A Dynamic Link Speed Mechanism for Energy Saving in Interconnection Networks

A DISSERTATION PRESENTED

BY

NGUYEN HOANG HAI

TO

THE COMPUTER ARCHITECTURE & OPERATING SYSTEMS DEPARTMENT

THESIS SUBMITTED UNDER THE SUPERVISION OF

DR. DANIEL FRANCO PUNTES

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

IN THE SUBJECT OF

COMPUTER SCIENCE

UNIVERSITY AUTONOMA DE BARCELONA

BELLATERRA, BARCELONA

JULY 2014

A Dynamic Link Speed Mechanism for Energy Saving in Interconnection Networks

Thesis submitted by Nguyen Hoang Hai for the degree of Philosophiae Doctor by University Autonoma de Barcelona with the supervision of Dr. Daniel Franco Puntes, at the Computer Architecture and Operating Systems Department, under the High Performance Computing doctoral program.

Barcelona, July 2014

Supervisor	PhD Student
Dr. Daniel Franco Puntes	Nguyen Hoang Hai

Abstract

The growing processing power of parallel computing systems requires interconnection networks a higher level of complexity and higher performance, thus they consume more energy. A larger amount of energy consumed leads to many problems related to cost, cooling infrastructure and system stability. Link components contribute a substantial proportion of the total energy consumption of the networks.

Several proposals have been approaching a better link power management. In this thesis, we leverage built-in features of current link technology to dynamically adjust the link speed as a function of traffic. By doing this, the interconnection network consumes less energy when traffic is light. We also propose a link speed aware routing policy that favors high-speed links in the process of routing packets to boost the performance of the network when the energy saving mechanism is deployed.

The evaluation results show that the networks deploying our energy saving mechanism reduce the amount of energy consumption with the expense of an increase in the average packet latency. However, with the link speed aware routing policy proposal, our mechanism incurs a less increase in the average packet latency while achieving similar energy saving, compared with other conventional approaches in literature.

Resumen

El crecimiento de la potencia de procesamiento de los sistemas de computación paralelos requiere redes de interconexión de mayor nivel de complejidad y un mayor rendimiento, por lo que consumen más energía. Mayor cantidad de energía consumida conduce a muchos problemas relacionados con los costos, la infraestructura y la estabilidad del sistema de refrigeración. Los enlaces de la red contribuyen en una proporción sustancial al consumo total de energía de dichas redes.

Varias propuestas se han ido acercando a una mejor administración de la energía de los enlaces. En esta tesis, aprovechamos ciertas características integradas en la tecnología actual de los enlaces para ajustar dinámicamente la velocidad de los mismos en función del tráfico. De esta manera, la red de interconexión consume menos energía cuando el tráfico es ligero. También proponemos una política de enrutamiento que tiene en cuenta la velocidad del enlace que favorece las conexiones de alta velocidad en el proceso de enrutamiento de paquetes para aumentar el rendimiento de la red cuando se despliega el mecanismo de ahorro de energía.

Los resultados de la evaluación muestran que las redes que usan el mecanismo de ahorro de energía aquí propuesto, reducen el consumo de energía a expensas de un aumento en el promedio de latencia de paquetes. Sin embargo, con la propuesta de política de enrutamiento que tiene en cuenta la velocidad del enlace, nuestro mecanismo incurre en un menor aumento en el promedio de la latencia de paquetes mientras se logra un ahorro de energía similar, en comparación con otros enfoques convencionales de la literatura.

Contents

Listing of figures

Acknowledgments

It would not have been possible to write this doctoral thesis without the help and support of the kind people around me, to only some of whom it is possible to give particular mention here.

First and foremost, I would like to show my special appreciation and my deepest gratitude to my academic advisor Dr. Daniel Franco, who has given me great support, constructive feedback and thorough understanding to keep me diligent and moving forward in my years of doing PhD.

I also would like to thank Dr. Emilio Luque for your inspiration and guidance from the first day when I came to the department, thank you for encouraging my research and for allowing me to grow as a research scientist. I would like to send my special thanks to Dr. Dolores Rexachs for the kindness, caring and selfless support for all the time I have been in the department.

I feel grateful to Gonzalo Zarza, who has given me a lot of hands-on experience in network simulations and always first came up my mind every time I had problem with my work when I was in the first year of the PhD program. And thank you Carlos Nunez for your advices and for investing so much time and effort helping me in my research work.

It is my pleasure to thank Arindam Choudhury, Javier Panadero, Claudio Marquez, Alejandro Chacon, Zhongwei Xu for being the friends I spend most of the time with in the last 3 years with many healthy conversations about research and

career prospective. I thank all the colleagues and staff members of Computer Architecture and Operating Systems Department for all the help and support. I feel fortunate and privileged to have a chance to work in such a supportive and encouraging environment like CAOS.

Last but not least, I would like to thank my family and Vietnamese friends for their support and encouragement during my time at UAB as well as for my future endeavors.

The interconnection network is the heart of parallel architecture.

Chuan-Lin and Tse-Yun Feng

1 Introduction

1.1 Context

The study field of this thesis work is interconnection networks. Interconnection networks can be thought as programmable physical communication systems. These systems comprise links and routers that connect each other to perform the communication needed by various components of communication systems. With the ever-increasing in the density of transistors, the ever-increasing power consumption and the diminishing returns in performance of uniprocessor architectures, interconnection networks are critical in nowadays' digital world since they are the bottleneck to increase performance of modern digital systems [46, Chapter 1] [14]. Interconnection networks play an increasingly important role because not only they provide external connectivity, they also connect many components inside the box, such as among microprocessors. Interconnection networks traditionally have

been used to connect multi-computer parallel distributed systems, but today it is common to see the interconnection networks appearing in single computing system to provide connectivity at many levels: I/O units, boards, chips, modules and blocks inside chips [31].

Interconnection networks can be grouped into 4 major networking domains, depending on the number and the proximity of devices to be interconnected: OCNs, SANs, LANs, and WANs.

- On-chip networks (OCNs), a.k.a. network-on-chip (NoC): Interconnect microarchitecture functional units, register files, caches, compute tiles, processor and IP cores, chips or multichip modules. The number of interconnected devices around 10 devices (in future, possibly 100s), interconnect distance on the order of centimeters.

- System/storage area networks (SANs): Multiprocessor and multicomputer systems, with hundreds to thousands of devices interconnected, interconnect distance typically on the order of tens of meters, but some with as high as a few hundred meters.

- Local area networks (LANs): Interconnect autonomous computer systems with hundreds of devices interconnected (1,000s with bridging). Maximum interconnect distance on the order of few kilometers, but some with distance spans of a few tens of kilometers.

- Wide area networks (WANs): Interconnect systems distributed across the globe, many millions of devices interconnected with maximum interconnect distance of many thousands of kilometers.

Single thread performance improvement has been limited in the past several years. In contrast, the demand for performance continues to increase. Large-scale parallel computers have opened the door to many grand challenge problems [1]. Large-scale computers increase the number of processors, thus larger interconnection networks are being designed for those systems.

Interconnection networks should be designed to transfer the maximum amount of information within the least amount of time (and cost, power constraints) so as not to bottleneck the system. This work focuses on reducing energy consumption of the interconnection network systems, taking into account the impacts on network performance.

1.2 Motivation

The growing scale of computer systems and data centers has deteriorated issues related to energy consumption and cooling infrastructures. Recently, the design of computer systems has changed the focus purely from performance to good performance at lowest possible energy consumption because the need for energy efficient networking has become evident [37, 63, 64]. This leads to an intensive effort in the research to reduce the energy consumption of many computer systems including servers, networking infrastructure and storage. Traditionally, the high proportion of energy consumption associates with the computing and storage devices. The increasing in connectivity, the proliferation the networking devices with higher bandwidth increase the contribution of energy consumption of networking devices, which is expected to account for up to 30% of the total energy consumption [39]. The term "green computing" appears more ubiquitous and now it becomes a requirement in the computer system design world [18].

The performance of the computing system is measured by *FLOPS* (Floating Point Operations Per Second). The list of top 500 world fastest computer [6] with the fastest computer reaches 33.86 petaflop. A picture of a supercomputing system named Titan [2] is shown in Fig. 1.2.1 with a massive number of processing units connected together by means of communication medium. The energy concern has received growing attention from computing system designers, the greenness of the computing system is measured by *FLOPS per watt*. The list of top green computing system which promotes systems with high value of *FLOPS per watt* [5] raises the awareness of the growing importance of energy efficient computer systems.

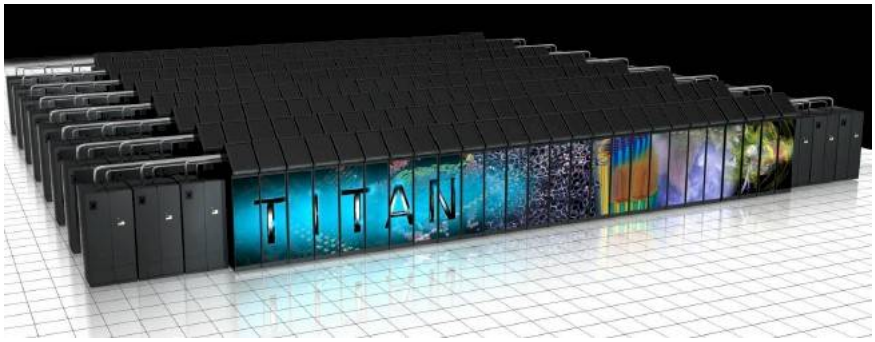


Figure 1.2.1: Titan Supercomputer

In the world of embedded systems or on chip networks, a better energy efficient management can prolong the life of the battery and the self-sustainability [27]. In the high performance computing world, energy dissipation significantly affects the operational costs and also the reliability [23]. Empirical data show that a 10-degree increase in temperature results in a doubling rate of system failure, which reduces the reliability of the system. The energy cost and the heat dissipation problems in interconnection networks necessitate future network systems be built with much more efficient power than today and it has become a priority design requirement for implementing scalable interconnected parallel systems [18].

1.3 Description of the Problem

Interconnection networks account for a significant portion of the energy consumed by communication systems. Gunaratne et al. [29] estimated that in USA just the network interface controllers (NICs) consume hundreds of millions of US dollars in electricity every year. The studies reported in [13] also stated that the energy consumed by the infrastructure of communication networks in the United States was around 5 to 24 *TWh/year*, with an exorbitant cost associated. There are several examples showing the significant contribution of the interconnection networks to the total energy consumption of the systems. For example, a 8-port switch 12X InfiniBand IBM estimates to consume 31W with the link components consume 64% (20W) [42]. The new specifications for point to point interconnections in new ar-

chitectures manufacturers like Intel (Quickpath) [4] and AMD (HyperTransport) [3] also define mechanisms for energy management. These systems soon will be common in most computer systems at the global level.

Vetter and Mueller show that applications scale more efficiently to large number of processors using point-to-point communication if the number of destination is relatively small [60]. It means that many applications use just a small portion of the network resources [51].

Among many components, links account for a substantial portion of the total energy usage of an interconnection network [54]. The average link utilization in many communication systems - already quite low - tended to go down as the link speed increases [35]. For most traffic patterns, link utilization distributes non-uniformly over an interconnection network system, in both temporal and spatial terms [55]. During operation, some phases require high link usage due to the high communication volume, whereas in other phases links almost idle. At a given time, links also separate into two groups: one group includes links that are in use, the other one includes those are idle.

In current commercial systems, a link consumes the amount of energy that is practically insensitive to the load it carries; it burns the same amount of energy even when it idles. For this reason, while around $15 - 30\%$ of the energy is allocated for the network [39, 44], this number can go to 50% when processors are not highly in use [8]. Thus, link components consume energy proportionally to the traffic load become a desired behavior [8].

1.4 Thesis Outline

This thesis organizes its content with different chapters as following:

Chapter 2: Objectives

This chapter formulates the objectives of the thesis work.

Chapter 3: Thesis Background & Previous Work

This chapter introduces some basic concepts about interconnection networks. Then, it overviews the state of the art, and the approaches for link power management.

Chapter 4: Performance-Aware Dynamic Link Speed Mechanism

This chapter describes the methodology and the proposals which compose the dynamic link speed energy saving mechanism.

Chapter 5: Evaluation

This chapter shows & analyzes experimental evaluation with different network scenarios.

Chapter 6: Conclusions

This chapter concludes the work and presents some open lines following the work in this thesis.

Man is still the most extraordinary computer of all.

John F. Kennedy

2 Objectives

2.1 Objectives

Chapter 1 overviews the context, highlights the increasing necessity in designing the energy efficient interconnection networks in different systems - from data centers to inside-the-box systems. Link component accounts for a significant portion of the total amount of energy consumption by interconnected systems. This thesis aims to reduce the energy consumed by link component of the interconnection network, taking into account the impacts to the performance. The energy reduction comes from the adjust of the link speed and corresponding energy consumption, thus making it proportional as a function of traffic. More specifically, this work aims to propose an **energy saving mechanism** that achieves following objectives:

- Leveraging current technology to minimize the introduction of extra hard-

ware components

- Maximizing the use of current network infrastructure without introducing new changes such as topologies, routing algorithms.
- Minimizing the degradation of the network performance when the energy saving mechanism is deployed.

With current technology where a physical link composes of several lanes, the adjusting of link speed does not require additional hardware components since the number of active lanes directly translates to the speed of the link. The topology of the network does not change, thus we do not need to provide a technique to guarantee the connectivity of the network and the routing algorithms can remains unchanged. This thesis also proposes several approaches to minimize the impact of the energy saving mechanism to the performance of the network. The thesis also illustrates the effectiveness of the mechanism by carrying out and analyzing network experiments.

To hypothesize and formulate the mechanism that achieves these objectives. The work in this thesis follows the research work method as stated in the following sections.

2.2 Research Method

The research in this thesis is oriented to the design, implementation and evaluation of a dynamic link speed energy saving mechanism, taking into account the network performance. The research work is framed in the academic program of applied research of the Autonomous University of Barcelona.

1. **Existing theories and observations**. Pose the question in the context of existing knowledge, theory and observations.
2. **Hypothesis**. Formulate a hypothesis as a tentative answer.
3. **Predictions**. Deduce consequences and make predictions.

4. **Test and new observations**. Test the hypothesis in a specific experiment/theory field.

5. **Old theory confirmed within a new context or new theory proposed**. When consistency is obtained the hypothesis becomes a theory and provides a coherent set of propositions that define a new class of phenomena or a new theoretical concept.

As a rule, the loop 2—3—4 is repeated with modifications of the hypothesis until the agreement is obtained, which leads to 5. If major discrepancies are found the process must start from the beginning. The results of stage 5 have to be published. Theory at that stage is subject of process of natural selection among competing theories. The process can start from the beginning, but the state 1 has to be changed to include the new theory/improvements of old theory [20].

Indeed, as system complexity and integration continues to increase, many designers are finding it more efficient to route packets, not wires.

Bill Dally

3

Thesis Background & Previous Work

3.1 Thesis Background

The network designer must implement the *topology*, *routing* and *flow control* of the network within technology constraints. The efficiency of interconnection networks derives from the fact that the network resources are shared. Instead of dedicated channels between every pair of nodes to deliver messages, they are delivered by a set of shared routers and shared links. The disposition of nodes defines the network *topology*. A message is delivered between nodes by making several *hops* across shared channels. Given the topology, a message can take several paths to go from a particular node to their destination. *Routing* decides which path from those paths the message actually follows. *Flow control* dictates which messages are given network resources over time.

3.1.1 Topology

The way nodes are arranged and connected by channels constitutes the topology of the network. It provides means for messages traveling from the origins to the destinations. The topology is modeled as a graph with vertices represent nodes of the networks, and edges represent links connecting them. Following the classification of Duato et al. [22, Ch. 1] and Dally & Towles [17, Ch. 3], network topologies are divided into three main groups: *shared-medium networks, direct networks, and indirect networks.*

Shared-medium networks

In this type of networks, the physical medium is shared by all communication devices. There are two major classes of shared-medium networks: *Local Area Networks* and *Backplane bus*. *Local Area Networks* typically constructs of computer networks that spread in less then a few kilometers with three main technology: *Contention Bus, Token Bus* and *Token Ring* as shown in Fig. 3.1.1. Some examples of shared-medium networks are shown in Fig. 3.1.2. These networks were first used in early parallel computing systems but soon no longer in use due to the performance and scalability issues.

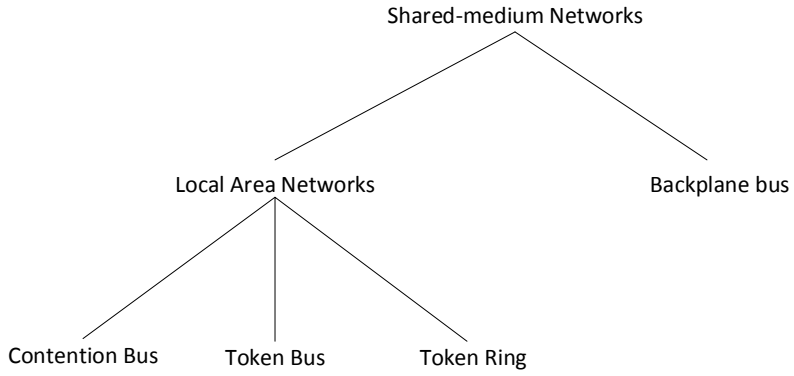
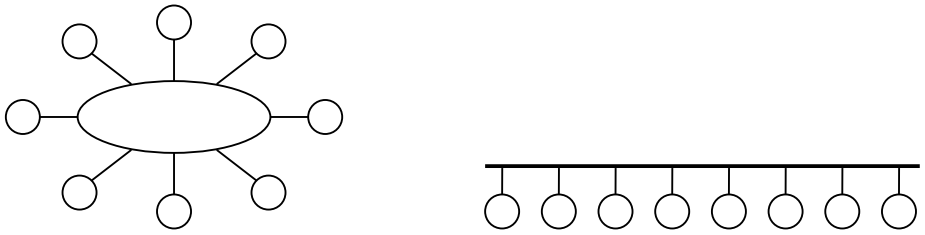


Figure 3.1.1: Classification of shared-medium network topologies



(a) Local Area Network **(b)** Bus-Based Network

Figure 3.1.2: Examples of shared-medium network topologies

Direct networks

Shared-medium networks are not scalable because the shared-medium becomes bottleneck when more processing nodes are added. The *direct network* or *point-to-point network* scales much better to a large number of processing nodes [22, Ch. 1]. This type of networks consists of a set of nodes; each node directly connects to a number (often small) of other nodes in the network. Each node is a programmable computer system that has its own processor, local memory and supporting devices. A common component of this network is a *router*, which processes messages among nodes. Every router has direct links to neighboring routers. Because of this, this type of network is also known as *router-based networks*.

Typically *direct network* is often modeled as a graph *G(N,C)*, where *N* vertices of the graph represents the set of processing nodes and *C* represents the set of communication links. Most of implemented direct networks have *orthogonal* topologies where nodes are arranged in an orthogonal n-dimensional space, and every link is arranged in a way that produces a displacement in a single dimension. *Orthogonal* topology further divides into *strictly orthogonal* topology - where every node has at least one link crossing each dimension such as *n-dimensional mesh* or *k-ary n-cube*, and *weakly orthogonal* topology - where some nodes may not have any link in some dimensions such as a *binary tree*. Fig. 3.1.3 and Fig. 3.1.4 show the classification and examples of direct networks.

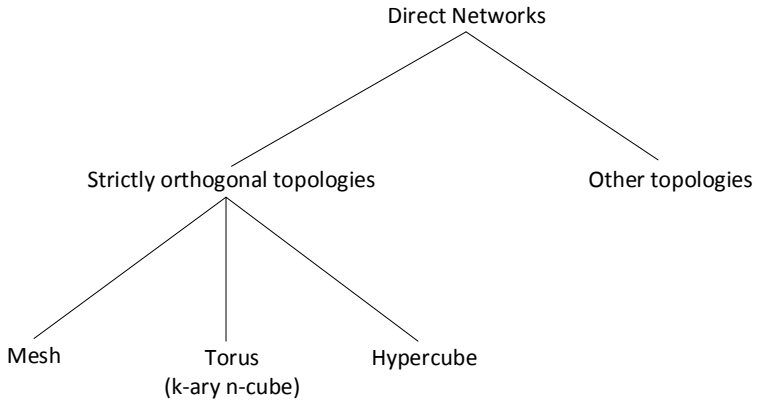
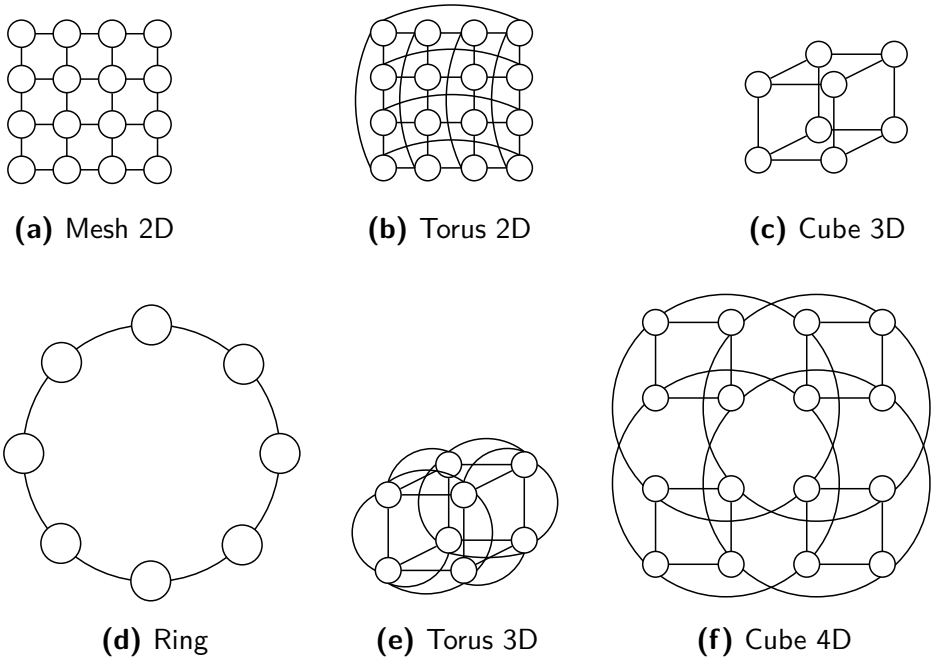


Figure 3.1.3: Classification of direct network topologies



(a) Mesh 2D **(b)** Torus 2D **(c)** Cube 3D

(d) Ring **(e)** Torus 3D **(f)** Cube 4D

Figure 3.1.4: Examples of direct network topologies

Indirect networks

Instead of providing a direct connection among some nodes, the communication between any two nodes can be carried out by means of one or more *switches*. Each

node has a network adapter to connect to a network switch. Each switch has a set of *ports*. Each port has one input and one output link. Fig. 3.1.5 and Fig. 3.1.6 shows the classification and examples of this type of network.

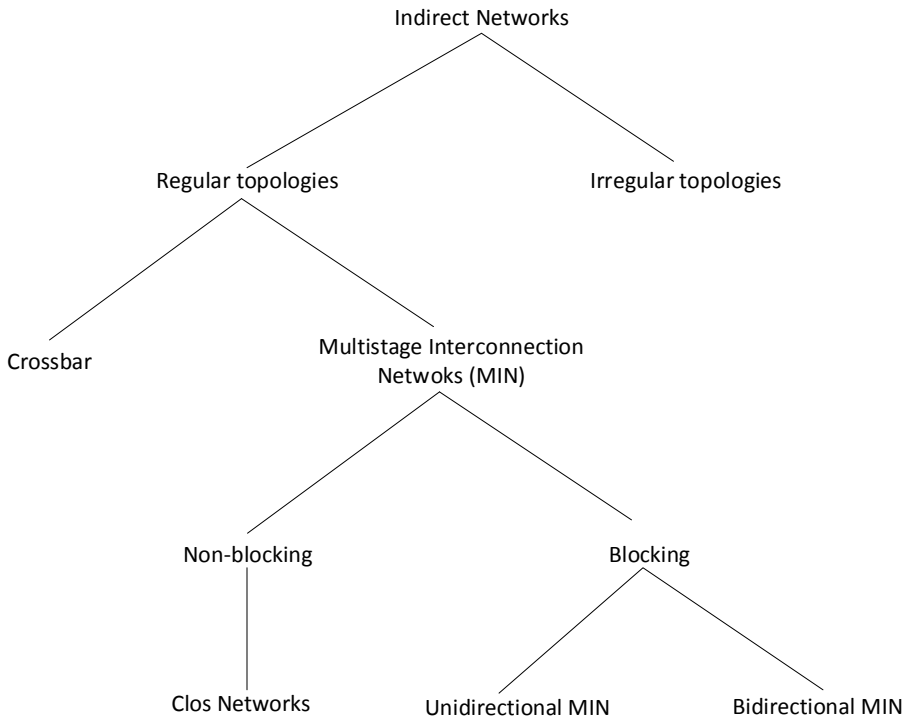
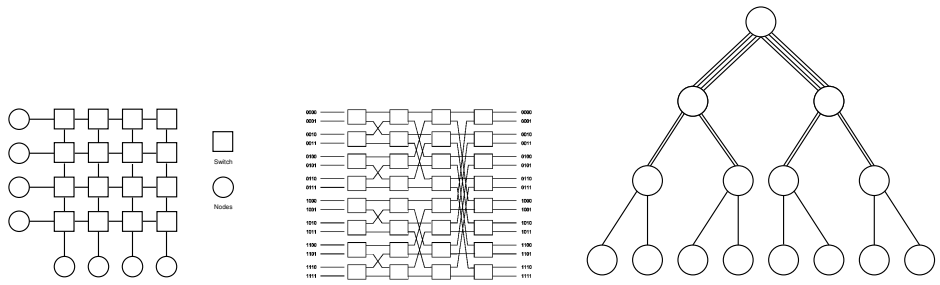


Figure 3.1.5: Classification of indirect network topologies

3.1.2 Routing

Routing is the process of specifying the path for a packet to take from a source terminal node to a destination terminal node. Each path is an ordered list of intermediate routers and links connecting them. For every pair of source-destination the routing algorithm may produce a set of possible paths. Duato et al. [22, Ch. 1] classified routing algorithms into four main groups based on *number of destina-*



(a) Crossbar **(b)** MIN Butterfly **(c)** Fat tree

Figure 3.1.6: Examples of indirect network topologies

tions, routing decisions, implementation and *adaptivity* as shown in Fig. 3.1.7 and Fig. 3.1.8.

Number of destinations

Routing algorithms provide paths for packets that are due for one destination is called *unicast* routing algorithms, while those having multiple destinations are called *multicast* routing algorithms.

Routing decisions

Routing algorithms in this group differentiated by *whom* and *where* the routing decisions are made. If those decisions are made by a centralized controller then the routing algorithm is called *centralized*. If the decisions are made in a non-centralized manner, then if the decision is made by the source node before packet injection then the routing algorithms are called *source-based* routing, if the decision is made in a distributed manner by intermediate routers then the routing algorithms are called *distributed* routing. *Multiphase* routing is the type of routing algorithms that combine both *source-based* routing and *distributed* routing.

Implementation

The routing decision might be made based on the information that is stored in advance, called *routing table*. The routing decision also might be a *routing function* determining the path for each source-destination pair.

Adaptivity

Adaptivity of a routing algorithm refers to how the routing algorithms determine the path between a source-destination pair. For each pair of source-destination, *deterministic* routing algorithm always chooses the same path, even there might be several paths; *oblivious* routing algorithm picks a path without regarding to the network state; *adaptive* routing algorithm determines paths taking into account the state information of the network at the moment the decision is made.

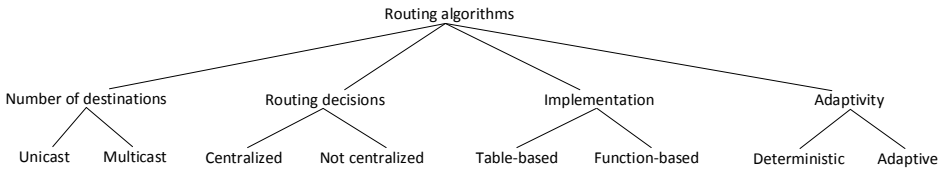


Figure 3.1.7: Routing Algorithm

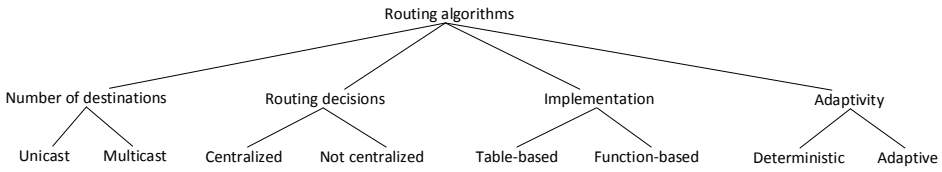


Figure 3.1.8: Adaptive Routing Algorithm

3.1.3 Flow Control

Flow control manages the allocation of resources to packets as they progress along intermediate routers. The key resources in the interconnection networks are the *links* and the *buffers*. The flow control is a protocol that is used for transmitting and receiving data, employing request/ acknowledgement signaling to ensure successful transfer and availability of buffer space at the receiver [22, Ch. 2], [17, Ch. 13]. The most common protocols are *credit-based* and *on/off*.

Credit-based Flow Control

The sender keeps track of the number of free *flits* buffers, or *credits* of the buffer space at the far end of the link. Each time a flit is sent, the credit count is reduced. If the count is zero, then flits can not move forward, they need to wait until the count number increases, meaning that there is available space at the buffer of the receiver.

On/Off Flow Control

The sender sends flits and stops sending flits based on a signal sent from the receivers, when a receiver can not accept more flits they send the sender an *off* signal that hints the sender to wait until the receiver processes flits on their queues. Until then, the receiver sends an *on* signal permitting the sender to send more flits.

3.2 Previous Work

Several works have investigated the power model of interconnection networks, profiling the energy behaviors of network routers and links [47, 61, 62].

Hoefler et. al. [32] combined the hardware & software techniques to throttle the use of network resources. The hardware techniques are those that are widely used in other components of communication systems such as microprocessors, more specifically the methods proposed were: dynamic link-speed reduction, receiver modification, deep sleep states.... The software techniques involve algorith-

mic power saving options to overlap the communication and computation which leads to a steady use of the interconnect, and in turn reduces the required bandwidth and/or improves application performance and wait times. The two techniques together reduce the time to solution, avoid bursty traffic and thriftily use network resources. Yong Dong & Juan Chen [21] save the energy consumption of the interconnection networks by a dynamic model that estimates the network energy consumption of various MPI algorithms at run time and recommends an algorithm with the least energy consumption.

Following the software approach, Li Shang et. al. [53] uses routers as controllers to constraint network energy consumption. For routers, the local view of networks states limits the applicability of this proposal. Soteriou et. al. [56] show a possible severe performance degradation of hardware approaches and propose the use of parallelizing compilers to manage the energy of links. However, this approach is deemed to be less effective. Besides, compilers do not have enough information about input dependent message flow of an application thus can not manage the energy effectively for such applications.

Conner et. al. [16] demonstrate the link shutdown opportunities during collective communications in 3-D Torus networks to save energy, reducing the overall system energy by approximately 15-28%. Laros et. al. [40] illustrate results on potential energy saving using CPU and network scaling by post-processing the data collected from the monitoring system of Cray XT machines. Their works using real systems (instead of simulations) and real applications show the importance and potential of network energy management for interconnected computing systems.

Jian Li et. al. [42] proposed that the energy reduction from the interconnection components can also be absorbed by the associated processing units. The processing units increase their processor frequency for higher performance.

The studies on this thesis focuses on the link energy management with the dynamic link speed approach, which differs from other proposals that might applied on different network components (e.g. switch buffers, allocators, routing units...). In general, there have been 3 main approaches for link energy management as de-

scribed in following sections:

3.2.1 Dynamic Voltage Scaling

Technique Description

Dynamic Voltage Scaling (DVS) is a popular energy saving mechanism which originates for microprocessors. The DVS mechanism exploits the different requirements for frequency and voltage of the microprocessors corresponding to different workloads. Workload also varies in the process of running applications in interconnection networks. Thus this mechanism is adopted to save energy in interconnection network systems [38, 52]. The heart of the mechanism is the transition policy which prescribes when to adjust the voltage and how much to adjust. Link bandwidth is linearly proportional to the link frequency, as described in the Eq. 3.1. Where b is the link bandwidth, w is the width of the link and f is the link frequency.

$$b = w.f \tag{3.1}$$

The reduction in link frequency leads to the reduction in the link bandwidth, which can degrade the network latency and the throughput. As illustrated in Fig. 3.2.1, the graph of the network latency over the offered traffic shifts up and left as the link frequency decreases.

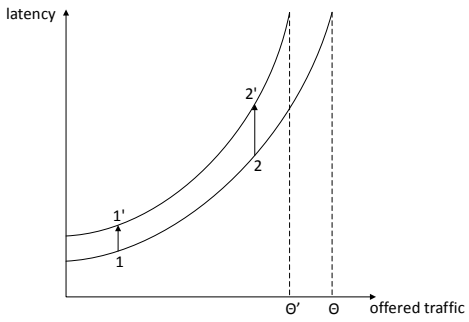


Figure 3.2.1: Latency and throughput penalty as a result of voltage scaling

This mechanism requires a transition policy that balances the tradeoff between energy saving and performance penalty. A distributed history-based Dynamic Volage Scaling policy is proposed in Eq. 3.2.

$$LU = \frac{\sum_{t=1}^{H} A(t)}{H} \tag{3.2}$$

Where A(t) = $\begin{cases} 1 \text{ if traffic passes the link in the link cycle } t \\ 0 \text{ if no traffic passes the link in the link cycle } t \end{cases}$

and H is the history window size, in other words the mechanism monitors the last H cycles to calculate the LU.

$$BU = \frac{\sum_{t=1}^{H} F(t)/B}{H} \tag{3.3}$$

Where $F(t)$ is the number of input buffers that are occupied at time t, and B is the size of input buffer.

$$Par_{predict} = \frac{weight * Par_{current} + Par_{past}}{weight + 1} \tag{3.4}$$

Where $Par_{predict}$ is the predicted indicator, $Par_{current}$ is the indicator of the current history period and Par_{past} is the indicator of the past history period.

Each port monitors the workload history, predicts the future workload and dynamically adjusts its frequency and voltage to the minimum required by the predicted future workload. Link Utilization, Input Buffer Utilization are monitored in 3.2, 3.3 and then predicted with a weighted relation in 3.4. A pair of thresholds, *TH_low* and *TH_high* are used as the criteria to adjust the link speed, *B_congested* is used as the threshold of buffer utilization to determine whether the network is predicted to be congested.

A gate-level netlists combined with real-delay timing simulation is carried out to evaluate the proposed mechanism. The self-similar traffic pattern [57] was injected to the network. On average, a 4.6X energy reduction is achieved with a 15.2% increase in average packet latency and 2.5% throughput reduction.

Critique of the technique

The technique has a significant energy saving ($4.6X$ on average). It also bases on local information thus hardware overhead is lower than other policies that require global network information. Using self-similar traffic which has been empirically reported to represent real network traffic [48, 59, 65] further support the effectiveness of the DVS policy.

The disadvantages of the DVS is the complexity of hardware components with adaptive power-supply regulators and frequency synthesizers, which results in a high cost and limits the applicability of the DVS mechanism.

3.2.2 Dynamic Link Shutdown

Technique Description

The idea of the DVS mechanism in section 3.2.1 can be put to the extreme by the Dynamic Link Shutdown (DLS) mechanism. This mechanism turns off some underutilized links when the traffic load is light, and turns them on again when the traffic increases as proposed in [10, 30, 50, 54, 58, 66]. Link utilization is monitored as described in Eq. 3.2. When the link utilization drops below the "off" threshold, it drains all the remaining flits and transitions to the off state. When the average link utilization of the remaining "on" links of a router exceeds the "on" threshold, an off link will be turned back on.

Turning a link on takes a significant transitioning time, the value of the transitioning time depends on the link technology. Thus degradation in network performance might be observed. The off links also reduce the network path diversity or even make the network disconnected. Hence the mechanism proposes a power-performance connectivity graph to select candidates for on/off links. The goal of the graph is to ensure that the entire network is still connected when all on/off link candidates are off. To avoid deadlock, the routing algorithm must takes into account the state of links, thus the routing algorithm is non-minimal. The hop count component of packet latency increases and is a function of the currently operational links.

Soteriou and Peh conduct an 8-ary 2-mesh topology in their network simulation to evaluate the DLS mechanism [54]. To guarantee connectivity, the topology specifies a maximum of 2 outgoing links per inner router as candidates for on/off links. Outer routers do not have any outgoing on/off link candidates. A physical link includes 2 virtual channels, with 1 virtual channel uses non-minimal east-last routing and the other uses non-minimal west-last routing. Uniform traffic was injected to the network, link transition time (t_{sw}) is set to 1000 cycles. The simulation observes a 34% energy saving with 48.5% increase in average packet latency. The impact of t_{sw} is studied with a varying of offered traffic. With t_{sw}=100 cycles, energy saving achieved is 35.9%, while t_{sw}=1000 cycles the energy saving decreases to 35.9%. When t_{sw}=10000 cycles, energy saving drops to 30.2%. Note that because of the varying of offered traffic in the second experiment, the energy saving is different from the first experiment.

Critique of the technique

Unlike the DVS technique, the on/off links do not consume energy when they are off. The DLS technique requires simpler hardware and it can operate faster. As a result, it is easier and cheaper to deploy. However, the requirement to maintain the entire network connected limits the theoretical energy saving of the mechanism. Although the deadlock free routing in simulation is fairly simple, a deadlock avoidance routing for real interconnection networks for the proposed method in [54] may complicate the system designs. In fact, both power-performance connectivity graph and deadlock avoidance routing are network type specific. Such specificity makes the DLS approach becomes difficult to adopt. A methodology to systematically generate candidates for on/off links and routing algorithms for a given network topology is required. The energy saving in this mechanism is highly dependent on the transition latency of the on and off links. We can notice the energy saving and performance number are less impressive than those stated in the DVS technique in the previous section. However, note that the experiments in this section use a network simulator, whereas in the previous section use gate-level

netlist. Also, the experiments used different traffic patterns and might use different network configuration for the evaluation. Thus, we should not make any direct comparison. In addition, uniform traffic does not represent realistic network traffic. As a result, the latency increase and energy saving numbers stated here do not reflect exactly in a real world environment.

3.2.3 Dynamic Link Width

Technique Description

The Dynamic Link Width (DLW) mechanism dynamically adjusts the bandwidth of links by narrowing or increasing the width of the links, thus adjust the energy consumption [9]. For simplicity and productivity, the mechanism adjusts the link speed by either doubling or halving the current width. The mechanism monitors the link utilization as described in Eq. 3.2 and adjusts the link width with the two "on" and "off" threshold values. The hardware model implementation of this mechanism is shown in Fig. 3.2.2.

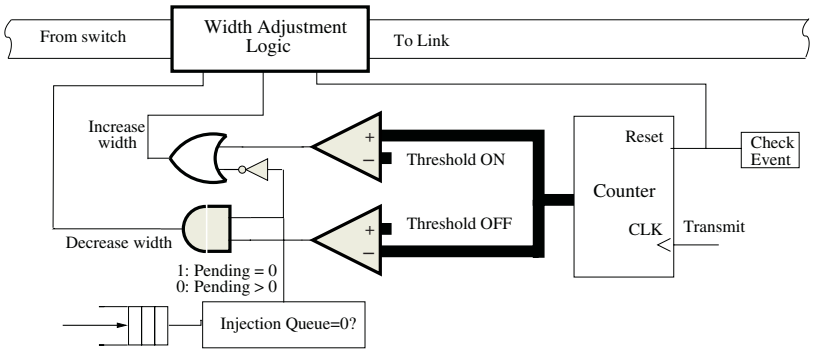


Figure 3.2.2: Dynamic Link Width Hardware Model Implementation

The simulations with 32-ary 2-cube (with 1024 nodes) and an 8-ary 3-cube torus (512 nodes) were carried out to evaluate the mechanism. The networks configure a

minimal adaptive routing and wormhole flow control. With light traffic load, most of the link utilization is around 25%, resulting a 4x deduction in energy (26% of the energy consumption of default systems), with a 3x increase in average packet latency. With a higher traffic load, a 2x deduction is obtained with a 2x increase in latency. When the network is highly congested, the energy saving approaches to zero and the latency matches the default network systems.

Critique of the technique

The DLW technique has several advantages when compared to the DLS technique. First, unlike the DLS technique where routing is restricted because of off-links, here we can always use the same routing algorithms. Thus, we do not need to create a new connectivity graph and a new complex on/off algorithms to avoid deadlock, which simplifies hardware design. Second, the minimal hop count unchanged also makes the latency and energy estimation easier. Third, this technique has a more granular energy saving compared to the on/off techniques. Forth, it offers a lower switching overhead than the DLS technique. If we put the DLW in comparison with the DVS, the DLW also has the advantage that the hardware design is simpler. Overall, the DLW achieves a satisfactory energy saving with a simple design. However, this technique has the disadvantage in the latency penalty. The reported 3x increase in latency might not be tolerable in many applications.

3.2.4 Discussion of different techniques

Summary of critiques of existing techniques

All three techniques described in previous sections are related to each other. The DLS technique can be considered as an extreme case of either DVS or DLW. In the case of DVS, if the technique alternates the link frequency between the maximum value and zero, then DVS turns into DLS. Similarly, in the case of DLW, if the technique uses two values for the width of a link: maximum width and zero width, then the DLW is becomes DLS.

All three techniques make a tradeoff between performance and energy saving. All of them decrease throughput and increase the average packet latency of the network by reducing the width or the frequency and consequently the bandwidth of a link. DLS also worsens the average minimal hop counter H_{min} in the network since some packets have to bypass the off links by using non-minimal routing.

However, each technique has several different strong points and weak points. Fig. 3.2.3 shows these three techniques grouped by different metrics, which are all important characteristics for measuring the quality of the techniques.

Technique/Metric	DVS	DLS	DLW
Energy saving	Limited by the number of levels of voltage/frequency	Limited by topology because a number of links must be on to keep the network connected. This number is topology dependent	Limited by the number of available link width configuration
Latency Penalty	Low penalty for lightly loaded network	Heavy penalty for lightly loaded network	Heavy penalty for lightly to modestly loaded network
Design Complexity	High – To adjust voltage/ frequency	Complex – Because of new routing algorithms, deadlock avoidance	Low with naïve implementation
Hardware Overhead	High – To adjust voltage/ frequency	Low	Low
Fault Tolerance Impacts	Minimal impact	Reduce path diversity	Minimal impact
Routing Impacts	Minimal impact	High Impact	Minimal Impact

Figure 3.2.3: Summary of technique critiques

Proposed technique

We follow the idea of the dynamic link width technique because this technique has some advantages. It has a finer granularity of the energy consumption of a link. The topology of the network remains unchanged, thus simplifying the routing algorithms. However, instead of adjusting the width of the link we leverage the new link technology. In new link technology, every link composes several in-

dividual lanes which can independently activate/ deactivate. We adjust the speed of the link by adjust the number of active lanes in the links. Our mechanism also takes into account the impacts on the performance when the network deploys the energy saving mechanism by introducing several proposals.

4

Performance-Aware Dynamic Link Speed Mechanism

While Ethernet continues its dominance in networking fabrics, it is not the only one this thesis focuses on. Infiniband is already available at 48 Gb/s with 12 lanes, denoted as 12*x* [12]. Since Infiniband was designed from ground up for datacenter networking fabrics, it is considered better suit than traditional TCP/IP/Ethernet for high performance computing infrastructures [34]. PCI-Express was designed as a replacement for the PCI interconnect [28] and has become universal in that space, PCI-Express links are available up to 16-lane configuration and denoted as 16*x*.

Fiber Channel remains dominant for storage in data centers and their speed already surpasses 10 Gb [45]. Other niches such as Myrinet [33] and Qsnet [49] also scale up in speed. Those technologies move forward the trend in which a link

composes of several individual lanes. Our mechanism leverages that built-in feature to dynamically adjust the number of active lanes to save energy. The number of active lanes directly translates to the link bandwidth or link speed. There is a strong correlation between the link speed and the energy consumption of the link. For example, the energy consumption of a 1 Gb/s Ethernet consumes only 1 watt while the energy consumption of 10 Gb/s links can easily exceed 10 watts [35].

The decisions to adjust link speed rest upon the counters reflected the amount of traffic being transferred on them. The bit-serial technology become more ubiquitous where every link comprises a number of individual lanes, supposed n lanes. Fig. 4.0.1 illustrates the case when $n = 4$.

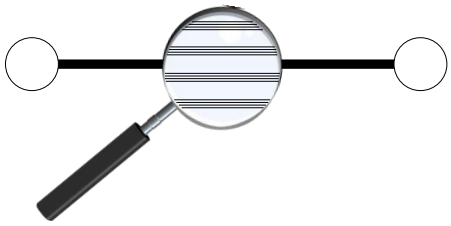


Figure 4.0.1: Bit-serial link with 4 lanes

4.1 Dynamic Link Speed Mechanism

When the traffic is intense, all of n lanes is turned on to provide a bandwidth as high as possible. When traffic decreases, the mechanism deactivates some of those lanes to save energy. At most $n - 1$ lanes can be deactivated; at least 1 lane is always kept on to keep the connectivity of the network and to avoid complicating the routing algorithm. The mechanism composes of two basic phases: *Monitoring* phase and *Decision Making* phase.

4.1.1 Monitoring phase

The main purpose of the *Monitoring* phase is to estimate the network traffic. There are several approaches to estimate the traffic: number of busy virtual channels of the link [43, 54], buffer occupancy [54] or the link utilization [11, 35, 52]. Equation 4.1 presents the mathematical description of link utilization (LU) for a link in a router.

$$LU = \frac{\sum_{t=1}^{H} A(t)}{H} \tag{4.1}$$

Where A(t) = $\begin{cases} 1 \text{ if traffic passes the link in the link cycle } t \\ 0 \text{ if no traffic passes the link in the link cycle } t \end{cases}$

and H is the history window size, in other words the mechanism monitors the last H cycles to calculate the LU. The value of H is one of the parameter of the mechanism that need to be configured.

The value of LU is a faithful measure of how busy a link is. A higher value of LU implies a higher traffic on the link, and thus is a strong indicator to keep the lanes of the link "on" to maximize the aggregate link bandwidth. Otherwise, the link becomes a candidate for the mechanism to reduce its number of active lanes and its speed.

4.1.2 Decision Making Phase

The mechanism predicts the value of LU based on recent link activities and then smoothes it out by combining the values of link utilization from the last two H cycles as described in Eq. 4.2.

$$LU_{predict} = \frac{weight * LU_{current} + LU_{past}}{weight + 1} \tag{4.2}$$

Where $LU_{predict}$ is the predicted value of LU, $LU_{current}$ is the link utilization of the last H cycles and LU_{past} is the link utilization of the H cycles before that; *weight* is the smoothing coefficient, the higher value of *weight* is, the more the mechanism regards the value of the recent activities of the link.

Links can not instantaneously adjust their speed, they need a transition time to adjust their speed level. The value of *transition_time* varies depending on the link technology, which is estimated to be around 100 cycles according to the best value reported in [52, 54]. Thus the decision making process takes place every *decision_period* link cycles, which exceeds *transition_time* to allow the newly-changed-speed links to stabilize.

The mechanism decreases the speed of a link if its $LU_{predict}$ drops below a threshold value of *threshold_low*; and increases the speed of the link if its $LU_{predict}$ exceeds a threshold value of *threshold_high*. Fig. 4.1.1 illustrates a link with 4 lanes, when it has 4 active lanes it is in the state S_4 with the bandwidth value B_4, if the predicted link utilization falls below the *threshold_low* it will deactivate 1 lane to the state S_3 with the bandwidth value $B_3 = \frac{3}{4}B_4$. Similar processes happen with other states of the link.

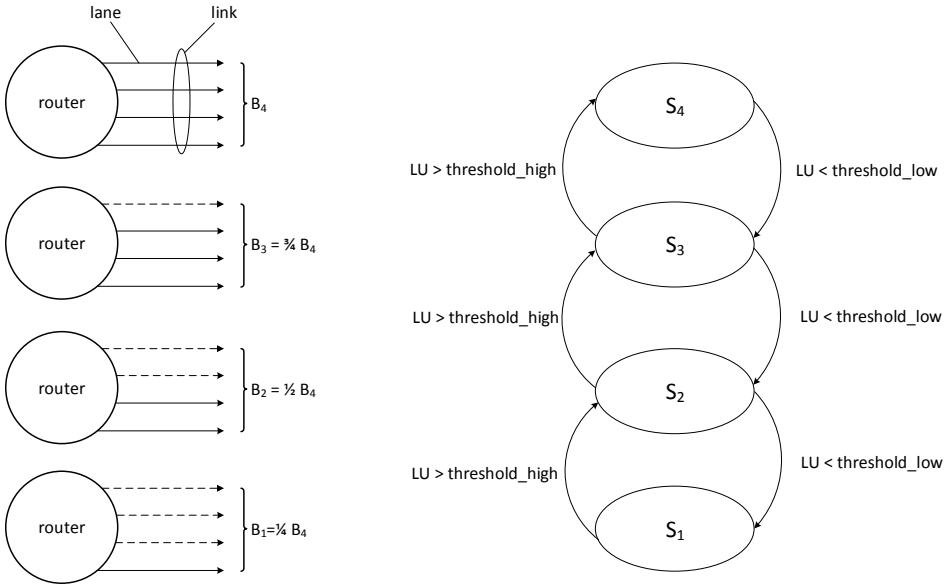


Figure 4.1.1: Overview of the change of link speeds

Algorithm 1 describes the pseudo code for the Dynamic Link Speed Mecha-

Algorithm 1 Basic Dynamic Link Speed Mechanism

```
Monitors the LU value for every link
Predicts the value of LU_predict
if current_time mod decision_period = 0 then
  if (LU_predict < threshold_low) then
    Decrease the link speed
  end if
  if LU_predict > threshold_high then
    Increase the link speed
  end if
end if
```

nism.

Sections 4.1.1 and 4.1.2 describe basic features of the mechanism. This thesis also proposes several improvement as describing in the following sections.

4.1.3 Two-level Threshold Policy

In the situation of congestion, packets mostly spend time filling up the buffer space and waiting for other packets ahead of them in the buffer queue to move before making any progress. In this case, the additional latency incurred by the energy saving mechanism can be hidden because the movement of packets are restricted by the availability of the buffer space, not the link speed itself; thus the impact of the saving mechanism on the packet latency is minimal. Hence, if the mechanism can detect the congestion then it can save energy more aggressively without having a bad effect on the performance of the network.

The buffer utilization is monitored as described in Eq. 4.3. This value is used to detect the congestion situation of the network.

$$BU = \frac{\sum_{t=1}^{H} F(t)/B}{H} \tag{4.3}$$

Where $F(t)$ is the number of input buffers that are occupied at time t, B is the size of input buffer and H is the history window size.

The mechanism predicts the value of the buffer utilization $BU_{predict}$ as described

in Eq. 4.4.

$$BU_{predict} = \frac{weight * BU_{current} + BU_{past}}{weight + 1} \tag{4.4}$$

Where $BU_{predict}$ is the predicted value of BU, $BU_{current}$ is the buffer utilization of the last H cycles and BU_{past} is the buffer utilization of the H cycles before that; *weight* is the smoothing coefficient, the higher value of *weight* is, the more the mechanism regards the value of the recent activities of the buffer of the link.

If the value of $BU_{predict}$ surpasses a threshold value of *buffer_threshold* then this is a signal indicating the congestion at the far end of the link. In that case, a different pair of thresholds with the values of *threshold_low_aggressive* and *threshold_high_aggressive* can be used to save more energy while negligibly impacts the average packet latency or performance. Fig. 4.1.2 depicts the state changes of these two threshold values.

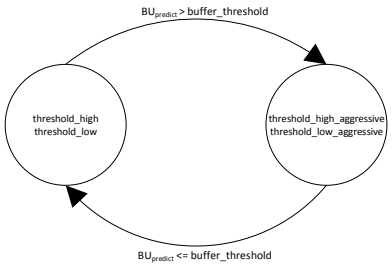


Figure 4.1.2: Two-level threshold policy

This two level threshold policy modifies the basic features of the mechanism by using extra more aggressive pairs of thresholds. The decision about which of the two threshold pairs to pick bases on the congestion level of the network. The pseudocode of the mechanism adding the two level threshold policy is shown in Algorithm. 2.

Algorithm 2 Dynamic Link Speed Mechanism with two-level threshold policy

Monitors the LU value for every link
Predicts the value of $LU_{predict}$
/*Check the congestion situation*/
if $BU_{predict} > buffer\_threshold$ **then**
  $threshold\_low = threshold\_low\_aggressive$
  $threshold\_high = threshold\_high\_aggressive$
else
  $threshold\_low = threshold\_low\_original$
  $threshold\_high = threshold\_high\_original$
end if
if $current\_time$ mod $decision\_period = 0$ **then**
  if $(LU_{predict} < threshold\_low)$ **then**
    Decrease the link speed
  end if
  if $LU_{predict} > threshold\_high$ **then**
    Increase the link speed
  end if
end if

4.1.4 Link Flip-Flop Avoidance

According to the policy above in section 4.1.1 and 4.1.2, when the mechanism changes a link in speed level l_1 to a speed level l_2, if the traffic remains the same its LU changes as described in Fig. 4.1.1 and Fig. 4.1.3, more specifically it will change $\frac{l_1}{l_2}$ times. For example, due to the low value of LU (below the value of *threshold_low*), a $2x$ link will increase its LU 2 times when the mechanism decreases its speed to $1x$. The 2-fold increase in LU makes its LU might exceed *threshold_high* and triggers the mechanism to increase the speed of the newly-decreased-speed link back to $2x$ level. The increase in the link speed from $1x$ to $2x$ in turn halves the value of the just-changed LU. This new value of LU drops below *threshold_low* as stated before, hence the mechanism will decrease the link speed in the next decision making phase. The same cyclic process happens again and thus causes flip-flop situation [9].

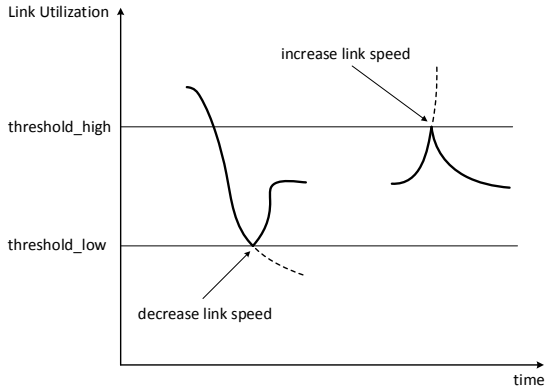


Figure 4.1.3: Link Utilization change with time

Fig. 4.1.4 highlights a scenario where *threshold_low*=0.3 and *threshold_high*=0.5. The link is in the state S_2 with 2 active lanes and it has the link utilization value of 0.28, which is lower than *threshold_low*. Thus the mechanism triggers the link decrease its active lane number to 1 and goes into the state S_1. Because of the decrease in the link speed, the link utilization goes up 2 times and reaches the value

of 0.56, which is higher than *threshold_high*. In this case, the mechanism increases the number of active lanes of the link back to 2. This increase makes the link utilization decreases to 0.28, which is lower than *threshold_low*. And the cyclic process starts again and causes the link flip flop situation.

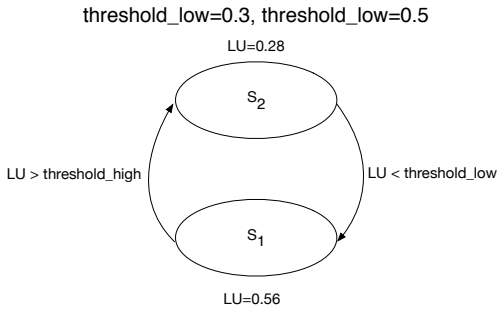


Figure 4.1.4: Flip Flop situation

To avoid the link flip-flop problem, the mechanism triggers a link to change the speed not only by comparing the value of LU with the threshold values of *threshold_low* and *threshold_high*, but it also needs to guarantee that the new value of LU falls between the values of the 2 thresholds. To achieve those behaviors, the mechanism imposes the following two rules:

1) A link decreases its speed level from l_1 to l_2 only if its LU drops below *threshold_low* **and** the new value of LU if the link speed changes does not exceed *threshold_high*. In other words:

$$\begin{cases} LU < threshold\_low \\ LU * \frac{l_1}{l_2} < threshold\_high \end{cases}$$

It implies that the value of LU would trigger the links to decrease their speed level if LU drops to a value less than the minimum of *threshold_low* and $\frac{l_2}{l_1} * threshold\_high$ as described below:

$$\begin{cases} LU < threshold\_low \\ LU * \frac{l_1}{l_2} < threshold\_high \end{cases} \Rightarrow \begin{cases} LU < threshold\_low \\ LU < \frac{l_2}{l_1} * threshold\_high \end{cases}$$

$$\Rightarrow LU < min(threshold\_low, \frac{l_2}{l_1} * threshold\_high)$$

2) Similarly, a link increases its speed level from l_1 to l_2 only if its LU exceeds

threshold_high **and** the new value of LU if the link speed changes does not drop below *threshold_low*. In other words:

$$\begin{cases} LU > threshold\_high \\ LU * \frac{l_1}{l_2} > threshold\_low \end{cases}$$

It implies that the value of LU would trigger the links to increase their speed level if LU exceeds to a value greater than the max of *threshold_high* and $\frac{l_2}{l_1} * threshold\_low$ as described below:

$$\begin{cases} LU > threshold\_high \\ LU * \frac{l_1}{l_2} > threshold\_low \end{cases} \Rightarrow \begin{cases} LU > threshold\_high \\ LU > \frac{l_2}{l_1} * threshold\_low \end{cases}$$

$$\Rightarrow LU > max(threshold\_high, \frac{l_2}{l_1} * threshold\_low)$$

This Flip-Flop Avoidance techniques extends the basic features of the dynamic link speed mechanism. A pseudocode of the mechanism with the flip-flop avoidance technique is described in Alg. 3.

Algorithm 3 Mechanism Improvement with Flip-Flop Avoidance Technique

```
Monitors the LU value for every link
Predicts the value of LU_predict
/*Check the congestion situation*/
if BU_predict > buffer_threshold then
  threshold_low = threshold_low_aggressive
  threshold_high = threshold_high_aggressive
else
  threshold_low = threshold_low_original
  threshold_high = threshold_high_original
end if
if current_time mod decision_period = 0 then
  if (LU_predict < min(threshold_low, (l2/l1)threshold_high)) then
    Decrease the link speed
  end if
  if LU_predict > max(threshold_high, , (l2/l1)threshold_low) then
    Increase the link speed
  end if
end if
```

4.1.5 Link Speed Change Policy

When the link utilization value falls out of the threshold intervals, the mechanism can apply several policies in the process of changing the link speed. Each policy impacts differently to the amount of energy saving and the latency behavior. There is a tradeoff between those two objectives.

- Fast Increase: The link speed drops every sequential speed level, but when its LU exceeds the value of *threshold_high* meaning it requires an increase in link speed, the mechanism triggers the link to the maximum speed level possible to avoid the latency penalty due to the low speed links as illustrated in Fig. 4.1.5. This policy is motivated by the average packet latency over energy consumption.

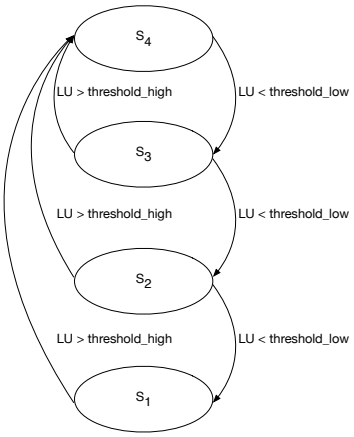


Figure 4.1.5: Fast Increase Link Speed Change Policy

- Fast Decrease: The link speed increases every sequential level, but when its LU drops below the value of *threshold_low*, the mechanism triggers the link to the minimum speed level possible to minimize the energy consumed as illustrated in Fig. 4.1.6. This prioritizes the energy saving over the average packet latency.

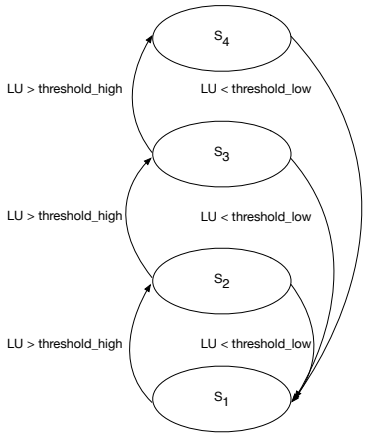


Figure 4.1.6: Fast Decrease Link Speed Change Policy

- Gradual Increase - Gradual Decrease: Links increase and decrease their speed levels one step at a time according to the value of LU as illustrated in Fig. 4.1.7. This policy is the mid way between the two extremes.

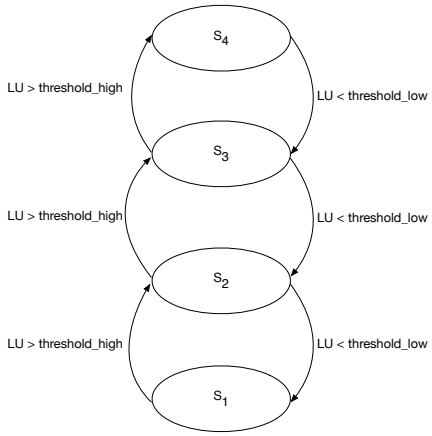


Figure 4.1.7: Gradual Increase - Gradual Decrease Link Speed Change Policy

To the best of our knowledge, previous publications follow the *Gradual Increase - Gradual Decrease* policy. However, the network can be configured with other policies depending on the objectives of the systems. Besides, parallel applications

often have many phases with different requirements, which might be predicted [15]. Thus the mechanism can apply a dynamic policies corresponding to different phases of the applications to gain the most efficiency.

The link speed change policies completes our proposals on the dynamic link speed mechanism. It provides the options for links to adjust their speed level. The pseudocode of the whole mechanism is presented in Alg. 4.

Algorithm 4 The Dynamic Link Speed Mechanism

```
Monitors the LU value for every link
Predicts the value of LU_predict
/*Check the congestion situation*/
if BU_predict > buffer_threshold then
  threshold_low = threshold_low_aggressive
  threshold_high = threshold_high_aggressive
else
  threshold_low = threshold_low_original
  threshold_high = threshold_high_original
end if
if current_time mod decision_period = 0 then
  if (LU_predict < min(threshold_low, (l2/l1)threshold_high)) then
    if FAST_DECREASE_POLICY then
      Decrease the link speed to the minimum level
    else
      Decrease the link speed 1 level
    end if
  end if
  if LU_predict > max(threshold_high, , (l2/l1)threshold_low) then
    if FAST_INCREASE_POLICY then
      Increase the link speed to the maximum level
    else
      Increase the link speed 1 level
    end if
  end if
end if
```

4.2 Performance Awareness

The relationship between the average packet latency and network performance can be quite complex. For example, for a reliable transport protocol such as *TCP*, higher value of average packet latency directly lowers the performance. In general cases, an increase in average packet latency results in an increased processing unit stall and thus degrades application performance. If the applications are computation-bounded, the increase in packet latency might only partly visible to the processing units (CPU). The estimated result is expressed in the following equation for the performance λ in terms of average packet latency L_a [36]:

$$\lambda = \frac{CU_{CPU}}{1 + \zeta L_a} \tag{4.5}$$

Where C and ζ are appropriate constants that depend on the applications and network infrastructures, U_{CPU} is the power of processing units.

The impacts of latency on performance depend on applications and workload. However, Eq. 4.5 shows that the performance is proportional to the reciprocal value of average packet latency. Thus we limit our discussion mostly on average packet latency which directly translates to performance, i.e a lower a value of average packet latency implies a higher performance and vice versa.

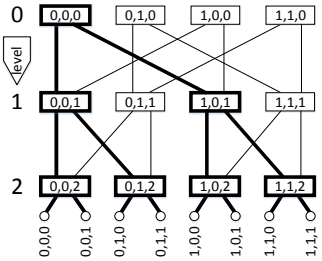
The latency of a packet is the elapsed time since it is generated till it is received at the destination. An interconnection network system might apply the Energy Saving Mechanism described in section 4.1 to reduce the energy consumed by adjusting the link speed. In this system, the average packet latency increases because some packets move on lower bandwidth links, which deteriorate the network performance.

In this section, we propose a method to boost the performance (means to reduce the average packet latency) on top of the above-mentioned energy saving mechanism. Henceforth, we assume the network is coupled with the Dynamic Link Speed energy saving mechanism described in section 4.1 when introducing the proposal below.

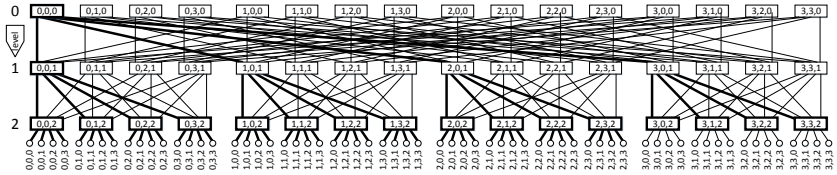
4.2.1 Link Speed Aware Routing Policy

We introduce a Link Speed Aware Routing Policy that prioritizes output ports coupled with links that are in high-speed level. This policy deploys on top of any reasonable adaptive routing algorithm and thus it is topology-and-routing-algorithm independent.

Fig. 4.2.1 shows examples of the fat tree topologies; Fig. 4.2.2b shows examples of the cube topologies. All of them have path redundancy for fault tolerance and load balancing purposes. With the path redundancy in modern network infrastructure, from a sender to a destination there are many paths for packets to take.



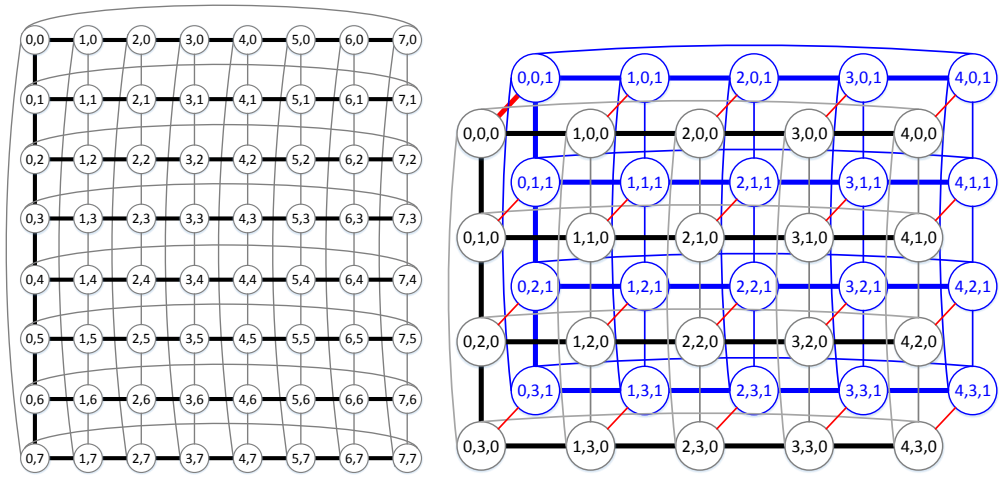
(a) 2-ary 3-tree



(b) 4-ary 3-tree

Figure 4.2.1: k-ary n-tree topology network

When a packet arrives at a router, typically the routing algorithm produces a set of several compatible virtual channels and output ports for the packet to forward toward a destination. The packet will bid for a virtual channel from this set of virtual channels and output ports in the process of virtual channel allocation. The Virtual Channel Allocator, according to its policy, will select a virtual channel in a



(a) 8-ary 2-cube **(b)** 5,4,2-ary 3-cube

Figure 4.2.2: k-ary n-cube topology network

port on which the packet will go through, taking into account the priority of the virtual channels and output ports.

At every routing step, the Link Speed Aware Routing Policy picks one output port from the pool of compatible ports $p_1, p_2, \ldots, p_k$. The ports couple with links corresponding to speed levels of $l_1, l_2, \ldots, l_k$ as described in fig. 4.2.3. The picked port will be given a higher priority compared with other ports in the Virtual Channel Allocation process and thus this port has a higher probability for the packet to go through.

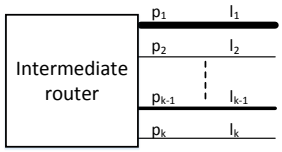


Figure 4.2.3: A pool of compatible output ports

The probability for a port p_i to be picked from a pool of compatible ports corresponds to the speed level l_i and the buffer occupancy b_i of the link coupled with that port. The higher the value of l_i and the lower value of b_i translate to the higher

probability this routing policy will pick port p_i. Mathematically speaking, a port p_i has the probability of $\rho(p_i)$ as described in Equation 4.6

$$\rho(p_i) = \frac{l_i * (1 - b_i)}{\sum_{p=1}^{k} l_p * (1 - b_p)} \tag{4.6}$$

Where l_i is the link speed level coupled with the port number i, b_i is the buffer occupancy level associated with the port number i.

As a result, the ports coupled with links in high-speed levels and links that have low value of buffer occupancy have high priority in the virtual channel allocation process. The presence of the value b_i in the formula facilitates the load balancing in the process of routing packets.

Conventional adaptive routing algorithms route packets taking into account the buffer occupancy level. Our proposed routing policy expands the criteria to route packet with the link speed level factor. This link speed aware routing policy favors high-speed links, thus it contributes to reduce the average packet latency and thus boosting the performance of the network systems with different level of link speeds.

4.3 Mechanism Overview

Fig. 4.3.1 overviews the components that compose the mechanism. The dotted boxes represent parameters that the mechanism needs to configure. The parameters are picked based on the objectives of the interconnection networks, i.e which kinds of applications (latency sensitive or non-sensitive) are running in the networks, prioritizing energy saving or average packet latency. The solid boxes represent the logical components of the mechanism.

As we can see the *Decision Maker* is the center of the mechanism. Other components feed information related to the link utilization, link speed change policy, different thresholds. The decision making process takes place every *decision_period* link cycles and produces an appropriate link speed level for the link. The link speed

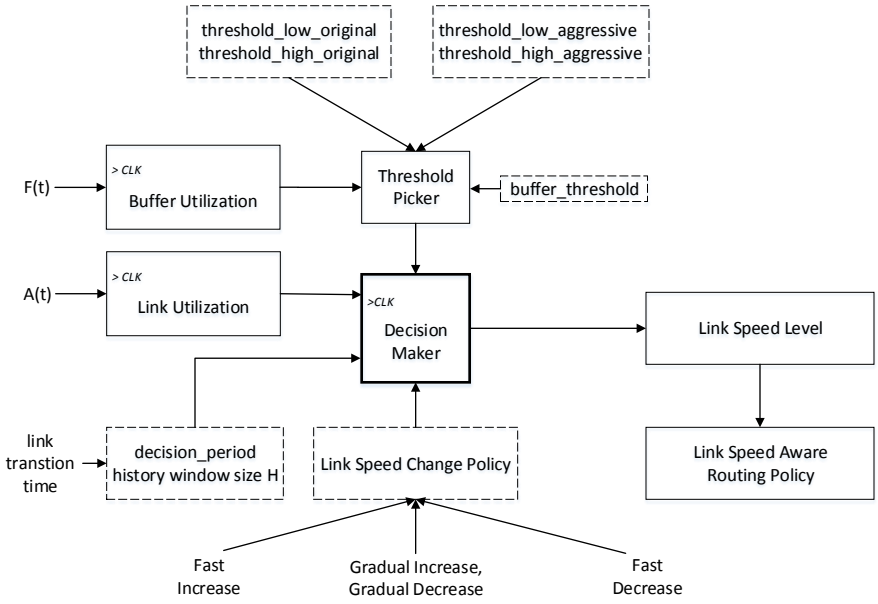


Figure 4.3.1: Mechanism Overview

level in turn impacts the routing process with the link speed aware routing policy.

4.4 Hardware model prototype

Fig. 4.4.1 illustrates the hardware model prototype for the Dynamic Link Speed energy saving mechanism in this thesis. The shaded boxes represent components that are introduced by this thesis, white boxes represent the conventional components in the interconnection network systems. The Routing Unit introduces an additional *Link Speed Aware Routing Policy* component to prioritize high-speed links from a set of compatible links for a given packet as described in section 4.2.1. The *Dynamic Link Speed Adjustment* component is modified with extra features. The heart of the proposal is presented in the center of the figures with several modules corresponding to the above-mentioned proposals - The *Link & Buffer Utilization* module monitors the link & buffer usage frequency. The predicted value of link usage is compared with the pair of threshold values (which are set according to the

two-level threshold policy by the *Threshold Picker* module). The Decision Making process prescribes whether to adjust the link speed.

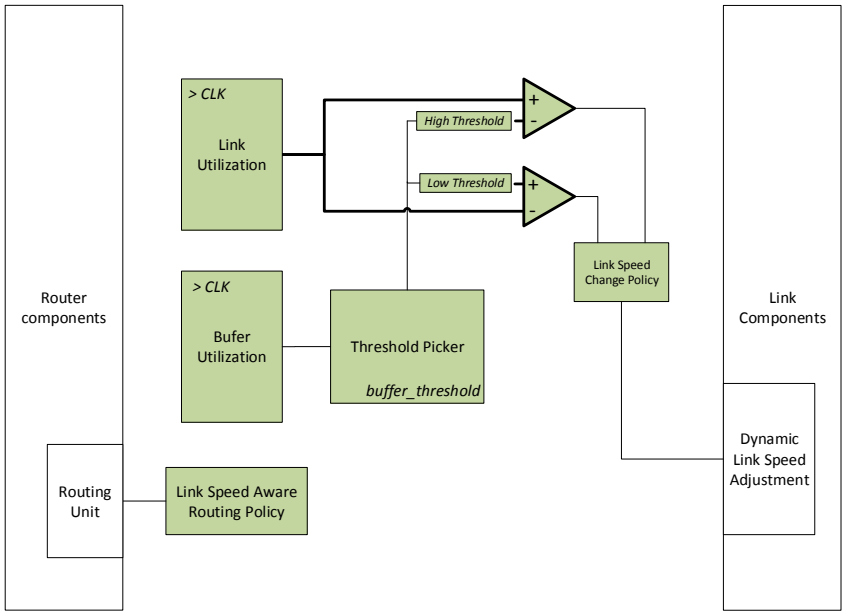


Figure 4.4.1: Hardware model implementation

4.5 Discussions

The impacts on the energy saving and network performance of the mechanism depend on the configured parameters. Parameters *threshold_high* and *threshold_low* directly impacts how the network performs. They form two characteristics for the mechanism, which are *Aggressiveness* and *Responsiveness* as described below:

Aggressiveness and Responsiveness

- Aggressiveness: $aggressiveness = \frac{threshold\_high + threshold\_low}{2}$. This characteristic defines how aggressively the mechanism works to save energy. The higher the value of *aggressiveness* is, the more easily the mechanism decreases the

link speed, in turn the network achieves more energy saving with the expense of a possible increase in average packet latency.

- Responsiveness: $responsiveness = threshold\_high - threshold\_low$. This hysteresis band defines how responsively the mechanism reacts with the variance of the traffic. The higher the value of *responsiveness* the more traffic variance required to trigger the mechanism to change the link speed.

The mechanism needs to be configured with the values of *threshold_low* and *threshold_high* and thus indirectly specifies the *aggressiveness*. These configured threshold values are set according to the objectives of the network. If networks provision applications that are latency insensitive then the mechanism configures the threshold parameters so that the *aggressiveness* being at a high value to save more energy. Otherwise, threshold parameters are tuned to lower the *aggressiveness* to prioritize network performance. The mechanism is also fed the threshold parameters to get an appropriate value of the *responsiveness*. In other words, the parameters minimize the number of change in link speed while maintaining the dynamic of the mechanism in response to the traffic fluctuation.

Comparison with previous works

Compared with other previous works, our mechanism focuses on link energy management with the dynamic link speed feature. This mechanism follows similar ideas of the dynamic link width approach. It has the advantages over the dynamic link shutdown in that the topology remains unchanged, thus the routing algorithm stays the same. There is also no need for new deadlock avoidance technique and connectivity guarantee. Our mechanism relies on the built-in features of links to adjust the number of active lanes and the link speed instead of narrowing and increasing link width to adjust the link bandwidth. Thus this mechanism introduces fewer new hardware components.

Our mechanism also differs from other methods by introducing several proposals to improve the energy saving mechanism: the two-level threshold policy

to further increase energy saving when congestion occurs, the link flip-flop avoidance technique to reduce the overhead of link speed transition time, the link speed change policy option to further tune the behaviors of the mechanism based on the network objectives. Furthermore, the link speed aware routing policy prioritizes high speed links to reduce the serialization latency, thus boosting the performance of the network when the energy saving mechanism is deployed. Fig. 4.5.1 shows the dynamic link width (DLW) technique and the proposed mechanism with respect to different important metrics.

Technique/Metric	**DLW**	**Proposed mechanism**
Energy saving	Limited by the number of available link width configuration	Limited by the number of lanes in a physical link
Latency Penalty	Heavy penalty for lightly to modestly loaded network	Lower penalty for lightly to modestly loaded network
Design Complexity	Low with naïve implementation	Low
Hardware Overhead	Low	Low
Fault Tolerance Impacts	Minimal impact	Minimal impact
Routing Impacts	Minimal Impact	Link Speed Aware to Boost Network Performance

Figure 4.5.1: Comparison with previous works

5 Evaluation

In this chapter we evaluate the behaviors of the proposed mechanism mentioned in previous chapters by conducting sets of simulations on an extended version of the booksim simulator [7].

We evaluate the energy saving percentage and their impact on the network performance. We then compare the behaviors of the interconnection network with our proposals with similar proposals in the literature (dynamic on/off mechanism, conventional dynamic link speed mechanism) and the default system (without applying energy saving mechanisms).

We assume that links consume energy proportionally with its speed level. The first metric of the sets of simulations is the *relative link power consumption*. The *relative link power consumption* is the ratio of the energy consumed by links in the network deploying an energy saving mechanism, to the energy consumed by links in the network of default systems. We only take into account link energy consump-

tion since we work exclusively on this particular network component; hence we evaluate the impact of our proposal to this component, which differs from other proposals that might applied on different network components (e.g. switch buffers, allocators, routing units...). The second metric of those sets of simulations is the *average packet latency*. Network performance derives directly from the *average packet latency* as described in section 4.2, thus we minimize the impact of the mechanism to the performance of the network by minimizing the increase of *average packet latency* in comparing with default systems.

Sections 5.1, 5.2 and 5.3 describe the configurations for the simulations. Section 5.4 shows the effectiveness of the mechanism in general. Sections 5.5, 5.6 and 5.7 study the impacts of internal parameters to the mechanism. Section 5.8 highlights the advantages of our proposed mechanism compared with other mechanisms in literature. And finally, section 5.9 shows the versatility of the mechanism to different traffic patterns and network topologies.

5.1 Network model

Our simulator models a flit-level wormhole-switching network. Each router has a routing unit, a crossbar, an allocator and several ports coupled with physical links. These components are typical in conventional router microarchitecture [17]. Every physical link contains a number of individual lanes and the link can activate/deactivate those lanes dynamically. All routers in the network are configured with the same parameters.

5.2 Traffic patterns

Traffic pattern directly impacts how the proposed mechanism performs. Traffic pattern is made of 3 different factors: the flit generation rate at every node, the packet size and destination.

We conduct simulations with 2 kinds of traffic pattern. First kind of traffic pattern is *uniform* traffic where every sender sends to other nodes with the same prob-

ability at a constant rate. This traffic pattern does not have any locality or repetitiveness, thus the mechanism will work at its lowest effectiveness. Another traffic pattern is the *Lower-Upper Gauss-Seidel solver* traffic [19]. The task graph and intensity of communication of this traffic is shown in Fig. 5.2.1. As we can see, this traffic pattern represents a class of traffic that has locality and repetitiveness property since this traffic pattern has a majority of traffic focusing across the diagonals.

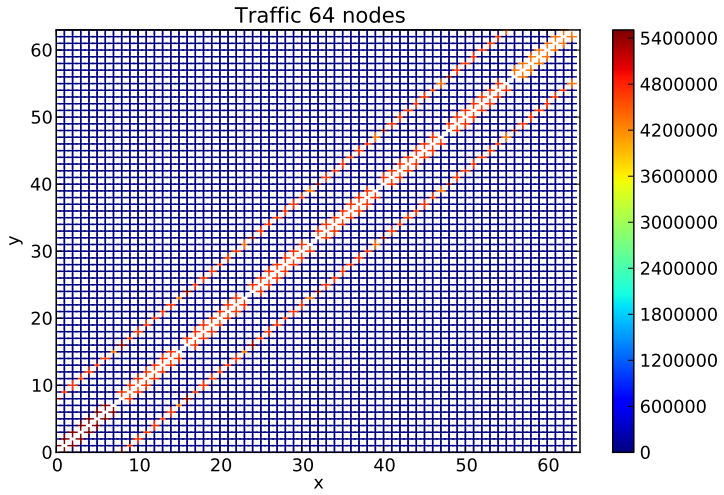


Figure 5.2.1: Lower-Upper Gauss-Seidel solver traffic task graph

5.3 General parameters for the evaluation

For general scenarios in the evaluation process, the interconnection network is configured with 64 processing nodes arranged in the 4-ary 3-n fat-tree topology [41]. The network deploys virtual channel credit-based flow control. Packets are 8 flits in size.

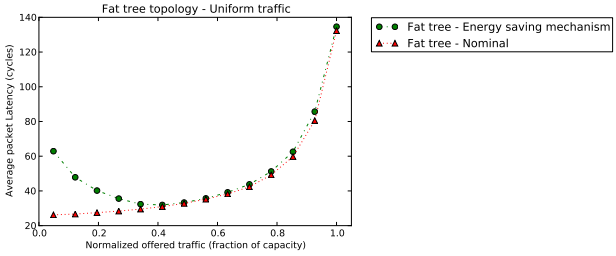
Every link consists of 8 virtual channels and has 12 individual lanes which can independently activated or deactivated. Links can not instantaneously adjust their speed, they need a period of *transition_time* cycles. Its value varies depending on the link technology, which is estimated to be around 100 cycles according to the

best value reported in [52, 54]. The value of the parameter *decision_period* needs to exceeds the value of *transition_time* to ensure the stability of the link after changing its speed. For the purpose of illustration, we adjust it to have the value of 400 cycles. The history window size H is configured equaling to half the value of *decision_period* with 200 link cycles.

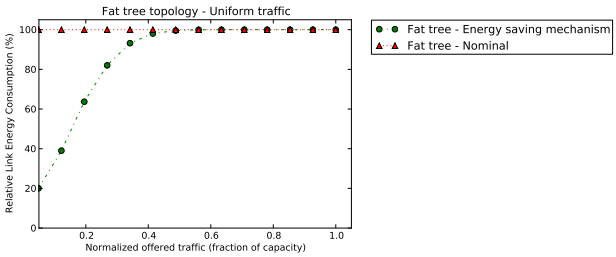
5.4 Mechanism overview

This section overviews the impacts of the mechanism on the *relative link power consumption* and the *average packet latency* with typical mechanism parameters described in 5.3.

Fig. 5.4.1 and 5.4.2 show the *average packet latency* and *relative link power consumption* for the fat tree topology with *uniform* traffic and *Lower-Upper Gauss-Seidel solver* traffic respectively.

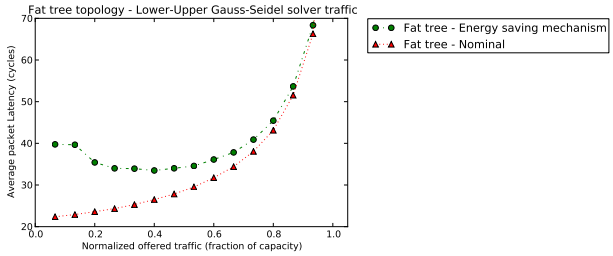


(a) Average Packet Latency - Uniform traffic

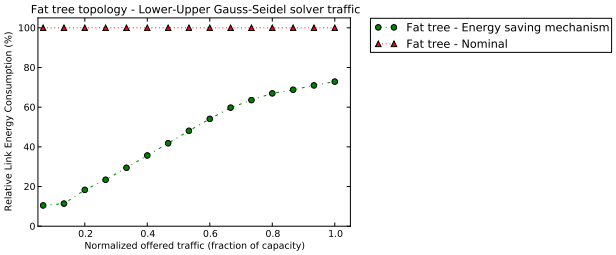


(b) Relative Link Energy Consumption - Uniform traffic

Figure 5.4.1: Mechanism overview with fat tree topology - Uniform traffic



(a) Average Packet Latency - Gauss-Seidel solver traffic



(b) Relative Link Energy Consumption - Gauss-Seidel solver traffic

Figure 5.4.2: Mechanism overview with fat tree topology - Lower-Upper Gauss-Seidel solver traffic

The interconnection networks in this set of simulations are deployed with the *Dynamic Link Speed* mechanism coupled with the proposals described in the previous chapters. The behavior of the default system without energy saving mechanism is labeled as *"Fat tree - Nominal"*, and the behavior of the network system with dynamic link speed mechanism taking into account network performance is labeled as *"Fat tree - Energy Saving Mechanism"*. The offered traffic varies in a range from light traffic until almost saturated traffic.

There is a clear trend: when the traffic is light, the mechanism saves energy by reducing the link speed (the lowest relative link energy consumption values are 21.86% in the case of uniform traffic, and 11.75% in the case of Lower-Upper Gauss-Seidel solver traffic). This comes with the expense of an increase in packet latency compared with the default system. When the traffic is high, the mechanism does not reduce the link speed as its utilization is high. As a result, the network saves

less (or not at all) energy with a less (or not at all) increase in the average packet latency. Similar scenarios and similar behaviors for torus topology is illustrated later in section 5.9.1. Thus, we show that the mechanism works as expected.

5.5 Link Speed Aware Routing Algorithm Impacts

We investigate the link speed change routing policy impacts by running 3 different sets of scenarios: The first scenario is the network deploying the dynamic link speed energy saving mechanism coupled with the Link Speed Aware routing policy from our proposal. The second scenario is the network deploying the conventional dynamic link speed energy saving mechanism without the Link Speed Aware routing policy. And the third scenario is the default network system without any energy saving mechanism.

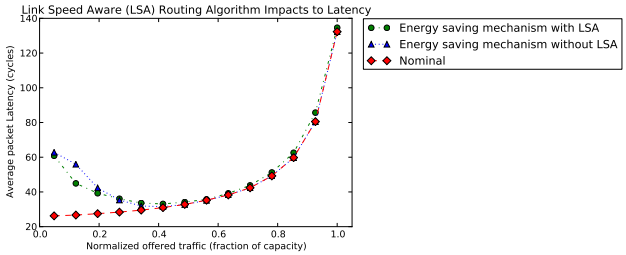
To compare the efficiency of the mechanisms, we calculate the ratio of the relative energy saving (S) and the percentage of latency increase (I). We get the relative energy saving of the system with the assumption that apart from links, other components of the interconnection networks account for 17.6% of the total energy consumption as described in [11]. This ratio S/I is a metric to measure the efficiency of energy saving mechanisms. The higher value of this ratio implies a higher efficiency of the mechanism.

Fig. 5.5.1 shows the *average packet latency*, the *relative link energy consumption* & the ratio S/I behaviors with a range of increasing traffic for uniform traffic.

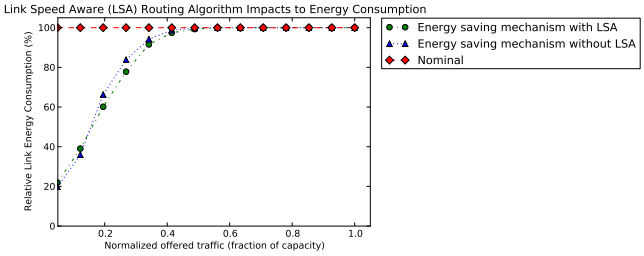
As can be seen from the figure 5.5.1, the energy saving mechanism coupled with the link speed aware routing policy has the latency curve below the conventional energy saving mechanism without the proposed routing policy.

Similarly, Fig. 5.5.2 shows the *average packet latency*, the *relative link energy consumption* & the ratio S/I behaviors with a range of increasing traffic for Lower-Upper Gauss-Seidel solver traffic.

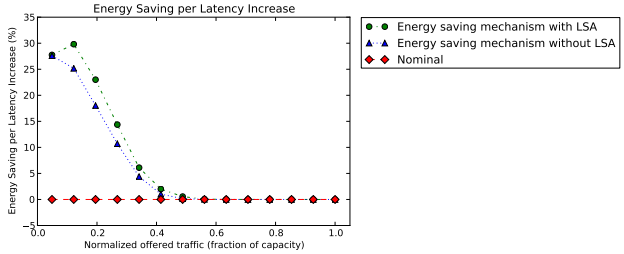
Fig. 5.5.2 shows that the *average packet latency* behaviors for the *Lower-Upper Gauss-Seidel solver* traffic is similar to what have been stated in the case of uniform



(a) Average Packet Latency



(b) Relative Link Energy Consumption

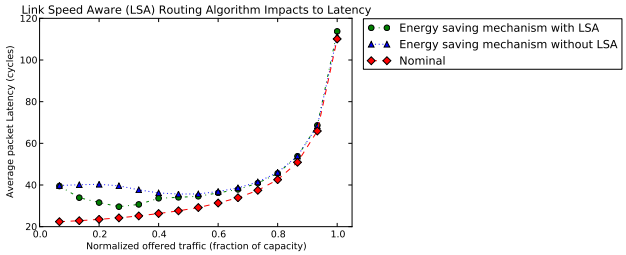


(c) Ratio of Energy Saving and Percentage Latency Increase

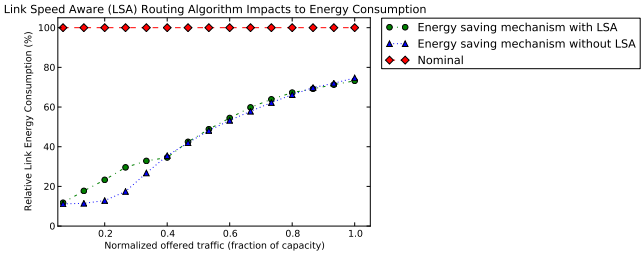
Figure 5.5.1: Link Speed Aware Routing Algorithm Impacts - Uniform traffic

traffic. We can observe that the latency curve of the link speed aware routing policy lays below the latency curve of the conventional mechanism without the proposed routing policy.

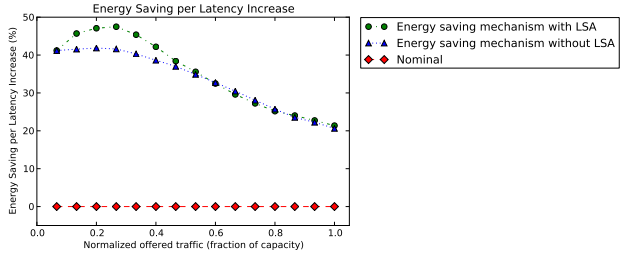
In both of the 2 scenarios, the mechanisms coupled with the link speed aware routing policy have similar relative link energy consumption curves. Thus, as can be seen in Fig. 5.5.1c and 5.5.2c the ratio S/I of the mechanism deployed with the proposed routing policy has a better behaviors compared with the conventional



(a) Average Packet Latency



(b) Relative Link Energy Consumption



(c) Ratio of Energy Saving and Percentage Latency Increase

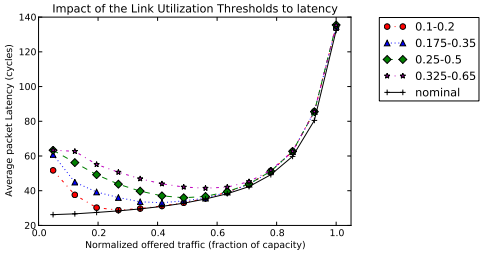
Figure 5.5.2: Link Speed Aware Routing Algorithm Impacts - Gauss-Seidel solver traffic

energy saving mechanism without the proposed routing policy (its S/I curve lays above the S/I curve of the conventional mechanism). It means that the energy saving mechanism deployed with the link speed aware routing policy outperforms the conventional energy saving mechanisms without the proposed routing policy.

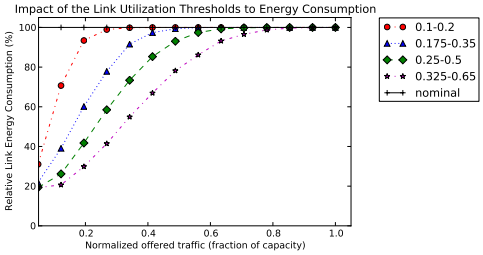
5.6 Link Utilization Thresholds Impacts

The *threshold_low* and *threshold_high* control the aggressiveness and responsiveness of the mechanism. We conduct a set of simulations with an increasing range of aggressiveness with 4 pairs of thresholds labeled as *threshold_low - threshold_high*.

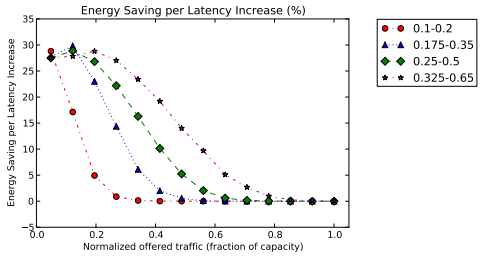
First scenario, we pick the value of *threshold_high* doubles the value of *threshold_low* as illustrated by Fig. 5.6.1.



(a) Average Packet Latency



(b) Relative Link Energy Consumption

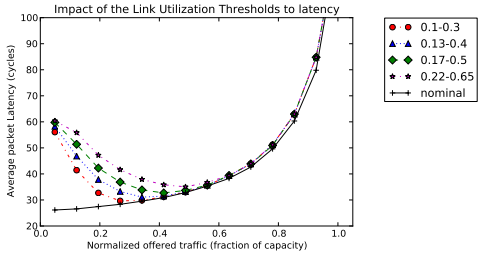


(c) Ratio of Energy Saving and Relative Latency Increase

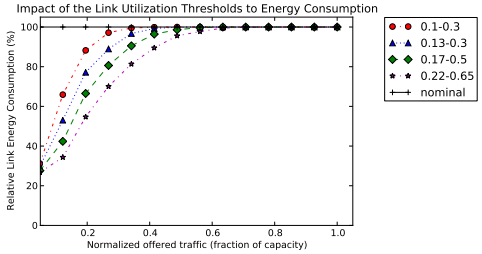
Figure 5.6.1: LU Thresholds Impacts - *threshold_high= 2 * threshold_low*

In this scenario, we witness the impacts of the increase of aggressiveness to the network.

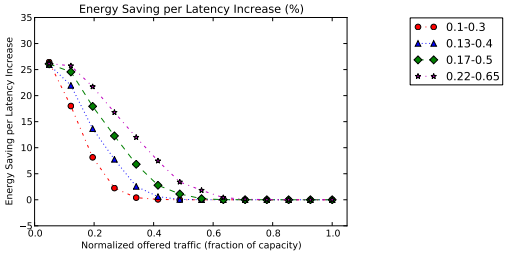
Second scenario, we reduce the responsiveness of the mechanism by increasing the distance between *threshold_low* and *threshold_high* by picking *threshold_high* = 3 * *threshold_low*. In this scenario as shown in Fig. 5.6.2, we witness the impacts of the decrease in the responsiveness of the mechanism since the distance between the low and high thresholds is lager compared with the first scenario.



(a) Average Packet Latency



(b) Relative Link Energy Consumption



(c) Ratio of Energy Saving and Relative Latency Increase

Figure 5.6.2: LU Thresholds Impacts - *threshold_high= 3 * threshold_low*

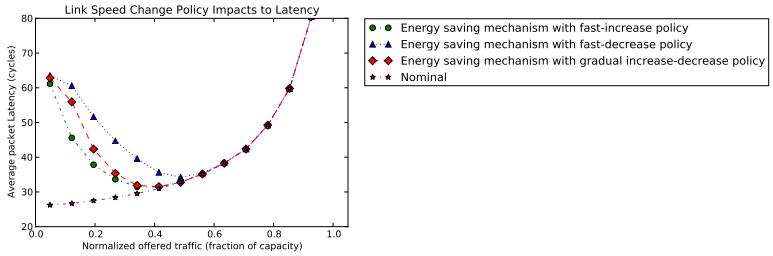
Typical parameters as described in section 5.3 were configured for these sets of simulations. The metrics under observation are the *relative link energy consumption*, the *average packet latency* and the *ratio of relative energy saving per latency increase* S/I. A higher value of aggressiveness (higher values of the threshold pairs) implies a higher saving in term of energy consumption as shown in both two figures. At the same time, the average packet latency has a higher value accordingly because packets have to go through lower speed links. The ratio S/I shows the effectiveness of the mechanism. As can be seen from the two figures, the ratio corresponding to the case where $threshold\_high = 2 * threshold\_low$ shows a better shape than in the case where $threshold\_high = 3 * threshold\_low$. We also observe that the pair of threshold $0.175 - 0.35$ yields a good balance between the *relative link energy consumption* and the *average packet latency*, thus it is picked to use in many of our simulations to evaluate other parameters of the mechanism.

5.7 Link speed change policy impacts

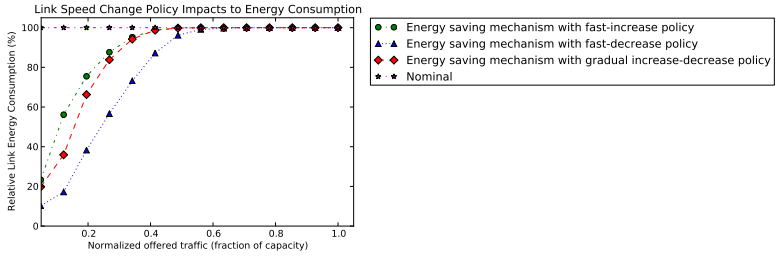
We investigate the link speed change policy impacts by running 3 different sets of scenarios. In all scenarios the interconnection networks deploy the proposed energy saving mechanism. However, in each scenario the mechanism changes the link speed with different policies: *fast increase, fast decrease, gradual increase - gradual decrease*. The traffic ranges from very light to almost saturated. Typical parameters as described in section 5.3 were configured for these sets of simulations.

Fig. 5.7.1 illustrated the impacts of different policies to the *relative link power consumption, average packet latency* and the *ratio of relative energy saving per latency increase* of the network. The first 3 labels in the figures correspond to 3 different link speed change policies, the behaviors of the default network system is also illustrated and labeled as *"Nominal"*.

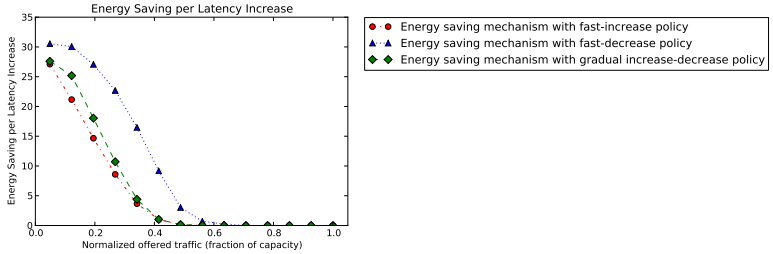
The figure confirms that the *fast increase* policy prioritizes performance over energy saving, and *fast decrease* policy prioritizes energy saving over performance and the *gradual increase, gradual decrease* policy balances between the two.



(a) Average Packet Latency



(b) Relative Link Energy Consumption



(c) Ratio of Energy Saving and Relative Latency Increase

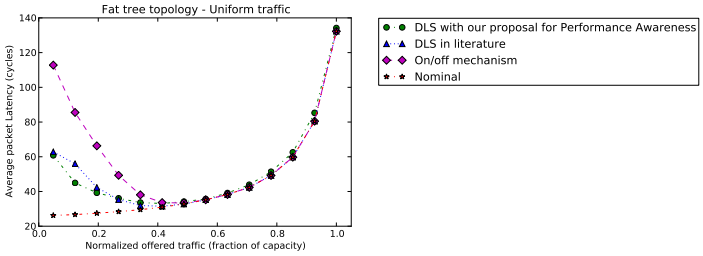
Figure 5.7.1: Link Speed Change Policy Impacts

5.8 Mechanism Comparisons

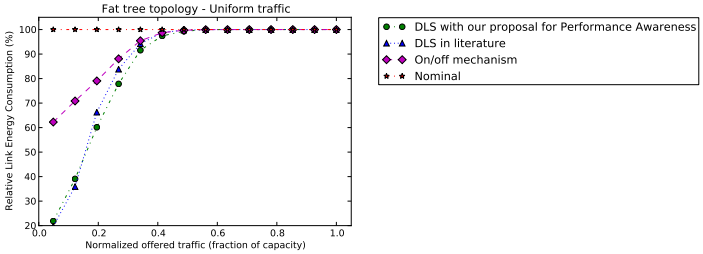
There have been several alternatives dealing with the energy saving in interconnection networks as described in section 3.2. More specifically, the dynamic voltage scale mechanism and the dynamic link width mechanism ultimately adjust the link speed, thus we categorize it into the *dynamic link speed* group; another group of mechanisms is the *on/off mechanism*.

We compare the effectiveness of our proposed Dynamic Link Speed Energy

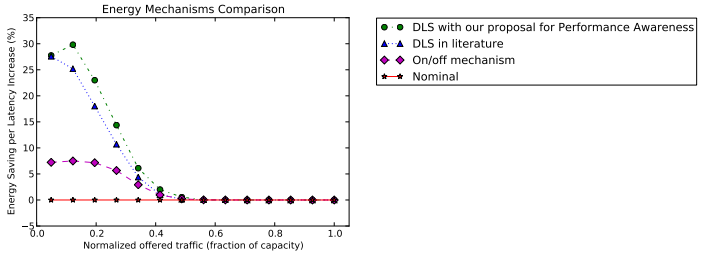
Saving mechanism taking into account the performance with those two groups of mechanisms and the default system without any energy saving mechanism. Fig. 5.8.1 shows the behaviors of the network systems.



(a) Average Packet Latency



(b) Relative Link Energy Consumption



(c) Ratio of Energy Saving and Percentage Latency Increase

Figure 5.8.1: Comparison among mechanisms

Typical parameters as described in section 5.3 were configured for these sets of simulations. The offered traffic ranges from very low load until almost saturated load. We observe the *relative link power consumption, average packet latency* and the

ratio of relative energy saving per latency increase values corresponding with these scenarios. In the figure, the *dynamic link speed* group that is described in section 3.2 labeled as *DLS in literature* and the on/off mechanism is labeled as *On/off mechanism*. The mechanism described in this thesis is labeled as *DLS with our proposal for Performance Awareness*. And the default network system without any energy saving mechanism is labeled as *Nominal*.

As can be seen from the figure, the Dynamic Link Speed Energy Saving mechanism taking into account the performance has the average packet latency curve below the average packet latency curves of the two other mechanisms while the relative link energy consumption curves follow approximately with the conventional dynamic link speed mechanism. Thus the *ratio of relative energy saving per latency increase* corresponding to the Dynamic Link Speed Energy Saving mechanism taking into account the performance lays above other mechanisms, which implies that it outperforms them.

5.9 Different Topology & Scalability

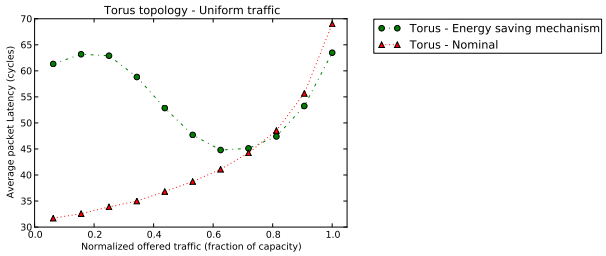
The dynamic link speed energy saving mechanism described in this thesis works in the granularity of links and their associated components, thus it is versatile and independent with different topologies and it scales well with large networks.

To test the versatility and scalability we conduct different scenarios that are different than those configured in previous sections. More specifically we carry out 2 other experiments: the first experiment is configured with another popular topology - the torus topology and the second experiment is a scenario with significantly more processing nodes. We then compare the behaviors of the network systems in this section with the behaviors in previous sections.

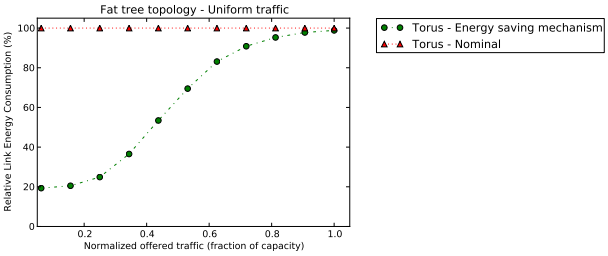
5.9.1 Different Topology

We carried out a set of simulations with typical parameters as described in section 5.3, the topology of the networks in these simulations are 4-ary 3-n torus. Two kinds of traffic patterns are *uniform* and *Lower-Upper Gauss-Seidel solver traffic*.

Fig. 5.9.1 and 5.9.2 show the *relative link power consumption* and *average packet latency* for the torus topology with uniform and Lower-Upper Gauss-Seidel solver traffic respectively with an increasing range of traffic from low to almost saturated. The behaviors of the networks deployed our proposed energy saving mechanism are labeled as *Torus - Energy saving mechanism*, and the default network systems are labeled as *Torus - Nominal*.



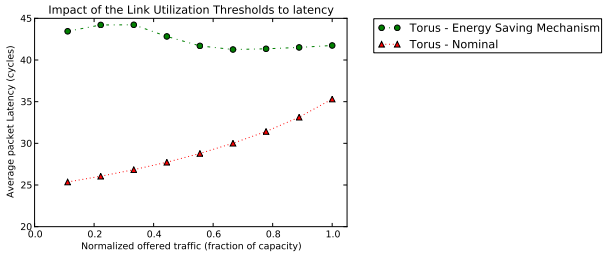
(a) Average Packet Latency - Uniform traffic



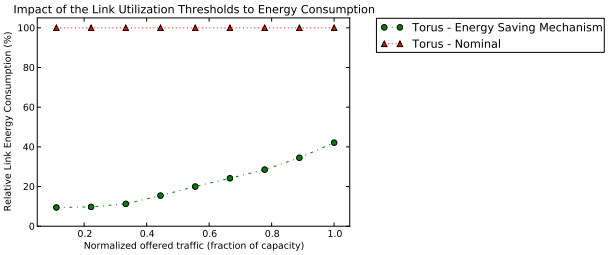
(b) Relative Link Energy Consumption - Uniform traffic

Figure 5.9.1: Mechanism overview with Torus topology - Uniform traffic

The trend for the torus topology is similar to the trend of the fat tree topology in section 5.4: the low traffic triggers the mechanism to save energy with an increase in average packet latency. This trend confirms that the mechanism works in a way that is versatile to network topologies.



(a) Average Packet Latency - Gauss-Seidel solver traffic



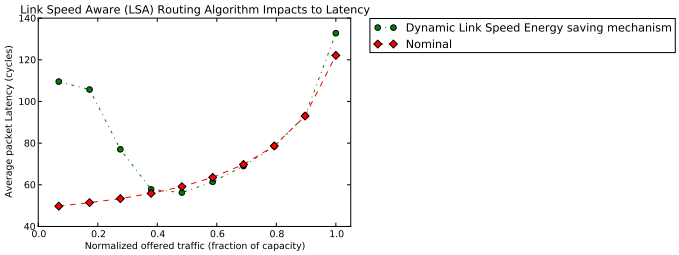
(b) Relative Link Energy Consumption - Gauss-Seidel solver traffic

Figure 5.9.2: Mechanism overview with torus topology - Gauss-Seidel solver traffic

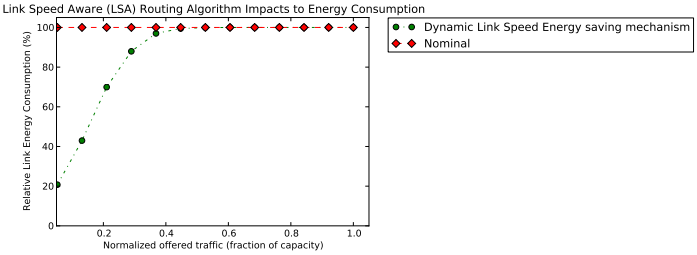
5.9.2 Scalability

We conduct the simulation with uniform traffic with a much larger number of processing nodes to test the scalability of the mechanism (512 nodes compared with 64 nodes in the previous sections). Fig. 5.9.3 and Fig. Fig. 5.9.4 show the behaviors for fat tree topology and torus topology respectively.

A similar trend happens for the *relative link power consumption* and *average packet latency*: the mechanism saves energy in light traffic situation, coupled with an increase in the average packet latency. This trend illustrates that the mechanism relies on the monitoring and decision making phases for links, which work in a distributed fashion. Thus, this mechanism scales well.

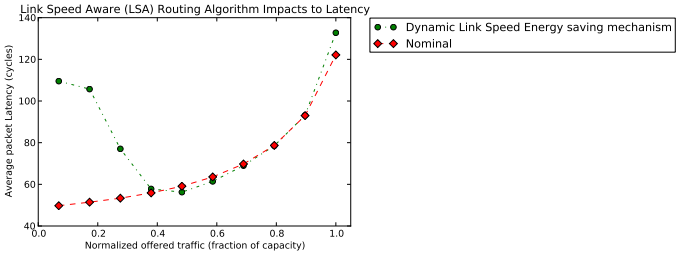


(a) Average Packet Latency - Fat Tree Topology

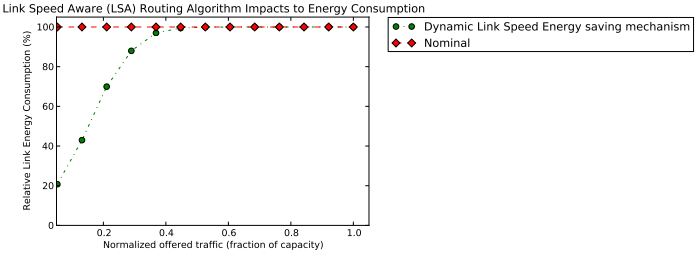


(b) Relative Link Energy Consumption - - Fat Tree Topology

Figure 5.9.3: Scalability Test - 8-ary 3-n fat tree topology with 512 nodes



(a) Average Packet Latency - Torus Topology



(b) Relative Link Energy Consumption - - Torus Topology

Figure 5.9.4: Scalability Test - 8-ary 3-n torus topology with 512 nodes

6

Conclusions

6.1 Final Conclusions

The main contribution of the work described in this thesis composes of two aspects. The first aspect is the proposals that leverages the multi-lane link technology to dynamically adjust the number of active lanes and corresponding link speed to save energy when the traffic is low. The second aspect is the introduction of link speed aware routing policy to boost the performance of the network when the dynamic link speed energy saving mechanism is deployed. Hence, this work contributes toward the trend of the *energy proportional computing system* where the energy consumption is more proportional to the amount of traffic it carries. The mechanism works at the granularity of links and the system node they attach to, it is independent of the choice of network topology and routing schemes.

More specifically, the mechanism achieves its objectives with different propos-

als and policies:

- The mechanism monitors the link and buffer utilization in a history-based distributed manner; the prediction of the link and buffer utilization is smoothed out by a *weight* value. The prediction of the link utilization faithfully reflexes the amount of traffic presenting on a link. The prediction of the buffer utilization plays a role as the litmus test to detect the congestion situation of the network.

- The mechanism makes use of several parameters, including the link utilization thresholds, buffer thresholds. Those parameters are fully flexible and configurable to achieve different objectives of the network.

- The mechanism introduces several policies, such as the two-level threshold policy, the link flip flop avoidance technique, the link speed change policy options to further tune the energy saving mechanism as compared to those in literature.

- The Link Speed Aware routing policy deploys on top of another routing algorithm to give preference for links with high speed to boost the performance of the networks deploying the energy saving mechanism. These networks operate with links at different speed levels.

We have conducted the experiments with different scenarios to evaluate the effectiveness of the proposed dynamic link speed energy saving mechanism and the impacts of different parameters and policies under a range of traffic load. Our energy saving mechanism takes into account the performance and outperforms other conventional energy saving mechanisms in literature. The monitoring and decision making phases of the mechanism happen in a distributed history-based manner. Thus it is well-scalable and versatile to different network topologies and routing schemes. We make no assumption about underlying network infrastructure since our mechanism is universal for different domains of interconnection networks - ranging from on-chip networks to wide-area networks...

6.2 Further Work & Open lines

The work presented in this thesis raises several issues and deserves greater attention. For example, the *link transition time* depends on the link technology and is subject to change over time. This transition time influences the value of *decision_period* and the *history window size H*, thus they need to be better-configured for the mechanism to perform at its best. Besides, the *two-level threshold* and the *link speed aware routing* policies base only only local information. Some kind of feedback providing a broader view of the network states like [24–26] will help those policies perform better on energy efficient network systems.

6.3 Publications

1. Hai Nguyen, Gonzalo Zarza, Daniel Franco, Emilio Luque - *"History Aware Routing Algorithm For Energy Saving in Interconnection Networks"*, (PDPTA'13)
2. Hai Nguyen, Daniel Franco, Emilio Luque - *"Minimum Spanning Tree For Energy Saving in Interconnection Networks"* – (PDPTA'13)
3. Hai Nguyen, Daniel Franco, Emilio Luque - *"Performance-Aware Energy Saving Mechanism in Interconnection Networks for Parallel Systems"* - (ICCS'14)
4. Hai Nguyen, Daniel Franco, Emilio Luque - *"A Dynamic Link Speed Mechanism for Energy Saving in Interconnection Networks"* - (The Scientific World Journal - *submitted*)

References

[1] Grand challenges. URL http://en.wikipedia.org/wiki/Grand_Challenges.

[2] Titan supercomputer. URL https://www.olcf.ornl.gov/titan/.

[3] Hypertransport i/o link specification, 2008. URL http://www.hypertransport.org/docs/twgdocs/HTC20051222-00046-0028.pdf.

[4] Intel quickpath architecture. a new system architecture for unleashing the performance of future generations of intel multi-core microprocessors, 2008. URL http://www.intel.com/technology/quickpath/whitepaper.pdf.

[5] The green500 list, 2014. URL http://www.green500.org.

[6] Top500 supercomputer sites, 2014. URL http://www.top500.org.

[7] Booksim interconnection network simulator., April 2014. URL https://nocs.stanford.edu/cgi-bin/trac.cgi/wiki/Resources/BookSim.

[8] Dennis Abts, Michael R. Marty, Philip M. Wells, Peter Klausler, and Hong Liu. Energy proportional datacenter networks. *SIGARCH Comput. Archit. News*, 38(3):338–347, June 2010. ISSN 0163-5964.

[9] M. Alonso, J. Martinez, V. Santonja, and P. Lopez. Reducing power consumption in interconnection networks by dynamically adjusting link width. In *International European Conference on Parallel Processing (Euro-Par 2004)*, volume 3149 of *Lecture Notes in Computer Science*, pages 882–890. Springer Berlin / Heidelberg, 2004. ISBN 978-3-540-22924-7.

[10] M. Alonso, S. Coll, J.M. Martinez, V. Santonja, P. Lopez, and J. Duato. Dynamic power saving in fat-tree interconnection networks using on/off links. In *International Parallel and Distributed Processing Symposium (IPDPS 2006)*, page 8 pp., april 2006.

[11] Marina Alonso, Salvador Coll, Juan-Miguel Martinez, Vicente Santonja, Pedro Lopez, and Jose Duato. Power saving in regular interconnection networks. *Parallel Computing*, 36(12):696 – 712, 2010. ISSN 0167-8191. doi:http://dx.doi.org/10.1016/j.parco.2010.08.003. URL http://www.sciencedirect.com/science/article/pii/S0167819110001109.

[12] Infiniband Trade Association. Infiniband architecture specification 1.2.1, vols 1 & 2. URL http://www.infinibandta.org/specs.

[13] Nordman B. Energy efficient ethernet: Outstanding questions. Technical report, Lawrence Berkeley National Laboratory, 2007.

[14] Keren Bergman, Shekhar Borkar, Dan Campbell, William Carlson, William Dally, Monty Denneau, Paul Franzon, William Harrod, Jon Hiller, Sherman Karp, Stephen Keckler, Dean Klein, Robert Lucas, Mark Richards, Al Scarpelli, Steven Scott, Allan Snavely, Thomas Sterling, R. Stanley Williams, Katherine Yelick, Keren Bergman, Shekhar Borkar, Dan Campbell, William Carlson, William Dally, Monty Denneau, Paul Franzon, William Harrod, Jon Hiller, Stephen Keckler, Dean Klein, Peter Kogge, R. Stanley Williams, and Katherine Yelick. Exascale computing study: Technology challenges in achieving exascale systems peter kogge, editor & study lead, 2008.

[15] Carlos Heriberto Nunez Castillo. *Predictive and Distributed Routing Balancing for High Speed Interconnection Networks*. PhD thesis, Universitat Autonoma de Barcelona, July 2013.

[16] S. Conner, S. Akioka, M.J. Irwin, and P. Raghavan. Link shutdown opportunities during collective communications in 3-d torus nets. In *Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International*, pages 1–8, March 2007. doi:10.1109/IPDPS.2007.370534.

[17] William Dally and Brian Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003. ISBN 0122007514.

[18] DARPA. Ubiquitous High Performance Computing (UHPC), March 2010. URL https://www.fbo.gov/spg/ODA/DARPA/CMO/DARPA-BAA-10-37/listing.html. Broad Agency Announcement (BAA).

[19] NASA Advanced Supercomputing Division. Nas parallel benchmarks. URL http://www.nas.nasa.gov/publications/npb.html.

[20] Gordana Dodig-Crnkovic. Scientific methods in computer science. In *Conference for the Promotion of Research in IT at New Universities and at University Colleges in Sweden*, April 2002. URL http://www.es.mdh.se/publications/375-.

[21] Yong Dong and Juan Chen. Network energy optimization for mpi operations. In *Intelligent Computation Technology and Automation (ICICTA), 2012 Fifth International Conference on*, pages 221–224, Jan 2012. doi:10.1109/ICICTA.2012.62.

[22] Jose Duato, Sudhakar Yalamanchili, and Ni Lionel. *Interconnection Networks: An Engineering Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2002. ISBN 1558608524.

[23] Wu-chun Feng and Kirk Cameron. The green500 list: Encouraging sustainable supercomputing. *Computer*, 40(12):50–55, December 2007. ISSN 0018-9162. doi:10.1109/MC.2007.445. URL http://dx.doi.org/10.1109/MC.2007.445.

[24] D. Franco, I. Garces, and E. Luque. Distributed routing balancing for interconnection network communication. In *High Performance Computing, 1998. HIPC '98. 5th International Conference On*, pages 253–261, Dec 1998. doi:10.1109/HIPC.1998.737996.

[25] D. Franco, I. Garcés, and E. Luque. A new method to make communication latency uniform: Distributed routing balancing. In *Proceedings of the 13th International Conference on Supercomputing*, ICS '99, pages 210–219, New York, NY, USA, 1999. ACM. ISBN 1-58113-164-X. doi:10.1145/305138.305195. URL http://doi.acm.org/10.1145/305138.305195.

[26] D. Franco, I. Garces, and E. Luque. Avoiding communication hot-spots in interconnection networks. In *Systems Sciences, 1999. HICSS-32. Proceedings*

of the 32nd Annual Hawaii International Conference on, volume Track8, pages 10 pp.–, Jan 1999. doi:10.1109/HICSS.1999.773072.

[27] Kinshuk Govil, Edwin Chan, and Hal Wasserman. Comparing algorithm for dynamic speed-setting of a low-power cpu. In *Proceedings of the 1st Annual International Conference on Mobile Computing and Networking*, MobiCom '95, pages 13–25, New York, NY, USA, 1995. ACM. ISBN 0-89791-814-2. doi:10.1145/215530.215546. URL http://doi.acm.org/10.1145/215530.215546.

[28] PCI Special Interest Group. Pci express base 3.0 specification. URL www.pcisig.com/specifications/pciexpress/base3.

[29] C. Gunaratne, K. Christensen, B. Nordman, and S. Suen. Reducing the energy consumption of ethernet with adaptive link rate (alr). *Computers, IEEE Transactions on*, 57(4):448–461, April 2008. ISSN 0018-9340. doi:10.1109/TC.2007.70836.

[30] Brandon Heller, Srini Seetharaman, Priya Mahadevan, Yiannis Yiakoumis, Puneet Sharma, Sujata Banerjee, and Nick McKeown. Elastictree: saving energy in data center networks. In *Proceedings of the 7th USENIX conference on Networked systems design and implementation*, NSDI'10, pages 17–17, Berkeley, CA, USA, 2010. USENIX Association. URL http://dl.acm.org/citation.cfm?id=1855711.1855728.

[31] John L. Hennessy and David A. Patterson. *Computer Architecture, Fifth Edition: A Quantitative Approach - Appendix F*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 5th edition, 2011. ISBN 012383872X, 9780123838728.

[32] T. Hoefler. Software and hardware techniques for power-efficient hpc networking. *Computing in Science Engineering*, 12(6):30–37, Nov 2010. ISSN 1521-9615. doi:10.1109/MCSE.2010.96.

[33] Myricom Inc. home page. URL http://www.myricom.com.

[34] K. Kant. Towards a virtualized data center transport protocol. In *INFOCOM Workshops 2008, IEEE*, pages 1–6, April 2008. doi:10.1109/INFOCOM.2008.4544645.

[35] K. Kant. Multi-state power management of communication links. In *Communication Systems and Networks (COMSNETS), 2011 Third International Conference on*, pages 1–10, 2011.

[36] Krishna Kant and Youjip Won. Server capacity planning for web traffic workload. *IEEE Trans. on Knowl. and Data Eng.*, 11(5):731–747, September 1999. ISSN 1041-4347. doi:10.1109/69.806933. URL http://dx.doi.org/10.1109/69.806933.

[37] SameeUllah Khan, Lizhe Wang, LaurenceT. Yang, and Feng Xia. Green computing and communications. *The Journal of Supercomputing*, 63(3): 637–638, 2013. ISSN 0920-8542. doi:10.1007/s11227-011-0718-x. URL http://dx.doi.org/10.1007/s11227-011-0718-x.

[38] E.J. Kim, K.H. Yum, G.M. Link, N. Vijaykrishnan, M. Kandemir, M.J. Irwin, M. Yousif, and C.R. Das. Energy optimization techniques in cluster interconnects. In *International Symposium on Low Power Electronics and Design (ISLPED 2003)*, pages 459–464, aug. 2003.

[39] Peter M. Kogge. Architectural Challenges at the Exascale Frontier (Invited talk). In *Simulating the Future (STF 2008): Using One Million Cores and Beyond*, September 2008.

[40] James H. Laros, III, Kevin T. Pedretti, Suzanne M. Kelly, Wei Shu, and Courtenay T. Vaughan. Energy based performance tuning for large scale high performance computing systems. In *Proceedings of the 2012 Symposium on High Performance Computing*, HPC '12, pages 6:1–6:10, San Diego, CA, USA, 2012. Society for Computer Simulation International. ISBN 978-1-61839-788-1. URL http://dl.acm.org/citation.cfm?id=2338816.2338822.

[41] Charles E. Leiserson. Fat-trees: universal networks for hardware-efficient supercomputing. *IEEE Trans. Comput.*, 34(10):892–901, October 1985. ISSN 0018-9340. URL http://dl.acm.org/citation.cfm?id=4492.4495.

[42] Jian Li, Wei Huang, Charles Lefurgy, Lixin Zhang, Wolfgang E. Denzel, Richard R. Treumann, and Kun Wang. Power shifting in thrifty interconnection network. In *Proceedings of the 2011 IEEE 17th International Symposium on High Performance Computer Architecture*, HPCA '11, pages 156–167, Washington, DC, USA, 2011. IEEE Computer Society. ISBN

978-1-4244-9432-3. URL http://dl.acm.org/citation.cfm?id=2014698.2014904.

[43] Pedro Lopez and Jose Duato. Deadlock-free adaptive routing algorithms for the 3d-torus: Limitations and solutions. In Arndt Bode, Mike Reeve, and Gottfried Wolf, editors, *PARLE '93 Parallel Architectures and Languages Europe*, volume 694 of *Lecture Notes in Computer Science*, pages 684–687. Springer Berlin Heidelberg, 1993. ISBN 978-3-540-56891-9. doi:10.1007/3-540-56891-3_59. URL http://dx.doi.org/10.1007/3-540-56891-3_59.

[44] Priya Mahadevan, Puneet Sharma, Sujata Banerjee, and Parthasarathy Ranganathan. Energy aware network operations. In *Proceedings of the 28th IEEE International Conference on Computer Communications Workshops*, INFOCOM'09, pages 25–30, Piscataway, NJ, USA, 2009. IEEE Press. ISBN 978-1-4244-3968-3. URL http://dl.acm.org/citation.cfm?id=1719850.1719855.

[45] Z. Meggyesi. Fiber channel overview. URL http://hsi.web.cern.ch/hsi.

[46] Li-Shiuan Peh Natalie Enright Jerger. *On-Chip Networks - Synthesis Lectures on Computer Architecture*, volume 4. Morgan & Claypool, 2009. doi:10.2200/S00209ED1V01Y200907CAC008. URL http://www.morganclaypool.com/doi/abs/10.2200/S00209ED1V01Y200907CAC008.

[47] G.S. Patel, S.M. Chai, S. Yalamanchili, and D.E. Schimmel. Power constrained design of multiprocessor interconnection networks. In *Computer Design: VLSI in Computers and Processors, 1997. ICCD '97. Proceedings., 1997 IEEE International Conference on*, pages 408–416, Oct 1997. doi:10.1109/ICCD.1997.628902.

[48] Vern Paxson and Sally Floyd. Wide area traffic: The failure of poisson modeling. *IEEE/ACM Trans. Netw.*, 3(3):226–244, June 1995. ISSN 1063-6692. doi:10.1109/90.392383. URL http://dx.doi.org/10.1109/90.392383.

[49] Fabrizio Petrini, Wu-chun Feng, Adolfy Hoisie, Salvador Coll, and Eitan Frachtenberg. The quadrics network (qsnet): High-performance clustering technology. In *Proceedings of the The Ninth Symposium on High Perfor-*

mance Interconnects, HOTI '01, pages 125–, Washington, DC, USA, 2001. IEEE Computer Society. ISBN 0-7695-1357-3. URL http://dl.acm.org/citation.cfm?id=572719.876332.

[50] Andreas G. Savva, Theocharis Theocharides, and Vassos Soteriou. Intelligent on/off dynamic link management for on-chip networks. *JECE*, 2012:6:6–6:6, January 2012. ISSN 2090-0147.

[51] John Shalf, Shoaib Kamil, Leonid Oliker, and David Skinner. Analyzing ultra-scale application communication requirements for a reconfigurable hybrid interconnect. In *Proceedings of the 2005 ACM/IEEE Conference on Supercomputing*, SC '05, pages 17–, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 1-59593-061-2. doi:10.1109/SC.2005.12. URL http://dx.doi.org/10.1109/SC.2005.12.

[52] Li Shang, Li-Shiuan Peh, and N.K. Jha. Dynamic voltage scaling with links for power optimization of interconnection networks. In *International Symposium on High-Performance Computer Architecture (HPCA-9 2003)*, pages 91 – 102, feb. 2003.

[53] Li Shang, Li-Shiuan Peh, and N. K. Jha. Powerherd: A distributed scheme for dynamically satisfying peak-power constraints in interconnection networks. *Trans. Comp.-Aided Des. Integ. Cir. Sys.*, 25(1):92–110, November 2006. ISSN 0278-0070. doi:10.1109/TCAD.2005.852438. URL http://dx.doi.org/10.1109/TCAD.2005.852438.

[54] V. Soteriou and Li-Shiuan Peh. Dynamic power management for power optimization of interconnection networks using on/off links. In *Symposium on High Performance Interconnects*, pages 15 – 20, aug. 2003.

[55] V. Soteriou and Li-Shiuan Peh. Design-space exploration of power-aware on/off interconnection networks. In *IEEE International Conference on Computer Design: VLSI in Computers and Processors (ICCD 2004)*, pages 510 – 517, oct. 2004.

[56] Vassos Soteriou, Noel Eisley, and Li-Shiuan Peh. Software-directed power-aware interconnection networks. *ACM Trans. Archit. Code Optim.*, 4(1), March 2007. ISSN 1544-3566. doi:10.1145/1216544.1216548. URL http://doi.acm.org/10.1145/1216544.1216548.

[57] Murad S. Taqqu, Walter Willinger, and Robert Sherman. Proof of a fundamental result in self-similar traffic modeling. *SIGCOMM Comput. Commun. Rev.*, 27(2):5–23, April 1997. ISSN 0146-4833. doi:10.1145/263876.263879. URL http://doi.acm.org/10.1145/263876.263879.

[58] Ehsan Totoni, Nikhil Jain, and Laxmikant V. Kale. Toward runtime power management of exascale networks by on/off control of links. In *Proceedings of the 2013 IEEE 27th International Symposium on Parallel and Distributed Processing Workshops and PhD Forum*, IPDPSW '13, pages 915–922, Washington, DC, USA, 2013. IEEE Computer Society. ISBN 978-0-7695-4979-8. doi:10.1109/IPDPSW.2013.191. URL http://dx.doi.org/10.1109/IPDPSW.2013.191.

[59] G. Varatkar and R. Marculescu. Traffic analysis for on-chip networks design of multimedia applications. In *Design Automation Conference, 2002. Proceedings. 39th*, pages 795–800, 2002. doi:10.1109/DAC.2002.1012731.

[60] Jeffrey S. Vetter and Frank Mueller. Communication characteristics of large-scale scientific applications for contemporary cluster architectures. In *In International Parallel and Distributed Processing Symposium*, 2002.

[61] Hang-Sheng Wang, Li-Shiuan Peh, and S. Malik. A power model for routers: modeling alpha 21364 and infiniband routers. In *High Performance Interconnects, 2002. Proceedings. 10th Symposium on*, pages 21–27, 2002. doi:10.1109/CONECT.2002.1039253.

[62] Hang-Sheng Wang, Xinping Zhu, Li-Shiuan Peh, and S. Malik. Orion: a power-performance simulator for interconnection networks. In *Microarchitecture, 2002. (MICRO-35). Proceedings. 35th Annual IEEE/ACM International Symposium on*, pages 294–305, 2002. doi:10.1109/MICRO.2002.1176258.

[63] Lizhe Wang and SameeU. Khan. Review of performance metrics for green data centers: a taxonomy study. *The Journal of Supercomputing*, 63(3):639–656, 2013. ISSN 0920-8542. doi:10.1007/s11227-011-0704-3. URL http://dx.doi.org/10.1007/s11227-011-0704-3.

[64] Lizhe Wang, SameeU. Khan, and Jai Dayal. Thermal aware workload placement with task-temperature profiles in a data center. *The Journal of Super-*

computing, 61(3):780–803, 2012. ISSN 0920-8542. doi:10.1007/s11227-011-0635-z. URL http://dx.doi.org/10.1007/s11227-011-0635-z.

[65] W. Willinger, M.S. Taqqu, R. Sherman, and D.V. Wilson. Self-similarity through high-variability: statistical analysis of ethernet lan traffic at the source level. *Networking, IEEE/ACM Transactions on*, 5(1):71–86, Feb 1997. ISSN 1063-6692. doi:10.1109/90.554723.

[66] Jieming Yin, Pingqiang Zhou, Anup Holey, Sachin S. Sapatnekar, and Antonia Zhai. Energy-efficient non-minimal path on-chip interconnection network for heterogeneous systems. In *Proceedings of the 2012 ACM/IEEE international symposium on Low power electronics and design*, ISLPED '12, pages 57–62, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1249-3.