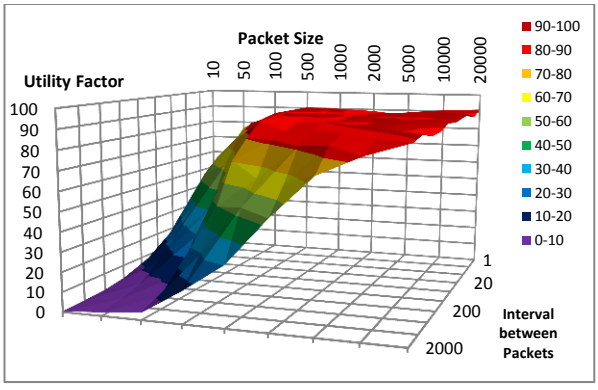


Figure 73: The Processor Utility of a cubic ball computer with different packet sizes and intervals between packets.

The results for the spherical container is also listed in Table 20 and graphically plotted in Figure 74. Based on the results in these tables and figures a cubic container yields just a little bit better performances compared to a spherical container. It should be mentioned that because the containers are not very large and also because of the hexagonal topology used for both cases, the actual shape of the network is neither perfect cubic nor perfect spherical.

Packet Size / Time Interval	10	50	100	500	1000	2000	5000	10000	20000
1	27.64	78.75	86.32	90.34	92.20	87.19	91.84	91.63	92.46
5	16.48	63.47	79.12	87.63	89.12	86.50	89.39	89.05	92.13
10	14.42	55.91	78.82	82.34	88.70	89.57	86.92	91.60	92.12
20	12.72	52.12	73.26	87.12	81.51	82.60	92.06	91.38	89.84
50	8.07	36.68	59.58	86.21	89.46	82.36	89.14	89.53	91.96
100	5.38	24.60	42.92	82.61	85.67	90.07	90.97	92.28	91.75
200	3.11	14.86	28.42	73.94	83.36	84.96	87.02	91.39	89.61
500	1.44	6.63	13.35	50.48	69.28	80.12	88.45	90.53	91.49
1000	0.72	3.45	6.99	30.14	47.08	70.11	83.26	88.67	90.60
2000	0.37	1.87	3.56	15.82	30.51	53.41	73.94	85.38	89.57

Table 20: Processor Utility (%) of a spherical ball computer consisting of 783 nodes.

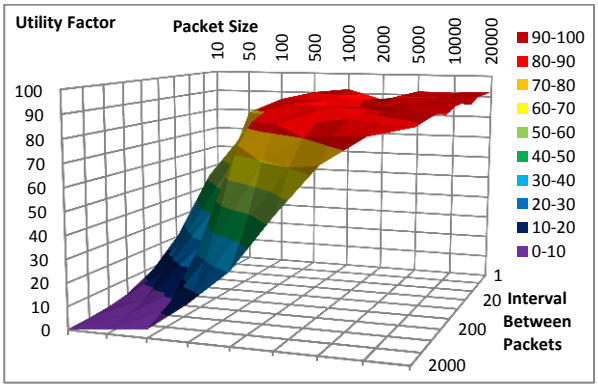


Figure 74: The Processor Utility of a spherical ball computer with different packet sizes and packet intervals.

The average number of neighbours per node in a cubic container of 800 nodes is 10.02. The same number for a spherical container of 783 nodes is 10.25.

This explains why the differences between results in Table 19 and Table 20 are not significant. The size of two networks in these experiments are a bit different; therefore, it can be assumed that the slightly different size of the networks cannot make a big difference in these experiments. Since the spherical containers do not have a meaningful advantage over cubic containers, cubic containers are used for their simplicity.

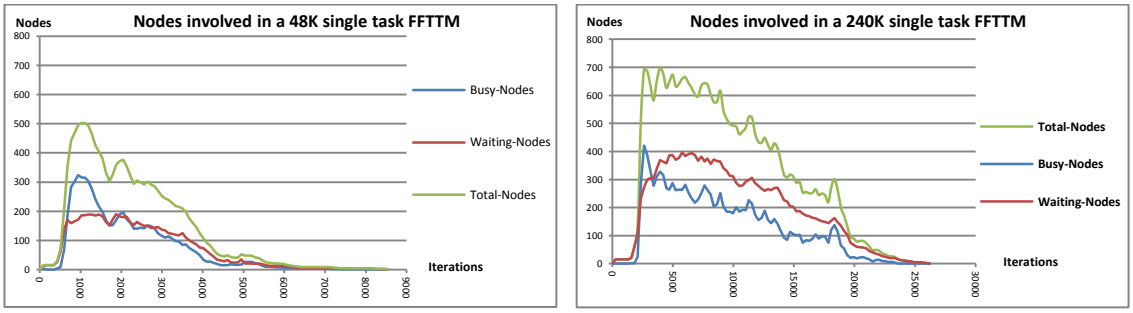
8.4) Multi-Tasking and Multiple Workloads

The performance of a divide-and-conquer task-model like FFTTM may be enhanced by using its idle times for other tasks. To find the extent of this improvement a set of tests are running on FFTTM. Two different data sizes are assigned to the starter nodes: 48K

and 240K Bytes. By choosing these data sizes we will not have a wide picture of how multiple instances of FFTTM react to different data sizes; they are just examples to show how that task-model deals with small and medium sizes of data.

Up to 8 independent workloads are used in this set of experiments. Regarding the cubic shape of the network it is a reasonable choice to have 8 workloads. Workloads are located near the eight corners of the cubic container of the network which gives balance and symmetry to the workloads. A 3D network of size 10*10*8 nodes is used.

For the first set of tests a single FFTTM workload are applied to single-task nodes. Then the same tests are repeated with multi-task nodes. The data rates of all links in all tests in this series of experiments are 10 Gb/s. Also all nodes execute 32 million instructions per second. The single task results are shown in Figure 75[33].



(a) **(b)**

Figure 75: Nodes involved in a single-task FFTTM with a) 48K bytes, Max Nodes Involved= 547, b) 240K Bytes Max Nodes Involved= 799.

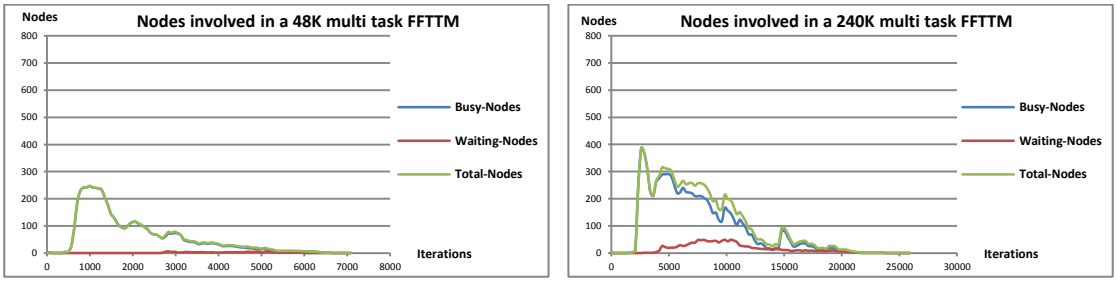
The important issue is the number of waiting nodes which in both cases are significantly high. The figure shows that during a large part of the simulation time a large number of nodes are just waiting for results from other nodes' without actively participating in the task given. This is interpreted as inefficiency of the single-task FFTTM.

When the multi-tasking feature is added to nodes they produced results which can be found in Figure 76[34]. What is apparent in these graphs is the sharp drop in the number of waiting nodes. This means that the idea of recruiting waiting nodes for other tasks has worked. The figures show the simulation time does not change very much while the number of nodes involved and the waiting nodes in particular are dropped significantly.

The tasks on different nodes do not finish at the same time. This means that at some parts of simulation time the number of nodes that are involved in the task can be very low. This can be seen in Figure 75 and Figure 76. In both single-task and multi-task nodes the number of nodes involved in a task rises quickly and then falls. This is the right time for a new task to reuse the nodes that have just finished their old task. We

[33] Each test is repeated three times. The variance of results are 3076.22 (48K) and 29400.89 (240K).
[34] Each test is repeated three times and the variance of results are 231889.56 and 383308.67 for 48K and 240K respectively.

have already seen the effect of multi-tasking with single workloads; but it is anticipated to show its real strength when combined with multiple workloads.



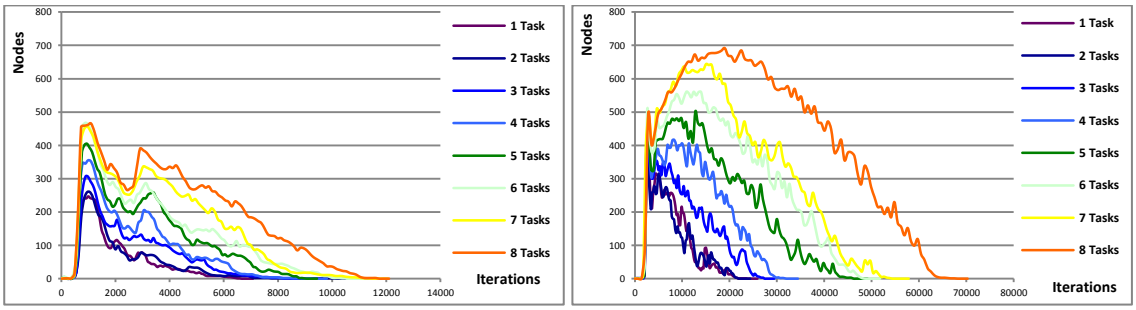
(a) **(b)**

Figure 76: Nodes involved in a multi-task FFTTM with a) 48K bytes, Max Nodes Involved= 247, b) 240K Bytes Max Nodes Involved= 386.

To test the effect of using multiple workloads a set of experiments are run using FFTTM with data sizes of 48K Bytes and 240K Bytes. These two data sizes are chosen to see if the task-model's behaviour changes with a change in data size. The number of workloads varies between 1 and 8. For the first test there is only one workload located near one of the corners of the network. Then for the second test another workload is placed near another corner of the network. This repeats until in the eighth experiment there are 8 workloads each located near one of the corners of the network.

The number of nodes involved in an FFTTM is measured with two data sizes and different number of tasks per node (Figure 77). Figure 77.a[35] - in particular - shows that the number of nodes involved stays reasonably low when new workloads are added.

Both graphs (especially part b[36]) show the task execution time does not increase dramatically when new workloads are introduced. When the data size increases the number of nodes involved gets close to its maximum number but with low data sizes even with multiple workloads the number of nodes involved is not very significant.



(a) **(b)**

Figure 77: Nodes involved in a set of multi-task FFTTM with a) 48K bytes, b) 240K Bytes.

[35] Each test is repeated three times and the variance of results are 3748.67, 361243.56, 244322.89, 106568.22, 660404.22, 319970.67, 769690.89, 407490.89 for 1 to 8 workloads respectively.

[36] Each test is repeated three times and the variance of results are 485810.67, 22190.89, 33950.00, 20393.56, 409482.67, 1763764.67, 100002.89, 144697.56 for 1 to 8 workloads respectively.

The effect of multi-tasking on the performance of the FFTTM is also studied in a separate set of experiments in which the network consists of 800 (8*10*10) nodes; each node does 32 million instructions per second and the links' data rate is 10 Gb/s. The task used is the original FFTTM (with no load balancing). The results (Table 21) show for the majority of cases a node yields better results when it accommodates more tasks. In these experiments the network's processor utility (Eq. 13) is measured.

Data Size (KB)	Tasks/ Node	Nodes Involved	Packet Size	Processor Utility (%)	Data Size (KB)	Tasks/ Node	Nodes Involved	Packet Size	Processor Utility (%)
48000	1	170	282	**8.76**	2400000	1	800	3000	**39.26**
	2			**11.13**		2			**45.60**
	4			**10.60**		4			**39.38**
	8			**10.64**		8			**67.44**
240000	1	800	300	**6.82**	4800000	1	800	6000	**43.2**
	2			**6.85**		2			**39.35**
	4			**13.24**		4			**56.76**
	8			**12.09**		8			**67.50**
480000	1	800	600	**13.87**	9600000	1	800	12000	**39.70**
	2			**12.17**		2			**44.97**
	4			**13.76**		4			**57.74**
	8			**22.10**		8			**70.57**

Table 21: The effect of multi-tasking in an FFTTM with different data sizes.

It should be noticed that for a majority of the tests the whole network is saturated by the task and all the nodes are involved in the test except for the smallest data size (48KB). That is because the task-model is designed to have a lower limit on the data size for a local FFT. When the original data size is small it splits into smaller number of pieces. Figure 78 is the graphical representation of data presented in Table 21. In addition to testing the idea of multi-tasking, the graph also reinforces the results derived from other tests in which higher performances are usually achieved with larger data sizes.

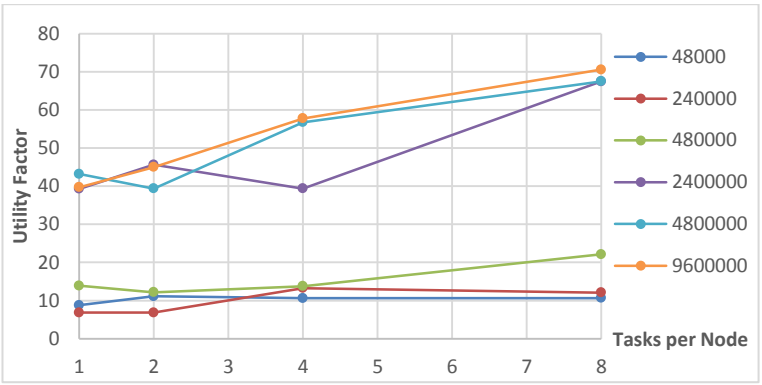
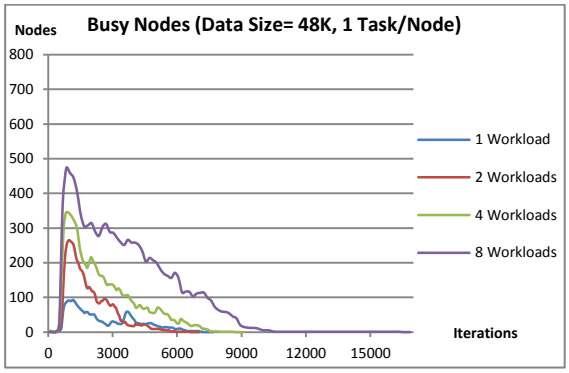


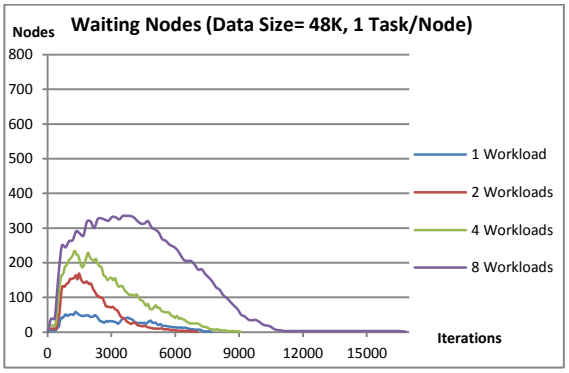
Figure 78: The improvement of performance as the number of tasks per node increases

Studying the number of nodes involved in a multi-tasking FFT is the subject of another set of tests. Two different data sizes (48KB and 240KB) are used in this series of tests. There are four different experiments for each data size in which the number of tasks per node is chosen from the values 1, 2, 4 and 8. The number of workloads is also 1, 2, 4

and 8. In all experiments the number of busy nodes and the waiting nodes are measured during the execution time of the simulation.

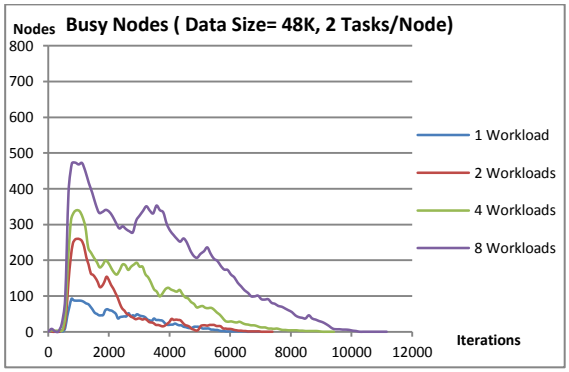


(a)

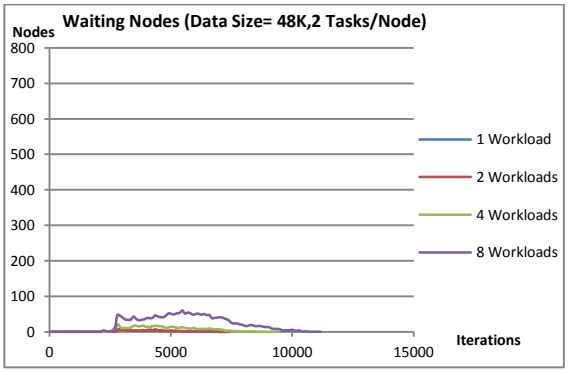


(b)

Figure 79: (a) Busy and (b) Waiting nodes in a 48K FFTTM when each node has 1 task.

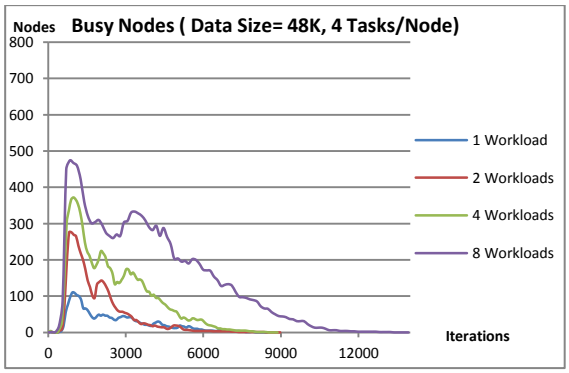


(a)

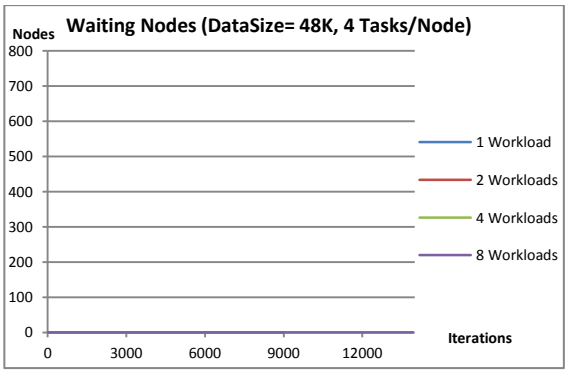


(b)

Figure 80: (a) Busy and (b) Waiting nodes in a 48K FFTTM when each node has 2 tasks.

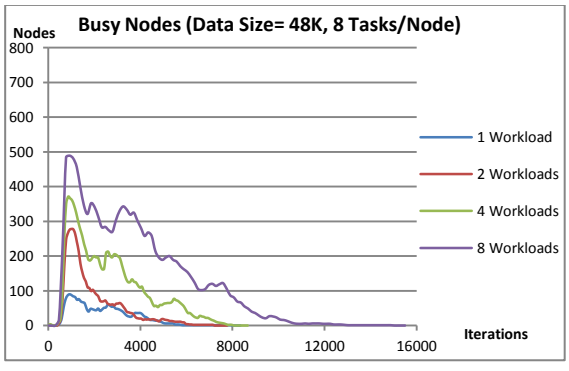


(a)

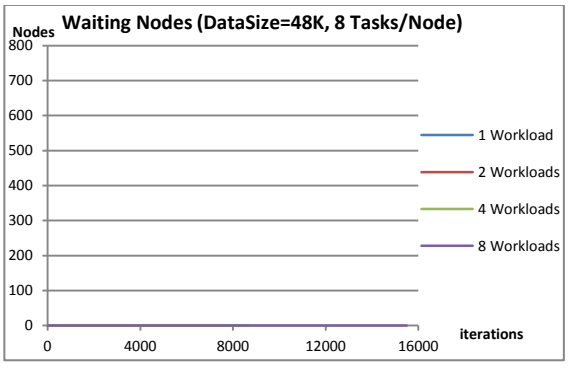


(b)

Figure 81: (a) Busy and (b) Waiting nodes in a 48K FFTTM when each node has 4 tasks.



(a)

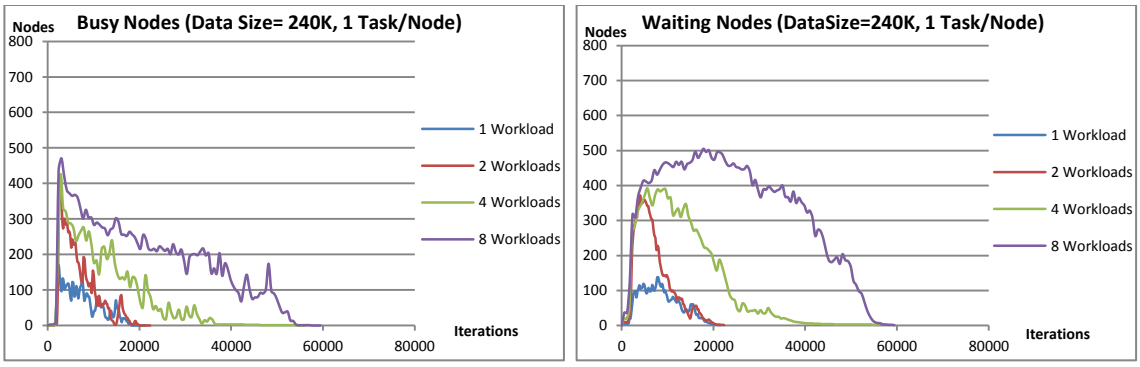


(b)

Figure 82: (a) Busy and (b) Waiting nodes in a 48K FFTTM when each node has 8 tasks.

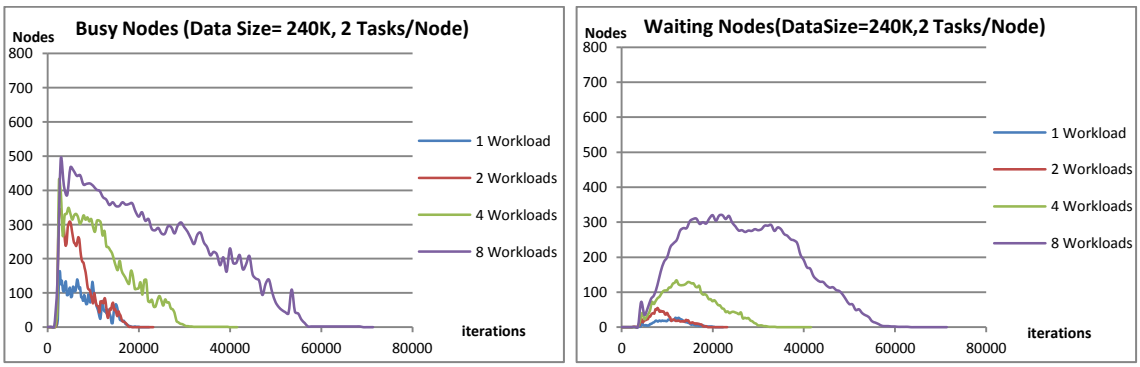
Figure 79, Figure 80, Figure 81 and Figure 82 show the busy and waiting nodes for an FFTTM of size 48K. The network is a 3D hexagonal grid of size 800 (8*10*10), the data rate of links is 10 Gb/s and each node can perform 32 million operations per second.

Figure 83, Figure 84, Figure 85 and Figure 86 plot the busy and waiting nodes for FFTTMs of size 240K. Part (b) of all these figures show that the number of waiting nodes sharply reduces when multi-tasking is involved. This is because there are many waiting nodes in a single-task node that will have the chance to get busy with another task in a multi-tasking node. This changes their status from a waiting node to a busy one and therefore the number of waiting nodes reduces with a rise in the number of tasks per node. This reduction intensifies when the number of tasks per node increases. Figure 82.b and Figure 86.b show that the number of waiting nodes is almost zero when each node handles 8 tasks at the same time regardless of the number of workloads.



(a) **(b)**

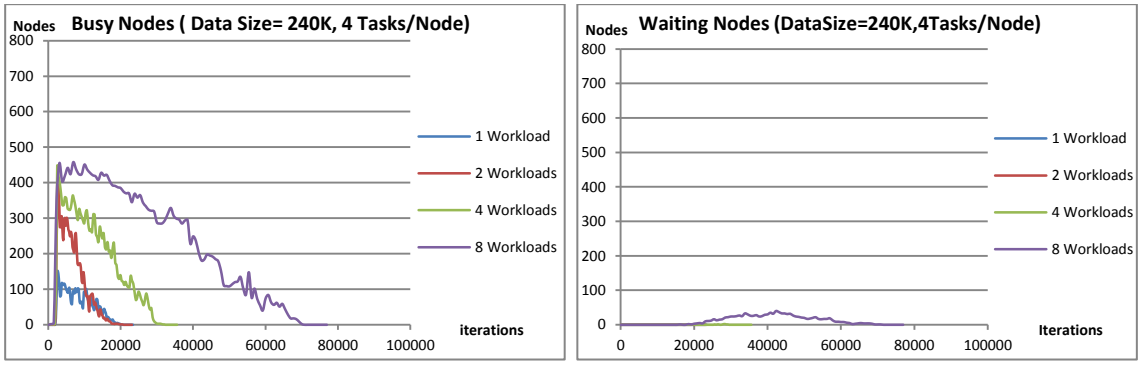
Figure 83: (a) Busy and (b) Waiting nodes in a 240K FFTTM when each node has 1 task.



(a) **(b)**

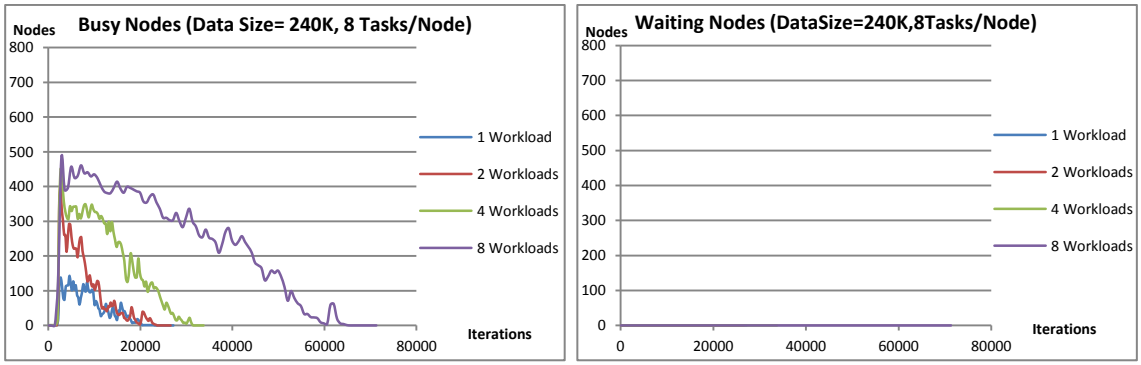
Figure 84: (a) Busy and (b) Waiting nodes in a 240K FFTTM when each node has 2 tasks.

On the other hand the number of busy nodes plotted on part (a) of above figures are almost unaffected by the number of tasks per node while it is affected by the number of workloads. When the number of workloads is fixed the number of busy nodes is solely determined by the internal mechanism of the divide-and-conquer task (FFT in this example). The speedup factor (as defined by Eq. 12) increases when either the number of tasks per node or the number of workloads increase.



(a) **(b)**

Figure 85: (a) Busy and (b) Waiting nodes in a 240K FFTTM when each node has 4 tasks.



(a) **(b)**

Figure 86: (a) Busy and (b) Waiting nodes in a 240K FFTTM when each node has 8 tasks.

There are seemingly some anomalies in some parts of graphs especially when comparing 1 workload against 2 workloads. Sometimes during their execution times it happens that the number of busy nodes with one workload increases that of two workloads. This happens just for a fraction of the execution time of the tasks. However, this cannot be regarded as a real anomaly since there is no strong relation between the statuses of the network in a given iteration of the simulation in those two cases. As an example, in Figure 79.an approximately around iteration 4000 there are more busy nodes with 1 workload (blue graph) than that of 2 workloads (red graph). This is not a permanent and/or dominant situation and can be regarded as a temporal fluctuation in the number of busy nodes caused by the randomness of transactions in both simulations. Not all workloads start their execution right from the start of the simulation. This is to have a lag between workloads to let them start after the peak of operations of the previous one is passed. The workload execution timing is based on an estimation and this thesis has not tried to optimise the timing of the workloads execution. Therefore, a non-optimal workload execution timing may contribute in minor temporal anomalies in results.

In general these experiments show that a multi-tasking node in BC platform running FFTTM can help reducing the number of waiting nodes although it cannot make a big change in the execution time since multiple tasks engaging with different workloads

occupy their share of CPU time. Because the CPU computational ability is unchanged in these two cases (single-task and multi-task), the overall time needed for running all workloads is expected to have no meaningful change.

8.5) Task-model Modifications and Load Balancing

In this part of the thesis it will be tested that to what extent the simple load balancing technique adopted by this thesis has improved the efficiency of the network. In theory the performance should improve when the workload is shared between different sub-trees of a dependency tree proportionate to their sizes. In practice two different versions of FFTTM (with and without load balancing) are run on the same network with the same parameters and with the same data size. The results endorse the theoretical conclusion that the load balancing technique improves the performance. Testing other versions of load balancing algorithm is left for future work.

In this set of experiments two 3D network of 1000 nodes (10*10*10) and 2028 nodes (12*13*13) are used. The transfer rate of all links is set to 10 Gb/s. All nodes can process 32 million instructions per second. Only one workload is applied to the network in each experiment. The performance of the network is measured with different data sizes (i.e. 100K Bytes and 10M Bytes). These two values are chosen only to represent low and high data sizes. Therefore, this particular experiment cannot be regarded as a comprehensive study on the effect of data size on load balancing. The results of the experiments are listed in Table 22. The processor utility factors (in percentages) are measured based on Eq. 13.

Network Size	Data Size	Load Balancing Included	Processor Utility (%)
1000 Nodes	100K Bytes	No	4.73
		Yes	10.13
	10MBytes	No	20.12
		Yes	96.27
2028 Nodes	100K Bytes	No	2.74
		Yes	2.23
	10MBytes	No	10.19
		Yes	87.31

Table 22: Comparing the performance of two versions of FFTTM; with and without load balancing.

As the table shows in almost all values for network size and data size the FFTTM with load balancing yields much better performances compared to the task-model without load balancing. The only exception is when the FFTTM with 100K Bytes of data is applied to a network of around 2K nodes. Regarding the larger size of the network (compared to 1K nodes), the smaller data size (compared to 10M Bytes) and the nature of the task-model (which engages all nodes) the performance is so poor that even load

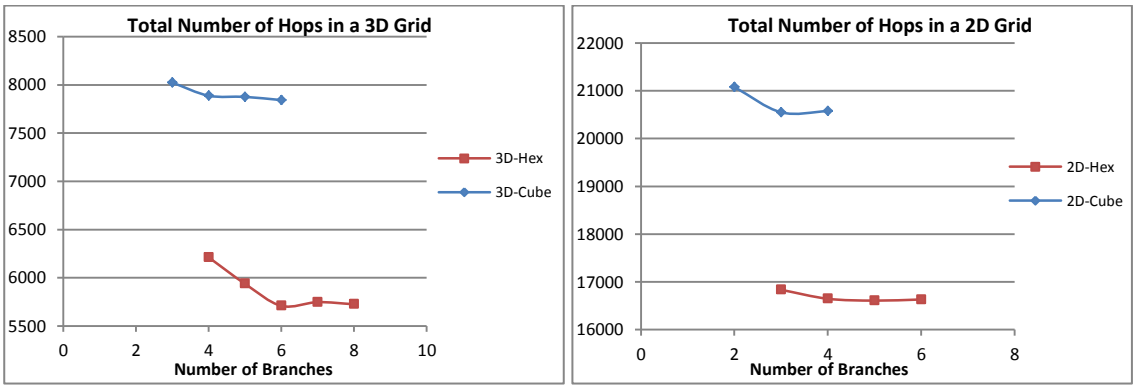
balancing cannot help improving it. The reason for slight decrease in performance with load balancing in this test is unknown at the moment.

The reason why the performance of the network under 2K nodes drops compared to the 1K node network is discussed later in chapter 9.

8.6) Tuning the Number of Branches per Node

A divide-and-conquer task-model like FFTTM follows a repetitive pattern of finding new candidates and sharing the task with them. The number of nodes each node finds determines the shape of the dependency tree of a task, which in its turn plays an important role on how fast the task finishes. This parameter should be tuned and optimised to maximise the performance of the task-model by decreasing the total number of hops.

A set of experiments is conducted to find the best number of branches per node. In this set of experiments an enhanced FFTTM is used. 2D and 3D grids are tested separately. The number of branches varies from 4 to 8 in 3D in a network of 1000 (10*10*10) nodes. Each 3D case is repeated 8 times. The number of branches varies from 3 to 6 in 2D in a network of 1024 (32*32) nodes. Each 2D case is repeated 4 times. Each 2D or 3D case is run on a hexagonal topology as well as a cube topology. In all tests the FFTTMs are applied to a data of size 100K bytes. The results are projected in Figure 87.



(a) **(b)**

Figure 87: Total number of hops in Hexagonal and cubic topologies in (a) 2D and (b) 3D.

The best network performance is achieved when the total number of hops is the lowest. The results show that the optimum number of branches for a divide-and-conquer task like FFT is 5 in 2D grids and 6 in 3D grids using hexagonal topology. The same numbers for a cubic topology are 3 and 6 respectively. With those values for branches the lowest total number of hops is achieved.

Chapter 9: Overall Study of the Behaviour of the Network

So far, the performance of the network (in terms of processor utility) has been measured in a fixed set of network attributes. The most important network attributes are network size, data size, transfer rate of links and the number of instructions each node executes in a second. Running experiments with such a pre-fixed set of attributes do not give a general understanding of the behaviour of the network. To have a broader view of how the network performs in different situations, it is important to test the network with different network attributes.

In the final set of experiments in this thesis the behaviour of the network is studied when the main four network attributes listed above are changing. Table 23 lists the range of values chosen for the aforementioned network attributes. In all these experiments what is measured is the network's processor utility factor (as defined in Eq. 13), link utility (Eq. 13), link's busy time (Eq. 13) and link wait time (Eq. 13). The range of values for these attributes start from rather low values which can be easily achieved with the current level of technologies with a rather low price (e.g. 10MIPS of processing ability, 100 Mb/s links and 100 KB of data). The higher values for the network attributes are not necessarily available with the state-of-the-art technology at the moment (e.g. processing ability of 100 GIPS and links with 100 Gb/s transfer rate). Such currently-unrealistic values are included to give an idea about the effect of future technological improvements on the performance of the proposed BC network.

Network Attributes	**Values**				
Number of Nodes	1000	2000			
Number of Instructions per Second	10 MIPS	100 MIPS	1 GIPS	10 GIPS	100 GIPS
Data Rate (bits per second)	100 Mb/s	1 Gb/s	10 Gb/s	100 Gb/s	
Data Size (Bytes)	100 KB	1 MB	10 MB	100 MB	

Table 23: Range of values for four major network attributes in the last set of experiments.

The simulations were running on an HPC grid partially owned by the University of York called the White Rose Grid (WRG)[37]. On each set of experiments 80 tasks were submitted to the grid. The results are presented and discussed in following sections.

9.1) FFTTM on 1000 Nodes

In the first part of these experiments the network size is 1000 nodes. There are 8 FFTTM workload in each test. A total number of 80 simulation tests produce results shown in Table 24 (a to d) and plotted in Figure 88 (a to d).

[37] http://www.wrgrid.org.uk/

Data Rate / Instr./Sec	100 Mb/s	1 Gb/s	10 Gb/s	100 Gb/s	100 Mb/s	1 Gb/s	10 Gb/s	100 Gb/s
10MIPS	1.08	7.81	14.1	14.90	1.21	21.48	67.29	71.06
100MIPS	0.11	0.95	6.57	14.93	0.097	1.80	17.08	68.69
1GIPS	0.01	0.12	0.64	8.41		0.11	1.55	18.60
10GIPS		0.01	0.093	1.47			0.20	1.38
100GIPS			0.01	0.13				0.22

(a) Data size= 100 KB **(b) Data size= 1 MB**

Data Rate / Instr./Sec	100 Mb/s	1 Gb/s	10 Gb/s	100 Gb/s	1 Gb/s	10 Gb/s	100 Gb/s
10MIPS	3.31	29.95	83.90	85.04	41.83	68.40	86.72
100MIPS		3.23	32.54	83.54		49.75	77.98
1GIPS			3.063	30.74			62.46
10GIPS				2.96			

(c) Data size= 10 MB **(d) Data size= 100 MB**

Table 24: Processor Utility in a set of FFTTMs on a network of 1000 nodes with different network attributes. Data sizes varies: (a) 100 KB, (b) 1MB, (c) 10MB and (d) 100MB.

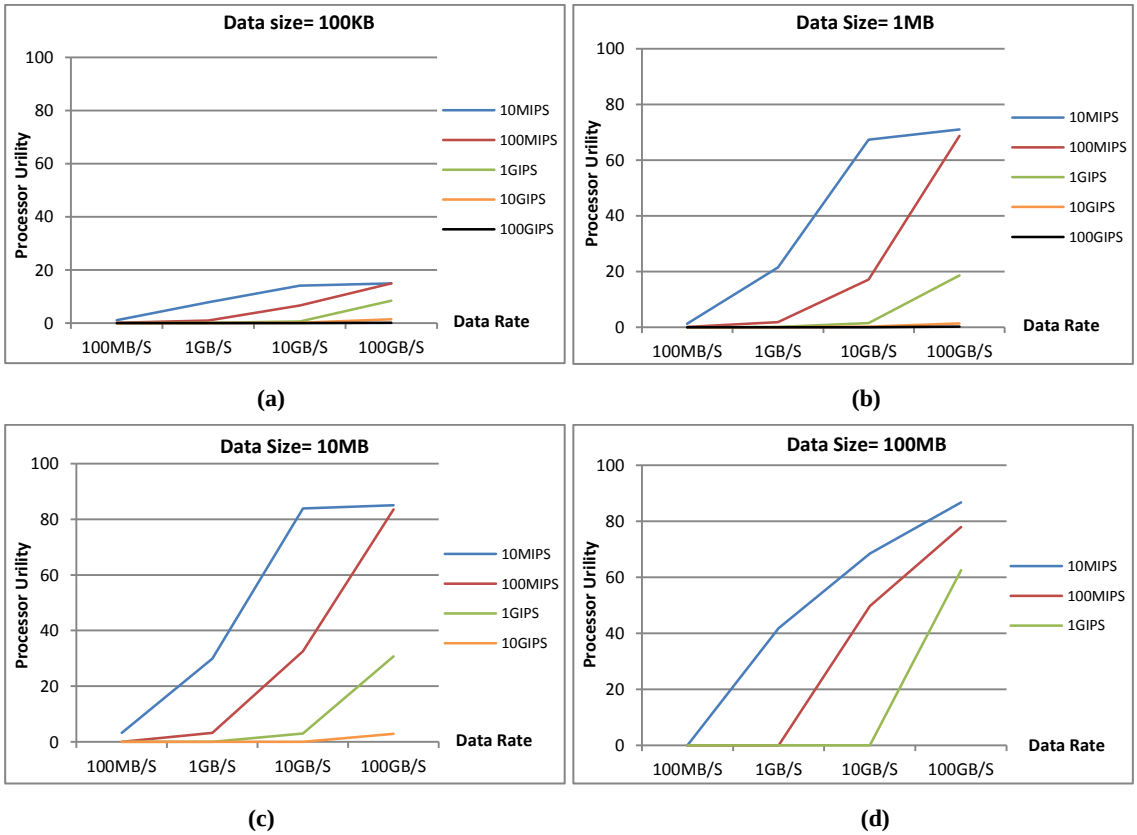


Figure 88: Processor Utility for FFTTM on 1000 nodes with different network attributes.

The results show that the performance of the network improves as the data size increases. Also, in general, higher transfer rates means better performances. Another point in the above tables is that the performance of the network relates with the ratio of the transfer rate to the number of instructions per second (called comm/comp ratio in this thesis). A higher value for such a ratio usually means high network performances[38].

[38]Some values in Table 24 and Figure 88 are missed. These numbers can be regarded as zero based on the log files keeping a record of the simulator's operations which showed very poor performances in those

OLBT and OLWT are also measured in this set of experiments (Table 25 and Table 26). The network parameters are chosen from Table 23 with only two exceptions: Data of size 100MB and computational ability of 100GIPS (except for two tests) are excluded from this part of the experiments.

Data Rate / Instr./Sec	100 Mb/s	1 Gb/s	10 Gb/s	100 Gb/s	100 Mb/s	1Gb/s	10 Gb/s	100 Gb/s
10MIPS	2490.51	263.73	57.95	43.30	49016.35	1745.95	154.76	42.52
100MIPS	2557.00	245.81	33.16	5.51	22270.63	2802.69	174.27	15.78
1GIPS		246.48	24.77	2.66		755.04	204.93	16.95
10GIPS		244.02	67.24	2.48		696.15	209.05	20.42
100GIPS			26.21	2.80				

(a) Data size= 100 KB **(b) Data size= 1 MB**

Data Rate / Instr./Sec	100 Mb/s	1 Gb/s	10 Gb/s	100 Gb/s
10MIPS	391004.23	16415.70	1294.13	163.35
100MIPS	63114.27	12568.77	1520.68	134.57
1GIPS		21604.61	5903.79	154.02
10GIPS		2172.28	401.24	41.68

(c) Data size= 10 MB

Table 25: OLBT (in mSec) measured in a set of FFTTMs on a network of 1000 nodes with different network attributes. Data sizes varies: (a) 100 KB, (b) 1MB and (c) 10MB.

Data Rate / Instr./Sec	100 Mb/s	1 Gb/s	10 Gb/s	100 Gb/s	100 Mb/s	1Gb/s	10 Gb/s	100 Gb/s
10MIPS	1976.85	210.04	39.07	8.99	63962.47	1593.80	19.66	1.51
100MIPS	2566.20	194.51	41.36	1.22	32928.22	4095.57	124.66	2.04
1GIPS		229.69	20.61	2.23		878.18	325.17	16.05
10GIPS		231.32	60.09	2.14		785.83	254.11	25.18
100GIPS			27.39	2.97				

(a) Data size= 100 KB **(b) Data size= 1 MB**

Data Rate / Instr./Sec	100 Mb/s	1 Gb/s	10 Gb/s	100 Gb/s
10MIPS	624374.68	10744.92	159.32	10.72
100MIPS	103566.50	27463.57	701.92	18.34
1GIPS		49977.70	9871.67	87.54
10GIPS		2340.13	787.21	114.01

(c) Data size= 10 MB

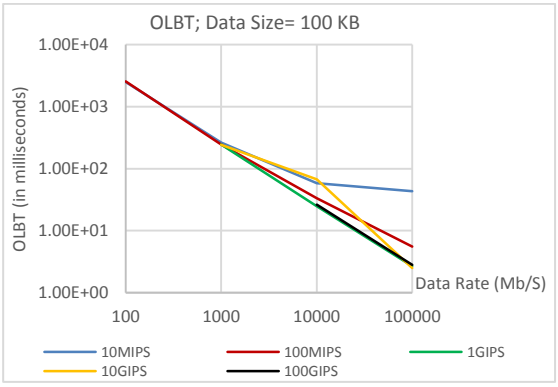
Table 26: LWT (in mSec) measured in a set of FFTTMs on a network of 1000 nodes. Data sizes varies: (a) 100 KB, (b) 1MB and (c) 10MB.

These two metrics are measured in mSec rather than simulation iterations because a simulation iteration represents different time periods in different tests. The tables show:

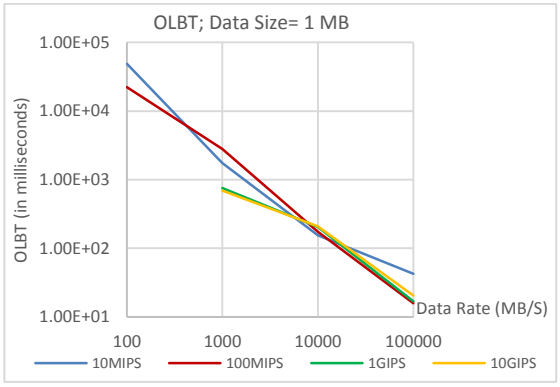
- With a fixed computational ability, both metrics reduce as data rate increases;
- With a fixed data rate, both metrics reduce as computational ability increases;
- It is not surprising that both metrics increase when data size increases.

---

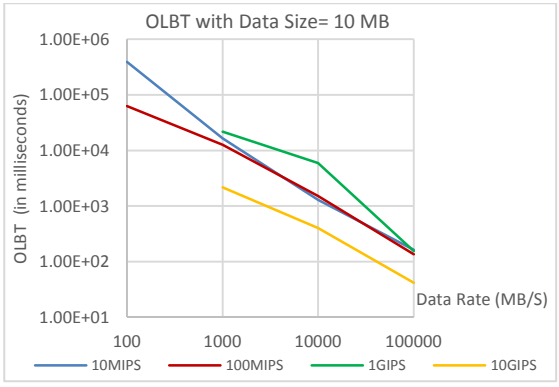
cases. In a high majority of those cases the simulation time exceeded a predefined threshold set by WRG (the platform used for simulations).



(a) Data size= 100 KB

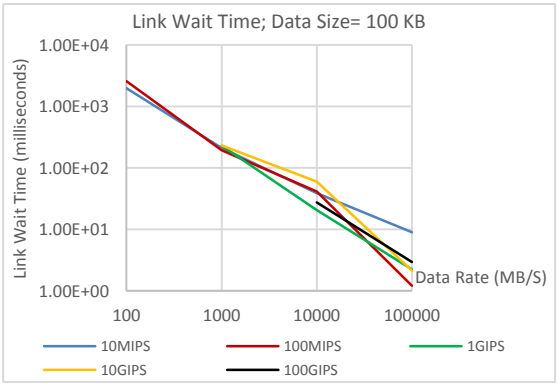


(b) Data size= 1 MB

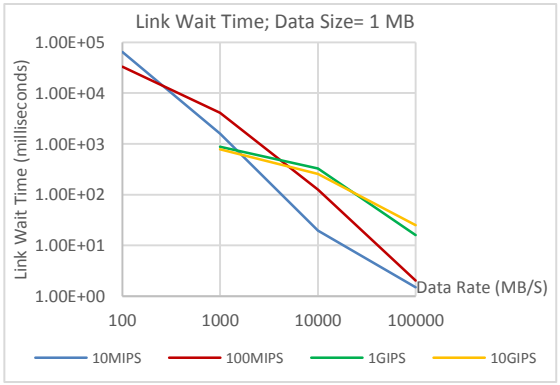


(c) Data size= 10 MB

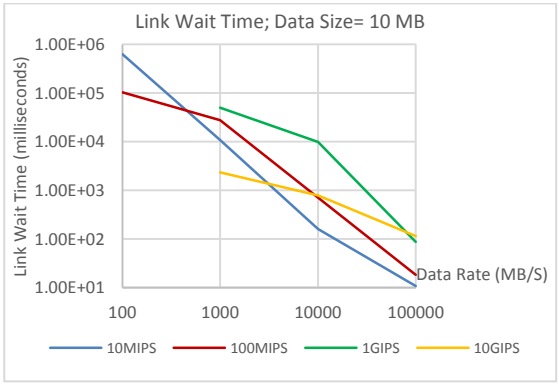
Figure 89: OLBT (in mSec) for FFTTM on 1000 nodes with different network attributes.



(a) Data size= 100 KB



(b) Data size= 1 MB



(c) Data size= 10 MB

Figure 90: Link Wait Time (in mSec) for FFTTM on 1000 nodes with different network attributes.

The results are also graphically presented in Figure 89 and Figure 90. The slope of reduction in both metrics varies with different values of network parameters but the

general trend is that both two link-related metrics decrease as the data rate and computational ability increase.

9.2) FFTTM on 2000 Nodes

To study the effect of network size on its performance a separate set of experiments are run in which the network size is increased to around 2000 nodes (2002=11*13*14 nodes to be more precise). The other three network attributes are changing the same as the experiments with 1000 nodes (section 9.1). There are 8 FFTTM workloads in each test. The measured processor utilities are listed in Table 27 and plotted in Figure 91.

Data Rate / Instr./Sec	100 Mb/s	1 Gb/s	10 Gb/s	100 Gb/s	100 Mb/s	1 Gb/s	10 Gb/s	100 Gb/s
10MIPS		2.26	4.97	4.73	0.40	6.08		48.01
100MIPS	0.03		2.27	4.75			7.85	43.31
1GIPS			0.16	2.23		0.09	0.60	7.14
10GIPS				0.21				0.47
100GIPS				0.02				0.12

(a) Data size= 100 KB **(b) Data size= 1 MB**

Data Rate / Instr./Sec	100 Mb/s	1 Gb/s	10 Gb/s	100 Gb/s	1 Gb/s	10 Gb/s	100 Gb/s
10MIPS	1.49	12.05	75.52	78.8	17.56	76.05	82.35
100MIPS		1.14	14.44	76.21		16.53	78.48
1GIPS			1.15	15			19.64
10GIPS				1.22			

(c) Data size= 10 MB **(d) Data size= 100 MB**

Table 27: Processor Utility (percentage) for FFTTM on a network of 2000 nodes with different network attributes. Data sizes varies: (a) 100 KB, (b) 1MB, (c) 10MB and (d) 100MB.

In section 9.1 it was observed that the performance of the network improves when the comm/comp ratio increases. The same trend can be detected in this set of experiments as well.

The results in the above table and graphs show that the performance of FFTTM is falling for the most of cases compared to the same network attribute values with 1000 nodes (in section 9.1). This may suggest that for an FFTTM and a given set of network attributes there is an optimum network size with which the performance of the network is in its peak. For networks smaller and bigger than this threshold the performance of the network is anticipated to be lower. This does not suggest that the size of a BC cannot increase beyond that size. Instead, it means that the tasks should not be given the grant to expand beyond that limit even if the size of a BC exceeds that threshold. The threshold is not a solid limit and depends on many network attributes (e.g. transfer rate and number of instructions per second). The results show that the performances of almost all the tests have fallen. This suggests that the threshold is likely not to depend

on the problem size (or at least is very loosely related to the problem size). More tests with different network sizes are needed on this matter.

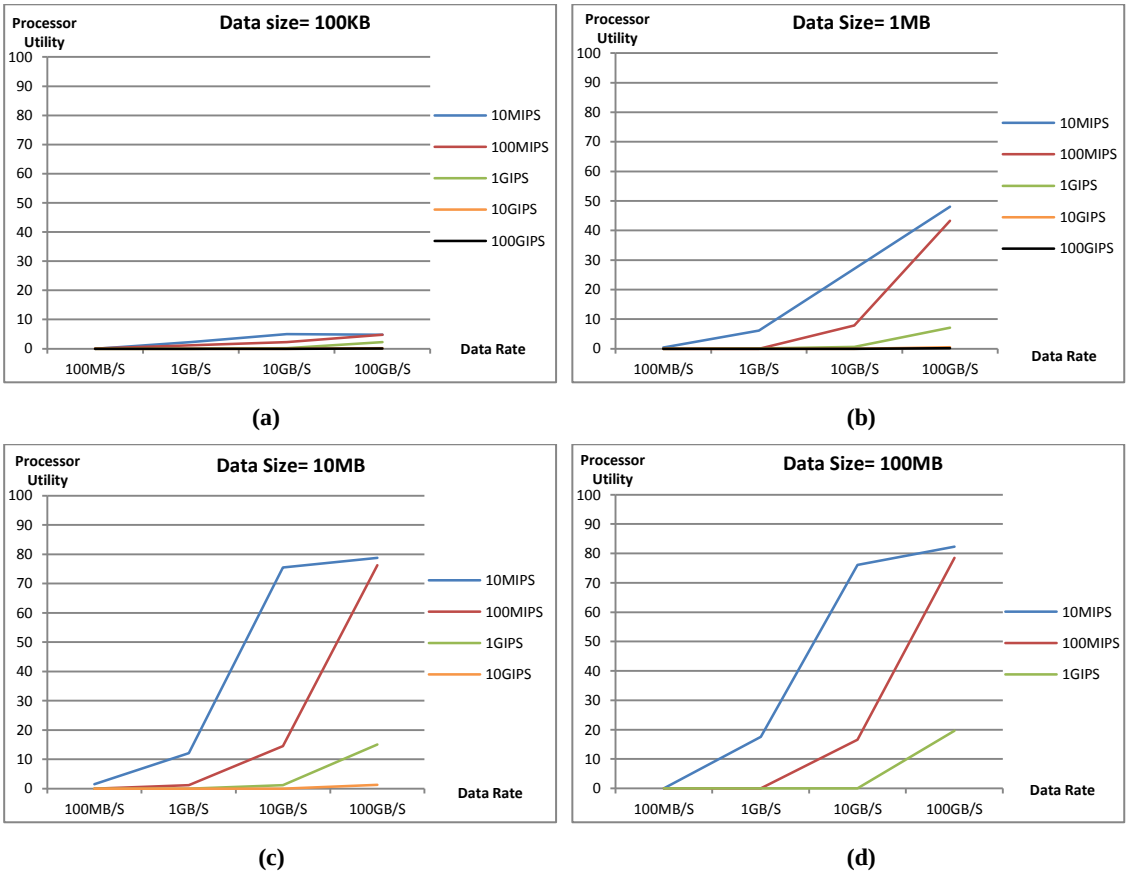


Figure 91: Processor Utility (percentage) of FFTTM on a 2000-node network with different sizes: (a) 100KB, (b) 1MB, (c) 10MB, and (d) 100MB.

Data Rate / Instr./Sec	100 Mb/s	1 Gb/s	10 Gb/s	100 Gb/s	100 Mb/s	1Gb/s	10 Gb/s	100 Gb/s
10MIPS	71811.93	5630.74	1121.81	356.23	284373.04	26589.06	2415.93	
100MIPS	49075.04	6690.20	521.41	114.61	403995.26	15712.56	2555.30	245.68
1GIPS		6338.44	463.88	121.97		81721.32	2133.83	192.15
10GIPS		4289.51	443.61	54.57		8543.12	3668.26	532.69

(a) Data size= 100 KB **(b) Data size= 1 MB**

Data Rate / Instr./Sec	100 Mb/s	1 Gb/s	10 Gb/s	100 Gb/s
10MIPS	5249580.19	200647.05	14788.25	2467.09
100MIPS	564259.93	608751.91	38538.20	1549.97
1GIPS		43383.73	21272.16	1973.62
10GIPS				2211.93

(c) Data size= 10 MB

Table 28: OLBT (in mSec) measured in a set of FFTTMs on a network of 2000 nodes with different network attributes. Data sizes varies: (a) 100 KB, (b) 1MB and (c) 10MB.

Measurements of OLBT and OLWT metrics are also included in this section (Table 27and Table 28). The network parameters are the same as FFTTM experiments on 1K-node networks.

Like the previous section, the metrics are measured in mSec rather than simulation iterations. The results show that like 1K-node networks, both link-related metrics

decrease when either transfer rate or computational ability increase. The results are also plotted in Figure 92 and Figure 93.

Data Rate / Instr./Sec	100 Mb/s	1 Gb/s	10 Gb/s	100 Gb/s
10MIPS	67235.48	5330.02	362.83	156.80
100MIPS	41437.43	6426.74	497.08	28.67
1GIPS		6282.90	323.74	115.05
10GIPS		3448.18	412.88	82.99
100GIPS				

(a) Data size= 100 KB

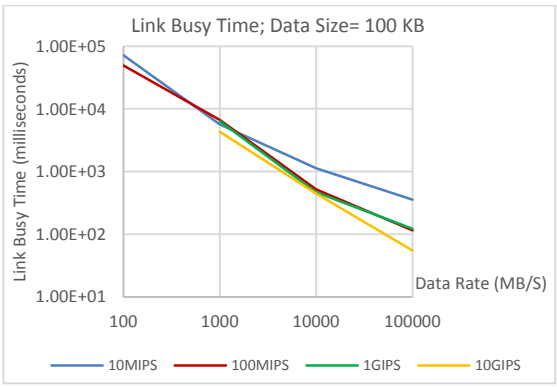
Data Rate / Instr./Sec	100 Mb/s	1Gb/s	10 Gb/s	100 Gb/s
10MIPS	391947.87	32019.35	390.41	286.66
100MIPS	615012.11	15020.16	3040.84	50.38
1GIPS		156112.66	2962.68	108.05
10GIPS		22306.67	5600.73	587.89
100GIPS				

(b) Data size= 1 MB

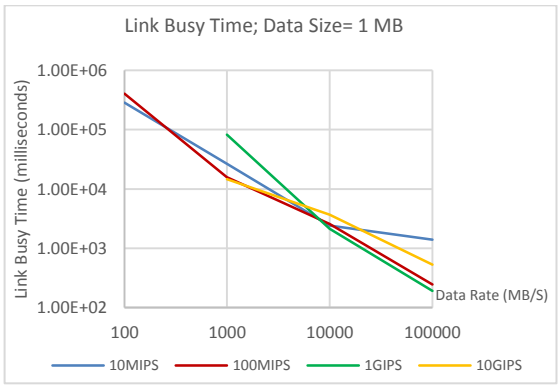
Data Rate / Instr./Sec	100 Mb/s	1 Gb/s	10 Gb/s	100 Gb/s
10MIPS	10928436.25	136347.72	1349.66	187.93
100MIPS	4565301.95	1280091.17	34766.51	146.27
1GIPS		344539.15	17038.94	2425.33
10GIPS				1821.37

(c) Data size= 10 MB

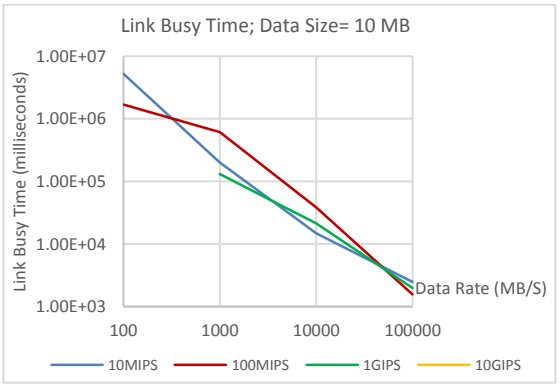
Table 29: Link Wait Time (mSec) measured in a set of FFTTMs on a network of 2000 nodes with different network attributes. Data sizes varies: (a) 100 KB, (b) 1MB and (c) 10MB.



(a) Data size= 100 KB

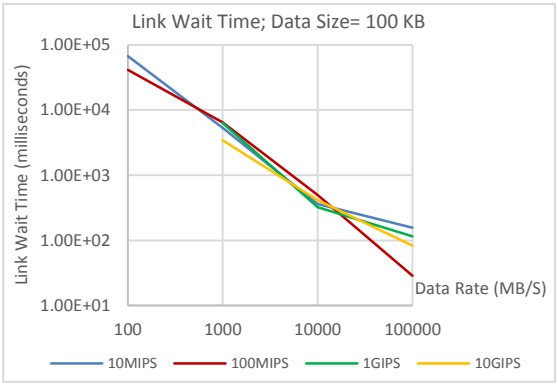


(b) Data size= 1 MB

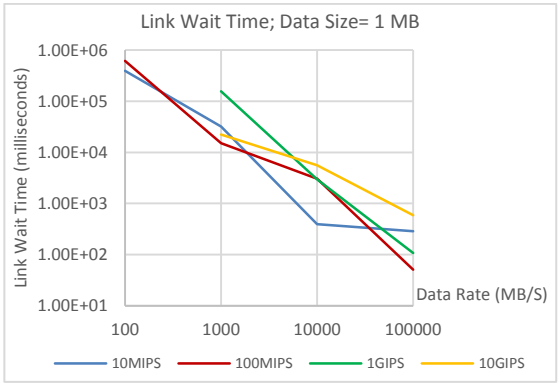


(c) Data size= 10 MB

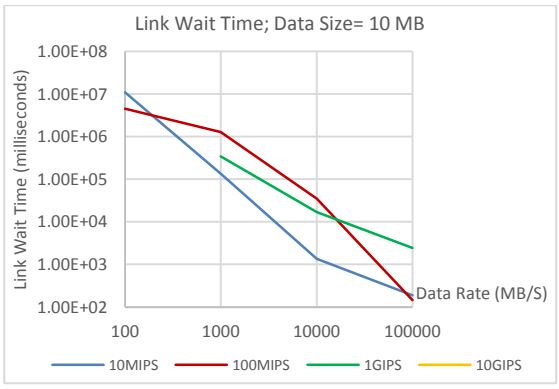
Figure 92: OLBT (in mSec) for FFTTM on 2000 nodes with different network attributes.



(a) Data size= 100 KB



(b) Data size= 1 MB



(c) Data size= 10 MB

Figure 93: Link Wait Time (in mSec) for FFTTM on 2000 nodes with different network attributes.

9.3) SPTM on 1000 Nodes

Earlier in this thesis it was discussed that the main difference between the two task-models is their degree of dependency between nodes. In a divide-and-conquer task like FFTTM, tasks on nodes depend on those of many other nodes while in an SPTM a task running on a node only depends on tasks running on some of its direct neighbours. The discussion about the degree of dependency leads to a conclusion that the network size does not affect the performance of a network when running SPTM. This assumption is tested against experimental results in this section and the next section. Table 30 shows the processor utilities on a network of size 1000 nodes all running SPTM.

The total data size has four different values: 800 KB, 8 MB, 80 MB and 800 MB. These values are chosen to match with the total size of data in the first two series of experiments (i.e. 8 FFTTM workloads of sizes 100 KB, 1 MB, 10 MB and 100 MB).

Data Rate / Instr./Sec	100 Mb/s	1 Gb/s	10 Gb/s	100 Gb/s
10MIPS	6.65	35.83	55.53	54.38
100MIPS	0.79	3.20	38.50	56.11
1GIPS	0.07	1.09	1.50	37.81
10GIPS	0.01	0.07	0.76	3.12
100GIPS		0.01	0.08	0.06

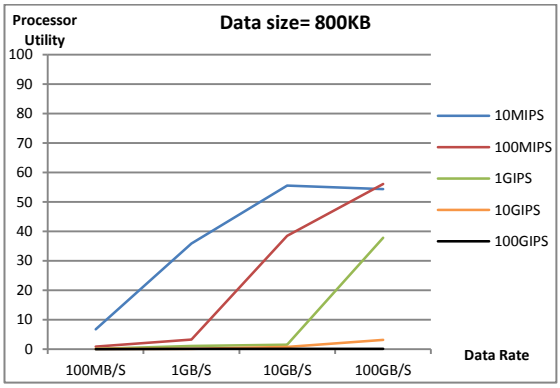
(a) Data size= 800 KB

Data Rate / Instr./Sec	100 Mb/s	1Gb/s	10 Gb/s	100 Gb/s
10MIPS	2.02	62.84	74.62	80.90
100MIPS	2.2	13.92	62.68	74.43
1GIPS	0.21	2.22	3.77	62.73
10GIPS	0.02	0.19	2.62	9.51
100GIPS		0.02	0.20	2.24

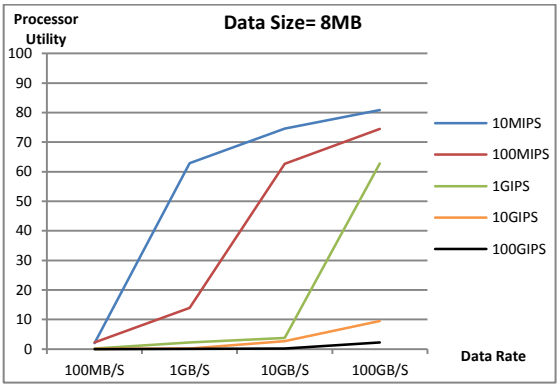
(b) Data size= 8 MB

Data Rate / Instr./Sec	100 Mb/s	1 Gb/s	10 Gb/s	100 Gb/s	100 Mb/s	1Gb/s	10 Gb/s	100 Gb/s
10MIPS	22.27	61.02	82.06	84.16	23.78	67.17	82.49	84.42
100MIPS	2.72	25.30	66.69	81.27	0.08	23.85	65.54	82.08
1GIPS	0.27	2.55	20.33	59.36		2.71	23.68	58.98
10GIPS		0.27	2.96	22.95			2.89	20.80
100GIPS			0.27	2.60				2.65
	(c) Data size= 80 MB				**(d) Data size= 800 MB**			

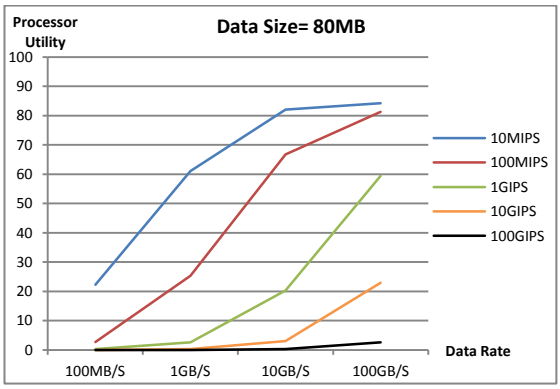
Table 30: Processor Utility measured in a set of experiments with SPTM on a network of 1000 nodes with different network attributes. Data sizes varies: (a) 800 KB, (b) 8MB, (c) 80MB and (d) 800MB.



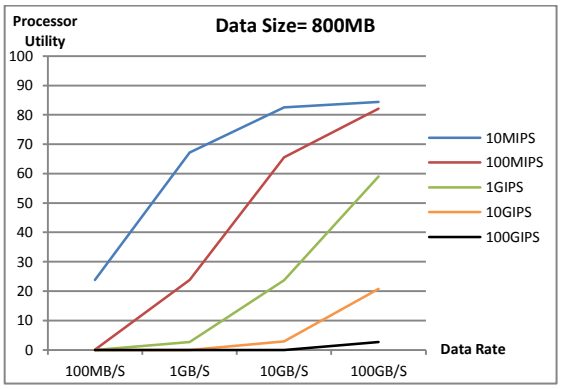
(a)



(b)



(c)



(d)

Figure 94: Processor Utility of Simple Parallel tasks on a 1000-node network with different sizes: (a) 800KB, (b) 8MB, (c) 80MB and (d) 800MB. Data sizes varies: (a) 100 KB, (b) 1MB, (c) 10MB and (d) 100MB.

In an SPTM all nodes are involved in sending and receiving packets. Therefore, in order to match the total data sizes in the first two series of tests in a network of size 1000 nodes each node needs to send a total data of size of 800 B, 8 KB, 80 KB and 800 KB. Like other sections, the comm/comp ratio is important in the network performance. Measurements of OLBT and OLWT metrics are also included in this section (Table 31 and Table 32). The network parameters are chosen from Table 23.

The metrics are measured in mSec to cancel out the effect of variation in time periods represented by a single simulation iteration in different tests. The results show that like FFTTM, link-related metrics decrease with an increase in either transfer rate or computational ability. The results are graphically presented in Figure 95 and Figure 96.

Instr./Sec \ Data Rate	100 Mb/s	1 Gb/s	10 Gb/s	100 Gb/s	100 Mb/s	1Gb/s	10 Gb/s	100 Gb/s
10MIPS	309.03	51.18	19.59	17.75	2072.15	470.31	242.17	227.59
100MIPS	316.62	31.04	5.08	1.95	1562.87	193.92	46.45	24.13
1GIPS	321.44	31.84	3.19	0.51	1807.97	157.92	19.61	4.55
10GIPS	321.98	32.13	3.18	0.31	1881.18	184.66	15.87	1.96
100GIPS	320.74	32.64	3.20	0.35		186.30	18.41	1.60

(a) Data size= 100 KB **(b) Data size= 1 MB**

Instr./Sec \ Data Rate	100 Mb/s	1 Gb/s	10 Gb/s	100 Gb/s	100 Mb/s	1 Gb/s	10 Gb/s	100 Gb/s
10MIPS	16949.58	4266.82	2448.16	2304.63	160306.46	41389.33	24296.26	23567.16
100MIPS	14050.83	1696.12	431.80	245.69	150758.87	16050.13	4116.93	2423.40
1GIPS	13627.58	1413.75	168.61	43.46		13674.94	1583.69	405.94
10GIPS		1349.96	141.71	16.76			1335.75	159.85
100GIPS			133.84	14.33				135.73

(c) Data size= 10 MB **(d) Data size= 100MB**

Table 31: OLBT (in mSec) measured in a set of SPTMs on a network of 1000 nodes with different network attributes. Data sizes varies: (a) 100 KB, (b) 1MB, (c) 10MB and (d) 100MB.

Instr./Sec \ Data Rate	100 Mb/s	1 Gb/s	10 Gb/s	100 Gb/s	100 Mb/s	1Gb/s	10 Gb/s	100 Gb/s
10MIPS	180.49	13.59	0.30	0.08	1207.23	150.41	6.15	1.23
100MIPS	245.21	18.32	1.36	0.03	937.38	101.43	14.75	0.62
1GIPS	259.39	24.20	1.96	0.15	1076.00	96.48	10.18	1.43
10GIPS	259.74	26.00	2.40	0.19	1162.42	111.73	9.54	1.04
100GIPS	257.68	26.34	2.53	0.29		114.17	11.23	0.98

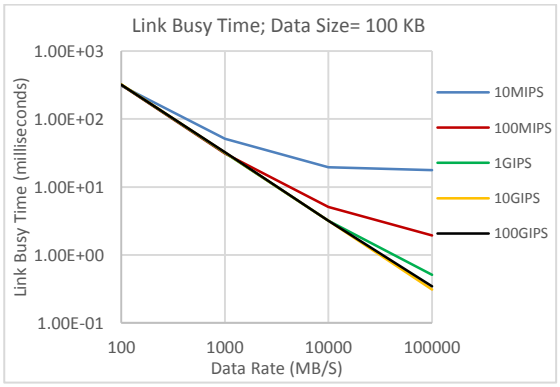
(a) Data size= 100 KB **(b) Data size= 1 MB**

Instr./Sec \ Data Rate	100 Mb/s	1 Gb/s	10 Gb/s	100 Gb/s	100 Mb/s	1 Gb/s	10 Gb/s	100 Gb/s
10MIPS	10070.12	1515.57	81.00	12.92	101311.41	15804.40	1038.42	148.33
100MIPS	8982.31	989.70	156.74	8.31	112739.08	9987.68	1544.66	99.91
1GIPS	9520.80	907.98	98.54	15.82		9251.42	991.38	154.46
10GIPS		930.00	91.94	9.87			910.07	101.79
100GIPS			91.85	9.35				92.09

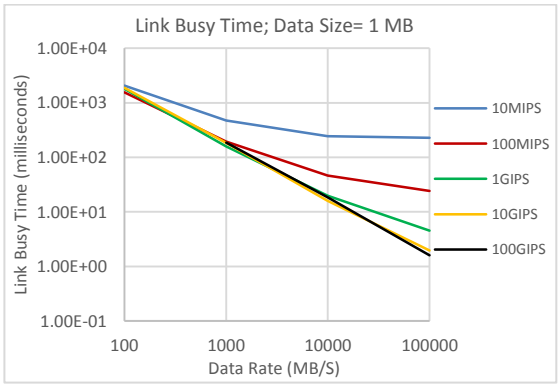
(c) Data size= 10 MB **(d) Data size= 100MB**

Table 32: Link Wait Time (mSec) measured in a set of SPTMs on a network of 1000 nodes with different network attributes. Data sizes varies: (a) 100 KB, (b) 1MB, (c) 10MB and (d) 100MB.

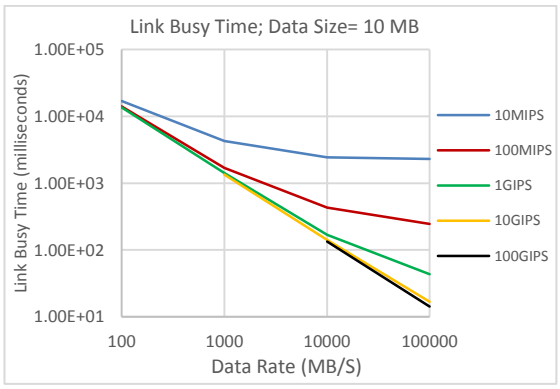
The slope of graphs plotted in link-related metrics differ from case to case. At the moment explaining those differences and analysing their reason are not part of the thesis. This can be a subject of further research on this field.



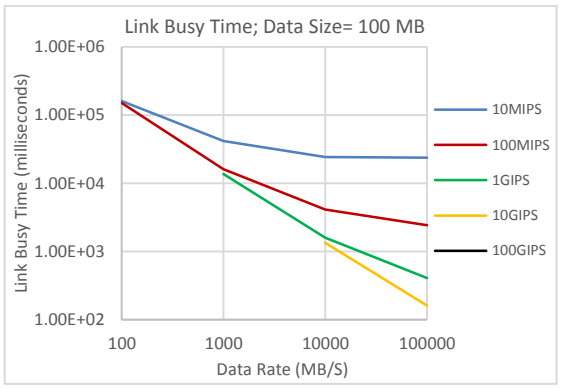
(a) Data size= 100 KB



(b) Data size= 1 MB

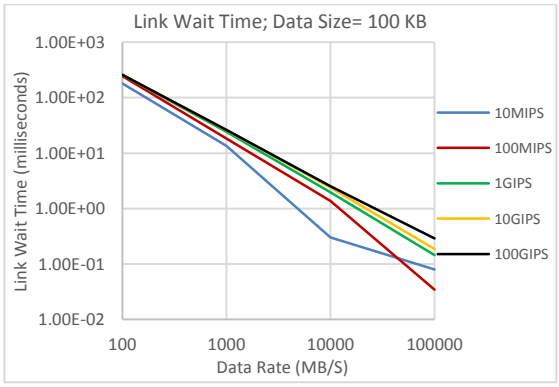


(c) Data size= 10 MB

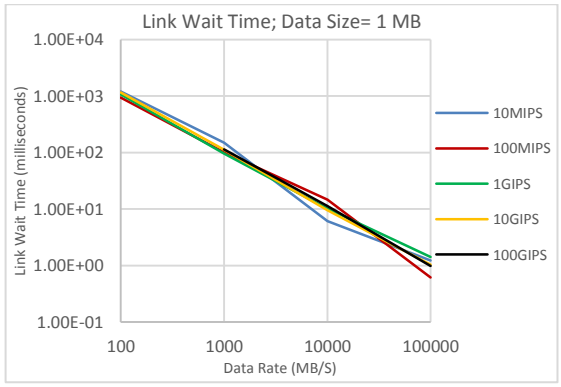


(d) Data size= 100 MB

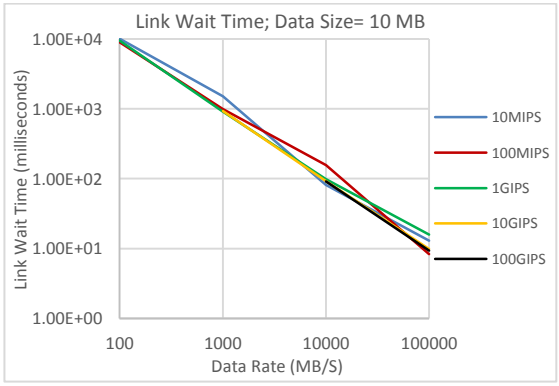
Figure 95: OLBT (in mSec) for SPTM on 1000 nodes with different network attributes.



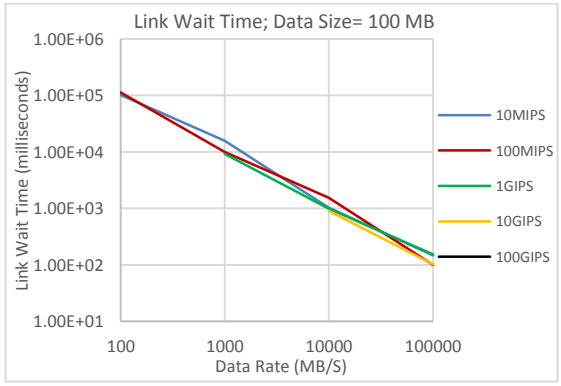
(a) Data size= 100 KB



(b) Data size= 1 MB



(c) Data size= 10 MB



(d) Data size= 100 MB

Figure 96: Link Wait Time (in mSec) for SPTM on 1000 nodes with different network attributes.

9.4) SPTM on 2000 Nodes

The nature of SPTM implies that the size of the network has no effect on its performance. To verify this assumption a final set of experiments are running in which a simulated BC of size 2000 nodes (precisely 12*13*14=2002 nodes) all running SPTM. The test is repeated 80 times with different values chosen from Table 23. The processor utility can be seen in Table 33.

Figure 97 plots the data in Table 33. The table and the figure show that the difference between a network of size 1000 nodes and a network of size 2002 nodes is less than 1.5% in almost all the cases which cannot be regarded as a meaningful difference.

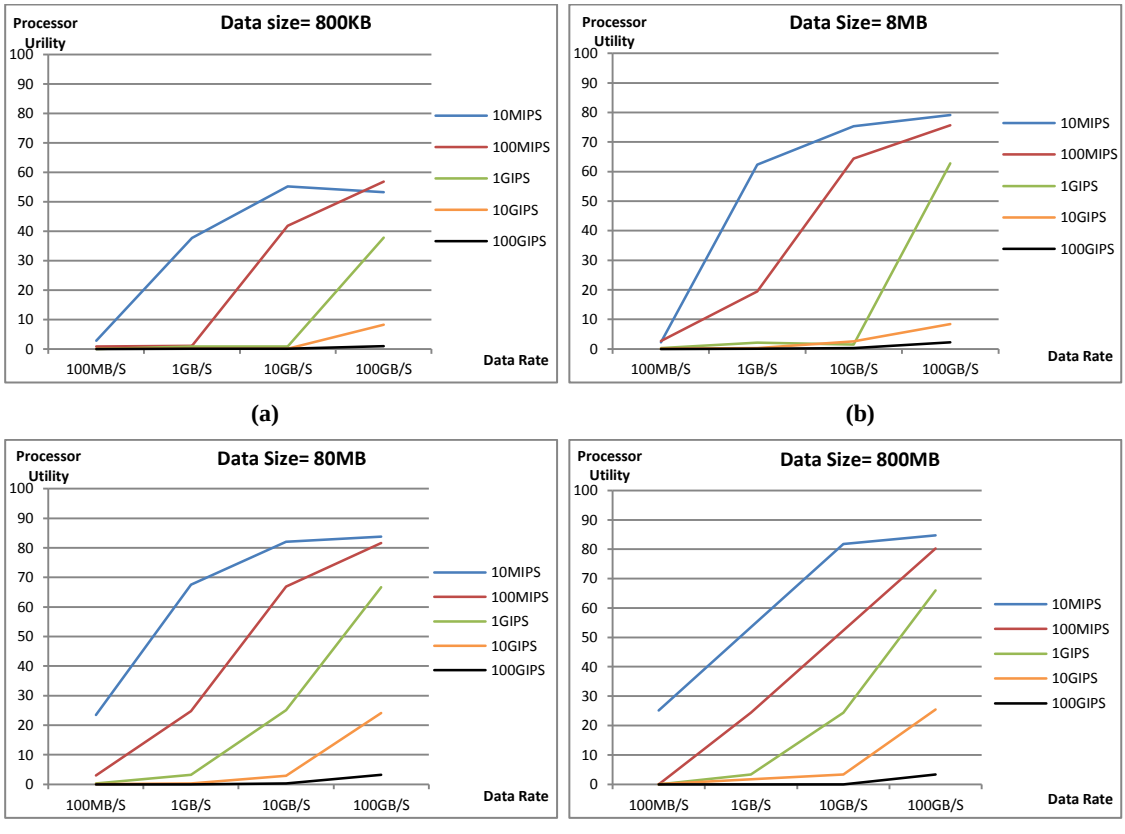
Data Rate / Instr./Sec	100 Mb/s	1 Gb/s	10 Gb/s	100 Gb/s	100 Mb/s	1Gb/s	10 Gb/s	100 Gb/s
10MIPS	2.75	37.74	55.18	53.25	2.25	62.36	75.36	79.07
100MIPS	0.80	1.042	41.77	56.86	2.66	19.50	64.41	75.67
1GIPS		0.80	0.77	37.75	0.25	2.08	1.51	62.79
10GIPS	0.01	0.09	0.048	8.20	0.02	0.24	2.51	8.41
100GIPS		0.01	0.08	0.93		0.02	0.23	2.24

(a) Data size= 800 KB **(b) Data size= 8 MB**

Data Rate / Instr./Sec	100 Mb/s	1 Gb/s	10 Gb/s	100 Gb/s	100 Mb/s	1Gb/s	10 Gb/s	100 Gb/s
10MIPS	23.44	67.50	82.02	83.75	25.13		81.77	84.70
100MIPS	3.03	24.71	66.84	81.60		24.37		80.24
1GIPS	0.29	3.16	25.13	66.69		3.33	24.39	65.96
10GIPS	0.03	0.29	2.93	24.08	0.04		3.33	25.48
100GIPS			0.30	3.15				3.31

(c) Data size= 80 MB **(d) Data size= 800 MB**

Table 33: Processor Utility measured in a set of experiments with SPTM on a network of 2000 nodes with different network attributes. Data sizes varies: (a) 800 KB, (b) 8MB, (c) 80MB and (d) 800MB.



(a) **(b)**

(c) **(d)**

Figure 97: Processor Utility of SPTM on a 2002-node network with different sizes: (a) 800KB, (b) 8MB, (c) 80MB and (d) 800MB.

Data Rate / Instr./Sec	100 Mb/s	1 Gb/s	10 Gb/s	100 Gb/s	100 Mb/s	1Gb/s	10 Gb/s	100 Gb/s
10MIPS	644.70	103.02	39.78		4075.54	951.01		459.47
100MIPS	664.44		10.37	3.94		395.23		48.32
1GIPS		66.55	6.60	1.03		321.67	40.54	9.43
10GIPS	667.40	66.68	6.83	0.64	3788.90	377.43	31.91	3.98
100GIPS	667.44	66.95	6.64	0.66		381.22	37.25	3.19

(a) Data size= 100 KB **(b) Data size= 1 MB**

Data Rate / Instr./Sec	100 Mb/s	1 Gb/s	10 Gb/s	100 Gb/s	1 Gb/s	10 Gb/s	100 Gb/s
10MIPS	34347.25	8675.91	4908.76	4651.91		48910.40	47145.57
100MIPS	28415.88	3405.85	874.18	495.12	32293.55		4868.80
1GIPS	26842.49	2834.96	341.12	86.67	27174.87	3175.08	823.61
10GIPS	32223.97	2695.53	284.19	34.12		2713.06	321.62
100GIPS			269.72	28.35			273.12

(c) Data size= 10 MB **(d) Data size= 100MB**

Table 34: OLBT (in mSec) measured in a set of SPTMs on a network of 2000 nodes with different network attributes. Data sizes varies: (a) 100 KB, (b) 1MB, (c) 10MB and (d) 100MB.

This is mainly due to the very limited dependency range in SPTM. In other words, nodes do not care about any nodes other than their direct neighbours. The experimental results confirm the pre-experiment assumption of independence of SPTM from the network size.

Measurements of OLBT and OLWT metrics are also included in this section (Table 34 and Table 35). The network parameters are chosen same as SPTM for 1K nodes.

Again, the metrics are measured in mSec rather than simulation iterations.

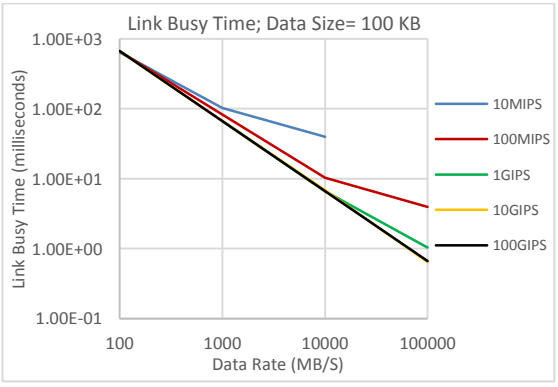
Data Rate / Instr./Sec	100 Mb/s	1 Gb/s	10 Gb/s	100 Gb/s	100 Mb/s	1Gb/s	10 Gb/s	100 Gb/s
10MIPS	357.86	25.10	0.61		2109.95	283.95		2.07
100MIPS	477.12		2.54	0.06		197.44		1.05
1GIPS		47.87	3.80	0.25		187.78	20.82	2.84
10GIPS	498.59	50.05	5.05	0.35	2218.89	219.21	18.59	2.00
100GIPS	499.83	50.09	4.99	0.48		224.52	21.71	1.87

(a) Data size= 100 KB **(b) Data size= 1 MB**

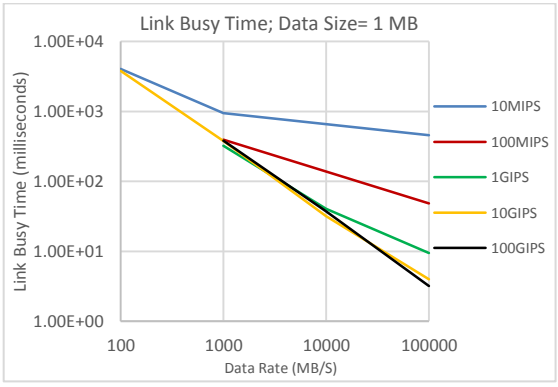
Data Rate / Instr./Sec	100 Mb/s	1 Gb/s	10 Gb/s	100 Gb/s	1 Gb/s	10 Gb/s	100 Gb/s
10MIPS	19508.58	2920.90	140.81	21.89		1810.51	247.10
100MIPS	17980.85	1928.87	294.21	14.11	20043.84		174.05
1GIPS	18050.63	1786.42	193.69	29.50	18319.78	1907.19	296.24
10GIPS	20889.94	1822.18	179.39	19.51		1826.65	196.52
100GIPS			180.11	17.79			187.63

(c) Data size= 10 MB **(d) Data size= 100MB**

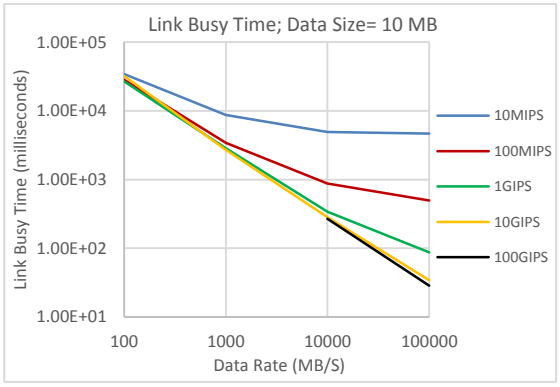
Table 35: Link Wait Time (mSec) measured in a set of SPTMs on a network of 2000 nodes with different network attributes. Data sizes varies: (a) 100 KB, (b) 1MB, (c) 10MB and (d) 100MB.



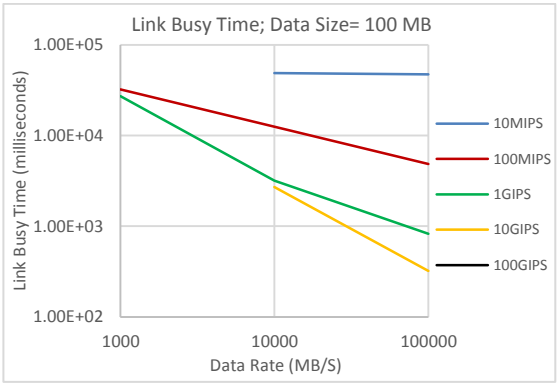
(a) Data size= 100 KB



(b) Data size= 1 MB



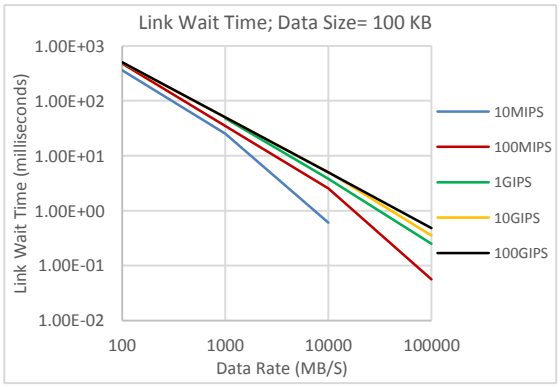
(c) Data size= 10 MB



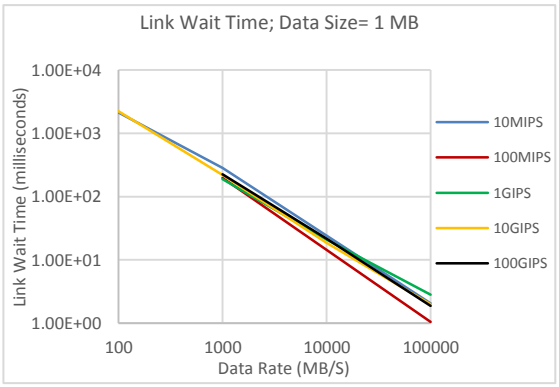
(d) Data size= 100 MB

Figure 98: OLBT (in mSec) for SPTM on 2000 nodes with different network attributes.

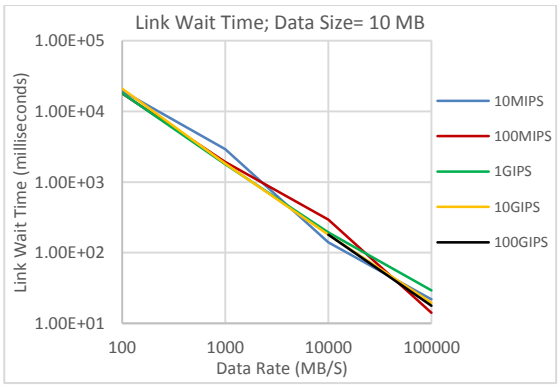
The results show that like 1K-node networks, both link-related metrics decrease (with different slopes) with an increase in either transfer rate or computational ability in SPTM on a 2K-node networks. Graphical representation of the results are also included in this thesis (Figure 98 and Figure 99).



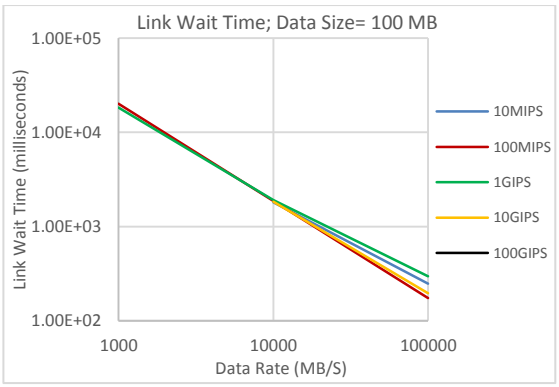
(a) Data size= 100 KB



(b) Data size= 1 MB



(c) Data size= 10 MB



(d) Data size= 100 MB

Figure 99: Link Wait Time (in mSec) for SPTM on 2000 nodes with different network attributes.

9.5) Network Size

The main question in this section is if the size of the network affects its behaviour. The results presented in previous sections are used to study both FFTTM and SPTM. The data in sections 9.1 and 9.2 is rearranged and some of them are plotted in Figure 100 which confirms classical literature in parallel computing (e.g. Figure 3). Some data in

aforementioned tables are too low in value to be used in this comparison. The data plotted below shows that when running an FFTTM for all network attribute values the performance of a 1000-node network is better than that of a 2000-node network.

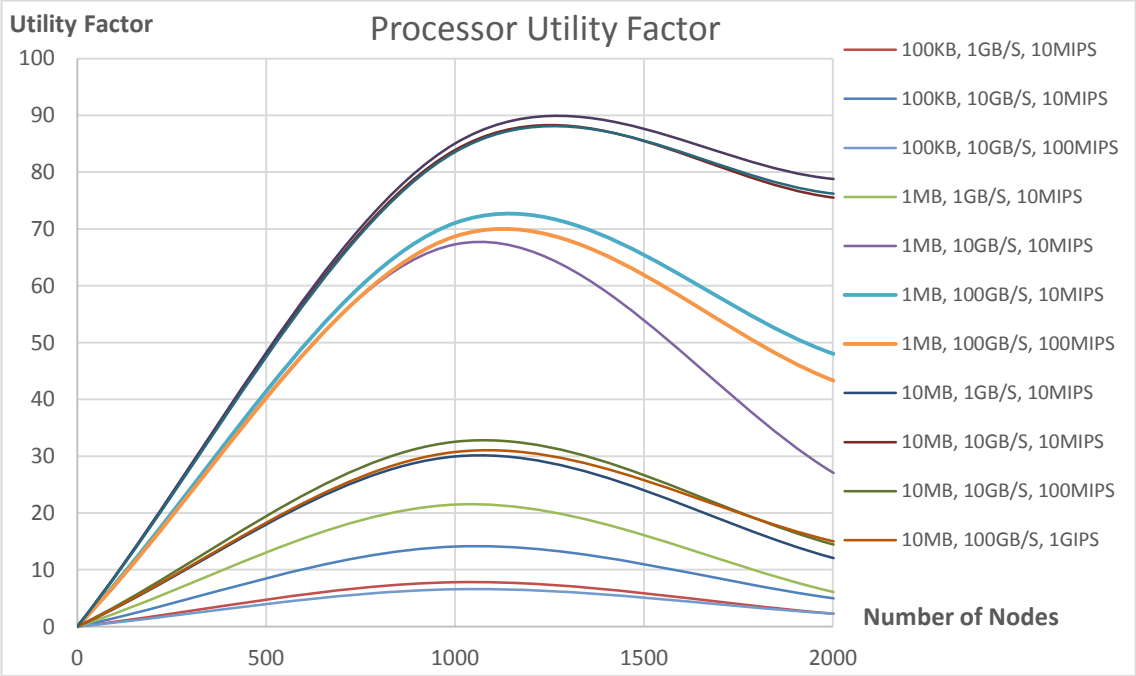


Figure 100: The effect of network size on the performance of an FFTTM

The same comparison for an SPTM brings completely different results. The experiment results show that when a network runs an SPTM the performance of the network does not relate with its size. This is not a surprise result because a node with a task-model like SPTM does not build a large network of task dependencies and therefore the execution time of SPTM on a node only depends on its direct neighbours rather than the whole network.

9.6) Transfer Rate

Transfer rate influences the performance of both FFTTM and SPTM for different reasons. In SPTM the experiment results presented in Figure 94 show that when the transfer rate is higher the network processor utility is higher just because the data propagates faster between nodes. This fact can be clearly observed is all graphs plotted in Figure 94.

But with FFTTM the link's data rate has a different influence on the network performance. A task like an FFT operates on the rule that if there are high chances to finish the task remotely in parallel rather than sequentially and locally then the task will be divided into smaller parts and sent to other nodes for a parallel execution of FFT. This means that when the data transfer rate is high, nodes are more likely to lean to

share the task with their neighbours which consequently increases the degree of parallelism. All graphs plotted in Figure 88 and Figure 91 confirm this assumption.

It should be noticed that in both task-models the best network utilities are achieved when the data rates are very high (10 Gb/s and 100 Gb/s). Such data rates are not available in off-the-shelf products. Even just a few scientific researches on wireless technologies yield such a high transfer rate. This can cast doubt over an imminent possibility of building a BC with high processor utility. A lower data rate like 1 Gb/s is more realistic at the moment. This means that based on the data presented in Table 24 and Table 27 the best performance for an FFTTM with a 1 Gb/s data link is below 42% and 18% respectively. Such a number for SPTM with a 1 Gb/s is less than 62% according to Table 30. But as data rates of range 10 Gb/s are anticipated in near future (and even higher data rates in a longer perspective), the network performances of an FFTTM can improve to less than 84% and 77% (according to Table 24 and Table 27). Such a data link also enhances the performance of an SPTM to less than 83%. As a conclusion, as far as it is concerned with data rates, although the current data rate of wireless links are promising but to achieve processor utility comparable to current modern parallel computers, radio links need just a little bit of more improvement in their data rates.

9.7) Computational Ability

The effect of number of instructions a member of BC wireless network can execute per second on the processor utility is very interesting. In the first glance the higher the computational ability of nodes the higher the parallel capability. But the results shown in previous sections doubts such a sharp verdict. Figure 101 shows that the processor utility of a network of size 1000 nodes decreases in running FFTTM when the computation capability of nodes of a network increases. This can happen because the tasks that are supposed to be executed in parallel are now running tasks sequentially because of high number of instructions executed by processors and relatively low data rates of radio links. In other words, when a node sends a packet to a neighbour there are chances that during a second packet transmission the result in the first neighbour is ready to return to the original node. This ends up in a semi-sequential execution of tasks. The same trend is shared by a network of 2000 nodes running the same task-model and a 1000- and 2000-node network running an SPTM.

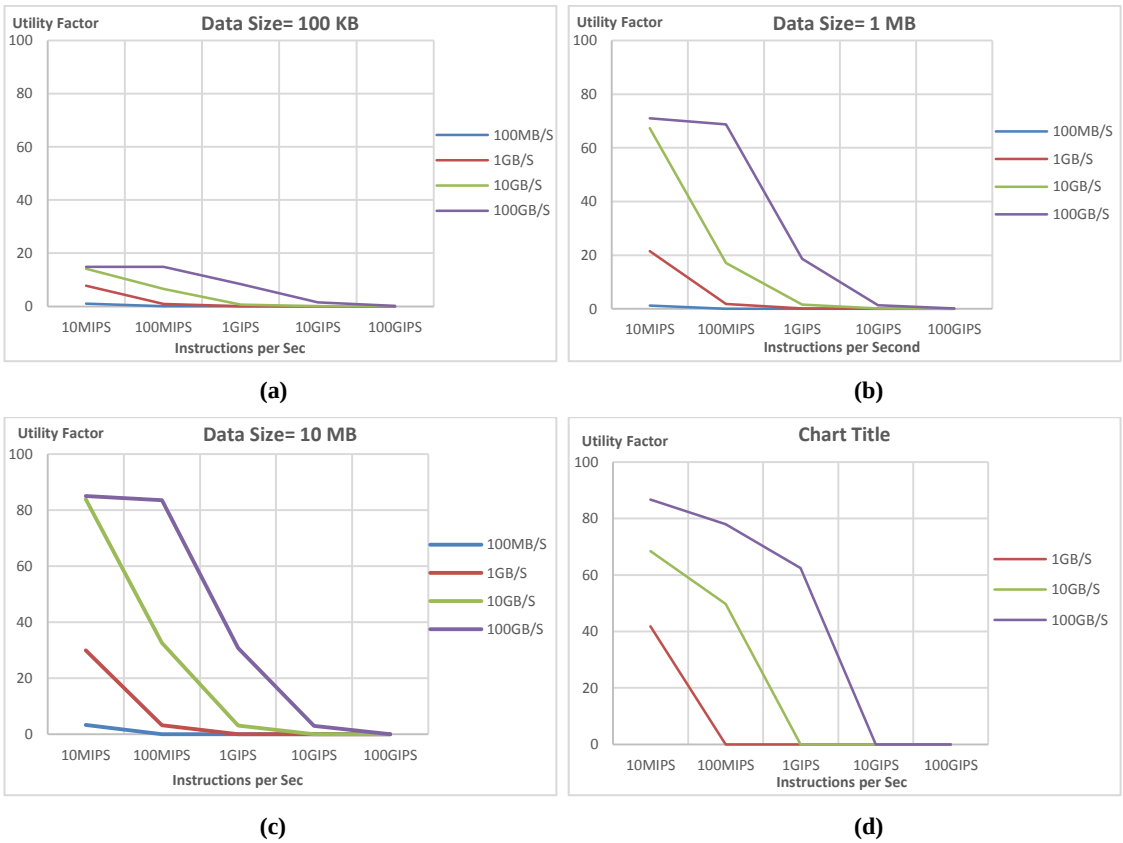


Figure 101: Processor utility of a network of size 1000 nodes running FFT task-model when the computation capability changes.

As expected, the comm/comp ratio has a more significant role than the data rate and computational ability separately. It can be seen in all tables and figures presented in this chapter that the absolute computational ability does not play a major role in a high processor utility; instead, it is the ratio of the computational ability to the data rate that decides how high the processor utility of a BC network is.

It is observed in all tables of this chapter that when the communication to computation ratio of two test results are the same the network utilities are almost unchanged. The results show that the processor utility in BC platform is proportionate to the comm/comp ratio. Therefore, to have a network of high processor utility it is better to choose processors and radio modules so that the aforementioned ratio is maximised.

As a consequence, with a higher margin on the radio data rates, an increase in the overall processing ability of a parallel computer can be achieved by increasing the number of processors rather than pushing for increasing the computational ability of processors. This guarantees the comm/comp ratio is high enough for an HPC machine.

The role of comm/comp ratio can be easily observed when the data in Table 24 and Table 27 are rearranged based on their communication to computation ratio. The results can be seen in Table 36 and Table 37. This observation is not restricted to FFTTM only.

Results derived from an SPTM have the same characteristics (see Table 30 and Table 33).

Comm/Comp / Data Rate	0.1	1	10	100	1000	10000	1	10	100	1000	10000
10MIPS			1.08	7.81	14.09	14.90		1.21	21.48	67.29	71.06
100MIPS		0.11	0.95	6.57	14.93		0.10	1.80	17.08	68.69	
1GIPS	0.01	0.12	0.64	8.41			0.11	1.55	18.60		
10GIPS	0.01	0.09	1.47				0.20	1.38			
100GIPS	0.01	0.13					0.22				

(a) Data Size= 100 KB **(b) Data Size= 1 MB**

Comm/Comp / Data Rate	10	100	1000	10000	100	1000	10000
10MIPS	3.31	29.95	83.90	85.04	41.83	68.40	86.72
100MIPS	3.23	32.54	83.54		49.75	77.98	
1GIPS	3.06	30.74			62.46		
10GIPS	2.96						
100GIPS							

(c) Data Size= 10 MB **(d) Data Size= 100 MB**

Table 36: Processor Utility (%) measured for a network of size 1000 nodes running an FFTTM.

Comm/Comp / Data Rate	1	10	100	1000	10000	1	10	100	1000	10000
10MIPS			2.26	4.97	4.73		0.40	6.08	27.05	48.01
100MIPS	0.03	1.15	2.27	4.75				7.85	43.31	
1GIPS		0.16	2.23			0.09	0.60	7.14		
10GIPS		0.21					0.47			
100GIPS	0.02					0.12				

(a) Data Size= 100 KB **(b) Data Size= 1 MB**

Comm/Comp / Data Rate	10	100	1000	10000	100	1000	10000
10MIPS	1.49	12.05	75.52	78.80	17.56	76.05	82.35
100MIPS	1.14	14.44	76.21		16.53	78.48	
1GIPS	1.15	15.00			19.64		
10GIPS	1.22						

(c) Data Size= 10 MB **(d) Data Size= 100 MB**

Table 37: Processor Utility (%) measured for a network of size 2000 nodes running an FFTTM.

9.8) Data Size

Tables and figures in this chapter show that in almost all the cases the size of the real data in a task-model is proportionate to the processor and network utilisation. It is much expected as with each bulk of real data in a packet there are some extra bytes used to guarantee a safe and secure stream of packets. This extra data (packet overhead) usually has a constant size.

Given a constant size of packet overhead, the best network performance will be achieved when the size of real data sent by each packet is as large as possible. In other

words, when the size of real data is small a major part of the packet is occupied by extra data (packet overhead) which is apparently not an efficient way of data transmission. In case of sending a large size of data in a single packet the packet overhead is just a small part of the actual packet sent which consequently increases the transmission efficiency which in its turn contributes in increasing the network and processor utilisation. The experimental results derived from both task-models with different network attributes confirm this theoretical conclusion.

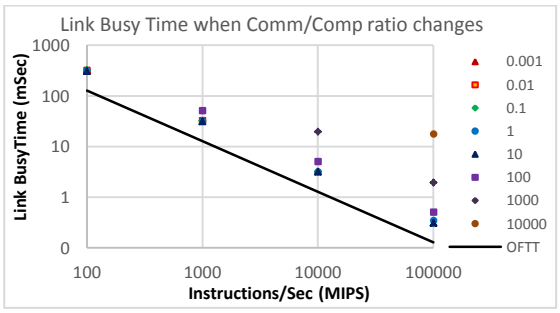
9.9) Link Busy Time

OLBT is introduced in this thesis as one of the metrics to measure the performance of the wireless links in a BC platform. The experiment results presented in sections 9.1 to 9.4 include OLBT measurements. To the knowledge of the author of this thesis there is no similar wireless platform to compare BC's results with; instead, these results can be compared to an ideal situation in which no transmission overhead exists. The transmission overheads include packet's header and footer, network discovery packets, network management packets, acknowledgment packets and retransmission of packets due to acknowledgment timer expire. This ideal situation is called *Overhead-Free* situation and for each node consists of:

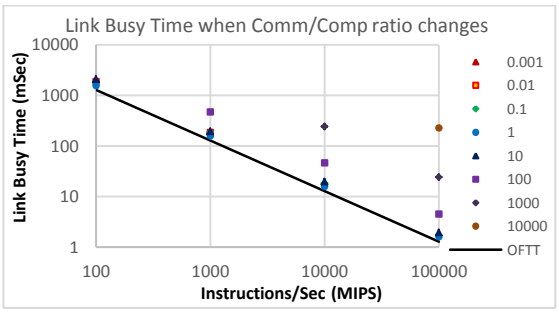
- For SPTM: Six packets of data from the node to its neighbours and six result packets. Each of these packets have the same fixed size.
- For FFTTM: Six packets to send task code to its neighbours; six data packets to selected neighbours and six result packets from those neighbours. The size of all packets are fixed and the size of data and result packets are the same.

All the network parameters and data sizes are known before running a simulation; therefore, the Overhead-Free transmission time (OFTT) can be determined (which may vary from experiment to experiment). Comparing the experimental measures for OLBT with OFTT can be a good measure for the performance of the network. OLBT and OFTT are compared for different task-models and network parameters. Figure 102 plots OLBT and OFTT in a network of 1K nodes running SPTM under different network parameters. Regarding the role comm/comp ratio plays in determining OFTT, the data points are separated based on this ratio. A similar results are measured with a 2K-node network (Figure 103). In both sets of experiments the total data size is 100KB, 1MB, 10MB and 100MB. The difference between the actual and ideal cases (OLBT and OFTT) are bigger with smaller data sizes. With large data sizes packet headers and ACK packets are smaller compared to the real data; therefore, the results are closer to

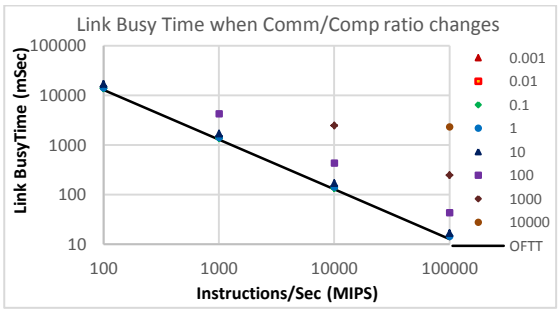
the ideal situation in parts c and d of both figures. Also when the computational ability is higher the difference between OLBT and OFTT is bigger. All times in this section are in milliseconds rather than simulation iterations (which is a variable unit).



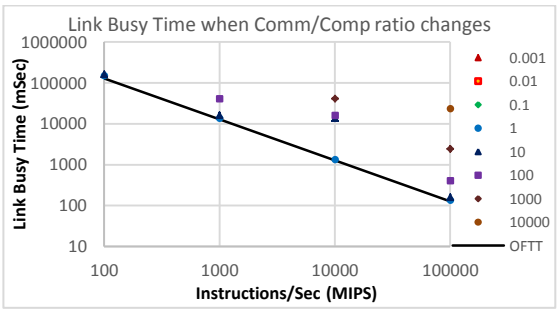
(a) Data Size= 100KB



(b) Data Size= 1MB

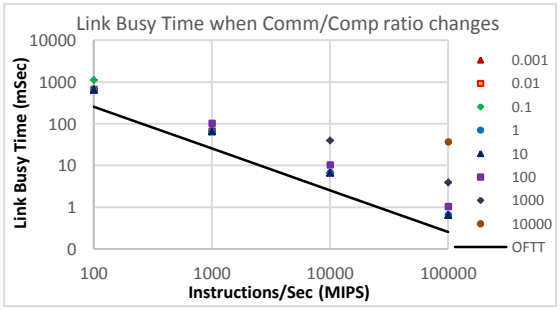


(c) Data Size= 10MB

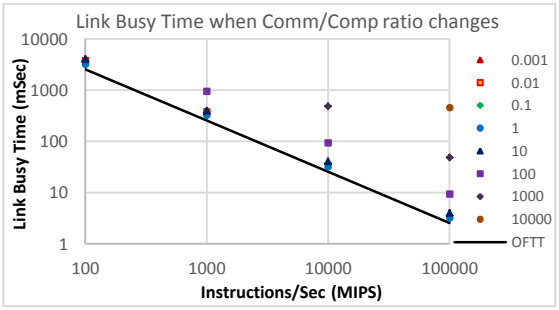


(d) Data Size= 100MB

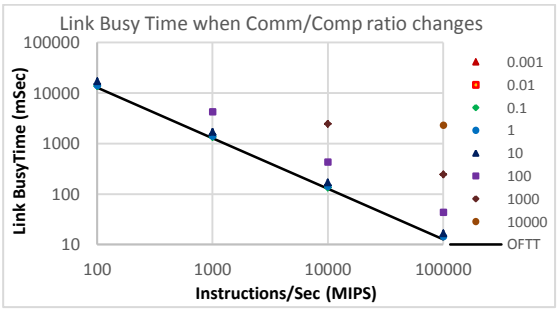
Figure 102: Comparison between OLBT and OFTT in a 1K-node network running SPTM



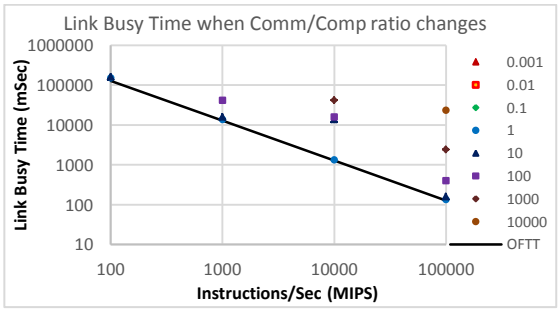
(a) Data Size= 100KB



(b) Data Size= 1MB



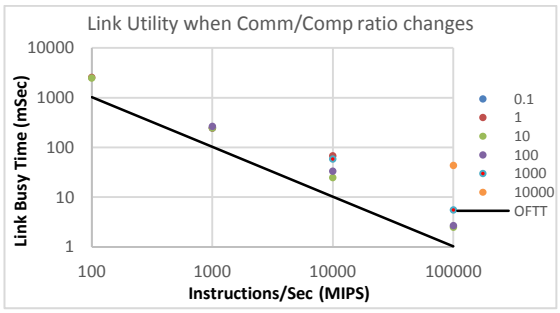
(c) Data Size= 10MB



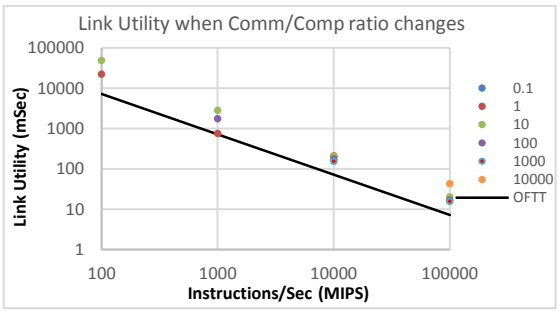
(d) Data Size= 100MB

Figure 103: Comparison between OLBT and OFTT in a 2K-node network running SPTM

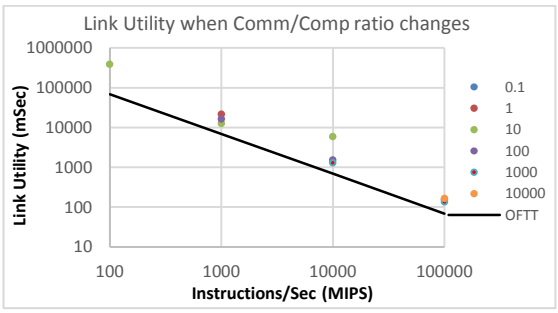
The same measurements are repeated with FFTTM on networks of size 1K and 2K nodes. The number of workloads are 8 to match the cubic form of the network containers.



(a) Data Size= 100KB

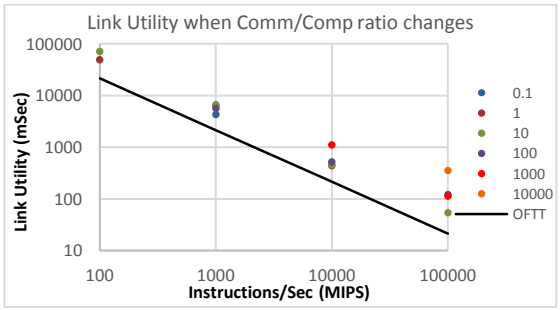


(b) Data Size= 1MB

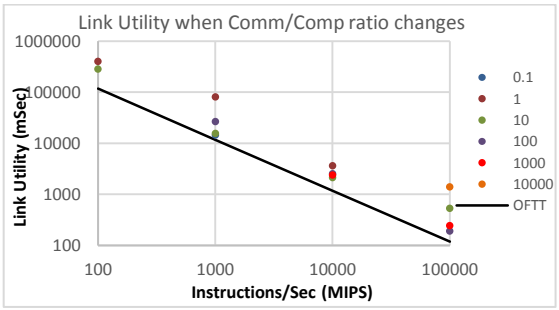


(c) Data Size= 10MB

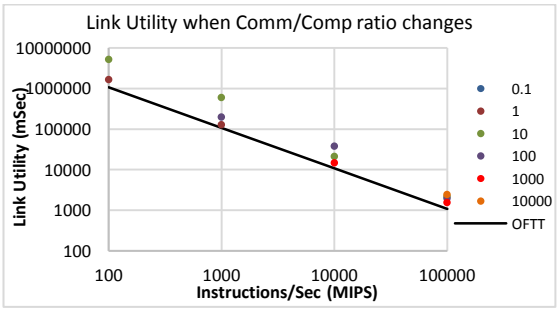
Figure 104: Comparison between OLBT and OFTT in a 1K-node network running 8 FFTTM workloads



(a) Data Size= 100KB



(b) Data Size= 1MB



(c) Data Size= 10MB

Figure 105: Comparison between OLBT and OFTT in a 2K-node network running 8 FFTTM workloads

The data sizes are 100KB, 1MB and 10MB. Results are plotted in Figure 104 and Figure 105 and similarities can be found between them and those from SPTM. The number of high OLBTs (compared to OFTTs) are slightly higher with FFTTM compared to SPTM. This can be because FFTTM is more susceptible to packet retransmission regarding its larger tree of dependencies. In both SPTM and FFTTM for some values for transfer rate, computational ability and data size OLBT and OFTT

values can be close which implies that the BC platform had handled communications efficiently in those cases.

9.10) Link Utility

Link utility is measured for both task-models under different values for network parameters. As mentioned before, link utility is an estimation for how much of a link's time during a simulation test is dedicated to communication (including both real data and overhead). This is an average figure and does not resemble any specific link's situation. All values in this section are expressed in percentages.

The first set of results (Table 38) are derived from the raw data presented in section 9.1 and concerns running 8 FFTTM workload on a network of 1K nodes. As expected the link utility is small when the comm/comp ratio is high. A high comm/comp ratio means that many bits of data can be transmitted over a small number of simulation iterations (an iteration is equal to a node's internal clock cycle). In a situation like this most of a node's time will be used for processing the tasks it has received and the actual transmission takes comparably shorter times. That is why the link utilisation is lower when comm/comp ratio is high. The data size is not playing a major role in link utility and this can be because a packet's processing time and communication time both relate to the packet size in FFTTM.

Data Rate / Instr./Sec	100 Mb/s	1 Gb/s	10 Gb/s	100 Gb/s	100 Mb/s	1 Gb/s	10 Gb/s	100 Gb/s
10MIPS	3.03	2.73	0.77	0.61	2.61	2.35	0.67	0.19
100MIPS	6.76	3.40	2.05	0.76	10.02	4.82	2.40	0.68
1GIPS		4.49	3.93	2.70		17.13	4.56	1.98
10GIPS		4.79	5.81	3.81		18.63	2.56	1.76

(a) Data Size= 100 KB **(b) Data Size= 1 MB**

Data Rate / Instr./Sec	100 Mb/s	1 Gb/s	10 Gb/s	100 Gb/s
10MIPS	5.14	2.31	0.26	0.03
100MIPS	2.02	3.25	0.83	0.26
1GIPS		3.37	2.36	1.11
10GIPS		14.92	6.32	1.76

(c) Data Size= 10 MB

Table 38: Average Link Utility (in percentages) of 8 FFTTMs running on a 1K-node network.

Table 39 shows how link utility changes in a network of 2K nodes running 8 FFTTM workloads. The range of results are almost the same as those of a network of size 1K. The same trend of reduction of link utility due to increase in comm/comp ratio can be seen in this set of results as well. It should be mentioned that any possible congestions occurring in communication hotspots may not be detected in these results since they are

only average results which means high volume of transmissions on a few hotspots can be covered by lower traffic over a vast majority of links. This imbalance between communication demands is intrinsic to FFTTM.

Data Rate / Instr./Sec	100 Mb/s	1 Gb/s	10 Gb/s	100 Gb/s	100 Mb/s	1 Gb/s	10 Gb/s	100 Gb/s
10MIPS	2.07	2.06	0.72	0.07	0.79	1.13	0.79	0.41
100MIPS	3.06	1.90	2.19	0.62	3.87	4.20	1.40	0.75
1GIPS		2.69	1.77	0.003		4.90	4.61	1.47
10GIPS		5.79	11.78	1.29		35.01	2.71	0.89

(a) Data Size= 100 KB **(b) Data Size= 1 MB**

Data Rate / Instr./Sec	100 Mb/s	1 Gb/s	10 Gb/s	100 Gb/s
10MIPS	2.66	1.23	0.58	0.10
100MIPS	8.81	5.56	1.17	0.60
1GIPS		7.87	1.24	2.38
10GIPS				1.37

(c) Data Size= 10 MB

Table 39: Average Link Utility (in percentages) of 8 FFTTMs running on a 2K-node network.

The same comparison is made for SPTM running on networks of size 1K and 2K nodes (Table 40 and Table 41). The results are based on raw data presented in sections 9.3 and 9.4. The first difference between FFTTM and SPTM results is the higher link utility values in SPTM compared to FFTTM. The range of changes with SPTM is also smaller compared to that of FFTTM. Another interesting point is how comm/comp ratio is related to link utility in SPTM. Despite a rather strong reverse effect of comm/comp ratio on link utility in FFTTM, no sign of such a strong influence can be detected with SPTM. Even in some cases link utility increases as comm/comp ratio increases which is in contrast to the observations with FFTTM.

When running SPTM it is expected to have less restrictions imposed by network attributes as each node only communicates with some of its direct neighbours. But in an FFTTM a rather large dependency tree (which in many experiments cover all the network) gives a chance to the network parameters to highly influence the task-model process.

This may play a role in a non-uniform response of an SPTM to variations in comm/comp ratio and other network parameters. More test and analysis are needed to have a more accurate explanation about different behaviour of SPTM and FFTTM in terms of link utility.

Data Rate / Instr./Sec	100 Mb/s	1 Gb/s	10 Gb/s	100 Gb/s	100 Mb/s	1 Gb/s	10 Gb/s	100 Gb/s
10MIPS	6.52	5.74	3.39	3.02	13.43	9.63	5.65	5.75

100MIPS	7.99	3.13	6.11	3.41	12.55	9.52	9.56	5.93
1GIPS	9.33	11.50	1.52	6.86	14.27	12.61	2.44	9.50
10GIPS	11.40	8.33	9.13	3.07	13.70	12.42	13.94	6.29
100GIPS	11.49	12.49	10.34	12.03		14.16	12.56	13.03

(a) Data Size= 100 KB **(b) Data Size= 1 MB**

Data Rate / Instr./Sec	**100 Mb/s**	**1 Gb/s**	**10 Gb/s**	**100 Gb/s**	**100 Mb/s**	**1 Gb/s**	**10 Gb/s**	**100 Gb/s**
10MIPS	12.63	8.69	3.17	3.03	13.32	9.33	3.18	3.11
100MIPS	6.74	7.01	4.72	3.15	6.88	6.53	4.55	3.16
1GIPS	6.05	6.32	5.83	4.26		6.51	6.67	4.04
10GIPS		6.04	7.09	6.56			6.80	5.78
100GIPS			6.29	6.43				6.60

(c) Data Size= 10 MB **(d) Data Size= 100MB**

Table 40: Average Link Utility (in percentages) of SPTM running on a 1K-node network.

Data Rate / Instr./Sec	**100 Mb/s**	**1 Gb/s**	**10 Gb/s**	**10 0Gb/s**	**100 Mb/s**	**1 Gb/s**	**10 Gb/s**	**100 Gb/s**
10MIPS	2.85	7.03	3.42	3.04	4.02	9.74	6.00	5.66
100MIPS	10.80	1.08	6.86	3.49	14.95	13.98	9.99	5.98
1GIPS	0.04	10.90	0.81	6.88	15.38	12.02	8.79	9.74
10GIPS	10.98	10.03	11.20	8.65	14.80	15.96	14.32	5.60
100GIPS	11.35	11.02	10.92	13.01		16.06	15.71	13.08

(a) Data Size= 100 KB **(b) Data Size= 1 MB**

Data Rate / Instr./Sec	**100 Mb/s**	**1 Gb/s**	**10 Gb/s**	**100 Gb/s**	**100 Mb/s**	**1 Gb/s**	**10 Gb/s**	**100 Gb/s**
10MIPS	13.60	9.75	6.35	6.08	13.85		6.33	6.24
100MIPS	15.17	14.35	9.70	6.38		13.90		6.19
1GIPS	13.70	15.29	15.08	9.62		15.26	13.94	9.13
10GIPS	14.78	13.98	14.83	14.53			16.10	13.99
100GIPS			14.33	15.17				17.14

(c) Data Size= 10 MB **(d) Data Size= 100MB**

Table 41: Average Link Utility (in percentages) of SPTM running on a 2K-node network.

9.11) Link Wait Time

OLWT is the last network performance metric in this thesis. This is particularly used to determine how good the store-and-forward routing and buffer management algorithm adopted for this thesis is. An efficient routing algorithm should be able to demonstrate its ability to minimise the time packets of different types stay in I/O queues during their transmission time. OLWT can be a good estimation of such a time in a real-world situation. The raw OLWT data (expressed in milliseconds) have been presented in sections 9.1 to 9.4.

To have a better idea about the performance of proposed BC platform from this point of view it is better to cancel out the effect of variation in simulation iteration time (modelled by an iteration of the simulation). Also it is better to consider the link busy

time when dealing with OLWT. For this reason, this thesis uses a ratio of OLBT over OLWT (called wait time ratio in this section) for analysing link wait time. 8 FFTTM workflows in a 1K-node network has produced wait time ratios listed in Table 42. The results show that in many cases OLWT is comparable to OLBT. Even in some cases OLWT is bigger than OLBT which means packets spend most of their times waiting in I/O queues to be sent. Only for high comm/comp ratio (e.g. 1000 and 10000) OLBT is considerably higher than OLWT.

Data Rate / Instr./Sec	100 Mb/s	1 Gb/s	10 Gb/s	100 Gb/s	100 Mb/s	1 Gb/s	10 Gb/s	100 Gb/s
10MIPS	1.26	1.26	1.48	4.82	0.77	1.10	7.87	28.22
100MIPS	1.00	1.26	0.80	4.52	0.68	0.68	1.40	7.74
1GIPS		1.07	1.20	1.19		0.86	0.63	1.06
10GIPS		1.05	1.12	1.16		0.89	0.82	0.81

(a) Data Size= 100 KB **(b) Data Size= 1 MB**

Data Rate / Instr./Sec	100 Mb/s	1 Gb/s	10 Gb/s	100 Gb/s
10MIPS	0.63	1.53	8.12	15.23
100MIPS	0.61	0.46	2.17	7.34
1GIPS		0.43	0.60	1.76
10GIPS		0.93	0.51	0.37

(c) Data Size= 10 MB

Table 42: OLBT to OLWT ratio of 8 FFTTMs running on a 1K-node network.

Data Rate / Instr./Sec	100 Mb/s	1 Gb/s	10 Gb/s	100 Gb/s	100 Mb/s	1 Gb/s	10 Gb/s	100 Gb/s
10MIPS	1.07	1.06	3.09	2.27	0.73	0.83	6.19	4.88
100MIPS	1.18	1.04	1.05	4.00	0.66	1.05	0.84	4.88
1GIPS		1.01	1.43	1.06		0.52	0.72	1.78
10GIPS		1.24	1.07	0.66		0.65	0.65	0.91

(a) Data Size= 100 KB **(b) Data Size= 1 MB**

Data Rate / Instr./Sec	100 Mb/s	1 Gb/s	10 Gb/s	100 Gb/s
10MIPS	0.48	1.47	10.96	13.13
100MIPS	0.37	0.48	1.11	10.60
1GIPS		0.38	1.25	0.81
10GIPS				1.21

(c) Data Size= 10 MB

Table 43: OLBT to OLWT ratio of 8 FFTTMs running on a 2K-node network.

The wait time ratio is also found for 8 FFTTM workloads on a network of size 2K nodes (Table 43). The observations are almost the same as those of 1K nodes. The only difference is a minor increase in values especially for high comm/comp ratios.

The raw OLWTs presented in section 9.3 are used to measure wait time ratios for SPTM workloads on a network of size 1K nodes (Table 44). There is a significant rise

in wait time ratio when comm/comp ratio is between 1000 and 10000. FFTTM has the same rise as well but the range of increase is much bigger with SPTM. For those comm/comp ratio values BC is very efficient running SPTM in terms of buffer management.

Data Rate / Instr./Sec	100 Mb/s	1 Gb/s	10 Gb/s	100 Gb/s	100 Mb/s	1 Gb/s	10 Gb/s	100 Gb/s
10MIPS	1.71	3.76	64.95	222.71	1.72	3.13	39.35	185.59
100MIPS	1.29	1.69	3.72	56.48	1.67	1.91	3.15	39.02
1GIPS	1.24	1.32	1.63	3.53	1.68	1.64	1.93	3.18
10GIPS	1.24	1.24	1.33	1.67	1.62	1.65	1.66	1.88
100GIPS	1.24	1.24	1.26	1.20		1.63	1.64	1.63

(a) Data Size= 100 KB **(b) Data Size= 1 MB**

Data Rate / Instr./Sec	100 Mb/s	1 Gb/s	10 Gb/s	100 Gb/s	100 Mb/s	1 Gb/s	10 Gb/s	100 Gb/s
10MIPS	1.68	2.82	30.22	178.41	1.58	2.62	23.40	158.88
100MIPS	1.56	1.71	2.75	29.57	1.34	1.61	2.67	24.26
1GIPS	1.43	1.56	1.71	2.75		1.48	1.60	2.63
10GIPS		1.45	1.54	1.70			1.47	1.57
100GIPS			1.46	1.53				1.47

(c) Data Size= 10 MB **(d) Data Size= 100MB**

Table 44: OLBT to OLWT ratio of SPTM running on a 1K-node network.

Data Rate / Instr./Sec	100 Mb/s	1 Gb/s	10 Gb/s	100 Gb/s	100 Mb/s	1 Gb/s	10 Gb/s	100 Gb/s
10MIPS	1.80	4.10	65.34	284.69	1.93	3.35	44.11	221.97
100MIPS	1.39	1.76	4.08	70.25	1.70	2.00	3.37	46.24
1GIPS	0.72	1.39	1.74	4.13	1.71	1.71	1.95	3.32
10GIPS	1.34	1.33	1.35	1.83	1.71	1.72	1.72	1.99
100GIPS	1.34	1.34	1.33	1.38		1.70	1.72	1.71

(a) Data Size= 100 KB **(b) Data Size= 1 MB**

Data Rate / Instr./Sec	100 Mb/s	1 Gb/s	10 Gb/s	100 Gb/s	100 Mb/s	1 Gb/s	10 Gb/s	100 Gb/s
10MIPS	1.76	2.97	34.86	212.52	1.63		27.01	190.80
100MIPS	1.58	1.77	2.97	35.09		1.61		27.97
1GIPS	1.49	1.59	1.76	2.94		1.48	1.66	2.78
10GIPS	1.54	1.48	1.58	1.75			1.49	1.64
100GIPS			1.50	1.59				1.46

(c) Data Size= 10 MB **(d) Data Size= 100MB**

Table 45: OLBT to OLWT ratio of SPTM running on a 2K-node network.

The same process is repeated with the data in section 9.4 for SPTM workloads on a 2K-node network (Table 45). The effect of comm/comp ratio on wait time ratio is the same as previous table. In both tables the best results are achieved with smaller data sizes (part a in both tables) and they slightly decrease as data size increases.

Chapter 10: Conclusion

Chapters 3 to 7 of this manuscript explained the process of designing and implementing a simulated wireless platform for a massively parallel computer known as Ball Computer (BC) in this thesis. Also the performance of BC platform is evaluated and analysed using series of simulation experiments. There are a number of major design questions involved in implementing such an idea some of which are tackled in this thesis. However, there are some important design issues that are left for future works. Two of these challenges are power delivery and heat dissipation.

Back to the research question of this thesis, we want to know if an effective solution exists for connectivity in a 3D wireless massively parallel computer. The main focus of the thesis is on solving hidden node problem and investigating some possibilities for routing algorithm and deadlock avoidance strategy.

The questions asked in section 3.1 of this thesis will be revisited later in this chapter to check if this thesis has been successful in answering those questions. The main questions in section 3.1 can be rephrased as:

- Do the current state-of-art wireless technologies suit the proposed platform in terms of the area they occupy, their transfer rates and energy they consume?
- Can packet collision be eliminated by proposing a multi-channel multi-radio platform enhanced with effective network partitioning and channel assignment algorithms?
- How much load balancing improves the performance of the BC platform?
- Does the save-and-forward algorithm implemented in this thesis yield good performances for at least a range of networks? What can be done to widen that range?
- How the performance of such a computer may look like?

10.1) Availability of Technology and Choice of Network Topology

The literature review chapter has shown that the main hurdle for wireless devices in the BC platform would be their energy consumption which is still high despite the significant reduction in energy demand of those technologies over recent years. This is a major problem especially because the main priority of modern HPC systems is to reduce their power consumption. The transfer rates of current wireless devices are shown to be not far from what is needed in the BC platform.

In comparison between different categories of wireless devices, on-chip millimetre-wave radio was selected to be used in the BC which has a large 3D hexagonal FCC topology. The nodes are in physical contact with their direct neighbours. Each node has a processor and eight on-chip radio transceivers. This topology is used to let the nodes have the highest number of neighbours.

As a conclusion, on-chip radio devices are still consuming too much energy which is too costly for a modern parallel platform; but this thesis has demonstrated that this can be solved in near future should the historical trend of reducing energy consumption of on-chip radios continues.

10.2) Packet Collision and Network Partitioning

Packet collision happens when a node is sending a packet to another node unaware that the receiver is already busy with receiving another signal. This problem is known to researchers as Hidden Node Problem. To solve this problem and avoid any packet collisions a two-stage network partitioning algorithm is introduced. This algorithm partitions the network into overlapping zones and assigns channels to them in a way that no signal interference can happen. The algorithm is presented, explained and its correctness is proven in section 5. Also none of the simulation experiments presented in chapters 8 and 9 show any traces of even one instance of packet collision which shows that the algorithm has passed the tests as well. The thesis has been successful in solving Hidden Node Problem for a multi-channel wireless network like the BC platform.

10.3) Routing and Deadlock Avoidance

A store-and-forward routing and buffer management algorithm is developed for this thesis. This decision is made mainly because of simplicity of implementation. The results presented in chapter 9 apparently shows that link-related performance metrics (i.e. OLBT, OLWT and ALU) were badly affected by such a decision. The OLWT metric which estimates the time packets spend in output queues before they have the chance to be sent over radios is directly affected by choosing store-and-forward over other methods. Results in chapter 9 show that apart from very big comm/comp ratios the OLWT values are comparable to OLBT values and even in some cases the wait times are bigger than transmission times. This is because the chosen algorithm was not the best for a direct network (i.e. without routers) like the BC network. This problem can be seen in results reported from both task-models and with most of values for network attributes.

This thesis strongly recommends that a wormhole method with virtual channels or at least a cut-through method replace the current store-and-forward algorithm. It is anticipated that the performance of the network will be improved to a great extends by making that replacement.

The deadlock avoidance algorithm used in this thesis is also designed only to explore some possibilities and cannot be regarded as the best algorithm of this type. But it has proven to be quite helpful as it successfully diverted heavy traffics from a communication hotspot towards its neighbouring nodes. However, as stated before, this algorithm cannot guarantee that a deadlock will not happen under any circumstances.

10.4) Load Balancing

A number of ideas for a load balancing algorithm has been discussed in section 6.4 and one of them is implemented for this thesis. The chosen algorithm was the easiest one to implement and is used just to explore some possibilities with load balancing. The proposed algorithm is used particularly for FFTTM since the other task-model has fixed load sizes and there is no room for optimising the size of workload a node dedicates to each of its neighbours.

The results reported in section 8.5 show how efficient the proposed load balancing algorithm was in increasing the performance of the BC network. This is despite a minor overhead it imposes by an extra network discovery process added to the task-model.

10.5) Final Analysis

A number of metrics has been introduced in this thesis to measure the performance of the proposed BC platform. The processor utility measures how effective the processors has been incorporated in a simulation experiment while OLBT, OLWT and ALU give an idea about how effectively radio links are used. Both FFTTM and SPTM are tested over a range of network sizes, transfer rates, computational abilities and data sizes.

The results show that an increase in the comm/comp ratio has the biggest effect on improving the processor utility in both task-models. Also when using FFTTM, this metric improves with an increase in data size. The results show that there is a cap on the size of FFTTM growth beyond which the performance of the task-model decreases. This cap is shown to be between 1000 nodes and 2000 nodes. The SPTM is less sensitive to network attributes when it comes to measuring processor utility metric.

The link-related metrics also are affected by changes in comm/comp ratio. The only exception is the ALU metric measured for SPTM. In this situation the results show that the performance does not always increase when comm/comp ratio increases.

In this thesis potential benefits and disadvantages of a 3D wireless parallel computer are investigated. It has been observed that there are some major obstacles. Energy consumption and – to a lesser extent – the transfer rate are emphasised as main concerns. The number of hops for large networks is also another issue which can be solved should an effective routing algorithm (e.g. wormhole with virtual channels) replaces current routing method. The scalability of the network and its simplicity in setup time are two major benefits. Another big benefit is the removal of all wirings between nodes which hugely reduces the complexity and cost of the network. Through extensive series of simulation experiments this thesis has shown that there are solid chances for the proposed BC platform to yield good performances under some types of networks and with certain types of tasks. This manuscript cannot suggest that there are enough technologies to make a wireless parallel computer right now; but it has shown that the there is a historical trend for pushing for coming up with faster radio links with lower energy demands that can end up in making a prototype for BC in the future.

The simulation and data visualisation tools designed and implemented for this thesis can be used for its next stages as well as other researchers interested in research on the same level of abstraction.

There are many interesting leads that need more investigations. Some of these possibilities for further expansions are listed and discussed in the next section.

10.6) Future Work

The future expansions and enhancement to this thesis can be put into three categories depending on the scope of expansion discussed in the following sections.

10.6.1) Current Simulator

The current simulation can be enhanced by introducing new task-models. Simulated tests with these new task-models give us a better understanding of how the network operates. Those task-models help covering more real-world tasks and identify how good the BC handles them.

Another thing that can be considered is to expand the simulated radio range of nodes so that in addition to its direct neighbours it covers the neighbours of its neighbours as well. This will have two effects. One is that it dramatically increases the number of direct links most of which are redundant which can be a good news. But it comes with the cost of sharp reduction in transfer rate dedicated to each channel. In the current version of the simulation it is assumed that the latter effect will overshadow the former and for this reason such an extended radio range were never tested. For the future it

would be a good idea to test if an increase in the number of links can compensate for the reduction of transfer rates for each of them. This is particularly of interest in light of the fact that in many biological systems (e.g. a human brain) it is not the speed of the links that are key to their success but the large number of connections per node is one of the main contributors to their strength.

Another important issue is to look for a more accurate cost-benefit analysis without which it would be very hard to head for further expansions in the simulation environment or the real world.

10.6.2) Extending the Simulator

A more accurate modelling of radio noises of different types definitely helps increase the accuracy of the network simulator. It is known that there are many sources of noise in a radio link which increases the path loss and ultimately decreases the transmission speed. At the current version of the simulator a very rough (and possibly not accurate enough) modelling of noise is imposed which plays its role mainly as a mild random nature of the packet transmission times. For the future, it should be determined what sources of noise are important enough to have an independent modelling. It should also be decided what is the most suitable form of modelling for those sources of noise regarding the level of abstraction of the simulator.

The current simulation tool needs a thorough change in its routing and buffer management mechanism. Based on the results presented and analysed in chapter 9, the store-and-forward method used by the simulator is not effective under a considerably wide range of network parameters. The main reason why the platform cannot have higher link utilities and lower link wait times is believed to be the delays imposed by the store-and-forward method. It is expected that replacing it with a cut-through method or a wormhole method with virtual channels helps improve both link utility and link wait times.

Another possibility is to investigate accommodating heterogeneous nodes in the network. This means having either processors with different computational abilities or radio modules with different transfer rates or even radio ranges. This is particularly interesting because it makes the BC look more like a biological system in which different parts of the system are better in doing a given sort of tasks rather than being a general purpose machine.

Another reason for thinking about heterogeneous nodes is dictated by the options for power delivery. If the BC is going to have wireless solution for power delivery there are chances that not all the nodes receive the same amount of electrical energy. This means

that some nodes may receive less amount of electricity than others which consequently either impairs their computational capability or reduces their data transfer rates. For this reason it is recommended to study such a possibility to know how this may affect the overall performance of the network.

10.6.3) Beyond Simulation

The ultimate goal of this thesis is to build a prototype of a 3D wireless parallel computer in real world. All the simulations done thus far or anticipated for future are in fact helping us having a better and more accurate understanding of how such a machine may look like and what that computer is capable of. Therefore, we are gathering more and more data through simulation experiments to realise that goal.

Whether that prototype is a pure wireless system (as sought in our most radical plan) or there will be some sort of wirelines included for performing minor tasks are yet to be understood. A more accurate cost-benefit analysis is vital for making decisions of that sort.

Up to now making decisions on issues like power delivery and heat dissipation have been deliberately postponed to let us focus on other important aspects of the thesis. Before any attempts on making a prototype, issues like those two need to be thoroughly researched. These are very important issues that would be investigated in the future stages.

Appendices

Appendix A: Summary of short-range on-chip wireless technologies

A summary of the surveys done in this research thesis is listed in the following tables. They are mainly focused on different short-range on-chip wireless technologies. Other wireless technologies are excluded since they do not satisfy all the criteria for the platform proposed in this thesis.

Ref.	Date	Data Rate (Gb/s)	Range (μm)	Power (mW)	Energy (pJ/bit)	Comments
[61]	**2004**	**1.2**	**300**	**43Tx- 2.5Rx**		**More than 3 stacked chips**
[62], [156]	**2004, 2005**	**1.25**	**60**	**43Tx-2.6Rx**		**Chip thickness= 240** μm
[62], [156]	**2004, 2005**	**1.25**	**60**	**1.1**		**Chip thickness= 30** μm
[62], [156]	**2004, 2005**	**5**	**60**	**21**		**Chip thickness= 30** μm
[63]	**2005**	**195**	**15,28,36,43**	**4,9,15,19**		**195 lines 1Gbps each**
[132]	**2006**	**195**	**15, 30, 45**	**1.2W, 2.2W, 4.1W**		**2, 3, and 4 stacked chips-195 lines 1Gbps each**
[46]	**2006**	**1000**	**15**	**3W(2.2WTx, 0.6WRx, 0.2WClk)-3mW/Gb**	**3**	**1GHz - 1024 channels 1Gbps each, BER< 10^{-12}**
[65]	**2006**	**1**	**1 – 5**	**2.4**		**4-stage Daisy Chain x 2 4-phase TDMA, BER< 10^{-12}**
[157]	**2006**	**1.2**	**300**	**46**		**In a 3D stack**
[158]	**2007**	**1**	**15**		**0.33 Data -3.5Clk**	**1GHz - 90nm CMOS – BER < 10^{-12}**
[158]	**2007**	**1**	**15**		**0.14 Data**	**1GHz - 180nm CMOS – BER < 10^{-12}**
[159]	**2007**	**0.02**	**1.2mm**	**CLK: 14.3Tx-10.4Rx DATA: 0.5Tx-8.1Rx**		**BER < 10^{-10}**
[160]	**2007**	**1**	**70**	**1.31TX – 0.04Rx**		
[68], [161]	**2008, 2009**	**11-10.5-8.5**	**15-30-45**		**1.4-1.5-1.8**	**0.18μm CMOS- with burst transmission-BER <10^{-14}**
[68]	**2008**	**30**	**15**		**0.11**	**90nm CMOS (Simulated)**
[68]	**2008**	**8.5**	**45**		**NA**	**0.18μm CMOS**
[162]	**2008**	**2**	**50**		**0.5**	**BER<10^{-12}, Bi-directional Transceivers with Differential Inductors**
[163]	**2008**	**1**	**15**		**0.065**	**BER<10^{-12}**

[164]	**2008**	**0.48**	**800-1200**	**NA**	**NA**	
[165], [166]	**2009**	**2**	**120**	**307**	**15**	**2 channels - Programmable Bus - BER < 10^{-12}**
[167]	**2009**	**19.2**	**120**		**1**	**18 Channels between a 90nm CMOS Processor and a 65nm CMOS SRAM**
[168]	**2009**	**3.5**	**30**	**23.68**	**6.77**	
[169]	**2009**	**4.7**	**25**	**NA**	**NA**	**Inductive coupling interposer – BER <10^{-12}**
[170]	**2009**	**7.2**	**95**		**7.6**	**1.5GHz - 16 transceivers and 16 processors in a chip**
[171]	**2009**	**0.15**	**200**	**NA**	**NA**	**BER <10^{-12} - Interleaved power and data transmission**
[172], [173]	**2009**	**19.2**	**120**		**1**	**600 MHz - 18 Channels between a 90nm CMOS Processor and two 65nm CMOS SRAMs**
[172], [173]	**2009**	**19.2**	**210**		**1**	**600 MHz - 18 Channels between a 90nm CMOS Processor and two 65nm CMOS SRAMs**
[70]	**2010**	**8000**	**25**	**8W**	**1**	**BER<10^{-16}, 1024 links of 8Gb/s**
[174]	**2010**	**2**	**2200-3600**		**1.8**	**128-Die NAND-Flash Memory Stacking**
[175]	**2010**	**2.5**	**1000**		**6**	**BER < 10^{-12}**
[176]	**2010**	**6**	**250**			**BER<10^{-14}-Simultaneous power and data transmission**
[69]	**2010**	**1.1**	**22**		**0.02**	**BER<10^{-12}-With wireless clock transfer**
[59]	**2011**	**1.1**	**1000(Estimate)**	**1**	**0.91**	**Supply voltage = 0.5**
[59]	**2011**	**1.7**	**1000(Estimate)**	**1**	**0.91**	**Supply voltage = 0.75**
[76]	**2011**	**19.2**	**2200-3600 (By comparison)**		**0.9**	**BER < 10^{-12} – 8 Channels 2.4 Gb/s each**
[67]	**2011**	**12.5**	**500**	**NA**	**NA**	**BER < 10^{-13} – Single Channel**
[67]	**2011**	**12**	**1000**	**NA**	**NA**	**BER < 10^{-13} – Single Channel**
[177]	**2011**	**30**	**30**	**2.1W**	**7**	**10 Channels 3 Gb/s each**
[45]	**2005**	**1**	**100**	**0.95**		
[178]	**2008**	**5**	**30**	**6**		
[179]	**2008**	**0.2**	**30000**		**20**	
[179]	**2008**	**0.2**	**59000**		**32**	
[180]	**2009**	**1**	**100**			
[181]	**2009**	**0.05**	**70000**		**0.475TX+1.3Rx**	**BER < 10^{-3}**
[182]	**2010**	**0.0071**	**30000**		**5.76**	**BER < 10^{-3}**
[58]	**2010**	**0.05**	**50000**		**0.475TX+0.825Rx**	**BER < 10^{-3}**
[183]	**2011**	**1.1**	**22**		**0.01**	**BER<10^{-12}-With wireless clock transfer**
[184]	**2011**	**1.2**	**NM**		**4.7**	
[185]	**2012**	**1000**	**20**		**1**	**Each link has 8Gbps - BER<10^{-16}**

Table 46: Comparing inductive coupling data transfer technologies

Ref.	Date	Data Rate (Gb/s)	Range (μm)	Power (mW)	Energy (pJ/bit)	Comments
[60]	**2003**	**1.27**	**1-2**	**3**		
[50]	**2007**	**2.63**	**1**		**0.080**	**1.7-2.46GHz - Mono-Directional-108 Links – min propagation delay= 380ps**
[50]	**2007**	**1.79**	**1**		**0.120**	**1.8GHz - Bi-Directional - BER<10^{-12}-108 Links -min propagation delay= 560ps**
[186]	**2008**	**100Kb/s**	**150? - 0.5 – 1.1 (by Comparison)**	**9.73μWTX-0.97μWRX**	**107**	
[186]	**2008**	**0.008**	**150? - 0.5 – 1.1 (by Comparison)**			
[187]	**2005**	**0.795-0.975-0.9**	**1**	**0.11-0.14-13mW(?) ? - 0.128 - ? mW**		**25*25 - 15*15 - 8*8μm^2**
[188]	**2007**	**1.23**	**0.5 – 1.1**	**0.17mW**	**0.14**	**BER<10^{-13}**
[189]	**2007**	**1.2**	**0.5 – 1.1**		**0.041**	**900MHz - BER<10^{-13}–15*15μm electrodes- Synchronous**
[189]	**2007**	**1.23**	**0.5 – 1.1**		**0.08**	**1.7GHz - BER<10^{-13}–8*8μm electrodes- Asynchronous**
[53]	**2009**	**32**	**0.5 – 1.1**	**1.12**		**250MHz -128 Channels**
[51]	**2004**	**1.35**	**20**	**3.6**	**3.9**	**BER<10^{-10}–16 Channels total=21.6Gb/s**
[190], [191]	**2005, 2006**	**3**	**0.5 – 1.1 (by Comparison)**	**5Tx-10Rx**		**BER<10^{-12}**
[54]	**2007**	**10**	**3**	**2.7**	**0.27**	**BER<10^{-14}**
[54]	**2007**	**11**	**3**	**4.3**	**0.39**	**25GHz - BER<10^{-14}**
[55]	**2010**	**15**	**4**	**7**	**0.47**	**BER<10^{-10}**
[192]	**2011**	**1**	**0.5 – 1.1 (by Comparison)**	**4.4**		**BER<10^{-12}**
[56]	**2007**	**1.8**	**9.5**		**3**	**144 channels total= 260Gb/s - BER<10^{-15}**
[56]	**2007**	**1.6**	**10.5**	**NA**	**NA**	**144 channels total= 230Gb/s - BER<10^{-14}**
[52]	**2012**	**6**	**0.5 – 1.1 (by Comparison)**		**0.015**	**Two Channels**
[193]	**2011**	**0.9**	**12**	**2.11Tx+5.18Rx**		**BER <10^{-13}**
[194]	**2009**	**2**	**4**	**4.87Tx+1.63Rx**		**BER <10^{-10}**

Table 47: Comparing capacitive coupling data transfer technologies

Ref.	Date	Transferred Power (mW)	Range (µm)	Comments
[195]	**2006**	**2.5**	**600 (Based on Graph)**	**Possibility of increase to 300mW**
[196]	**2007**	**36-22-3**	**15-50-250**	**(Based on Graph)**
[171]	**2009**	**56**	**200**	**33.6 V Ripple**
[197]	**2009**	**48**	**15**	**33% increase in efficiency**
[176]	**2010**	**10**	**250**	
[198]	**2011**	**6000**	**50-320**	**65mV – 8 power transmit channels**
[199]	**2012**	**15**	**50**	

Table 48: Wireless Power transfer technologies based on inductive coupling

Ref.	Date	Data Rate (Gb/s)	Range (mm)	Power (mW)	Energy per bit (pJ/bit)	Area (mm^2)	Comments
[200]	**2004**	**0.01**	**2**	**0.44**		**-**	
[200]	**2004**	**0.05**	**2**	**2.5**		**-**	
[201]	**2005**	**0.001**	**950**	**0.7Tx+4.0Rx**		**0.035Tx+0.38Rx**	**BER<10^{-5}**
[201]	**2005**	**0.001**	**1000**	**0.7Tx+4.0Rx**		**0.035Tx+0.38Rx**	**BER<10^{-3}**
[201], [202]	**2005**	**0.001**	**1000**	**1**		**0.035Tx+0.38Rx**	**BER<10^{-3}– LNA bias switching**
[203]	**2007**	**0.75**	**NA**		**12**	**0.045Tx**	**External Antenna - A Tx only**
[203], [204]	**2007**	**0.75**	**NA**		**41**	**0.29Tx**	**Embedded Ant - A Tx only**
[205]	**2007**	**0.3**	**NA**		**247**	**16.7**	**BER<10^{-3}**
[205]	**2007**	**0.4**	**NA**		**247**	**16.7**	**BER<10^{-2}**
[206]	**2007**	**0.01**	**NA**		**47**	**0.08**	**A Tx only**
[207]	**2009**	**0.2**	**0.5**	**43**		**0.54**	**Integrated antenna**
[208]	**2009**	**2**	**50**	**183Tx+103Rx**		**0.43Tx+0.68Rx**	**On-board antenna - BER<10^{-12}**
[208]	**2009**	**2.5**	**40**	**183Tx+103Rx**		**0.43Tx+0.68Rx**	**On-board antenna - BER<10^{-12}**
[208]	**2009**	**3**	**20**	**183Tx+103Rx**		**0.43Tx+0.68Rx**	**On-board antenna - BER<10^{-12}**
[80]	**2009**	**2-6**	**5-40**	**117**	**17**	**0.62**	**Bond-Wire ant – BER=4*10^{-13}**
[209]	**2009**	**4**	**1000**	**308(170Tx+138Rx)**	**77**	**6.875**	**BPSK - BER<10^{-11}**
[210]	**2009**	**1-2**	**300**	**32(26Tx+6Rx)**	**32**	**0.45**	**OOK – On-chip antenna**
[211]	**2010**	**3.5**	**1000 (by compare)**	**156Tx+108Rx**	**(45Tx,31Rx)38**		

[82]	**2010**	**16**	**20**	**90**		**1.12**	
[212]	**2010**	**1.2**	**50**	**51**		**3.8 (with on-chip ant)**	**BER<10^{-3}–On-chip ant-Rx only**
[212]	**2010**	**1.8**	**50**		**28**	**3.8 (with on-chip ant)**	**BER<10^{-3}–On-chip ant-Rx only**
[213]	**2010**	**16**	**15**	**11.6**			**Simulation- Rx only**
[146]	**2010**	**11**	**14**	**29Tx+41Rx**	**6.4**	**0.06Tx+0.07Rx**	**Bond-wire antennas**
[135]	**2010**	**9**	**NA**	**80.9**			**BER=10^{-9}**
[134]	**2011**	**10**	**NA**	**200**		**1.55**	**Tx only**
[147]	**2011**	**16**	**15 (By compare)**	**26.7**			**BER<10^{-15}**
[214]	**2011**	**1.7**	**2740**	**186Tx+106Rx**		**7.3**	**BPSK - Within 2.16GHz-BW**
[214]	**2011**	**3.52**	**2700**	**186Tx+106Rx**		**7.3**	**QPSK - Within 2.16GHz-BW**
[214]	**2011**	**5.28**	**200**	**186Tx+106Rx**		**7.3**	**8PSK - Within 2.16GHz-BW**
[214]	**2011**	**7.04**	**170**	**186Tx+106Rx**		**7.3**	**16QAM - Within 2.16GHz-BW**
[214]	**2011**	**8**	**2700 (By compare)**	**186Tx+106Rx**		**7.3**	**QPSK - With a wider-BW**
[214]	**2011**	**11**	**170 (By compare)**	**186Tx+106Rx**		**7.3**	**16QAM - With a wider-BW**
[83]	**2011**	**25**	**120**	**140**	**5.6**	**0.41**	**Full duplex – Plastic waveguide - BER<10^{-12}**
[215]	**2012**	**10**	**5000 (Not stated)**	**45**	**4.5**	**1.7**	**BER<10^{-12}**
[86]	**2012**	**10**	**10**	**21**	**2.1**	**0.05**	**Chip-to-chip**
[84]	**2012**	**26**	**120**	**52Tx+85Rx**		**0.16Tx+0.26Rx**	**Plastic waveguide - BER<10^{-12}**
[84]	**2012**	**20**	**5**	**52Tx+85Rx**		**0.16Tx+0.26Rx**	**Free Space - BER<10^{-12}**
[216]	**2012**	**2**	**34**	**150**	**75**	**0.44Tx+0.81Rx**	**BER<10^{-3}**
[217]	**2012**	**0.001**	**NA**	**1.8**		**0.76*0.56**	**Rx only**
[136]	**2012**	**10**	**NA**	**28**		**0.11**	
[81]	**2008**	**3.5**	**2000(from** [218]**)**	**374**		**1.28*0.81**	
[77]	**2006**	**0.63**	**10000**	**500 (526)**		**3.4*1.7Rx + 4*1.6Tx**	**QPSK-OFDM**
[77]	**2006**	**1**	**8000**	**800Tx+500Rx**		**3.4*1.7Rx + 4*1.6Tx**	
[77]	**2006**	**2**	**2500**	**800Tx+500Rx**		**3.4*1.7Rx + 4*1.6Tx**	**QPSK**
[219]	**2010**	**NA**	**NA**	**50.2**		**1.196**	**Rx only**
[220]	**2001**	**NA**	**NA**	**150Tx+162Rx**		**4*4.5**	
[133]	**2004**	**10**	**NA**	**NA**	**NA**	**3Tx+3Rx**	**Large Antenna**
[78]	**2007**	**0.001**	**<10000**	**3.8 to 9.1Tx - 0.5 to 2.6Rx**	**3800Tx+500Rx**	**0.27**	**BER<10^{-3}**
[221]	**2007**	**1.5**	**10000**	**420Tx+450Rx**		**4*3Tx+5.5*4Rx**	
[222]	**2007**	**4**	**NA**	**36.9BA+31LNA+30VCO**		**1.65*1.5**	
[223]	**2007**	**0.96**	**200**	**NA**	**NA**	**NA**	**Theoretically up to 2Gbps**
[224]	**2007**	**1.5**	**2000**	**37**	**<25**	**3**	
[225]	**2007**	**0.8**	**2500**	**1100**		**26*18**	**Tx only - BER=10^{-11}**
[226]	**2007**	**2.4**	**NA**	**85**		**NA**	**DQPSK Rx only - BER=10^{-9}**

[227]	**2008**	**0.000005**	**1800**	**1.1**		**NA**	
[228]	**2007**	**0.016(Implied)**	**1000**	**6.6 (implication)**		**0.23*0.22**	
[229]	**2008**	**7**	**NA**	**173Tx+189Rx**		**1.75*1.5Tx + 2.25*1.7Rx**	**QPSK**
[229]	**2008**	**15**	**NA**	**173Tx+189Rx**		**1.75*1.5Tx + 2.25*1.7Rx**	**16QAM**
[230]	**2008**	**0.001**	**NA**	**19.3*1.2Tx+29.7*1.2Rx**		**3**	**GFSK**
[230]	**2008**	**0.002**	**NA**	**19.3*1.2Tx+29.7*1.2Rx**		**3**	**DPSK**
[230]	**2008**	**0.003**	**NA**	**19.3*1.2Tx+29.7*1.2Rx**		**3**	**8PSK**
[231]	**2008**	**2.6**	**NA**	**133Tx+206Rx**		**1.5Tx+1.9Rx**	**QPSK**
[232]	**2008**	**0.1**	**5 (expected)**	**152**		**3**	
[233]	**2008**	**1**	**NA**	**55**			**Rx only**
[234]	**2008**	**2**	**NA**	**19.2**	**9.6**	**0.41**	**Rx only**
[235]	**2008**	**8**	**NA**	**NA**		**0.61*0.3**	
[236]	**2008**	**1.5**	**NA**	**NA**		**4*3Tx + 5.5*4Rx**	
[237]	**2008**	**1.5**	**NA**	**7.2**		**o.18**	**Demodulator only**
[238]	**2009**	**2.5**	**NA**	**6**		**0.3**	**Demodulator only**
[239]	**2009**	**6**	**2000**	**374 or 232**		**1.28*0.81**	**BPSK**
[79]	**2008**	**4**	**1150**	**1500**		**1.44**	
[240]	**2010**	**1.5**	**380**	**183Tx+103Rx**		**0.43Tx+0.68Rx**	**BER<10^{-12}**
[240]	**2010**	**1**	**610**	**183Tx+103Rx**		**0.43Tx+0.68Rx**	**BER<10^{-12}**
[240]	**2010**	**3.3**	**20**	**183Tx+103Rx**		**0.43Tx+0.68Rx**	**BER<10^{-12}**
[241], [242]	**2010**	**1.6**	**540**	**3800**		**6.5*6.75**	**802.15.3C- 16 elements – OFDM – Nominal bias conditions**
[243]	**2010**	**2**	**100**	**1.2*21.9**	**13.2**	**22*13**	
[244]	**2010**	**0.00025**	**50**	**0.0625Tx+0.045Clk +0.00625Rx**	**600**	**0.62*0.55**	
[244]	**2010**	**0.001**	**50**	**0.254Tx+0.045Clk +0.028Rx**	**373**	**0.62*0.55**	
[244]	**2010**	**0.016**	**50**	**1.75**		**0.62*0.55**	
[245]	**2010**	**0.015**	**12.5**	**1.2*(13.3Tx+20.2Rx)**		**0.014Tx+ 0.11Rx+1**	**ASK - BER<10^{-5}**
[245]	**2010**	**0.020**	**5**			**0.014Tx+ 0.11Rx+1**	**BER<10^{-3}**
[14]	**2010**	**0.64**	**NA**	**675**		**10.37**	**LDPC**
[14]	**2010**	**0.14**	**NA**	**337**		**10.37**	**LDPC**
[14]	**2010**	**0.14**	**NA**	**570**		**10.37**	**Turbo**
[246]	**2010**	**1**	**1040**	**280Tx+150Rx+70PLL**		**1.4*0.9**	**BER<10^{-12}**
[246]	**2010**	**1.5**	**660**	**280Tx+150Rx+70PLL**		**1.4*0.9**	**BER<10^{-12}**
[246]	**2010**	**2**	**410**	**280Tx+150Rx+70PLL**		**1.4*0.9**	**BER<10^{-12}**

[247]	**2010**	**1**	**NA**	**51Tx+116Rx**		**0.85Tx+1.92Rx**	**Repeater**
[242]	**2010**	**1.6**	**540**	**6400**		**6.5*6.75**	**802.15.3C- 16 elements – OFDM – increased PA bias conditions**
[242]	**2010**	**5.3**	**540**	**NM**		**6.5*6.75**	**802.15.3C-16 elements– 16QAM**
[248]	**2010**	**1.5**	**NA**	**31.2**		**1.08**	**ASK - Tx only**
[249]	**2010**	**1**	**NA**	**11.9Tx+11.4Rx**		**1.9**	**GFSK**
[250]	**2010**	**2.5**	**NA**	**83.5**		**1.275*1.19**	**Rx only – OOK-BER<10^{-12}**
[250]	**2010**	**3.5**	**NA**	**83.5**		**1.275*1.19**	**Rx only –BPSK-BER<10^{-9}**
[250]	**2010**	**1.3**	**NA**	**83.5**		**1.275*1.19**	**Rx only–DBPSK-BER<10^{-12}**
[251]	**2010**	**10**	**2000000(Estimate)**	**NA**		**20*8*25 mm^3**	**BER<10^{-10} – Large Antenna**
[252]	**2010**	**10**	**NA**	**8.7**		**NA**	**Tx only**
[253], [254]	**2011**	**5**	**200**	**14.7**	**2.94**	**0.32**	**Rx only**
[255]	**2011**	**5 (Approx.)**	**9000**	**2700**		**NA**	**16 elements**
[256]	**2011**	**7.14**					**WirelessHD**
[256]	**2011**	**6.76**					**802.11ad**
[256]	**2011**	**4**	**10000**	**895**		**8.95*8.12Src+8.64*8.93Snk**	**Tx only - BER<10^{-11} - 8Tx**
[256]	**2011**	**4**	**10000**	**1820**		**8.95*8.12Src+8.64*8.93Snk**	**Tx only - BER<10^{-11} - 32Tx**
[256]	**2011**	**4**	**10000**	**711**		**8.95*8.12Src+8.64*8.93Snk**	**Rx only - BER<10^{-11} – 4Rx**
[256]	**2011**	**4**	**10000**	**1250**		**8.95*8.12Src+8.64*8.93Snk**	**Rx only - BER<10^{-11} – 32Rx**
[256]	**2011**	**3.8**	**50000**	**NA**		**8.95*8.12Src+8.64*8.93Snk**	**32Src+32Snk - LOS**
[256]	**2011**	**1.9**	**184000**	**NA**		**8.95*8.12Src+8.64*8.93Snk**	**32Src+32Snk – LOS**
[256]	**2011**	**0.95**	**390000**	**NA**		**8.95*8.12Src+8.64*8.93Snk**	**32Src+32Snk - LOS**
[256]	**2011**	**3.8**	**16000**	**NA**		**8.95*8.12Src+8.64*8.93Snk**	**32Src+32Snk - NLOS**
[256]	**2011**	**1.9**	**58000**	**NA**		**8.95*8.12Src+8.64*8.93Snk**	**32Src+32Snk – NLOS**
[256]	**2011**	**0.95**	**123000**	**NA**		**8.95*8.12Src+8.64*8.93Snk**	**32Src+32Snk - NLOS**
[256]	**2011**	**3.8**	**13000**	**NA**		**8.95*8.12Src+8.64*8.93Snk**	**8Src+32Snk - LOS**
[256]	**2011**	**1.9**	**47000**	**NA**		**8.95*8.12Src+8.64*8.93Snk**	**8Src+32Snk – LOS**
[256]	**2011**	**0.95**	**101000**	**NA**		**8.95*8.12Src+8.64*8.93Snk**	**8Src+32Snk - LOS**
[256]	**2011**	**3.8**	**4000**	**NA**		**8.95*8.12Src+8.64*8.93Snk**	**8Src+32Snk - NLOS**
[256]	**2011**	**1.9**	**15000**	**NA**		**8.95*8.12Src+8.64*8.93Snk**	**8Src+32Snk – NLOS**
[256]	**2011**	**0.95**	**32000**	**NA**		**8.95*8.12Src+8.64*8.93Snk**	**8Src+32Snk - NLOS**
[257]	**2011**	**2**	**100**	**26.28**	**13.2**	**13*22*1.4 mm^3**	**Tx only + ant -1.4*10^{-9}<BER< 5.3*10^{-3} depending on Rx angle**
[258]	**2011**	**2**	**1000**	**14.4**		**1.18*0.82**	**Tx only**
[254]	**2011**	**2**	**200**	**14.4**		**21.2*11.6**	**Tx only**
[254]	**2011**	**0.648**	**200**			**21.2*11.6Tx+25.5*10.5Rx**	

[259]	**2011**	**2.5**	**1000**	**212Tx+166Rx**		**0.6Tx+0.8Rx**	**BER<10^{-11} – QPSK**
[259]	**2011**	**2.5**	**1000**	**202Tx+125Rx**		**0.6*0.8Tx+0.8Rx**	**BER<10^{-9} – BPSK**
[260]	**2011**	**3.8**	**1000**	**1090Tx+454Rx**		**3.05*2.75**	**16QAM OFDM**
[261], [262]	**2011**	**10**	**1000**	**181Tx+138Rx**		**4.8TRx+1.2PLL**	**13502 points -QPSK-BER<10^{-3}**
[261], [262]	**2011**	**16**	**200**	**181Tx+138Rx**		**4.8TRx+1.2PLL**	**42024 points-16QAM-BER<10^{-3}**
[261], [262]	**2011**	**3.52**	**1000**	**181Tx+138Rx**		**4.8TRx+1.2PLL**	**9506 points-QPSK-BER<10^{-3}**
[261], [262]	**2011**	**7.04**	**200**	**181Tx+138Rx**		**4.8TRx+1.2PLL**	**19912 points-16QAM-BER<10^{-3}**
[262]	**2011**	**1.76**	**5-2740**	**186Tx+106Rx**		**3.5Tx+3.8Rx**	**1585 points - BPSK-BER<10^{-3}**
[262]	**2011**	**5.28**	**5-200**	**186Tx+106Rx**		**3.5Tx+3.8Rx**	**4755 points - 8PSK- BER<10^{-3}**
[263]	**2012**	**8**	**2700**	**186Tx+106Rx**		**3.5Tx+3.8Rx**	**QPSK - BER<10^{-3}**
[263]	**2012**	**11**	**170**	**186Tx+106Rx**		**3.5Tx+3.8Rx**	**16QAM - BER<10^{-3}**
[264]	**2011**	**2**	**200**	**14.4**		**21*12*0.5 mm^3**	**Tx only – on-chip antenna**
[264]	**2011**	**5**	**200**	**14.7**	**2.94**	**21*12*0.5 mm^3**	**Rx only – on-chip antenna**
[87]	**2012**	**0.522**	**30**	**1.2*(83.6Tx+185.3Rx)**	**190Tx+430Rx**	**56**	
[265]	**2011**	**3**	**2000**	**30.5+54**		**1.04*0.47**	**Rx only - QPSK - BER= $9*10^{-9}$ – Large Antenna**
[266]	**2002**	**1.25**				**82x53x7mm^3**	
[267]	**2004**	**0.011**	**NA**	**3*(70Tx+60Rx)**		**3*2.5**	
[268]	**2005**	**1.047**	**7000**	**10**		**9.5*(27.2Tx + 22.0Rx)**	**BER=10^{-12}– Large Antenna**
[268]	**2005**	**1.285**	**7000**	**10**		**9.5*(27.2Tx + 22.0Rx)**	**BER=10^{-12}– Large Antenna**
[269]	**2006**	**0.1**	**3400**	**NA**			**QPSK**
[269]	**2006**	**0.1**	**2000**	**NA**			**8PSK**
[269]	**2006**	**0.2**	**1.5**	**NA**			**16QAM**
[270]	**2002**	**1.5**	**NA**	**NA**		**1.47*1.28Rx+1.4*1.45Ant**	**Rx only**
[271]	**2007**	**0.15**	**NA**	**50**		**0.39*0.48**	**Tx only – ASK**
[272]	**2007**	**2**	**NA**	**2.7*(4Tx+12Rx)**		**0.03Tx+0.02Rx**	
[273]	**2009**	**0.48**	**NA**	**172Tx+230Rx**		**3.3*3.3**	
[274]	**2009**	**12**	**NA**	**1200**		**1.9*1.1**	**Large Antenna**
[275]	**2009**	**1.5**	**100**	**42**	**28**	**1.38+1.43**	**BER<10^{-3} –Rx only– on-chip ant**
[276]	**2010**	**5.3**	**4500**	**1800**		**6.08*6.2**	**16QAM**
[277]	**2010**	**5**	**10000**	**500**		**14.5**	**32 element phased-array**
[278]	**2010**	**10**	**2000000**	**850Tx – 650Rx**		**2*2Tx+2*2Rx**	**120GHz-BER<10^{-10} – Large**

							Ant
[279]	**2010**	**18**	**NA**	**1220**		**1.9*1.1**	**Loop-back – Large Antenna**
[280]	**2010**	**2**	**NA**	**14.4**		**1.18*0.82**	**Tx only**
[280]	**2010**	**5**	**1000 (Implied)**	**14.7**	**2.94**	**0.85*0.37**	**Rx only**
[281]	**2006**	**0.165**	**200**	**21Tx+99Rx**		**2.32*1.75**	
[282]	**2007**	**2**	**3500**	**822Tx+547Rx**		**4*1.6Tx+3.3*1.7Rx**	**MSK**
[283]	**2011**	**2.222**	**750**	**28Tx**		**0.7*0.5Tx**	**Tx only**
[218]	**2011**	**4.8**	**15000**	**1300**		**3.15Tx**	**Tx only - OFDM signal with 16QAM**
[284]	**2011**	**0.002**	**13**	**0.003**	**0.33**	**1.1*1.1**	**For Implanted Neural Sensors**
[285]	**2011**	**5**	**NM**	**29**		**2.5*3.5**	
[137]	**2012**	**10**	**NM**	**117.9Tx+0.00005*1.2Rx**		**2.9*0.6Tx+0.35*0.95Rx**	
[286]	**2010**	**7**	**50**				
[286]	**2010**	**8**	**9**				
[286]	**2010**	**7**	**100**				
[286]	**2010**	**11**	**14**				

Table 49: Comparing short range radio data transfer technologies

Appendix B: More Detailed Architectural Data about Simulation

Network Partitioning and Channel Assignment

The structure of the data saved by the network partitioning for the main simulation is as follows:

The network partitioning program produces series of data for each node which is saved in a number of text files. The first set of data includes the data concerning the basic information about nodes. Table 50 shows the first row of such a data.

Node Id	X	Y	Z	Row	Col	Level	Task Type	Initial Step	Task Lag Time	Data Size	List of channels

Table 50: The first row of the data saved by network partitioning algorithm.

The second row includes a list of nodes which fall inside the node's interference range. The rest of the nodes contain the data about the nodes in the communication range. Each row has a structure shown in Table 51. Each pair of neighbours can contact each other on at most two distinct channels. The two channels are listed for each neighbour of a node.

Neighbour Id	Channel 1	Channel 2

Table 51: The third row of data saved by network partitioning algorithm.

This is almost everything we need from a network partitioning software to produce for the main simulator. But to make it easier for the simulation and to shorten the start-up time of the simulator a number of redundant pieces of data are also saved on separate text files. The first set of redundant data is the data concerning the zones of a node as shown in Table 52. A zone consists of at least two and at most four nodes. In a contiguous 3D network each node has at least one and at most eight zones.

Zone Id	Assigned Channel	Member 1	Member 2	Member 3	Member 4

Table 52: Information about zones of a node.

The first two files accommodate data used by node objects in the main simulator. A third set of text files stores redundant data which is used by channel objects in the simulator. Each file stores the data about a communication channel and zones operating on that channel. Table 53 shows what sort of data is stored by the network partitioning software to be used by the channel objects in the main simulator. The last part of the record is a list of nodes that are close enough to detect the data sent on a particular channel by a particular node.

Zone Id	Node Id	X	Y	Z	List of Interference Nodes

Table 53: Data of a zone stored by the network partitioning software to be used by channel objects.

Packet Structure

To let the simulator perform all its designated functions a simulated packet needs extra fields in addition to its real data. Some of these fields carry information that are just meaningful to the simulator and is used by it. In a real world prototype of a BC these fields are not used. However, there are some other fields that are needed even in a real world implementation of the network.

A basic structure is used for packets in the simulator. This basic structure is extended to new structures to fit particular functionalities in some parts of the simulator. Table 54 shows the fields included in the basic packet data structure. Extra fields are added to this structure to support multi-hop communication. Distinctions between the *source* and the *sender* of the packet on one side of the transaction and the *receiver* and the *destination* of the packet from the other side are made to facilitate this capability. The "*source task index*" and the "*destination task index*" fields are also added to support multi-tasking. To keep the track of timed out packets the "*packet retry count*" field is added. Field "*Request Id*" is used to distinguish between two different packets of the same type, same sender and same receiver. The "*Hop*" field is added for future use in case of implementation of a load balancing algorithm based on the distance of the destination nodes from the source node. The field "*creation time*" is used for simulator's internal operations and can be omitted when implemented for a real world network. Also the "*internal status*" byte keeps some status of the packet all related to the internal operation of the simulator and is irrelevant in a real world prototype of the network.

Sndr	Rcvr	Src	Dst	SrcTskIdx	DstTskIdx	Type	Data	RqstID	Hop	CrtnTm	#PktRtry	IntrStts
Int	Int	Int	Int	Byte	Byte	Byte	String	Int	Int	Int	Byte	Byte

Table 54: The basic structure of a packet implemented for the simulator.

Where:

Sndr: Sender of packet;

Rcvr: Receiver of packet;

Src: (Original) Source of packet;

Dst: (Ultimate) Destination of packet;

SrcTskIdx: The index of the task in the source node that issued the packet;

DstTskIdx: The index of the task in the destination node for which the packet is issued;

Type: The type of packet (Like those listed in Table 15, Table 16 and Table 17);

Data: The real data of the packet;

RqstID: Request ID, working as a packet identifier;

Hop: The number of hops from the source node so far (not used at the present version of the simulator);

CrtnTm: The creation time of the packet. It is just used for simulator's internal operation;

#PktRtry: The number of times the packet is sent due to timer expiration;

IntrStts: Keeps status data used by the simulator only for its internal operation.

The basic packet structure is used in application layer (task-model). The structure introduced in Table 54 is extended in a number of occasions to better fit in different

situations (for different software objects or for different layers of node object). Three of these extensions and modifications are introduced below.

The separation of channel objects from the node objects is discussed in next section but in the older version of the simulator in which all the channel activities are implemented in a node's physical layer, a modified version of the basic packet structure was used. This modification was essential particularly because in the older version of the simulator the packets transmitted between nodes should be treated as buffered data from the simulator's point of view. The contents of the packet is susceptible to corruption as long as the transmission process is in progress. To guarantee the correct transmission of data and detection of possible packet collisions and simultaneous transmissions the packet fields should be kept in a buffer during transmission time. As shown in Table 55 a duplicated version of the basic packet is used to keep the main and buffered fields.

Packet structure (as defined in Table 54)	Buffered packet (with the same structure as main packet)

Table 55: Buffered packet structure designed for physical layer and used before separation of channel and node structures.

After separation of channel and node structures an extension to the basic packet structure is designed for current physical layer I/O queues. This packet structure supports multi-part packet delivery. This is discussed in more details in section 7.1.7. At that section it will be shown that the physical layer of all nodes are implemented inside the channel object for the simplicity of implementation. Although multi-part packets are not part of the current version of the simulator anymore, the structure is still in use particularly to use its "*status*" field. Bits of this field are used to help the sender and receiver nodes process packets faster. The structure of the aforementioned packet structure can be found in Table 56. Another extension to the basic packet structure is the structure that is currently used by MAC layers in node objects and is shown in Table 57.

Packet structure (as defined in Table 54)	MaxPartNumber	PartNumber	Status

Table 56: Packet structure used by channel objects which accommodates multi-part packet transmission (discussed in 7.1.7) as well.

Compared to the basic packet structure what this new structure has are the physical layer timer and acknowledgement wait timer. These timers are transparent to application layer in which the basic packet structure is used. As its name suggests the "*PHY Layer Retry Timer*" is in charge of checking if the physical layer (in channel objects) are busy with other transactions. The "*ACK Wait Timer*" is also used for retransmission of application layer packets if their acknowledgements are not received on time.

Packet structure (as defined in Table 54)	PhyLayerRetryTimer	ACKWaitTimer

Table 57: A new packet structure extended to support ACK timer expiration mechanism.

Appendix C: More Detailed Architectural Data about a Distributed Simulation

The state diagrams of the three main objects in this architecture are shown in Figure 106, Figure 107 and Figure 108. The synchronisations between the three main object types are facilitated via a signalling mechanism shown in following figures.

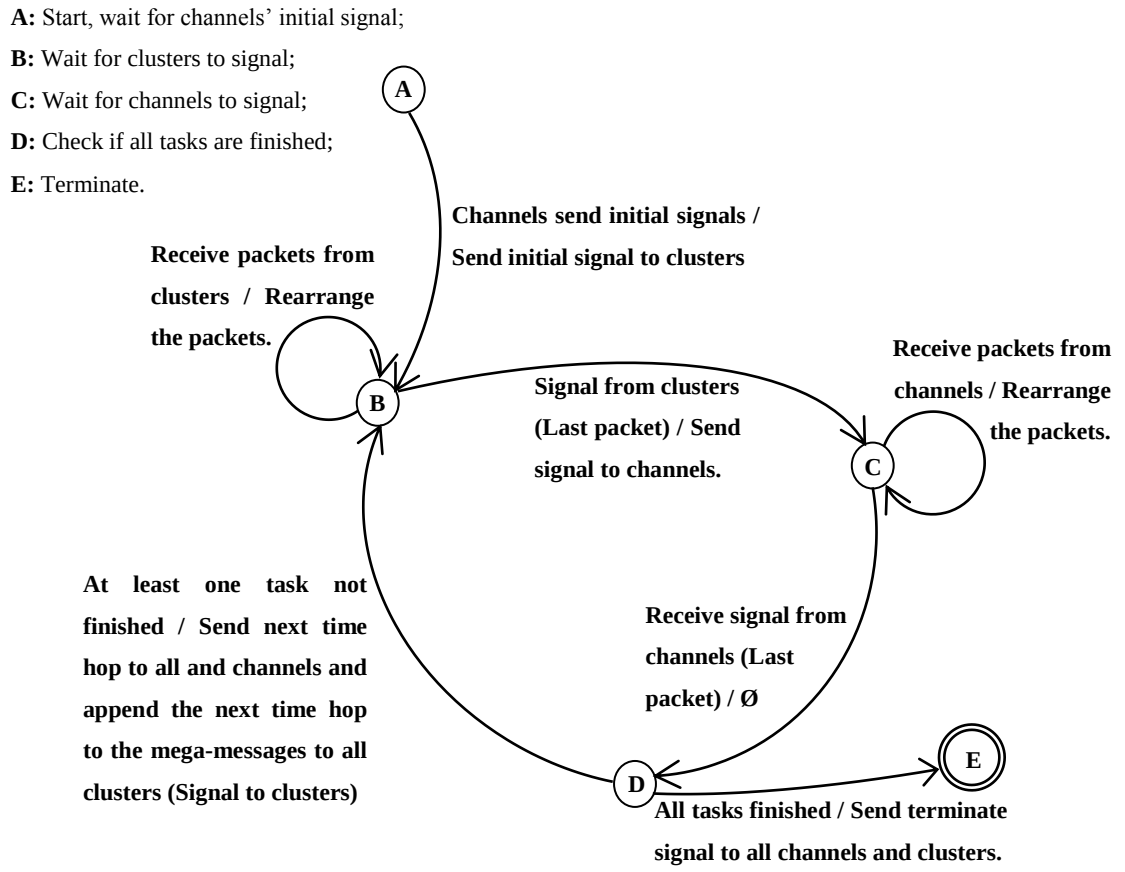


Figure 106: Main network simulator's state diagram. The simulator uses signals to other objects to trigger their activities.

The result is a multi-agent system in which each agent reacts to the inputs received from other agents and to changes in its own internal state. Table 58 describes how the major three agents of the simulation interact with each other at different stages of the simulation. A communication protocol is designed and implemented to let the agents interact between each other. A cluster agent aggregates all the packets sent by its members at the current timestamp and makes a big packet called mega-message in this thesis. The clusters also have their own mega-message packet using which all the packets that are transmitted at that timestamp are aggregated. The main simulator agent is in charge of splitting the channels' and the clusters' mega-messages and rearrange and sending them to the corresponding clusters and channels respectively. The operation described in Table 58 continues until all the tasks are finished.

Iteration	Current time = 1	Current time	Current time	Current time	Current time (**"X"**)
Next Hop	0	0	0	0	0
Action	The simulator Is created.	Clusters are created.	Channels are created. + Send initial signal to the simulator.	The simulator signals all the clusters.	Clusters send packets to their receiver nodes. + Nodes Process any input packets, do a bit of their tasks, check for any timer-expired packets and send their packets. + Clusters send their mega-messages to the simulator including a suggested time hop and an indication of the **number of finished tasks**. Last packet works as a signal to the simulator.
Simulator	A	A	A	A->B	B
Cluster	N/A	A->B	B	B	B->C
Channel	N/A	N/A	A->B	B	B
Iteration	Current time	Current time	Current time		Current time+= New Time Hop
Next Hop	Pending	Pending	Pending		0
Action	The simulator splits the mega messages from clusters + Processes the suggested time hops + Rearranges the packets into new mega-messages + Sends the mega-messages to the channels (Signal)	Channels split the mega messages + Apply the interference model to determine which new or old packets are corrupted and which one is still correct + Make a mega-message out of new node status and any possible received packets + Append their suggested next time hop to the mega-messages and send them to the simulator (Signal).	The simulator Splits the mega-messages from channels + Checks if all the tasks are finished. If so, it sends a terminate signal to all the nodes and channels Otherwise Works out what the next time hop would be + Rearranges the packets and node status in new mega-messages + Sends the next time hop to all channels + Appends the next time hop to all mega-messages and send them to all clusters (Signal).		If the simulator has sent a terminate signal, then all the agents are stopped. Otherwise The channels update their "C**urrent Time**" variables + Decrement the timer of on-air packets + append the packets with timer=0 to the mega-message. + The clusters split the mega-messages. + Update their "C**urrent Time**" variable. + The rest of the operations are the same as time slot **"X"**
Simulator	B->C	C->D	D->E **or** D->B		E **or** B
Cluster	C	C	C		C->D **or** C->B
Channel	B	B->C	C		C->D **or** C->B

Table 58: How the three groups of agents interact

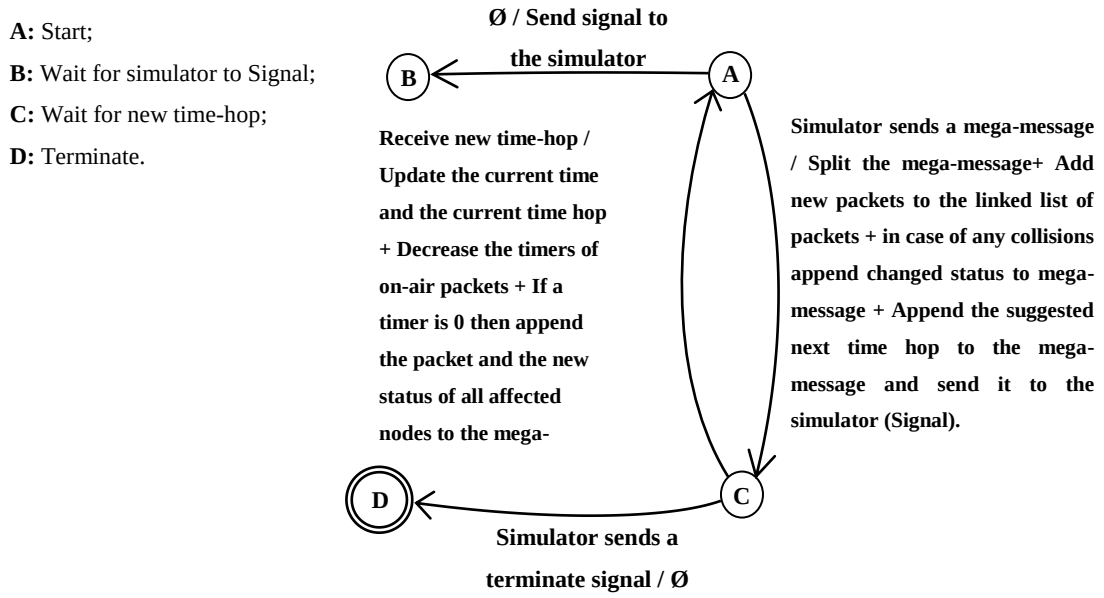


Figure 107: Channel's state diagram.

The internal operations of neither channels nor nodes in a cluster are affected by switching from centralised simulation to distributed simulation platform. Also the communication protocol those objects use for their interactions are not affected by the introduction of the new protocol.

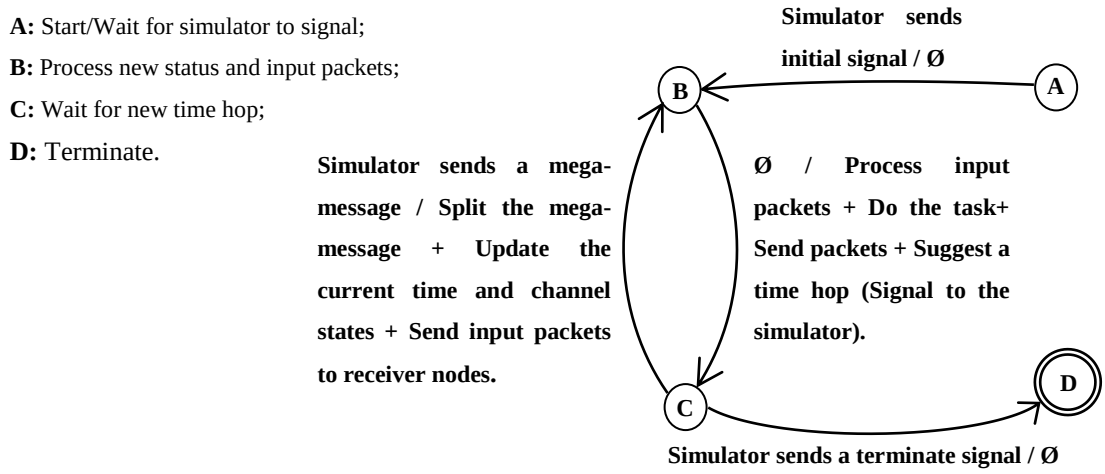


Figure 108: Node/cluster's state diagram.

The communication protocol used by the agents is shown below.

From Channel to simulator

Signal	Suggested_next_time_hop	Node_1_Status	Node_1_data	...	Node_n_Status	Node_n_data

Node_i_data can be either:

The IDs of both the sender and the receiver nodes:

Sender_node_id	Receiver_node_id

Or

A packet sent to 'i'th node.

The receiver node is the node that its status is changed and about to be informed of this change. The sender node is the node that has caused that change in the receiver's node status.

The state of the channel of a node is mentioned in the mega-message when it is recently changed; therefore, not all the nodes' state should necessarily be included in a message. Also the number of packets sent from a node can be either 0 or 1.

From Simulator to Channel

Terminate

Signal	Next_time_hop

Signal	ioBusyTime_1	Packet_to_node_1	ioBusyTime_2	Packet_to_node_2	...

The number of packets sent from a node can be either 0 or 1.

From Cluster to Simulator

Signal	Number_of_busy_tasks	Suggested_next_time_hop	Channel Id_1	ioBusyTime_1	Packet_from_node_1	...

From Simulator to Cluster

Terminate

Initial_signal

Signal	Next_time_hop	Node_id	Channel_id	Node_status	node_data	...

The definition of the field "node_data" is the same as the field with the same name sent from channels to main simulator. The state of a node's channel is mentioned in the mega-message when it is recently changed; therefore, not all the nodes and channels should necessarily be included in a message. Also the number of packets sent from a node can be between 0 and "n", where "n" is the number of channels used by a node.

Elements of a packet

A packet sent from a cluster to the simulator has the following elements (The order of the elements is important):

Data	Original node	Sender	Receiver	Final node	Task index	Type	Status

Except for the first two elements, the rest of them have been extracted from the contents of the original packet. The channel Id is determined at the transfer time depending on the availability of channels. The length of the message and the transfer rate determine the IO busy time.

Appendix D: Details about visualisation tool log files

Table 59 lists different types of data stored in the visualization log file. Some of them are used only once (e.g. data types "*B*" and "*N*"); while others can be used several times depending on the execution of task-model.

Type	Description
B	Basic data (size of the network)
C	Current step of a task
N	Neighbours of a node
I	Iteration completed
P	Packet sent or received
L	Memory location locked
U	Memory location unlocked

Table 59: Different types of data stored in the visualization log file.

Data type "*B*" (Table 60) is used to announce the number of a node. Since the network is static, this type of data only appears once for each node in the log file.

Parameter	Description
x	Number of nodes in X axis
y	Number of nodes in Y axis
z	Number of nodes in Z axis

Table 60: The fields logged by data type "*B*".

Data type "*C*" (Table 61) is used to record a change in the status of a node (more precisely a task in a node). Depending on the behaviour of the task-model this data type can be used several times for each task of each node.

Parameter	Description
N	Node Id
I	Active task
S	New step
T	Task identifier

Table 61: The fields logged by data type "*C*".

Data type "*N*" (Table 62) is only used once in the log file for each node. It includes a list of neighbours of a node which remains unchanged during the execution of the simulation.

Parameter	Description
N	Node Id
-----	A list of Ids of neighbouring nodes separated by " "

Table 62: The fields logged by data type "*N*".

Data type "*I*" (Table 63) is used at the end of each iteration of the simulator. It keeps a record of the network performance parameters. These parameters are mainly about the execution times, the delay times and the number of nodes involved in tasks.

Parameter	Description
I	Iteration number
TT	Transfer Time
CT	Compute Time
TWT	Transfer Wait Time
CWT	Computation Waist Time
BN	Busy Nodes
WN	Waiting Nodes
TF	Tasks Finished

Table 63: The fields logged by data type "*I*".

Data type "L" (Table 64) is used to log the creation of a task dependency between two nodes. The field "*M*" is obsolete at the present version of the visualizer and is not used by any of the main two task-models introduced in this manuscript.

Parameter	Description
S	Source (The node that has accepted the request)
D	Destination (The node that has asked to access memory)
M	Memory Location
T	Task index

Table 64: The fields logged by data type "*L*".

Data type "U" (Table 65) is the opposite of type "*L*" and is used to show the break of a dependency link between two nodes.

Parameter	Description
S	Source (The node that has accepted the request)
D	Destination (The node that had accessed the memory)
M	Memory Location
T	Task index

Table 65: The fields logged by data type "*U*"

Data type "P" (Table 66) is used to record different stages of a packet transaction. Different "*P*" type log data are recorded depending on the stage of the data transaction (From adding the packet to the application layer's I/O queue to the reception of data by the receiver).

Parameter	Description
S	Source of the packet
D	Destination of the packet
ST	Source's Task Index
DT	Destination's Task Index
T	Packet type
D	Packet's Data
F	Frequency channel Id
S/R	Send or receive? ("0": Send; "1": Receive)
Z	Packet size
Q	RequestID
R	Is a retry? ("1": Yes; "0": No)
C	Is a corrupt packet? ("1": Corrupt; "0": Not corrupt)
A	On Air? ("0": No; "1": Yes)
B	Blocked? ("0": Not blocked; "1": Blocked)
L	List of receivers separated by " ;"

Table 66: The fields logged by data type "*P*".

Glossary

BC	Ball Computer
CA	Channel Assignment
MRMC	Multi-Radio Multi-Channel
SMP	Symmetric Memory Processors
NUMA	Non-Unified Memory Access
TSV	Through-silicon-via (Through Si via)
HCP	Hexagonal Close-Packed
BER	Bit Error Rate
BAN	Body Area Network
RTS	Ready to Send
CTS	Clear to Send
FFT	Fast Fourier Transform
DTS	Discrete Time Simulator
DES	Discrete Event Simulator
WRG	White Rose Grid
HPC	High Performance Computing
ACK	Acknowledgment
FLOPS	Floating-point OPerations per Second
SPTM	Simple Parallel Task-Model
FFTTM	Fast Fourier Transform Task-Model
OLBT	Overall Link Busy Time
ALU	Average Link Utility
OLWT	Overall Link Wait Time
OFTT	Overhead-Free Transmission Time

Bibliography

1. **Hind, Richard.** *FEASIBILITY STUDY ON IMPLEMENTING THE "BALL COMPUTER".* Department of Computer Science, University of York. York : University of York, 2013. p. 113, MSc Thesis.

2. **Kürner, Thomas.** What's next? Wireless Communication Beyond 60 GHz. *IEEE 802 Plenary Tutorials.* [Online] 16 July 2012. [Cited: 23 September 2012.] https://mentor.ieee.org/802.15/dcn/12/15-12-0320-01-0thz-what-s-next-wireless-communication-beyond-60-ghz-tutorial-ig-thz.pdf.

3. *Chip to Chip Communications for Terabit Transmission Rates.* **Moore, B, et al.** Macao : IEEE, 2008. IEEE Asia Pacific Conference on Circuits and Systems, APCCAS. Vol. 1, pp. 1558-1561.

4. dipartimento di Matematica. *The University of Milan.* [Online] [Cited: 17 June 2014.] www.mat.unimi.it/users/pavarino/par_comp/3networks.ppt.

5. *Enabling distributed throughput maximization in wireless mesh networks: a partitioning approach.* **Brzezinski, Andrew, Zussman, Gil and Modiano, Eytan.** Los Angeles : ACM, 2006. 12th annual international conference on Mobile computing and networking, MobiCom '06. Vol. 1, pp. 26-37.

6. *A topology control approach for utilizing multiple channels in multi-radio wireless mesh networks.* **Marina, Mahesh K. and Das, Samir R.** Boston : IEEE, 2005. The Second International Conference on Broadband Networks. Vol. 1, pp. 381-390.

7. *Centralized channel assignment and routing algorithms for multi-channel wireless mesh networks.* **Raniwala, Ashish, Gopalan, Kartik and Chiueh, Tzi-cker.** 2, New York : ACM, April 2004, Vol. 8, pp. 50-65.

8. *Architecture and algorithms for an IEEE 802.11-based multi-channel wireless mesh network.* **Raniwala, Ashish and Chiueh, Tzi-cker.** Miami : IEEE, 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM 2005. Vol. 3, pp. 2223-2234.

9. *Routing and Link-layer Protocols for Multi-Channel Multi-Interface Ad Hoc Wireless Networks.* **Kyasanur, Pradeep and Vaidya, Nitin H.** 1, New York : ACM, January 2006, ACM SIGMOBILE Mobile Computing and Communications Review, Vol. 10, pp. 31-43.

10. *Joint optimal channel assignment and congestion control for multi-channel wireless mesh networks.* **Mohsenian Rad, Hamed and Wong, Vincent W.S.** Istanbul : IEEE, 2006. ICC '06. IEEE International Conference on Communications 2006. Vol. 5, pp. 1984-1989.

11. **Culler, David E and Singh, Jaswinder Pal.** *Parallel Computer Architecture a Hardware/Software Approach.* [ed.] Denise E. M. Pentrose. San Fransisco : Murgan Kaufmann Publishers, INC., 1999. p. 7. ISBN 1-55860-343-3.

12. **National Science Foundation (US).** National Science Foundation (US). *National Science Foundation (US).* [Online] May 2011. [Cited: 5 March 2014.] https://www.nsf.gov/cise/aci/taskforces/TaskForceReport_GrandChallenges.pdf.

13. *Reversible logic for supercomputing.* **DeBenedictis, Erik P.** Ischia : ACM, 2005. 2nd conference on Computing frontiers, CF '05. Vol. 1, pp. 391-402.

14. *A 10.37 mm2 675 mW reconfigurable LDPC and Turbo encoder and decoder for 802.11n, 802.16e and 3GPP-LTE.* **Naessens, F, et al.** Honolulu : IEEE, 2010. IEEE Symposium on VLSI Circuits. Vol. 1, pp. 213-214.

15. June 2014 TOP500 Supercomputing Sites. *TOP500 Supercomputing Sites.* [Online] June 2014. [Cited: 23 June 2014.] http://www.top500.org/list/2014/06/.

16. **Almasi, George S. and Gottlieb, Allan.** *Highly Parallel Computing.* [ed.] Dan Jorranstad. 2nd Edition. Redwood City : Benjamin/Cummings Publishing Company Inc., 1994. p. 5. ISBN 0-8053-0443-6.

17. *Validity of the single processor approach to achieving large scale computing capabilities.* **Amdahl, Gene M.** s.l. : ACM, 1967. AFIPS '67. pp. 483-485.

18. **Hill, Mark D. and Marty, Michael R.** *Amdahl's Law in the Multicore Era.* 2008.

19. [Online] 27 July 2011. [Cited: 11 June 2014.] http://hyper-graphica.blogspot.co.uk/2011_07_01_archive.html.

20. OpenFOAM - Performance on Vilje. *confluence.* [Online] [Cited: 11 June 2014.] https://www.hpc.ntnu.no/display/hpc/OpenFOAM+-+Performance+on+Vilje.

21. *Reevaluating Amdahl's law.* **Gustafson, John L.** 5, New York : ACM, May 1988, Communications of the ACM, Vol. 31, pp. 532-533.

22. *Fat-trees: universal networks for hardware-efficient supercomputing.* **Leiserson, Charles E.** 10, s.l. : IEEE, October 1985, IEEE Transactions on Computers, Vol. 34, pp. 892-901.

23. **Hemsoth, Nicole.** *Full Details Uncovered on Chinese Top Supercomputer.* 2 June 2013. HPCwire.

24. *Technology-Driven, Highly-Scalable Dragonfly Topology.* **Kim, John, et al.** Beijing : IEEE, 2008. ISCA '08. 35th International Symposium on Computer Architecture, 2008. Vol. 1, pp. 77 - 88.

25. *From Hypercubes to Dragonflies a short history of interconnect.* **Dally, William J.** San Jose : s.n., 2008. IAA Interconnection Networks Workshop 2008.

26. *Flattened Butterfly : A Cost-Efficient Topology for High-Radix Networks.* **Kim, John, Dally, William J. and Abts, Dennis.** San Diego : ACM, 2007. The 34th annual international symposium on Computer architecture, ISCA '07. Vol. 1, pp. 126-137 .

27. **Alverson, Bob, et al.** Cray XC Series Network. *http://www.cray.com.* [Online] 1 11 2011. [Cited: 6 6 2014.] http://www.cray.com/Assets/PDF/products/xc/CrayXC30Networking.pdf.

28. The Gemini Network, Rev 1.1. *University of Chicago.* [Online] 17 August 2010. [Cited: 18 June 2014.] wiki.ci.uchicago.edu/pub/Beagle/SystemSpecs/Gemini_whitepaper.pdf.

29. An Introduction to the InfiniBand™ Architecture. [book auth.] Hai Jin, Toni Cortes and Rajkumar Buyya. [ed.] Stamatios V Kartalopoulos. *High Performance Mass Storage and Parallel I/O.* Piscataway : IEEE Press, 2002, pp. 617-632.

30. **IBTA.** Infiniband Roadmap. *Infiniband Trade Association.* [Online] IBTA, 2010. [Cited: 30 August 2012.] http://www.infinibandta.org/content/pages.php?pg=technology_overview.

31. **Lembke, Pamela and Milano, James.** IBM System Blue Gene Solution Blue Gene/Q Hardware Overview and Installation Planning. *IBM Redbooks.* [Online] 13 August 2012. [Cited: 26 November 2012.] http://www.redbooks.ibm.com/redbooks/pdfs/sg247872.pdf.

32. **Heath, Michael T.** Parallel Numerical Algorithms, Chapter 1 – Parallel Computing. [Online] 2013. [Cited: 12 June 2014.] http://courses.engr.illinois.edu/cs554/fa2013/notes/01_overview_8up.pdf.

33. *The IBM Blue Gene/Q Interconnection Network and Message Unit.* **Chen, Dong, et al.** Seattle : ACM, 2011. 2011 International Conference for High Performance Computing, Networking, Storage and Analysis, SC11. Vol. 1, pp. 1-10.

34. Switches, What are forwarding modes and how do they work? *Intel, Network Connectivity.* [Online] Intel, 17 April 2014. [Cited: 19 June 2014.] http://www.intel.com/support/express/switches/sb/cs-014410.htm.

35. **Barney, Blaise.** Introduction to Parallel Computing. *Lawrence Livermore National Laboratory.* [Online] Lawrence Livermore National Laboratory, 12 November 2014. [Cited: 13 November 2014.] https://computing.llnl.gov/tutorials/parallel_comp/#Models.

36. **El-Rewini, Hesham and Abd-El-Barr, Mostafa.** *Advanced Computer Architecture and Parallel Processing.* s.l. : Wiley-Blackwell, 2005. ISBN 9780471467403.

37. LMSC, LAN/MAN Standards Committee(project 802). *802 Committee website.* [Online] IEEE, 2012 йил 5-August. [Cited: 2012 йил 19-August.] http://www.ieee802.org/.

38. IEEE 802.15 Working Group for Wireless Personal Area Networks (WPANs). [Online] IEEE SA, 2010 йил 23-May. [Cited: 2010 йил 21-June.] http://www.ieee802.org/15/.

39. **Heile, Bob.** Wireless Sensors and Control Networks: Wireless Control That Simply Works Enabling New Opportunities . *ZigBee Alliance.* [Online] 3 April 2007. [Cited: 23 September 2012.] https://docs.zigbee.org/zigbee-docs/dcn/07/docs-07-4882-00-0mwg-wireless-sensors-and-control-networks-enabling-new-opportunities.pdf.

40. **Alliance, ZigBee.** ZigBee Alliance. *ZigBee Alliance.* [Online] ZigBee Alliance, 2012. [Cited: 19 August 2012.] http://www.zigbee.org/Home.aspx.

41. *The Official Bluetooth SIG Member Website.* [Online] Bluetooth SIG, 2012. [Cited: 20 August 2012.] https://www.bluetooth.org/apps/content/.

42. IEEE_802.11 - wikipedia, the free encyclopedia. *wikipedia, the free encyclopedia.* [Online] Wikipedia, 18 August 2012. [Cited: 20 August 2012.] http://en.wikipedia.org/wiki/IEEE_802.11.

43. **AKASAKA, YOlCHl.** Three-Dimensional IC Trends. *PROCEEDINGS OF THE IEEE.* December 1986, Vol. 74, 12, pp. 1703-1714.

44. *Inductive-Coupling Transceiver for 3D System Integration.* **Miura, Noriyuki and Kuroda, Tadahiro.** Austin : IEEE, 2007. IEEE International Conference on Integrated Circuit Design and Technology, ICICDT. Vol. 1, pp. 1-4.

45. *A 0.95mW/1.0Gbps Spiral-Inductor Based Wireless Chip-Interconnect with Asynchronous Communication Scheme.* **Sasaki, Mamoru and Iwata, Atsushi.** Kyoto : IEEE, 2005. IEEE Symposium on VLSI Circuits. Vol. 1, pp. 348-351.

46. *A 1Tb/s 3W Inductive-Coupling Transceiver for Inter-Chip Clock and Data Link.* **Miura, Noriyuki, et al.** San Francisco : IEEE, 2006. IEEE International Solid-State Circuits Conference. Vol. 1, pp. 1676 - 1685.

47. *ThruChip Interface (TCI) for 3D Integration of Low-Power System.* **Kuroda, Tadahiro.** San Francisco : IEEE, 2010. IEEE International ElectronDevices Meeting (IEDM). Vol. 1, p. 17.1.1.

48. *Capacitive coupling solves the known good die problem.* **Saltzman, Daniel A and Knight, Thomas F.** Santa Cruz : IEEE, 1994. IEEE Multi-Chip Module. Vol. 1, pp. 95-100.

49. **Matsuzawa, Akira.** RF-SoC—Expectations and Required Conditions. *IEEE TRANSACTIONS ON MICROWAVE THEORY AND TECHNIQUES,.* JANUARY 2002, Vol. 50, 1, pp. 245 - 253.

50. *3D Capacitive Interconnections with Mono- and Bi-Directional Capabilities.* **Fazzi, Alberto, et al.** San Fransisco : IEEE, 2007. IEEE International Solid-State Circuits Conference, ISSCC. Vol. 1, pp. 356-608.

51. *Proximity Communication.* **Drost, Robert J, et al.** 4, s.l. : IEEE, September 2004, IEEE JOURNAL OF SOLID-STATE CIRCUITS, Vol. 39, pp. 1529-1535.

52. *Design of Simultaneous Bi-Directional Transceivers Utilizing Capacitive Coupling for 3DICs in Face-to-Face Configuration.* **Aung, Myat Thu Linn, et al.** 2, s.l. : IEEE, June 2012, IEEE JOURNAL ON EMERGING AND SELECTED TOPICS IN CIRCUITS AND SYSTEMS, Vol. 2, pp. 257-265.

53. *System on Chip with 1.12mW-32Gb/s AC-Coupled 3D Memory Interface.* **Canegallo, Roberto, et al.** s.l. : IEEE, 2009. IEEE Custom Intergrated Circuits Conference (CICC). Vol. 1, pp. 463-466.

54. *Two 1OGb/s/pin Low-Power Interconnect Methods for3D ICs.* **Gu, Qun, et al.** San Fransisco : IEEE, 2007. IEEE International Solid-State Clircits Conference. Vol. 1, pp. 448-449-614.

55. *A High-Speed, Low-Power Capacitive-Coupling Transceiver for Wireless Wafer-Level Testing Systems.* **Kim, Gil-Su, et al.** Munich : IEEE, 2010. IEEE International 3D Systems Integration Conference (3DIC). Vol. 1, pp. 1-4.

56. *Circuit Techniques to Enable 430Gb/s/mm2 Proximity Communication.* **Hopkins, David, et al.** San Fransisco : IEEE, 2007. IEEE International Solid-State Circuits Conference. Vol. 1, pp. 368-369-609.

57. *Wireless Wafer-Level Testing of Integrated Circuits via Capacitively-Coupled Channels.* **Lee, Dae Young, Wentzloff, David D and Hayes, John P.** Cottbus : IEEE, 2011. IEEE 14th International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS). Vol. 1, pp. 99-104.

58. *A Low-Energy Inductive Coupling Transceiver With Cm-Range 50-Mbps Data Communication in Mobile Device Applications.* **Lee, Seulki, et al.** 11, s.l. : IEEE, November 2010, IEEE JOURNAL OF SOLID-STATE CIRCUITS, Vol. 45, pp. 2366-2374.

59. *An 0.5V, 0.91pJ/bit, 1.1Gb/s/ch Transceiver in 65nm CMOS for High-Speed Wireless Proximity Interface.* **Matsubara, Takeshi, et al.** Phoenix : IEEE, 2011. IEEE Radio and Wireless Symposium (RWS). Vol. 1, pp. 74-77.

60. *1.27Gb/s/pin 3mW/pin Wireless Superconnect (WSC) Interface Scheme.* **Kanda, Kouichi, et al.** San Francisco : IEEE, 2003. IEEE International Solid-State Circuits Conference. Vol. 1, pp. 186 - 487.

61. *A 1.2Gb/s/pin Wireless Superconnect Based on Inductive Inter-Chip Signaling (IIS).* **Mizoguchi, Daisuke, et al.** San Fransisco : IEEE, 2004. IEEE International Solid-State Circuits Conference. Vol. 1, pp. 142 - 517.

62. *Analysis and Design of Transceiver Circuit and Inductor Layout for Inductive Inter-chip Wireless Superconnect.* **Miura, Noriyuki, et al.** Honolulu : IEEE, 2004. Symposium On VLSl Circuit. Vol. 1, pp. 246-249.

63. *A 195Gb/s 1.2W 3D-Stacked Inductive Inter-Chip Wireless Superconnect with Transmit Power Control Scheme.* **Miura, Noriyuki, et al.** San Fransisco : IEEE, 2005. IEEE International Solid-State Circuits Conference. Vol. 1, pp. 264 - 597.

64. *A 20Gb/s Bidirectional Transceiver Using a Resistor-Transconductor Hybrid.* **Tomita, Yasumoto, et al.** San Fransisco : IEEE, 2006. IEEE International Solid-State Circuits Conference. Vol. 1, pp. 2102 - 2111.

65. *Daisy Chain for Power Reduction in Inductive-Coupling CMOS Link.* **Inoue, Mari, et al.** Honolulu : IEEE, 2006. Symposium on VLSI Circuits. Vol. 1, pp. 65 - 66.

66. *A 10-Gb/s receiver with series equalizer and on-chip ISI monitor in 0.11-μm CMOS.* **Tomita, Yasumoto, et al.** Honolulu : IEEE, 2004. Symposium On VLSl Circuits. Vol. 1, pp. 986 - 993.

67. *A 12Gb/s Non-Contact Interface with Coupled Transmission Lines.* **Takeya, Tsutomu, et al.** San Fransisco : IEEE, 2011. IEEE International Solid-State Circuits Conference. Vol. 1, pp. 492-493.

68. *An 11Gb/s Inductive-Coupling Link with Burst Transmission.* **Miura, Noriyuki, et al.** San Francisco : IEEE, 2008. IEEE International Solid-State Circuits Conference. Vol. 1, pp. 298-299-614.

69. *A 0.7V 20fJ/bit Inductive-Coupling Data Link with Dual-Coil Transmission Scheme.* **Miura, Noriyuki, et al.** Honolulu : IEEE, 2010. IEEE Symposium on VLSI Circuits. Vol. 1, pp. 201-202.

70. *An 8Tb/s 1pJ/b 0.8mm2/Tb/s QDR Inductive-Coupling Interface Between 65nm CMOS GPU and 0.1μm DRAM.* **Miura, Noriyuki, et al.** San Fransisco : IEEE, 2010. IEEE International Solid-State Circuits Conference. Vol. 1, pp. 436-437.

71. *Wireless Power Transfer Using Resonant Inductive Coupling for 3D Integrated ICs.* **Han, Sangwook and Wentzloff, David D.** Munich : IEEE, 2010. IEEE International 3D System Integration Conference (3DIC). Vol. 1, pp. 1-5.

72. *Performance Improvement of Resonant Inductive Coupling for Wireless 3D IC Interconnect.* **Han, Sangwook and Wentzloff, David D.** Toronto : IEEE, 2010. IEEE Antennas and Propagation Society International Symposium (APS/URSI). Vol. 1, pp. 1 - 4.

73. *In-Phase Resonant Inductive Coupling for Multi-Layer Vertical Communication in 3D-ICs.* **Han, Sangwook and Wentzloff, David D.** Chicago : IEEE, 2012. IEEE International Symposium on Antennas and Propagation (AP-S). Vol. 1, pp. 1-2.

74. *Capacitor-Shunted Transmitter for Power Reduction in Inductive-Coupling Clock Link.* **Kumar, Amit, Miura, Noriyuki and Kuroda, Tadahiro.** 4, s.l. : Japan Society of Applied Physics, 2008, Japanese Journal of Applied Physics, Vol. 47, pp. 2749–2751.

75. *Crosstalk Countermeasures for High-Density Inductive-Coupling Channel Array.* **Miura, Noriyuki, Sakurai, Takayasu and Kuroda, Tadahiro.** 2, s.l. : IEEE, FEBRUARY 2007, IEEE JOURNAL OF SOLID-STATE CIRCUITS, Vol. 42, pp. 410-421.

76. *A 2.7Gb/s/mm2 0.9pJ/b/Chip 1Coil/Channel ThruChip Interface with Coupled-Resonator-Based CDR for NAND Flash Memory Stacking.* **Miura, Noriyuki, et al.** San Fransisco : IEEE, 2011. IEEE International Solid-State Circuits Conference. Vol. 1, pp. 490-491.

77. *A Silicon 60-GHz Receiver and Transmitter Chipset for Broadband Communications.* **Reynolds, Scott K, et al.** 12, s.l. : IEEE, December 2006, IEEE JOURNAL OF SOLID-STATE CIRCUITS, Vol. 41, pp. 2820-2831.

78. *An Energy-Efficient OOK Transceiver for Wireless Sensor Networks.* **Daly, Denis C and Chandrakasan, Anantha P.** 5, s.l. : IEEE, May 2007, IEEE JOURNAL OF SOLID-STATE CIRCUITS, Vol. 42, pp. 1003-1011.

79. *A 140-GHz Double-Sideband Transceiver with Amplitude and Frequency Modulation Operating over a few Meters.* **Laskin, E, et al.** Monterey : IEEE, 2008. IEEE Bipolar / BiCMOS Circuits and Technology Meeting - BCTM. Vol. 1, pp. 178-181.

80. *A 6-Gb/s Wireless Inter-Chip Data Link Using 43-GHz Transceivers and Bond-Wire Antennas.* **Chen, Wu-Hsin, et al.** 10, s.l. : IEEE, October 2009, IEEE JOURNAL OF SOLID-STATE CIRCUITS, Vol. 44, pp. 2711-2721.

81. *A Zero-IF 60GHz Transceiver in 65nm CMOS with > 3.5Gb/s Links.* **Tomkins, A, et al.** San Jose : IEEE, 2008. IEEE CustomIntegrated Circuits Conference. Vol. 1, pp. 471-474.

82. *Enhancing Performance of Network-on-Chip Architectures with Millimeter-Wave Wireless Interconnects.* **Deb, Sujay, et al.** Rennes : IEEE, 2010. 21st IEEE International Conference on Application-specific Systems Architectures and Processors (ASAP). Vol. 1, pp. 73-80.

83. *A 12.5+12.5 Gb/s Full-Duplex Plastic Waveguide Interconnect.* **Fukuda, Satoshi, et al.** 12, s.l. : IEEE, December 2011, IEEE JOURNAL OF SOLID-STATE CIRCUITS, Vol. 46, pp. 3113-3125.

84. *A Versatile Multi-Modality Serial Link.* **Tanaka, Yusuke, et al.** San Fransisco : IEEE, 2012. IEEE International Solid-State Circuits Conference. Vol. 1, pp. 332-333.

85. *Direct intensity modulation and wireless data transmission characteristics of terahertz-oscillating resonant tunnelling diodes.* **Ishigaki, K, et al.** 10, s.l. : IEEE, 10 May 2012, Electronic Letters, Vol. 48, pp. 582-583.

86. *A 10GBits/s 2.1pJ/bit OOK demodulator at 60GHz for chip-to-chip wireless communication.* **Foulon, Samuel, et al.** Santa Clara : IEEE, 2012. IEEE Radio and Wireless Symposium. Vol. 1, pp. 291-294.

87. *A-70dBm-Sensitivity 522Mbps 0.19nJ/bit-TX 0.43nJ/bit-RX Transceiver for TransferJet™ SoC in 65nm CMOS.* **Miyashita, Daisuke, et al.** Honolulu : IEEE, 2012. IEEE Symposium on VLSI Circuits. Vol. 1, pp. 74-75.

88. *Power Reduction of CMP Communication Networks via RF-Interconnects.* **Chang, M-c. Frank , et al.** Lake Como : IEEE, 2008. 41st IEEE/ACM International Symposium on Microarchitecture, MICRO 41. Vol. 1, pp. 376-387.

89. **Kiesler Markey, Hedy and Antheil, George.** *Secret Communication system. 2292387* United States of America, 10 June 1941.

90. **Kaye, Roger and Donald, A Goerge.** Transmission of Multiplexed PAM Signals Over Multiple Channel and Diversity Systems. *IEEE TRANSACTIONS ON COMMUNICATION TECHNOLOGY.* October 1970, Vol. 18, 5, pp. 520-526.

91. **Brandenburg, L H and Aaron, D Wyner.** Capacity of the Gaussian Channel with Memory: The Multivariate Case. *Bell System Technical Journal.* 1974, Vol. 53, 5, pp. 745-778.

92. **van Etten, Wim.** An Optimum Linear Receiver for Multiple Channel Digital Transmission Systems. *IEEE Transactions on Communication.* Aug 1975, Vol. 23, 8, pp. 828 - 834.

93. **van Etten, Wim and de Jong, E.** Joint optimisation of transmitter and receiver for digital transmission over multiple-channel systems. *IEE Proceedings on Communications, Radar and Signal Processing.* February 1981, Vol. 128, 1, pp. 28-32.

94. **Golden, G. D., et al.** Detection algorithm and initial laboratory results using V-BLAST space-time communication architecture . *ELECTRONICS LETTERS.* 7 January 1999, Vol. 35, 1, pp. 14 - 16.

95. *Packet Switching in Radio Channels: Part I--Carrier Sense Multiple-Access Modes and Their Throughput-Delay Characteristics.* **Kleinrock, Leonard and Tobagi, Fouad A.** 12, s.l. : IEEE, December 1975, IEEE Transactions on Communications, Vol. 23, pp. 1400-1416.

96. *Performance analysis of carrier sense multiple access with collision detection.* **Tobagi, Fouad A. and Hunt, V. Bruce.** 1980. First Symposium on Local area Communications Networks. Vol. 4, pp. 245-259.

97. Hidden node problem. *Wikipedia.* [Online] Wikipedia, 21 September 2013. [Cited: 26 June 2014.] http://en.wikipedia.org/wiki/Hidden_node_problem.

98. IEEE 802.11™: Wireless LANs. *IEEE Standard Association.* [Online] IEEE, 2014. [Cited: 26 June 2014.] http://standards.ieee.org/about/get/802/802.11.html.

99. *MACAW: a media access protocol for wireless LAN's.* **Bharghavan, Vaduvur, et al.** London : ACM, 1994. The conference on Communications architectures, protocols and applications, SIGCOMM '94. Vol. 1, pp. 212-225.

100. *Channel Assignment Schemes for Cellular Mobile Telecommunication Systems: A Comprehensive Survey*. **Katzela, I and Naghshineh, M.** 3, s.l. : IEEE, June 1996, IEEE Personal Communications, Vol. 3, pp. 10-31.

101. *An overview of Channel Assignment methods for multi-radio multi-channel wireless mesh networks.* **Weisheng, Si, Selvakennedy, Selvadurai and Zomaya, Albert Y.** 5, May 2010, Journal of Parallel and Distributed Computing, Vol. 70, pp. 505–524.

102. *Distributed Graph Coloring: An Approach Based on the Calling Behavior of Japanese Tree Frogs.* **Hernández, Hugo and Blum, Christian.** 2, s.l. : Swarm Intelligence, June 2012, Swarm Intelligence, Vol. 6, pp. 117-150.

103. **Silberschatz, Abraham, Baer Galvin, Peter and Gagn, Greg.** *OPERATING SYSTEM PRINCIPLES.* 7TH. s.l. : Wiley India Pvt. Limited, 2006.

104. *System Deadlocks.* **Coffman, Edward G., Elphick, M. J. and Shoshani, Arie.** 2, New York, NY, USA : ACM, June 1971, ACM Computing Surveys (CSUR), Vol. 3, pp. 67-78.

105. Wikipedia, the free encyclopedia. *Deadlock.* [Online] Wikimedia Foundation, Inc., 24 November 2014. [Cited: 25 November 2015.] http://en.wikipedia.org/wiki/Deadlock#cite_note-embb-7.

106. **Rizzoli, Andrea Emilio.** SimulationTools bibliography. *Dalle Molle Institute for Artificial Intelligence, IDSIA.* [Online] Dalle Molle Institute for Artificial Intelligence, IDSIA, 1 September 2008. [Cited: 22 November 2014.] http://people.idsia.ch/~andrea/sim/simnet.html.

107. ns (simulator). *Wikipedia, the free encyclopedia.* [Online] Wikimedia Foundation, Inc, 19 October 2014. [Cited: 20 November 2015.] http://en.wikipedia.org/wiki/Ns_%28simulator%29.

108. ns version 1 - LBNL Network Simulator. *Lawrence Berkeley National Laboratory*. [Online] Lawrence Berkeley National Laboratory. [Cited: 20 November 2014.] http://ee.lbl.gov/ns/.

109. ns-3 Manual Release ns-3.21. *ns-3*. [Online] 4 November 2014. [Cited: 20 November 2014.] http://www.nsnam.org/docs/release/3.21/manual/ns-3-manual.pdf.

110. Riverbed Modeler. *Riverbed*. [Online] Riverbed, 2014. [Cited: 23 November 2014.] http://www.riverbed.com/products/performance-management-control/network-performance-management/network-simulation.html#Product_Models.

111. DATA SHEET: Riverbed Modeler. *Riverbed*. [Online] 2014. [Cited: 23 November 2014.] http://media-cms.riverbed.com/documents/9306_Riverbed_Modeler_DS_101314KC-2.pdf.

112. NetSim. *Wikipedia, the free encyclopedia*. [Online] Wikipedia, the free encyclopedia, 19 November 2014. [Cited: 23 November 2014.] http://en.wikipedia.org/wiki/NetSim.

113. SpiNNaker Home Page. *APT Advanced Processor Technologies Research Group, the University of Manchester*. [Online] the University of Manchester. [Cited: 27 June 2014.] http://apt.cs.manchester.ac.uk/projects/SpiNNaker/.

114. *The SpiNNaker Project*. **Furber, Steve B., et al.** 5, s.l. : IEEE, May 2014, Proceedings of the IEEE, Vol. 102, pp. 652-665.

115. *Large-scale model of mammalian thalamocortical systems*. **Izhikevich, Eugene M. and Edelman, Gerald M.** 9, s.l. : National Academy of Sciences of the United States of America, 4 March 2008, Proceedings of the National Academy of Sciences of the United States of America, Vol. 105, pp. 3593–3598.

116. *Understanding the interconnection network of SpiNNaker*. **Navaridas, Javier, et al.** Yorktown Heights : ACM, 2009. The 23rd international conference on Supercomputing, ICS '09. Vol. 1, pp. 286-295 .

117. **Nagpal, Radhika.** Towards a Programmable Material. *Amorphous Computing*. [Online] MIT, 1999. [Cited: 26 June 2014.] http://groups.csail.mit.edu/mac/projects/amorphous/Progmat/thesis/activecells.html.

118. Amorphous Computing. *Amorphous Computing*. [Online] MIT. [Cited: 26 June 2014.] http://groups.csail.mit.edu/mac/projects/amorphous/.

119. **Coore, Daniel, Nagpal, Radhika and Weiss, Ron.** Self Organising Communication Networks in an Amorphous Computer. *Amorphous Computing*. [Online] MIT. [Cited: 26 June 2014.] http://groups.csail.mit.edu/mac/projects/amorphous/Network/.

120. —. Self Organising Communication Networks in an Amorphous Computer. *Amorphous Computing*. [Online] MIT. [Cited: 26 June 2014.] http://groups.csail.mit.edu/mac/projects/amorphous/Network/.

121. **Beebee, Wes, et al.** HC11 Gunk. *Amorphous Computing*. [Online] MIT, 1 September 1998. [Cited: 26 June 2014.] http://groups.csail.mit.edu/mac/projects/amorphous/HC11/.

122. **Beal, Jacob.** A Robust Algorithm for Bootstrapping Communications. *Amorphous Computing.* [Online] MIT, July 2002. [Cited: 26 June 2014.] http://groups.csail.mit.edu/mac/projects/amorphous/Bootstrap/.

123. **Tilera.** TILEPro64™ Processor, Product Brief. *Tilera.* [Online] Tilera, 2011. [Cited: 28 June 2014.] http://www.tilera.com/sites/default/files/productbriefs/TILEPro64_Processor_PB019_v4.pdf.

124. Tile processor architecture overview for the TilePro series. *Tilera.* [Online] February 2013. [Cited: 28 June 2014.] http://www.tilera.com/scm/docs/UG120-Architecture-Overview-TILEPro.pdf.

125. **Tilera.** Tile-Gx8072 Processor, Product Vrief. *Tilera.* [Online] Tilera, 2014. http://www.tilera.com/sites/default/files/productbriefs/TILE-Gx8072_PB041-03_WEB.pdf.

126. Net-X. *Distributed Algorithms and Wireless Networking (DAWN) Group.* [Online] University of Illinois at Urbana-Champaign. [Cited: 27 June 2014.] http://www.crhc.illinois.edu/wireless/netx.html.

127. *Link-State Routing Protocol for Multi-Channel Multi-Interface Wireless Networks.* **Kim, Cheolgi, Ko, Young-Bae and Vaidya, Nitin H.** San Diego : IEEE, 2008. IEEE Military Communications Conference, 2008. MILCOM 2008. Vol. 1, pp. 1 - 7.

128. **Raman, Vijay and Vaidya, Nitin H.** A Static-Hybrid Approach for Providing Low Delay Routing for Real Time Applications. *Net-X.* [Online] August 2009. [Cited: 27 June 2014.] http://www.crhc.illinois.edu/wireless/papers/voip_multichannel.pdf.

129. *SHORT: A Static-Hybrid Approach for Routing Real Time Applications Over Multichannel, Multihop Wireless Networks.* **Raman, Vijay and Vaidya, Nitin H.** Luleå : ACM, 2010. 8th International Conference on Wired/Wireless Internet Communications (WWIC). Vol. 1, pp. 77-94 .

130. **Raman, Vijay and Vaidya, Nitin H.** Adjacent Channel Interference Reduction in Multichannel Wireless Networks Using Intelligent Channel Allocation. *Net-X.* [Online] August 2009. [Cited: 27 June 2014.] http://www.crhc.illinois.edu/wireless/papers/aci-channel-aug09.pdf.

131. **Raman, Vijay.** TRAFFIC-AWARE CHANNEL ALLOCATION AND ROUTING IN MULTICHANNEL, MULTI-RADIO WIRELESS NETWORKS. *Net-X.* [Online] 2012. [Cited: 27 June 2014.] http://www.crhc.illinois.edu/wireless/papers/vijay-phdthesis.pdf.

132. **Miura, Noriyuki, et al.** A 195-Gb/s 1.2-W Inductive Inter-Chip Wireless Superconnect With Transmit Power Control Scheme for 3-D-Stacked System in a Package. *IEEE JOURNAL OF SOLID-STATE CIRCUITS.* January 2006, Vol. 41, 1, pp. 23-34.

133. *120-GHz Tx/Rx Chipset for 10-Gbit/s Wireless Applications Using 0.1-pm-gate InP HEMTs.* **KOSUGI, Toshihiko, et al.** Monterey : IEEE, 2004. IEEE Compound Semiconductor Integrated Circuit Symposium. Vol. 1, pp. 171-174.

134. *A 120GHz 10Gb/s Phase-Modulating Transmitter in 65nm LP CMOS.* **Deferm, Noël and Reynaert, Patrick.** San Fransisco : IEEE, 2011. IEEE International Solid-State Circuits Conference. Vol. 1, pp. 290-291.

135. *A 120-GHz Transmitter and Receiver Chipset with 9-Gbps Data Rate using 65-nm CMOS Technology.* **Fujimoto, Ryuichi, et al.** Beijing : IEEE, 2010. IEEE Asian Solid State Circuits Conference (A-SSCC). Vol. 1, pp. 1-4.

136. *28mW 10Gbps Transmitter for 120GHz ASK Transceiver.* **Katayama, Kosuke, et al.** Montréal : IEEE, 2012. IEEE International Microwave Symposium. Vol. 1, pp. 1-3.

137. *A SiGe BiCMOS Transmitter/Receiver Chipset With On-Chip SIW Antennas for Terahertz Applications.* **Hu, Sanming, et al.** 11, s.l. : IEEE, November 2012, IEEE JOURNAL OF SOLID-STATE CIRCUITS, Vol. 47, pp. 2654-2664.

138. *Power Comparison between High-speed Electrical and Optical Interconnects for Inter-chip Communication.* **Cho, Hoyeol, Kapur, Pawan and Saraswat, Krishna C.** Burlingame : IEEE, 2004. IEEE International Interconnect Technology Conference. Vol. 1, pp. 116-118.

139. *Performance Comparisons Between Carbon Nanotubes, Optical, and Cu for Future High-Performance On-Chip Interconnect Applications.* **Koo, Kyung-Hoae, et al.** 12, s.l. : IEEE, December 2007, IEEE Transactions on Electron Devices, Vol. 54, pp. 3206-3215.

140. *On-chip optical interconnects versus electrical interconnects for high-performance applications.* **Stucchi, Michele, et al.** 2013, Microelectronic Engineering, Vol. 112, pp. 84–91.

141. *A 47 ￼10 Gb/s 1.4 mW/Gb/s Parallel Interface in 45 nm CMOS.* **O'Mahony, Frank, et al.** 12, s.l. : IEEE, December 2010, IEEE JOURNAL OF SOLID-STATE CIRCUITS, Vol. 45, pp. 2828-2837.

142. *A 40Gb/s TX and RX Chip Set in 65nm CMOS.* **Chen, Ming-Shuan, et al.** San Fransisco : IEEE, 2011. IEEE International Solid-State Circuits Conference (ISSCC). Vol. 1, pp. 146-147.

143. *A Novel SST Transmitter with Mutually Decoupled Impedance Self-Calibration and Equalization.* **Chen, Shuai, et al.** Rio de Janeiro : IEEE, 2011. IEEE International Symposium on Circuits and Systems (ISCAS). Vol. 1, pp. 173-176.

144. *Modeling, Optimization and Benchmarking of Chip-to-Chip Electrical Interconnects with Low Loss Air-clad Dielectrics.* **Kumar, Vachan, Bashirullah, Rizwan and Naeemi, Azad.** Lake Buena Vista : IEEE, 2011. IEEE 61st Electronic Components and Technology Conference (ECTC). Vol. 1, pp. 2084-2090.

145. *Device Requirements for Optical Interconnects to Silicon Chips.* **David, Miller A. B.** 7, s.l. : IEEE, July 2009, Proceedings of the IEEE, Vol. 97, pp. 1166-1185.

146. *A Millimeter-Wave Intra-Connect Solution.* **Kawasaki, Kenichi, et al.** 12, s.l. : IEEE, December 2010, IEEE JOURNAL OF SOLID-STATE CIRCUITS, Vol. 45, pp. 2655-2666.

147. *A Wideband Body-Enabled Millimeter-Wave Transceiver for Wireless Network-on-Chip.* **Yu, Xinmin, et al.** Seoul : IEEE, 2011. IEEE 54th International Midwest Symposium on Circuits and Systems (MWSCAS). Vol. 1, pp. 1-4.

148. **Gauss, Carl Friedrich.** *Besprechung des Buchs von L.A. Seeber: Intersuchungen über die Eigenschaften der positiven ternären quadratischen Formen usw.* s.l. : Göttingsche Gelehrte Anzeigen, 1831.

149. Close-packing of equal spheres-Wikipedia. *Wikipedia, the free encyclopedia.* [Online] Wikimedia Foundation, Inc. [Cited: 7 2 2013.] http://en.wikipedia.org/wiki/Close-packing_of_equal_spheres.

150. **Fountain, Terry J.** *Paralell computing: principles and practice.* Cambridge : Press syndicate of the University of Cambridge, 1994. ISBN 0521451310.

151. *Map-Colour Theorem.* **Heawood, Percy John.** Oxford : Oxford Journals, 1890, Quarterly Journal of Mathematics, Vol. 24, pp. 332–338.

152. *Every planar map is four colorable. Part I: Discharging.* **Appel, Kenneth and Haken, Wolfgang.** 3, s.l. : The Department of Mathematics at the University of Illinois at Urbana-Champaign, 1977, Illinois Journal of Mathematics, Vol. 21, pp. 429-490.

153. *Every Planar Map is Four Colorable Part II. Reducibility.* **Appel, Kenneth, Haken, Wolfgang and Koch, John.** 3, s.l. : The Department of Mathematics at the University of Illinois at Urbana-Champaign, 1977, Illinois Journal of Mathematics, Vol. 21, pp. 491–567.

154. *Solution of the Four Color Map Problem.* **Appel, Kenneth and Haken, Wolfgang.** 4, s.l. : Nature Publishing Group, October 1977, Scientific American, Vol. 237, pp. 108–121.

155. **Appel, Kenneth and Haken, Wolfgang.** *Every Planar Map is Four-Colorable.* Providence : American Mathematical Society, 1989. Vol. 98. ISBN 0-8218-5103-9.

156. *Analysis and Design of Inductive Coupling and Transceiver Circuit for Inductive Inter-Chip Wireless Superconnect.* **Miura, Noriyuki, et al.** 4, s.l. : IEEE, April 2005, IEEE JOURNAL OF SOLID-STATE CIRCUITS, Vol. 40, pp. 829-837.

157. *A 1.2 Gbps Non-contact 3D-Stacked Inter-Chip Data Communications Technology.* **MIZOGUCHI, Daisuke, et al.** 3, s.l. : IEICE, March 2006, IEICE TRansactions on Electronics, Vols. E89-C, pp. 320-324.

158. *A 0.14pJ/b Inductive-Coupling Inter-Chip Data Transceiver with Digitaly-Controlled Precise Pulse Shaping.* **Miura, Noriyuki, et al.** San Francisco : IEEE, 2007. IEEE International Solid-State Circuits Conference. Vol. 1, pp. 358-608.

159. *An Attachable Wireless Chip AccessI nterface for Arbitrary Data Rate Using Pulse-Based lnductive Coupling through LSI Package.* **Ishikuro, Hiroki, Sugahara, Toshihiko and Kuroda, Tadahiro.** San Fransisco : IEEE, 2007. IEEE International Solid-State Circuits Conference. Vol. 1, pp. 360-608.

160. *Interference from Power/Signal Lines and to SRAM Circuits in 65nm CMOS Inductive-Coupling Link.* **Niitsu, Kiichi, et al.** Jeju : IEEE, 2007. IEEE Asian Solid-State Circuits Conference. Vol. 1, pp. 131-134.

161. *A High-Speed Inductive-Coupling Link With Burst Transmission.* **Miura, Noriyuki, et al.** 3, s.l. : IEEE, March 2009, IEEE JOURNAL OF SOLID-STATE CIRCUITS, Vol. 44, pp. 947-955.

162. *A 2 Gb/s Bi-Directional Inter-Chip Data Transceiver With Differential Inductors for High Density Inductive Channel Array.* **Yoshida, Yoichi, Miura, Noriyuki and Kuroda, Tadahiro.** 11, s.l. : IEEE, November 2008, IEEE JOURNAL OF SOLID-STATE CIRCUITS, Vol. 43, pp. 2363-2369.

163. *A 65 fJ/b inductive-coupling inter-chip transceiver using charge recycling technique for power-aware 3D system integration.* **Niitsu, Kiichi, et al.** Fukuoka : IEEE, 2008. IEEE Asian Solid-State Circuits Conference. Vol. 1, pp. 97-100.

164. *A Wireless Real-Time On-Chip Bus Trace System Using Quasi-Synchronous Parallel Inductive Coupling Transceivers.* **Kawai, Shusuke, et al.** Fukuoka : IEEE, 2008. IEEE Asian Solid-State Circuits Conference. Vol. 1, pp. 113-116.

165. *A 2Gb/s 15pJ/b/chip Inductive-Coupling Programmable Bus for NAND Flash Memory Stacking.* **Sugimori, Yasufumi, et al.** San Fransisco : IEEE, 2009. IEEE International Solid-State Circuits Conference. Vol. 1, pp. 244-245.

166. *2 Gb/s 15 pJ/b/chip Inductive-Coupling Programmable Bus for NAND Flash Memory Stacking.* **Saito, Mitsuko, et al.** 1, s.l. : IEEE, January 2010, IEEE JOURNAL OF SOLID-STATE CIRCUITS, Vol. 45, pp. 134-141.

167. *An Inductive-Coupling Link for 3D Integration of a 90nm CMOS Processor and a 65nm CMOS SRAM.* **Niitsu, Kiichi, et al.** San Fransisco : IEEE, 2009. IEEE International Solid-State Circuits Conference. Vol. 1, pp. 480-481.

168. *Coupling Coefficient Improvement for Inductor Coupled Vertical Interconnect in 3D IC Die Stacking.* **Lu, Hsin-Chia, et al.** San Diego : IEEE, 2009. IEEE Electronic Components and Technology Conference. Vol. 1, pp. 1207-1212.

169. *A 4.7Gb/s Inductive Coupling Interposer with Dual Mode Modem.* **Kawai, Shusuke, Ishikuro, Hiroki and Kuroda, Tadahiro.** Kyoto : IEEE, 2009. IEEE Symposium on VLSI Circuits. Vol. 1, pp. 92-93.

170. *A scalable 3D Processor by Homogeneous Chip Stacking with Inductive-Coupling Link.* **Kohama, Yoshinori, et al.** Kyoto : IEEE, 2009. IEEE Symposium on VLSI Circuits. Vol. 1, pp. 94-95.

171. *Digital Rosetta Stone: A Sealed Permanent Memory with Inductive-Coupling Power and Data Link.* **Yuan, Yuxiang, et al.** Kyoto : IEEE, 2009. IEEE Symposium on VLSI Circuits. Vol. 1, pp. 26-27.

172. *3D System Integration of Processor and Multi-Stacked SRAMs by Using Inductive-Coupling Links.* **Osada, Kenichi, et al.** Kyoto : IEEE, 2009. IEEE Symposium on VLSI Circuits. Vol. 1, pp. 256-257.

173. *3-D System Integration of Processor and Multi-Stacked SRAMs Using Inductive-Coupling Link.* **Saen, Makoto, et al.** 4, s.l. : IEEE, April 2010, IEEE JOURNAL OF SOLID-STATE CIRCUITS, Vol. 45, pp. 856-862.

174. *A 2Gb/s 1.8pJ/b/chip Inductive-Coupling Through-Chip Bus for 128-Die NAND-Flash Memory Stacking.* **Saito, Mitsuko, Miura, Noriyuki and Kuroda, Tadahiro.** San Fransisco : IEEE, 2010. IEEE International Solid-State Circuits Conference. Vol. 1, pp. 440-441.

175. *A 2.5Gb/s/ch 4PAM Inductive-Coupling Transceiver for Non-Contact Memory Card.* **Kawai, Shusuke, Ishikuro, Hiroki and Kuroda, Tadahiro.** San Fransisco : IEEE, 2010. IEEE International Solid-State Circuits Conference. Vol. 1, pp. 264-265.

176. *Simultaneous 6Gb/s Data and 10mW Power Transmission using Nested Clover Coils for Non-Contact Memory Card.* **Yuan, Yuxiang, et al.** Honolulu : IEEE, 2010. IEEE Symposium on VLSI Circuits. Vol. 1, pp. 199-200.

177. *A 30 Gb/s/Link 2.2 Tb/s/mm Inductively-Coupled Injection-Locking CDR for High-Speed DRAM Interface.* **Take, Yasuhiro, Miura, Noriyuki and Kuroda, Tadahiro.** 11, s.l. : IEEE, November 2011, IEEE JOURNAL OF SOLID-STATE CIRCUITS, Vol. 46, pp. 2552-2559.

178. **Wang, Yanjie, et al.** A CMOS IR-UWB Transceiver Design for Contact-Less Chip Testing Applications. *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—II: EXPRESS BRIEFS.* April 2008, Vol. 55, 4, pp. 334-338.

179. *A 200Mbps 0.02nJ/b dual-mode inductive coupling transceiver for cm-range interconnection.* **Lee, Seulki, Yoo, Jerald and Yoo, Hoi-Jun.** Seattle : IEEE, 2008. IEEE International Symposium on Circuits and Systems, ISCAS. Vol. 1, pp. 1954-1957.

180. *An Extended XY Coil for Noise Reduction in Inductive-Coupling Link.* **Saito, Mitsuko, et al.** Taipei : IEEE, 2009. IEEE Asian Solid-State Circuits Conference. Vol. 1, pp. 305-308.

181. *A 1.3pJ/b Inductive Coupling Transceiver with Adaptive Gain Control for Cm-range 50Mbps Data Communication.* **Lee, Seulki, et al.** Taipei : IEEE, 2009. IEEE Asian Solid-State Circuits Conference. Vol. 1, pp. 297-300.

182. *A Real-time Compensated Inductive Transceiver for Wearable MP3 Player System on Multi-layered Planar Fashionable Circuit Board.* **Lee, Seulki, Paek, Seungwook and Yoo, Hoi-Jun.** Paris : IEEE, 2010. IEEE International Symposium on Circuits and Systems (ISCAS). Vol. 1, pp. 2778-2781.

183. *A 0.55 V 10 fJ/bit Inductive-Coupling Data Link and 0.7 V 135 fJ/Cycle Clock Link With Dual-Coil Transmission Scheme.* **Miura, Noriyuki, et al.** 4, s.l. : IEEE, April 2011, IEEE JOURNAL OF SOLID-STATE CIRCUITS, Vol. 46, pp. 965-973.

184. *A 0.6V Noise Rejectable All-Digital CDR with Free Running TDC for a Pulse-Based Inductive-Coupling Interface.* **Yun, Won-Joo, Ishikuro, Hiroki and Kuroda, Tadahiro.** Jeju : IEEE, 2011. IEEE Asian Solid-State Circuits Conference. Vol. 1, pp. 145-148.

185. *A 1 TB/s 1 pJ/b 6.4 ￼ ￼ ￼ ￼ ￼ ￼ ￼ ￼QDR Inductive-Coupling Interface Between 65-nm CMOS Logic and Emulated 100-nm DRAM.* **Miura, Noriyuki, Saito, Mitsuko and Kuroda, Tadahiro.** 2, s.l. : IEEE, June 2012, IEEE JOURNAL ON EMERGING AND SELECTED TOPICS IN CIRCUITS AND SYSTEMS, Vol. 2, pp. 249-256.

186. *A 107pJ/b 100kb/s 0.18µm Capacitive-Coupling Transceiver for Printable Communication Sheet.* **Liu, Lechang, et al.** San Francisco : IEEE, 2008. IEEE International Solid-State Circuits Conference. Vol. 1, pp. 292-614.

187. *A 0.14mW/Gbps High-Density Capacitive Interface for 3D System Integration.* **Fazzi, Alberto, et al.** San Jose : IEEE, 2005. IEEE CUSTOM INTEGRATED CIRCUITS CONFERENCE. Vol. 1, pp. 101-104.

188. *3-D Capacitive Interconnections for Wafer-Level and Die-Level Assembly.* **Fazzi, Alberto, et al.** 10, s.l. : IEEE, October 2007, IEEE JOURNAL OF SOLID-STATE CIRCUITS, Vol. 42, pp. 2270-2282.

189. *3D Capacitive Interconnections for High Speed Interchip Communication.* **Canegallo, Roberto, et al.** San Jose : IEEE, 2007. IEEE Custom Intergrated Circuits Conference (CICC). Vol. 1, pp. 1-8.

190. *3Gb/s AC-Coupled Chip-to-Chip Communication using a Low-Swing Pulse Receiver.* **Luo, Lei, et al.** San Fransisco : IEEE, 2005. IEEE International Solid-State Circuits Conference. Vol. 1, pp. 522-523-614.

191. *3 Gb/s AC Coupled Chip-to-Chip Communication Using a Low Swing Pulse Receiver.* **Luo, Lei, et al.** 1, s.l. : IEEE, January 2006, IEEE JOURNAL OF SOLID-STATE CIRCUITS, Vol. 41, pp. 287-296.

192. *Capacitively Coupled Non-Contact Probing Circuits for Membrane-Based Wafer-Level Simultaneous Testing.* **Daito, Mutsuo, et al.** 10, s.l. : IEEE, October 2011, IEEE JOURNAL OF SOLID-STATE CIRCUITS, Vol. 46, pp. 2386-2395.

193. *A 900 Mbps Single-Channel Capacitive I/O Link for Wireless Wafer-Level Testing of Integrated Circuits.* **Lee, Dae Young, Wentzloff, David D and Hayes, John P.** Jeju : IEEE, 2011. IEEE Asian Solid-State Circuits Conference. Vol. 1, pp. 153-156.

194. *A 25-mV-Sensitivity 2-Gb/s Optimum-Logic-Threshold Capacitive-Coupling Receiver for Wireless Wafer Probing Systems.* **Kim, Gil-Su, Takamiya, Makoto and Sakurai, Takayasu.** 9, s.l. : IEEE, September 2009, IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—II, Vol. 56, pp. 709-713.

195. *Chip-to-Chip Inductive Wireless Power Transmission System for SiP Applications.* **Onizuka, Kohei, et al.** San Jose : IEEE, 2006. IEEE Custom Intergrated Circuits Conference (CICC). Vol. 1, pp. 575 - 578.

196. *Non-Contact 10% Efficient 36mW Power Delivery Using On-Chip Inductor in 0.18-µm CMOS.* **Yuan, Yuxiang, Yoshida, Yoichi and Kuroda, Tadahiro.** Jeju : IEEE, 2007. IEEE Asian Solid-State Circuits Conference. Vol. 1, pp. 115-118.

197. *A 33% Improvement in Efficiency of Wireless Inter-Chip Power Delivery by Thin Film Magnetic Material for Three-Dimensional System Integration.* **Niitsu, Kiichi, et al.** 4C, s.l. : Japan Society of Applied Physics, 20 April 2009, Japanese Journal of Applied Physics, Vol. 48, pp. 04C073-1-04C073-5.

198. *6W/25mm2 Inductive Power Transfer for Non-Contact Wafer-Level Testing.* **Radecki, Andrzej, et al.** San Fransisco : IEEE, 2011. IEEE International Solid-State Circuits Conference. Vol. 1, pp. 230-231.

199. *0.61W/mm2 Resonant Inductively Coupled Power Transfer for 3D-ICs.* **Han, Sangwook and Wentzloff, David D.** San Jose : IEEE, 2012. IEEE Custom Integrated Circuits Conference (CICC). Vol. 1.

200. *TRANSCEIVER CIRCUITS FOR PULSE-BASED ULTRA-WIDEBAND.* **Terada, Takahide, et al.** Vancouver : IEEE, 2004. IEEE International Symposium on Circuits and Systems, 2004. ISCAS '04. Vol. 1, pp. 349-352.

201. *A CMOS IMPULSE RADIO ULTRA-WIDEBAND TRANSCEIVER FOR 1Mb/s DATA COMMUNICATIONS AND ±2.5cm RANGE FINDINGS.* **Terada, Takahide, et al.** Kyoto : IEEE, 2005. Symposium on VLSI Circuits. Vol. 1, pp. 30-33.

202. *A CMOS Ultra-Wideband Impulse Radio Transceiver for 1-Mb/s Data Communications and ±2.5-cm Range Finding.* **Terada, Takahide, et al.** 4, s.l. : IEEE, April 2006, IEEE JOURNAL OF SOLID-STATE CIRCUITS, Vol. 41, pp. 891-898.

203. *A 750Mb/s 12pJ/b 6-to-10GHz Digital UWB Transmitter.* **Kulkarni, Vishal, et al.** San Jose : IEEE, 2007. IEEE Custom Intergrated Circuits Conference (CICC). Vol. 1, pp. 647 - 650.

204. *A 750 Mb/s, 12 pJ/b, 6-to-10 GHz CMOS IR-UWB Transmitter With Embedded On-Chip Antenna.* **Kulkarni, Vishal K, et al.** 2, s.l. : IEEE, February 2009, IEEE JOURNAL OF SOLID-STATE CIRCUITS, Vol. 44, pp. 394-403.

205. *A 0.18μm CMOS Dual-Band UWB Transceiver.* **Zheng, Yuanjin, et al.** San Fransisco : IEEE, 2007. IEEE International Solid-State Circuits Conference. Vol. 1, pp. 114-115-590.

206. *A 47pJ/pulse 3.1-to-5GHz All-Digital UWB Transmitter in 90nm CMOS.* **Wentzloff, David D and Chandrakasan, Anantha P.** San Fransisco : IEEE, 2007. IEEE International Solid-State Circuits Conference. Vol. 1, pp. 118-119-591.

207. *A Single-Chip Ultra-Wideband Receiver With Silicon Integrated Antennas for Inter-Chip Wireless Interconnection.* **Sasaki, Nobuo, et al.** 2, s.l. : IEEE, February 2009, IEEE JOURNAL OF SOLID-STATE CIRCUITS, Vol. 44, pp. 382-393.

208. *A Low-Power Fully Integrated 60GHz Transceiver System with OOK Modulation and On-Board Antenna Assembly.* **Lee, Jri, et al.** San Fransisco : IEEE, 2009. IEEE International Solid-State Circuits Conference. Vol. 1, pp. 316-317.

209. *A 90nm CMOS Low-Power 60GHz Transceiver with Integrated Baseband Circuitry.* **Marcu, Cristian, et al.** San Fransisco : IEEE, 2009. IEEE International Solid-State Circuits Conference. Vol. 1, pp. 314-315.

210. *A Low Power 60GHz OOK Transceiver System in 90nm CMOS with Innovative On-Chip AMC Antenna.* **Lin, Fujiang, et al.** Taipei : IEEE, 2009. IEEE Asian Solid-State Circuits Conference. Vol. 1, pp. 349-352.

211. **Juntunen, Eric, et al.** A 60-GHz 38-pJ/bit 3.5-Gb/s 90-nm CMOS OOK Digital Radio. *IEEE TRANSACTIONS ON MICROWAVE THEORY AND TECHNIQUES.* February 2010, Vol. 58, 2, pp. 348-355.

212. *A 60-GHz OOK Receiver With an On-Chip Antenna in 90 nm CMOS.* **Kang, Kai, et al.** 9, s.l. : IEEE, September 2010, IEEE JOURNAL OF SOLID-STATE CIRCUITS, Vol. 45, pp. 1720-1731.

213. *Performance evaluation and receiver front-end design for on-chip millimeter-wave wireless interconnect.* **Yu, Xinmin, et al.** Chicago : IEEE, 2010. International Green Computing Conference. Vol. 1, pp. 555 - 560.

214. *A 60-GHz 16QAM/8PSK/QPSK/BPSK Direct-Conversion Transceiver for IEEE802.15.3c.* **Okada, Kenichi, et al.** 12, s.l. : IEEE, December 2011, IEEE JOURNAL OF SOLID-STATE CIRCUITS, Vol. 46, pp. 2988-3004.

215. *A 10 Gb/s 45 mW Adaptive 60 GHz Baseband in 65 nm CMOS.* **Thakkar, Chintan, et al.** 4, s.l. : IEEE, April 2012, IEEE JOURNAL OF SOLID-STATE CIRCUITS, Vol. 47, pp. 952-968.

216. *A 2Gb/s 150mW UWB Direct-Conversion Coherent Transceiver with IQ-Switching Carrier Recovery Scheme.* **Abe, Takayuki, et al.** San Fransisco : IEEE, 2012. IEEE International Solid-State Circuits Conference. Vol. 1, pp. 442-443.

217. *Highly Sensitive and Low Power Injection-Locked FSK Receiver for Short-Range Wireless Applications.* **Ye, Rong-Fu, Horng, Tzyy-Sheng and Wu, Jian-Ming.** Montréal : IEEE, 2012. IEEE Radio Frequency Integrated Circuits Symposium. Vol. 1, pp. 377-380.

218. *A Fully Integrated 60 GHz Transmitter Front–End in SiGe BiCMOS Technology.* **Glisic, Srdjan, et al.** Phoenix : IEEE, 2011. IEEE 11th Topical Meeting on Silicon Monolithic Integrated Circuits in RF Systems, (SiRF). Vol. 1, pp. 149-152.

219. *A 60 GHz CMOS receiver front-end with integrated 180° out-of-phase wilkinson power divider.* **Lee, Jen-How, Chen, Chi-Chen and Lin, Yo-Sheng.** 12, s.l. : Wiley Periodicals, Inc, December 2010, MICROWAVE AND OPTICAL TECHNOLOGY LETTERS, Vol. 52, pp. 2688-2694.

220. *A Single-Chip 2.4GHz Direct-Conversion CMOS Transceiver with GFSK Modem for Bluetooth Application.* **Lee, Seung-Wook, et al.** Kyoto : IEEE, 2001. IEEE Symposium on VLSl Circuits. Vol. 1, pp. 245-246.

221. *60 GHz Single-Chip Front-End MMICs and Systems for Multi-Gb/s Wireless Communication.* **Gunnarsson, Sten E, et al.** 5, s.l. : IEEE, May 2007, IEEE JOURNAL OF SOLID-STATE CIRCUITS, Vol. 42, pp. 1143-1157.

222. *A 60GHz Low-Power Six-Port Transceiver for Gigabit Software-Defined Transceiver Applications.* **Wang, Chi-Hsueh, et al.** San Fransisco : IEEE, 2007. IEEE International Solid-State Circuits Conference. Vol. 1, pp. 192-193-596.

223. *60 GHz SiGe-BiCMOS Radio for OFDM Transmission.* **Grass, Eckhard, et al.** New Orleans : IEEE, 2007. IEEE International Symposium on Circuits and Systems, ISCAS. Vol. 1, pp. 1979-1982.

224. *A Single-Chip 25pJ/bit Multi-Gigabit 60GHz Receiver Module.* **Sarkar, Saikat and Laskar, Joy.** Honolulu : IEEE, 2007. IEEE/MTT-S International Microwave Symposium. Vol. 1, pp. 475-478.

225. **Jung, Dong Yun, et al.** 60-GHz System-on-Package Transmitter Integrating Sub-Harmonic Frequency Amplitude Shift-Keying Modulator. *IEEE TRANSACTIONS ON MICROWAVE THEORY AND TECHNIQUES.* August 2007, Vol. 55, 8, pp. 1786-1793.

226. *A 2.2Gb/s DQPSK Baseband Receiver in 90-nm CMOS for 60 GHz Wireless Links.* **Chen, Minghui and Chang, Mau-Chung Frank.** Kyoto : IEEE, 2007. IEEE Symposium on VLSI Circuits. Vol. 1, pp. 56-57.

227. *A 5.2-GHz BFSK Transceiver Using Injection-Locking and an On-Chip Antenna.* **Popplewell, Peter, et al.** 4, s.l. : IEEE, April 2008, IEEE JOURNAL OF SOLID-STATE CIRCUITS, Vol. 43, pp. 981-990.

228. *Wideband Flexible Transmitter and Receiver Pair for Implantable Wireless Neural Recording Applications.* **Yin, Ming and Ghovanloo, Maysam.** s.l. : IEEE, 2007. IEEE Northeast Workshop on Circuits and Systems, NEWCAS. Vol. 1, pp. 85-88.

229. *A 90nm CMOS 60GHz Radio.* **Pinel, Stephane, et al.** San Fransisco : IEEE, 2008. IEEE International Solid-State Circuits Conference. Vol. 1, pp. 130-131-601.

230. *A Single-Chip CMOS Radio SoC for v2.1 Bluetooth Applications.* **Weber, David, et al.** San Fransisco : IEEE, 2008. IEEE International Solid-State Circuits Conference. Vol. 1, pp. 364-365-620.

231. *TX and RX Front-Ends for 60GHz Band in 90nm Standard Bulk CMOS.* **Tanomura, Masahiro, et al.** San Fransisco : IEEE, 2008. IEEE International Solid-State Circuits Conference. Vol. 1, pp. 558-559-635.

232. *A 24-GHz Transmitter With On-Chip Dipole Antenna in 0.13-μm CMOS.* **Cao, Changhua, et al.** 6, s.l. : IEEE, June 2008, IEEE JOURNAL OF SOLID-STATE CIRCUITS, Vol. 43, pp. 1394-1402.

233. *A 1Gbps Mixed-Signal Analog Front End for a 60GHz Wireless Receiver.* **Sobel, David A and Brodersen, Robert W.** Honolulu : IEEE, 2008. IEEE Symposium on VLSI Circuits. Vol. 1, pp. 156-157.

234. *19.2mW 2Gbps CMOS Pulse Receiver for 60GHz Band Wireless Communication.* **Oncu, Ahmet and Fujishima, Minoru.** Honolulu : IEEE, 2008. IEEE Symposium on VLSI Circuits. Vol. 1, pp. 158-159.

235. *8Gbps CMOS ASK Modulator for 60GHz Wireless Communication.* **ONCU, Ahmet, TAKANO, Kyoya and FUJISHIMA, Minoru.** Fukuoka : IEEE, 2008. IEEE Asian Solid-State Circuits Conference. Vol. 1, pp. 125-128.

236. *Integrated 60 GHz Circuits and Systems for High-Speed Communications.* **Abbasi, Morteza, et al.** Amsterdam : IEEE, 2008. 3rd European Microwave Integrated Circuits Conference. Vol. 1, pp. 9-12.

237. *A Low power 20 GHz 1.5 Gb/s CMOS Injection-Pulling FSK Modulator and Frequency Discriminator for 60GHz Links.* **Wen, Shon-Hang, Wang, Chao-Shiun and Wang, Chorng-Kuang.** San Jose : IEEE, 2008. IEEE Custom Intergrated Circuits Conference (CICC). Vol. 1, pp. 483-486.

238. *A 0.13µm CMOS 2.5Gb/s FSK Demodulator Using Injection- Locked Technique.* **Wang, Chao-Shiun, Chu, Kun-Da and Wang, Chorng-Kuang.** Boston : IEEE, 2009. IEEE Radio Frequency Integrated Circuits Symposium. Vol. 1, pp. 563-566.

239. *A Zero-IF 60 GHz 65 nm CMOS Transceiver With Direct BPSK Modulation Demonstrating up to 6 Gb/s Data Rates Over a 2 m Wireless Link.* **Tomkins, Alexander, et al.** 8, s.l. : IEEE, August 2009, IEEE JOURNAL OF SOLID-STATE CIRCUITS, Vol. 44, pp. 2085-2099.

240. *A Low-Power Low-Cost Fully-Integrated 60-GHz Transceiver System With OOK Modulation and On-Board Antenna Assembly.* **Lee, Jri, Chen, Yentso and Huang, Yenlin.** 2, s.l. : IEEE, February 2010, IEEE JOURNAL OF SOLID-STATE CIRCUITS, Vol. 45, pp. 264-275.

241. *A SiGe BiCMOS 16-Element Phased-Array Transmitter for 60GHz Communications.* **Valdes-Garcia, Alberto, et al.** SanFransisco : IEEE, 2010. IEEE International Solid-State Circuits Conference. Vol. 1, pp. 218-219.

242. *A Fully Integrated 16-Element Phased-Array Transmitter in SiGe BiCMOS for 60-GHz Communications.* **Valdes-Garcia, Alberto, et al.** 12, s.l. : IEEE, December 2010, IEEE JOURNAL OF SOLID-STATE CIRCUITS, Vol. 45, pp. 2757-2773.

243. *SiP-based 60GHz 4x4 Antenna Array with 90nm CMOS OOK Modulator in LTCC.* **Karim, M K, et al.** Anaheim : IEEE, 2010. IEEE International Microwave Symposium, IMS. Vol. 1, pp. 352-355.

244. *A fully integrated, 300pJ/bit, dual mode wireless transceiver for cm-range interconnects.* **Gambini, S, et al.** Honolulu : IEEE, 2010. IEEE Symposium on VLSI Circuits/Technical. Vol. 1, pp. 31-32.

245. *FDM-based Wireless Source Synchronous 15-Mbps TRx with PLL-less Receiver and 1-mm On-chip Integrated Antenna for 1.25-cm Touch-and-Proceed Communication.* **Ishizaki, Haruya, et al.** Honolulu : IEEE, 2010. IEEE Symposium on VLSI Circuits. Vol. 1, pp. 73-74.

246. *A 60-GHz FSK Transceiver with Automatically-Calibrated Demodulator in 90-nm CMOS.* **Wang, Huaide, et al.** Honolulu : IEEE, 2010. IEEE Symposium on VLSI Circuits. Vol. 1, pp. 95-96.

247. *1Gbps/ch 60GHz CMOS Multichannel Millimeter-Wave Repeater.* **Oncu, Ahmet, et al.** Honolulu : IEEE, 2011. IEEE Symposium on VLSI Circuits. Vol. 1, pp. 93-94.

248. *60 GHz Low Power 1.5 Gb/s ASK Transmitter in 90 nm CMOS with On-Board Yagi-Uda Antenna.* **Shin, Woorim, et al.** Paris : s.n., 2010. 40th European Microwave Conference. Vol. 1, pp. 272-275.

249. *A 0.8V 2.4GHz 1Mb/s GFSK RF Transceiver with On-Chip DC-DC Converter in a Standard 0.18μm CMOS Technology.* **Dal Fabbro, Paulo Augusto, et al.** s.l. : IEEE, 2010. IEEE ESSCIRC. Vol. 1, pp. 458-461.

250. **Chuang, Kevin, et al.** A 90 nm CMOS Broadband Multi-Mode Mixed-Signal Demodulator for 60 GHz Radios. *IEEE TRANSACTIONS ON MICROWAVE THEORY AND TECHNIQUES.* December 2010, pp. 4060-4071.

251. **Takahashi, Hiroyuki, et al.** 10-Gbit/s Quadrature Phase-Shift-Keying Modulator and Demodulator for 120-GHz-Band Wireless Links. *IEEE TRANSACTIONS ON MICROWAVE THEORY AND TECHNIQUES.* December 2010, pp. 4072-4078.

252. *A 10Gbps UWB Transmitter for Wireless Inter-chip and Intra-chip Communication.* **Afroz, Sadia, et al.** Dhaka : ICECE, 2010. 6th International Conference on Electrical and Computer Engineering, ICECE. Vol. 1, pp. 104-107.

253. *A 60 GHz 5 Gb/s Gain-Boosting OOK Demodulator in 0.13 μm CMOS.* **Byeon, Chul Woo, et al.** 2, s.l. : IEEE, February 2011, IEEE MICROWAVE AND WIRELESS COMPONENTS LETTERS, Vol. 21, pp. 101-103.

254. *A 60-GHz Transceiver System with Low-Power CMOS OOK Modulator and Demodulator.* **Byeon, Chul Woo, et al.** Daejeon : IEEE, 2011. IEEE MTT-S International Microwave Workshop Series on Intelligent Radio for Future Personal Terminals (IMWS-IRFPT). Vol. 1, pp. 1-2.

255. *Single-Element and Phased-Array Transceiver Chipsets for 60-GHz Gb/s Communications.* **Valdes-Garcia, Alberto, et al.** s.l. : IEEE, April 2011, IEEE Communications Magazine, pp. 120-131.

256. *A 60GHz CMOS Phased-Array Transceiver Pair for Multi-Gb/s Wireless Communications.* **Emami, Sohrab, et al.** San Fransisco : IEEE, 2011. IEEE International Solid-State Circuits Conference. Vol. 1, pp. 164-165.

257. **Karim, Muhammad Faeyz, et al.** Integration of SiP-Based 60-GHz 4 □4 Antenna Array With CMOS OOK Transmitter and LNA. *IEEE TRANSACTIONS ON MICROWAVE THEORY AND TECHNIQUES.* July 2011, Vol. 59, 7, pp. 1869-1878.

258. **Lee, Jae Jin and Park, Chul Soon.** 60-GHz Gigabits-Per-Second OOK Modulator With High Output Power in 90-nm CMOS. *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—II: EXPRESS BRIEFS.* May 2011, Vol. 58, 5, pp. 249-253.

259. *W-Band BPSK and QPSK Transceivers With Costas-Loop Carrier Recovery in 65-nm CMOS Technology*. **Huang, Shih-Jou, et al.** 12, s.l. : IEEE, December 2011, IEEE JOURNAL OF SOLID-STATE CIRCUITS, Vol. 46, pp. 3033-3046.

260. *A 65-nm CMOS Fully Integrated Transceiver Module for 60-GHz Wireless HD Applications.* **Siligaris, Alexandre, et al.** 12, s.l. : IEEE, December 2011, IEEE JOURNAL OF SOLID-STATE CIRCUITS, Vol. 46, pp. 3005-3017.

261. *A 60 GHz 16 Gb/s 16QAM Low-Power Direct-Conversion Transceiver Using Capacitive Cross-Coupling Neutralization in 65 nm CMOS.* **Asada, Hiroki, et al.** Jeju : IEEE, 2011. IEEE Asian Solid-State Circuits Conference. Vol. 1, pp. 373-376.

262. *A 60GHz 16QAM/8PSK/QPSK/BPSK Direct Coversion TRanceiver*. **Okada, Kenichi.** Jeju : IEEE, 2011. IEEE International SoC Design Conference, ISOCC. Vol. 1, pp. 20-23.

263. *A 60-GHz 16QAM 11Gbps Direct-Conversion Transceiver in 65nm CMOS.* **Minami, Ryo, et al.** Sydney : s.n., 2012. 17th Asia and South Pacific Design Automation Conference (ASP-DAC). Vol. 1, pp. 467-468.

264. *A Low Loss 60GHz Radio Integrating CMOS Circuits with LTCC AiP*. **Song, In Sang, et al.** Hanzhou : IEEE, 2011. IEEE Electrical Design of Advanced Packaging and Systems Symposium (EDAPS). Vol. 1, pp. 1-4.

265. *3-Gb/s 60-GHz Link With SiGe BiCMOS Receiver Front-End and CMOS Mixed-Mode QPSK Demodulator*. **Ko, Minsu, et al.** 4, s.l. : IEEE, December 2011, IEEE JOURNAL OF SEMICONDUCTOR TECHNOLOGY AND SCIENCE, Vol. 11, pp. 256-261.

266. *Wireless 1.25Gb/s Transceiver Module at 60GHz-Band.* **Ohata, Keiichi, et al.** San Fransisco : IEEE, 2002. IEEE International Solid-State Circuits Conference. Vol. 1, pp. 298-299-468.

267. *A 2.4GHz Fully CMOS Integrated RF Transceiver for 802.11b Wireless LAN Application.* **Kong, Weixin, Ye, Chianghua and Lin, H C.** Atlanta : IEEE, 2004. IEEE Radio and Wireless Conference. Vol. 1, pp. 475-478.

268. *60-GHz-band LTCC module technology for wireless gigabit transceiver applications.* **Maruhashi, Kenichi, et al.** Singapore : IEEE, 2005. IEEE International Workshop on Radio-Frequency Integration Technology: Integrated Circuits for Wideband Communication and Wireless Sensor Networks. Vol. 1, pp. 131-134.

269. **Shoji, Yozo, Choi, Chang-Soon and Ogawa, Hiroyo.** 70-GHz-Band OFDM Transceivers Based on Self-Heterodyne Scheme for Millimeter-Wave Wireless Personal Area Network. *IEEE TRANSACTIONS ON MICROWAVE THEORY AND TECHNIQUES.* October 2006, Vol. 54, 10, pp. 3664-3674.

270. *60-GHz Antenna and 5-GHz Demodulator MMICs for More Than 1-Gbps FSK Transceivers.* **Nakagawa, Tadao, et al.** Milan : IEEE, 2002. 32nd European Microwave Conference. Vol. 1, pp. 1-4.

271. *A 46-GHz Direct Wide Modulation Bandwidth ASK Modulator in 0.13-μm CMOS Technology.* **Chang, Hong-Yeh, et al.** 9, s.l. : IEEE, September 2007, IEEE MICROWAVE AND WIRELESS COMPONENTS LETTERS, Vol. 17, pp. 691-693.

272. *Multi-Mode Modulator and Frequency Demodulator Circuits for Gb/s Data Rate 60 GHz Wireless Transceivers.* **Valdes-Garcia, Alberto, Reynolds, Scott and Beukema, Troy.** San Jose : IEEE, 2007. IEEEE Custom Integrated Circuits Conference (CICC). Vol. 1, pp. 639-642.

273. *A 65nm CMOS Inductorless Triple-Band-Group WiMedia UWB PHY.* **Leenaerts, Domine, et al.** San Fransisco : IEEE, 2009. IEEE International Solid-State Circuits Conference. Vol. 1, pp. 410-411.

274. *A 12-Gb/s, Direct QPSK Modulation SiGe BiCMOS Transceiver for Last Mile Links in the 70-80 GHz Band.* **Sarkas, I, et al.** Greensboro : IEEE, 2009. Annual IEEE Compound Semiconductor Integrated Circuit Symposium, CISC. Vol. 1, pp. 1-4.

275. *A Power Efficient 60 GHz 90nm CMOS OOK Receiver with an On-chip Antenna.* **Kang, Kai, et al.** Singapore : IEEE, 2009. IEEE International Symposium on Radio-Frequency Integration Technology. Vol. 1, pp. 36-39.

276. *A 16-Element Phased-Array Receiver IC for 60-GHz Communications in SiGe BiCMOS.* **Reynolds, Scott K, et al.** Anaheim : IEEE, 2010. IEEE Radio Frequency Integrated Circuits Symposium. Vol. 1, pp. 461-464.

277. *A thirty two element phased-array transceiver at 60GHz with RF-IF conversion block in 90nm flip chip CMOS process.* **Cohen, Emanuel, et al.** Anaheim : IEEE, 2010. IEEE Radio Frequency Integrated Circuits Symposium. Vol. 1, pp. 457-460.

278. *10-Gbit/s QPSK Modulator and Demodulator for a 120-GHz-band Wireless Link.* **Takahashi, Hiroyuki, et al.** Anaheim : IEEE, 2010. IEEE MTT International Microwave Symposium. Vol. 1, pp. 632-635.

279. *An 18-Gb/s, Direct QPSK Modulation SiGe BiCMOS Transceiver for Last Mile Links in the 70–80 GHz Band.* **Sarkas, Ioannis, et al.** 10, s.l. : IEEE, October 2010, IEEE JOURNAL OF SOLID-STATE CIRCUITS, Vol. 45, pp. 1968-1980.

280. *Gbps 60GHz CMOS OOK Modulator and Demodulator.* **Lee, Jae Jin, et al.** Monterey : IEEE, 2010. IEEE Compound Semiconductor Integrated Circuit Symposium (CSICS). Vol. 1, pp. 1-4.

281. **Zheng, Yuanjin, Zhang, Yueping and Tong, Yan.** A Novel Wireless Interconnect Technology Using Impulse Radio for Interchip Commmunication. *IEEE Transactions on Microwave and Techniques.* April 2006, Vol. 54, 4, pp. 1912-1920.

282. *Second Generation 60-GHz Transceiver Chipset Supporting Multiple Modulations at Gb/s data rates.* **Reynolds, Scott, et al.** s.l. : IEEE, 2007. IEEE Bipolar/BiCMOS Circuits and Technology Meeting, BCTM. Vol. 1, pp. 192-197.

283. *A 60 GHz UWB impulse radio transmitter with integrated antenna in CMOS65nm SOI technology.* **Siligaris, A, et al.** Phoenix : IEEE, 2011. IEEE 11th Topical Meeting on Silicon Monolithic Integrated Circuits in RF Systems (SiRF). Vol. 1, pp. 153-156.

284. *A 1mm3 2Mbps 330fJ/b Transponder for Implanted Neural Sensors.* **Mark, M, et al.** Kyoto : Ieee, 2011. IEEE Symposium on VLSI Circuits. Vol. 1, pp. 168-169.

285. *A 65 nm CMOS 4-Element Sub-34 mW/Element 60 GHz Phased-Array Transceiver.* **Tabesh, Maryam, et al.** 12, s.l. : IEEE, December 2011, IEEE JOURNAL OF SOLID-STATE CIRCUITS, Vol. 46, pp. 3018-3032.

286. *Analysis of On-Board Antenna Modules for the Millimeter-Wave Intra-Connect System.* **Ohashi, Sho, et al.** Tokyo : IEEE, 2010. IEEE Components, Packaging, and Manufacturing Technology Symposium; CPMT. Vol. 1, pp. 1-4.

287. Role of Packaging in Moore's Law. *Ray Prasad Consultancy Group.* [Online] Ray Prasad Consultancy Group. [Cited: 3 October 2012.] http://www.rayprasad.com/home/page_7/columns/moore.html.

288. *A 750 Mb/s, 12 pJ/b, 6-to-10 GHz CMOS IR-UWB Transmitter With Embedded On-Chip Antenna.* **Kulkarni, Vishal V, et al.** 2, s.l. : IEEE, February 2009, IEEE JOURNAL OF SOLID-STATE CIRCUITS, Vol. 44, pp. 394-403.

289. *Wireless DC Voltage Transmission Using Inductive-Coupling Channelfor Highly-Parallel Wafer-Level Testing.* **Yoshida, Yoichi, et al.** San Fransisco : IEEE, 2009. IEEE International Solid-State Circuits Conference. Vol. 1, pp. 470-471.

290. **Mellor-Crummey, John.** https://www.clear.rice.edu/comp422/lecture-notes/comp422-2014-Lecture11-NetworkTopologies.pdf. *https://www.clear.rice.edu.* [Online] 2014. [Cited: 9 6 2014.] https://www.clear.rice.edu/comp422/lecture-notes/comp422-2014-Lecture11-NetworkTopologies.pdf.

291. Exposed node problem. *Wikipedia.* [Online] Wikipedia, 16 June 2014. [Cited: 26 June 2014.] http://en.wikipedia.org/wiki/Exposed_node_problem.

292. **Knuth, Donald Ervin.** *The art of computer programming, Volume 3: Sorting and searching.* s.l. : Addison-Wesley, 1998. Vol. 3. ISBN-13: 078-5342896855.

293. *Formal Proof—The Four-Color Theorem.* **Gonthier, Georges.** 11, s.l. : the American Mathematical Society, November 2008, Notices of the American Mathematical Society, Vol. 55, pp. 1382–1393.

294. **Westman, H. P.** *Reference data for radio engineers.* New York : s.n., 1957.

295. **MacDougall, Myron H.** *Simulating Computer Systems: Techniques and Tools.* s.l. : MIT Press, 1987.

296. **Delaney, William and Vaccari, Erminia.** *Dynamic Models and Discrete Event Simulation.* s.l. : Dekker INC, 1988.

297. **Pidd, Michael.** *Computer simulation in management science– fourth edition.* s.l. : Wiley, 1998.

298. **Banks, Jerry, et al.** *Discrete-event system simulation – fourth edition.* s.l. : Pearson, 2005.

299. **Nutaro, James J.** *Building software for simulation: theory and algorithms, with applications in C++.* s.l. : Wiley, 2010.

300. Discrete event simulation. *Wikipedia.* [Online] Wikimedia Foundation, Inc., 28 March 2014. [Cited: 18 October 2014.] http://en.wikipedia.org/wiki/Discrete_event_simulation#cite_note-1.

301. **Stanley, Gershwin B.** Introduction to Simulation. *MIT OpenCourseWare, OCW.* [Online] 2010. [Cited: 18 October 2014.] http://ocw.mit.edu/courses/mechanical-engineering/2-854-introduction-to-manufacturing-systems-fall-2010/lecture-notes/MIT2_854F10_sim.pdf.