UNIVERSITY OF CALIFORNIA,
IRVINE

# Design and Evaluation of High-Performance and Fault-tolerant Routing Algorithms for 3D-NoCs

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Electrical and Computer Engineering

by

Ronak Salamat

2018

# DEDICATION

To my beloved Misagh, for all his endless love, support, patience and encouragement not only in my academic endeavors but also in my everyday life. He has always been the first person I turn to in the face of challenges.
I would also like to dedicate this thesis to my parents Saeid and Roshank, who put my life and goals ahead of theirs, sacrificed a lot so I can follow my dreams, and raised me to be where I am.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my advisor, Professor Nader Bagherzadeh for all his support, encouragements and my achievements during the past five years. He is also a mentor of life.

I would also like to extend my sincere gratitude to an exceptional person both in my academic and personal life, Masoumeh Ebrahimi. She is both a dear friend of mine and the most caring motivator.

# CURRICULUM VITAE

## Ronak Salamat

### EDUCATION

| | |
|---|---|
| **Doctor of Philosophy in Computer Engineering** | **2018** |
| University of California, Irvine | *Irvine, California* |
| **Master of Science in Computer Engineering** | **2012** |
| University of Amirkabir, Tehran Polytechnic | *Tehran, Iran* |
| **Bachelor of Science in Computer Engineering** | **2009** |
| Iran University of Science and Technology | *Tehran, Iran* |

### RESEARCH EXPERIENCE

| | |
|---|---|
| **Graduate Research Assistant** | **2013–2018** |
| University of California, Irvine | *Irvine, California* |
| **Graduate Research Assistant** | **2009–2012** |
| University of Amirkabir, Tehran Polytechnic | *Tehran, Iran* |

### TEACHING EXPERIENCE

| | |
|---|---|
| **Organization of Digital Computer Lab** | **Winter 2017, Winter 2018** |
| University of California, Irvine | *Irvine, California* |
| **Organization of Digital Computer** | **Fall 2016, Fall 2017** |
| University of California, Irvine | *Irvine, California* |
| **C Programming** | **Spring 2015, Spring 2017** |
| University of California, Irvine | *Irvine, California* |
| **Advanced Computer Architecture** | **Fall 2014** |
| University of California, Irvine | *Irvine, California* |

## REFEREED JOURNAL PUBLICATIONS

**LEAD: An Adaptive 3D-NoC Routing Algorithm with Queuing-theory Based Analytical Verification** **2018**
IEEE Transaction on Computers

**A Resilient Routing Algorithm with Formal Reliability Analysis for Partially Connected 3D-NoCs** **2016**
IEEE Transaction on Computers

## REFEREED CONFERENCE PUBLICATIONS

**CoBRA: Low cost compensation of TSV failures in 3D-NoC** **2016**
IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology System

**An adaptive, low restrictive and fault resilient routing algorithm for 3d network-on-chip** **2015**
Euromicro International Conference on Parallel, Distributed, and Network-Based Processing

**Fault-tolerant assessment and enhancement in the reconfigurable network-on-chip** **2012**
International Symposium on Computer Architecture and Digital Systems

**Susceptibility Analysis of LEON3 Embedded Processor against Multiple Event Transients and Upsets** **2012**
International Conference on Computational Science and Engineering

**Fault-tolerance assessment and enhancement in SoCWire interface: A system-on-chip wires** **2011**
International On-Line Testing Symposium

# ABSTRACT OF THE DISSERTATION

Design and Evaluation of High-Performance and Fault-tolerant Routing Algorithms for 3D-NoCs

By

Ronak Salamat

Doctor of Philosophy in Electrical and Computer Engineering

University of California, Irvine, 2018

Professor Nader Bagherzadeh, Chair

2D Network-on-Chips (NoCs) have been the mainstream interconnection technology for multi-core systems. In this dissertation, different aspect of the alternative 3D-NoC technology have been investigated. The 3D technology compensates for the deficiencies of 2D-NoCs such as long latency, power overhead and lack of scalability. While the routers in a traditional 3D-NoC are fully-connected using Through-Silicon-Via (TSV), we consider partially-connected 3d-NoCs to mitigate the silicon area overhead of a fully-connected architecture. The TSV fault sources such as thermal stress, warpage, impurities and misalignment have been reviewed. We investigate the delicacies of designing routing algorithms for partially connected networks. Several high-performance, fault-tolerant and adaptive routing algorithms have been suggested and proved to be livelock- and deadlock-free. The proposed algorithms are capable of tolerating faults on vertical links. The algorithms are then extended to be reconfigurable to tolerate both fabrication-time and run-time TSV failures. Both simulation and analytical models are applied to evaluate the performance of the algorithms. An analytical model, tailored to the adaptivity of the algorithm and under low traffic scenarios, has been developed and the results have been verified by simulation. The algorithms are tested under different traffic patterns, different number of elevators and different elevator assignment mechanisms and shown to outperform the previous work in terms of both network latency and fault-tolerance.

# Chapter 1

# Introduction

On a billion transistors chip, it may not be possible to send a global signal across the chip within real-time bounds [4]. If the System-on-Chip (SoC) is synchronized by a global clock signal, the circuit will be more prone to Electromagnetic Interference (EMI) [1]. Synchronous SoC designs suffer from huge amount of power consumption due to their critical paths and clock trees. Besides, these clock trees are difficult to manage because of clock skew problem. On the other hand, designing an asynchronous system is a more complex problem although they are modular and do not suffer from issues such as clock skew, higher power consumption and EMI [5].

GALS (Globally Asynchronous and Locally Synchronous) have combined the ideas of synchronous and asynchronous designs. GALS divides a system into smaller, locally decoupled synchronous regions. These synchronous regions are connected to each other in an asynchronous way in which there is no need to have a single global clock. One GALS solution is NoC (Network-on-Chip) [4]. NoC improves design productivity by supporting modularity and reusing the complex cores.

Global interconnect does not scale well with the technology advancement and it has

| Bus Cons | NoC Pros |
|---|---|
| Parasitic capacitance increases as more units attached, therefore performance degrades with growth | Performance is not degraded with scaling |
| Bus timing is difficult in a deep submicron process | Network wires can be pipelined because links are point-to-point. |
| Bus arbitration can become a bottleneck as the number of masters grow | Routing decisions are distributed. |
| Bandwidth is limited and shared by all units attached | Aggregated bandwidth scales with the network size. |

Table 1.1: Bus versus NoC Pros and Cons

become as one of the major concerns in current and future high-performance SoC designs. Scalability, higher bandwidth, better throughput and lower power consumption of NoCs have encouraged researchers to consider NoCs as a promising alternative for conventional interconnects [6][7]. Table 1.1 summarizes the cons and pros of SoC versus NoC [8].

NoC consists of resources (Processing Elements) and switches (Routers) connected using channels in a mesh structure so they are communicating with each other by sending messages. According to Figure 1.1, a resource is either a computation or storage unit. A switch S is responsible for routing and buffering messages between resources. Each switch is connected to four other neighboring switches through input and output channels. A channel consists of two one-dimensional point-to-point buses between two switches or a resource and a switch. However, as the number of cores increases, two-dimensional NoC-based (2D-NoC) infrastructures suffer from long latency and power overhead. In other words, the geometric distance between cores dramatically grow when the network size increases.

Three dimensional ICs have attracted a lot of attention in the past few years [9] [10]. 3D-ICs provide better performance, more flexibility and higher throughput as compared with traditional ICs [11] [12], allowing for continued performance improvements

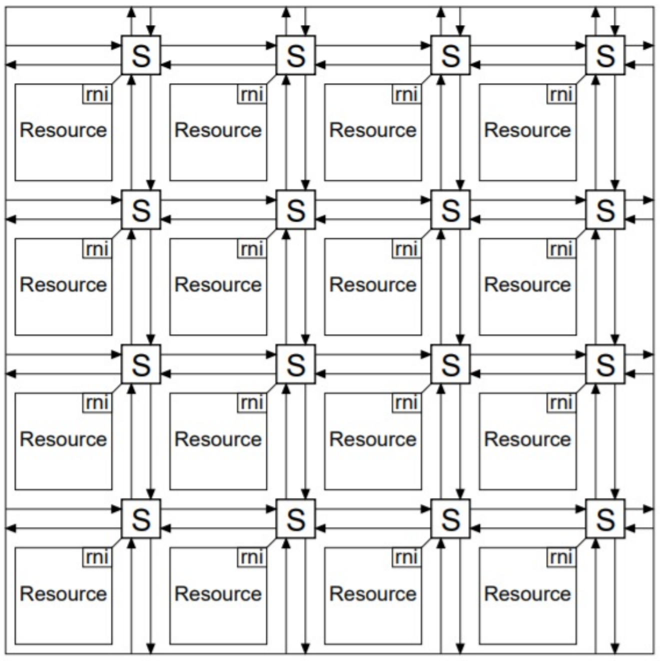


Figure 1.1: A NoC structure[1]

using CMOS technology [13]. Figure 1.2 illustrates a 3D-IC in which 2D layers are stacked on top of each other using *Through-Silicon-Via* (TSV). TSVs are conducting nails which extend out of the back-side of a thinned-down die and enable the vertical interconnect to another die [14] [15]. TSVs are high-density, low-capacity interconnects compared to traditional wire-bonds, and hence allow for many more interconnections between stacked dies, while operating at higher speeds and consuming less power [16].

Transistor density has increased in three dimensional (3D) ICs by vertically stacking multiple dies using a dense and high-speed die-to-die interconnection [17]. Because of the positive correlation between the length of long global wires and performance bottlenecks such as delay and power consumption, it is anticipated that a decrease in the wiring footprint leads to low latency and energy efficient 3D integration. Moreover, 3D-IC offers an opportunity to integrate heterogeneous processes in a more efficient manner. Potential applications include processor (stacking CPU and various levels

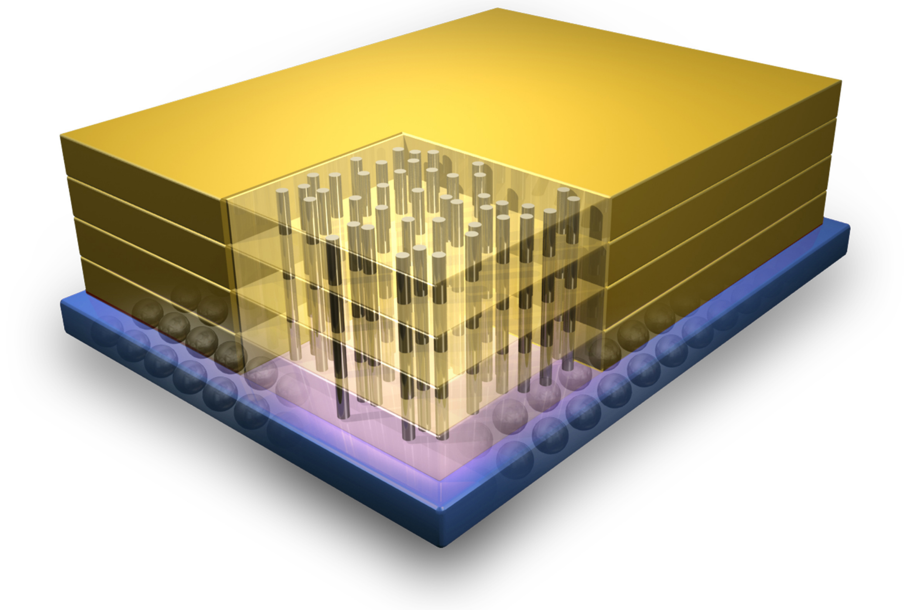


Figure 1.2: A 3D-IC representation

of caches) [18][19], memory (stacking SRAM, DRAM and/or Flash) [20][21][22]. Although there exist alternatives, like wire bonding and micro-bumping, using TSVs achieves higher interconnection density and better performance [23][24].

3D-NoC emerges as an improvement of 2D-NoCs aiming to reduce the length and number of global interconnections. Vertically stacked dies with TSV together with NoCs is a powerful solution to tackle the on-chip communication problem [25] [26]. As can be seen in Figure 1.3, layers of 2D-NoCs are connected to each other through vertical links located at the routers. To do so, every router is equipped with 7 ports for North, South, East, West, Up, Down and Local. TSVs are the most promising solution among other vertical interconnections, since they provide high density, high bandwidth and low power [27]. TSVs impose their own challenges. First, TSV pads consume considerably larger bonding area in each layer compared to horizontal wires[28] [17]. Second, TSV technology does not scale with feature size [29]. Therefore, transistor and wire shrinkage make the above problems even more severe. Third,

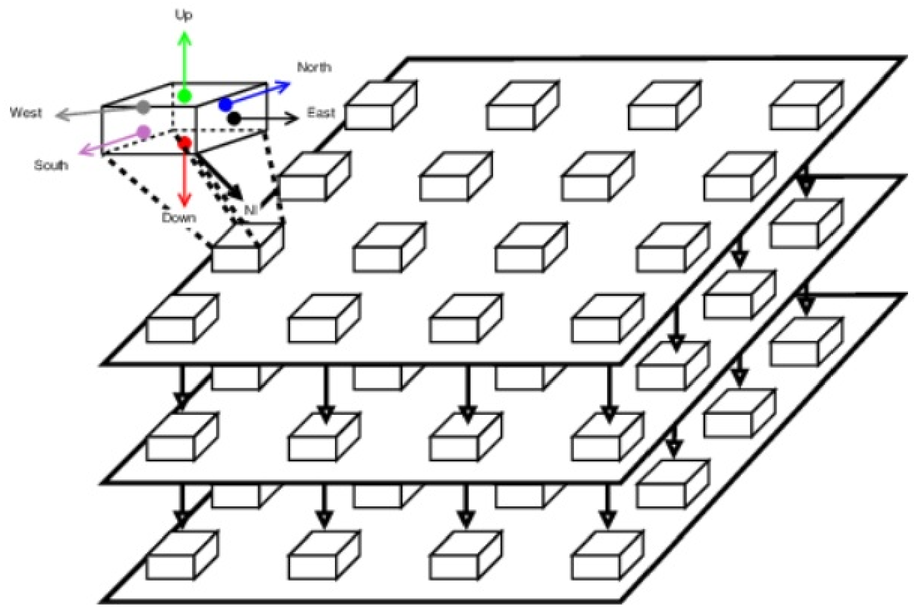


Figure 1.3: A 3D-NoC representation[2]

the TSV fabrication process suffers from low yield [14] [30]. The higher the number of TSVs, the lower is the yield [31]. The cost of high yield TSV manufacturing process is only justifiable in presence of a practical solution to counteract TSV related effects (such as voids in TSVs, TSV pinch-off, oxide defects such as pinholes, thermo-mechanical stress, cracks in micro-bumps, chip warpage, and impurities [32]) which may render the entire chip useless [33]. The low yield is caused by the wide range of chemical and mechanical properties of the materials used in the TSV fabrication process. Specifically, the non-uniformity of chemical and mechanical properties of different materials magnifies the conversion of thermal stress into mechanical stress during fabrication. While the mechanical stress itself can directly lead to mechanical failures, it can also indirectly affect device performance by altering carrier mobility in active silicon [34]. Finally, TSV parasitic parameters depend on the layout and properties of the insulating barrier [35]. Consequently, a non-optimized layout design or an improper insulating barrier material can take away TSV advantages such as high bandwidth and low power consumption [36].

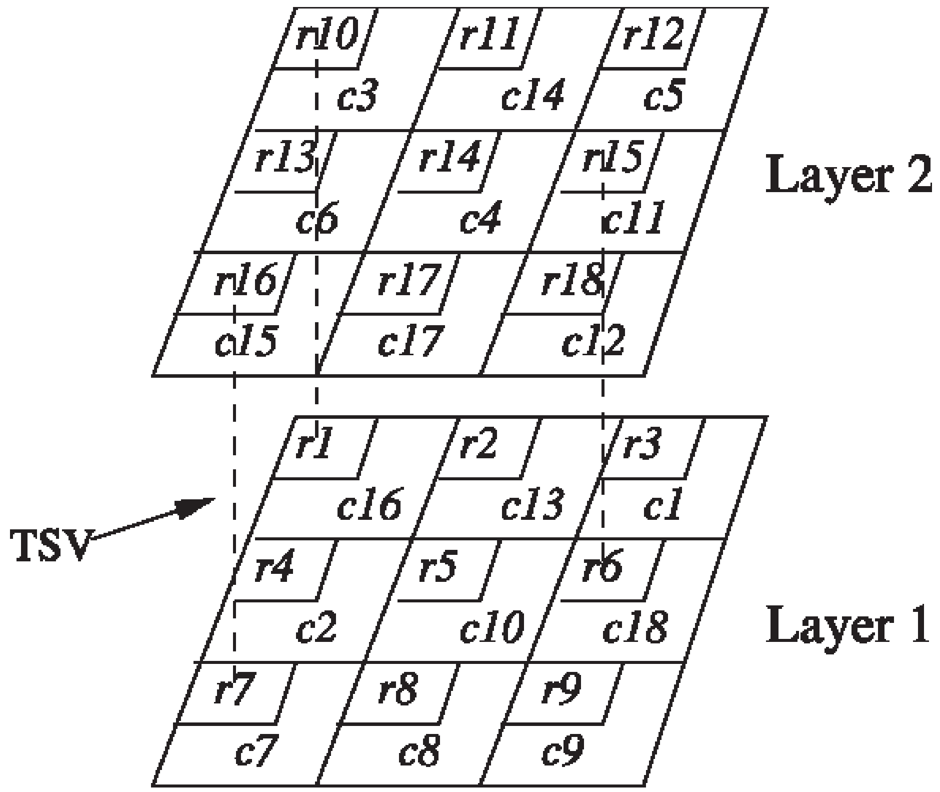


Figure 1.4: A two-layer partially connected 3D-NoC [3]

In short, although 3D-NoC exhibits higher speed and shorter wiring compared to 2D-NoC, employing a large number of TSVs degrades reliability and causes area overhead. To overcome some of these challenges, in *partially connected 3D-NoCs*, a subset of routers are connected to the upper/lower layers using TSVs while the routers in the same layer are connected using global links. This architecture takes advantage of 3D-NoC philosophy while mitigating the disadvantages of a fully-connected 3D-NoC. As it can be seen in Figure 1.4 where $r$ and $c$ represent routers and cores respectively, not all the routers in the 2D-plane are connected to the upper/lower layers through TSVs.

As it has been mentioned, NoC nodes use packets to communicate to each other. The process that is applied to forward the packets along appropriate directions in the network between a source and destination is called routing. Therefore, NoCs should be accommodated into the limited silicon area using efficient topology, routing algorithm and router implementation. A brief discussion of NoC routing basics is presented.

1. Network topology includes the study of the arrangement and connectivity of the routers. Shared-Bus, Ring, Crossbar, Torus and Mesh are among the most popular topologies for on-chip interconnections as in figure 1.5. In shared-bus topology, all PEs are connected to a common shared bus and they compete for exclusive access to the bus. A small modification to the shared-bus topology to allow more concurrent transactions is to create the ring topology where every PE has exactly two neighbors. The crossbar topology is a fully connected one which allows every PE to directly communicate with any other PE. Mesh and torus networks are widely used in multiprocessor architectures because of their simple connection and easy routing provided by adjacency. Both torus and mesh topologies are fully scalable.

2. The switching mechanism determines how messages traverse a route in a network. The goal is to effectively share the network resources among messages traversing the network. Basically, circuit switching and packet switching form the two extremes of switching mechanisms. In the packet switching domain, buffered flow control defines the mechanism that deals with the allocation of channels and buffers for the packets traversing between source and destination. The flow control mechanism is necessary when two or more packets compete to use the same channel, at the same time. Commonly three different buffered flow control strategies are used: store-and-forward, virtual cut through, and wormhole. In store-and-forward [37], each router along the path stores the entire packet in the buffer and then, the packet is forwarded to a selected neighboring router if the chosen neighboring router has enough empty buffering space available to hold the whole packet. In virtual cut-through [37], the router starts

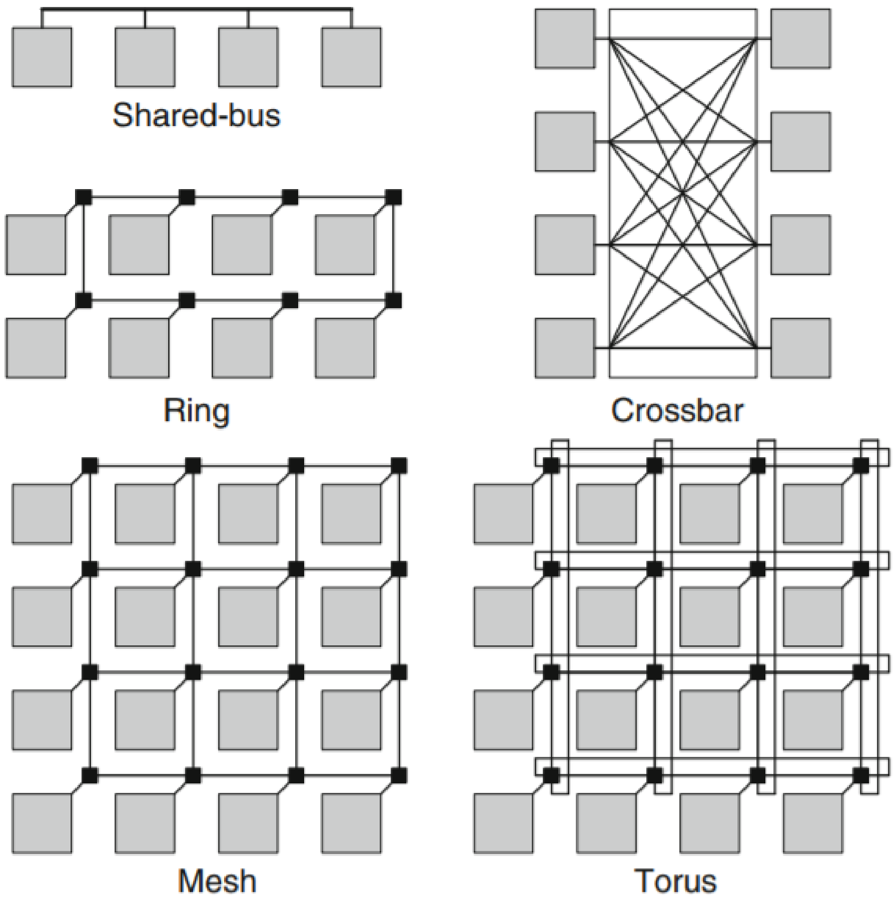


Figure 1.5: Network topologies

forwarding a packet before the whole packet has been received as opposed to the store-and-forward. Finally, in wormhole, a packet is divided into smaller segments called FLITs (FLow control digIT)[38].Then, the flits are routed through the network one after another in a pipelined fashion.

3. Virtual channels overcome the problem of blockages in the wormhole network. There is a possibility of blocking in the wormhole network when a packet reserves a channel along a path which is prevented to be used by other packets. Virtual channel implementation is accomplished by assigning several VCs, each with a separate flit queue, to each physical channel. For each VC, when the header flit arrives, a buffer will be assigned to the incoming packet, and is reserved until the trailer flit is transmitted. If a packet holding a VC gets blocked, other packets from other VCs can still traverse the physical channel. A typical router with virtual channels is represented in figure 1.6.

A routing algorithm can be seen as the routing function and the selection function. First, the routing function computes the set of admissible output channels towards which the packet can be forwarded to reach the destination. Then, the selection function is used to select one output channel from the set of admissible output channels returned by the routing function. In a router implementing a deterministic routing algorithm, the selection block is not present since the routing function returns only a single output port. In a router implementing an oblivious routing algorithm, the selection block takes its decision based solely on the information provided by the header flit.

The main problem in vertically partially connected 3D-NoC is the packet routing strategy where the traditional simple routing algorithms such as XYZ are not applicable. XYZ simply means that packets take the $X$, $Y$ and $Z$ dimensions consecutively. y following $X$, $Y$ and $Z$ dimensions in order, we might end up at node with no ver-

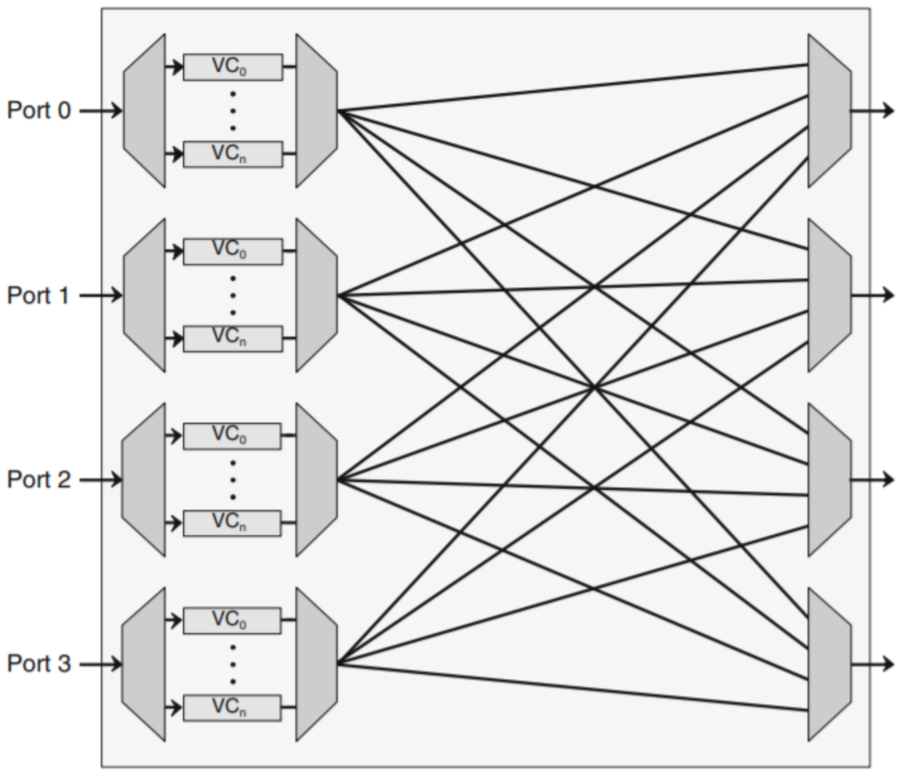


Figure 1.6: A router using VCs

tical link. An appropriate routing algorithm should be used to utilize a partially connected 3D-NoC. The absence of TSVs at certain points results in more load being pushed on the present TSVs, and the algorithm should be capable of distributing the load across the TSVs more uniformly to enhance the network performance and mitigate TSV aging. Routing algorithms are classified as deterministic and adaptive. While the former is simple, it is incapable of balancing the load across the links in non-uniform traffic [39]. The latter is applied to distribute load across the links and avoiding congested regions, and enhance the network performance [40].

Designing an adaptive deadlock-free routing algorithm in partially connected 3D-NoC is very challenging due to the possibility of forming a cycle within and between three planes (i.e. $XY$, $XZ$, and $YZ$) and the current state-of-the-art is still lacking a viable solution. Another concern with the use of TSVs is the ensuing reliability issues. While the reliability aspects of 2D networks have been extensively studied, investigation of TSV fault sources and their implication in terms of network performance is still a developing topic. Consequently, it is desirable to evaluate the reliability and performance of the designed routing algorithm under the influence of TSV faults and non-idealities.

Performance metrics of routing algorithms on a specific platform are typically evaluated through simulation in order to determine whether or not the algorithm satisfies application constraints [41] [42]. Unfortunately, simulation-based performance analysis is considerably time-consuming and the situation exacerbates as the network size increases. In addition, comparisons are challenging as different methods might be applied on different tools. Analytical models are an alternative approach to estimate performance metrics in a fraction of time. To be tractable, most analytical models rely on certain assumptions. The validity of an analytical model is directly related to how well the real-world scenario follows the assumptions. If a good enough analytical model is used to approximate the real-world situation, the desired performance metrics can be calculated efficiently and reliably. Typically, the analytical models are used to get the design within an acceptable vicinity of the desired outcome, and then simulations are performed for fine tuning and removing the effect of assumptions. This analysis-simulation sequence saves designers considerable time and helps them focus on improving the design by getting fast and accurate feedback from the proposed designs.

The rest of the dissertation is organized as follows. Chapter 2 summarizes the related work focuses 3D integration 2.1, TSV reliability 2.2, routing algorithms 2.3 and analytical verification 2.4 presented for 3D-NoC. In Chapter 3, ETW, a lightweight and adaptive routing algorithm for partially connected 3D-NoC is presented. ETW imposes specific rules on choosing elevators for vertical transmission which threatens the fault tolerance of the network. Moreover, reliability analysis is conducted on the TSVs of the partially connected 3D-NoC. CoBRA routing algorithm is presented in Chapter 4 which makes some modifications on ETW to enhance the fault tolerance of the network. Chapter 5 presents an adaptive routing algorithm named LEAD which has no specific rules on choosing elevators. Also, the analytical models are applied to estimate the performance of the network. Finally, Chapters 6 and 7 proposes possible research topics for future work and concludes the dissertation respectively.

# Chapter 2

# Related Work

## 2.1 3D Integration and 3D-NoC

In the past few years a large amount of research has been devoted to 3D IC design [9] [10]. The main driving force behind this effort is the higher density, better performance, more flexibility and higher throughput offered by 3D ICs as compared to the traditional ICs [11] [12] [17]. Vertical stacking provides multiple levels of of devices and multiple layers of on-chip metal interconnects. A promising method for 3D stacking is to first design and manufacture the 2D dies to be integrated vertically using the traditional planar methods and processes, and then further process them with additional steps such as substrate thinning and TSV filling [43]. Stacking process might be face-to-face or face-to-back. In the former, the via stubs from each metal layer of two candidate wafers are bonded directly. The Chemical-Mechanical Polishing (CMP) is applied to thin the back side of the the top die for exposing buried TSVs to connect to the I/O pads. In the latter, the back side silicon bulk of the top wafer is first thinned by CMP and then thermo-compression bonding is applied with the front side of the bottom wafer.

The modeling and performance evaluation of TSVs have been studied in different works [44][45]. Unfortunately, TSVs are expensive, impose large area overhead, and suffer from lower yield as compared to horizontal links. In order to take advantage of reduced interconnection latency offered by 3D ICs and to address the scalability and bandwidth bottleneck in NoC, many works [46] [13] [47] consider 3D-NoC with limited TSV as a realistic design option. Partially connected 3D-NoCs reach a compromise between the advantages and disadvantages of vertical interconnections. In other words, as the provision of TSVs introduces higher speed and shorter wiring as compared to 2D systems, a smaller number of TSVs mitigates the disadvantages such as degraded reliability and area issues of vertical interconnections.

## 2.2 TSV Reliability

An analytical model for reliability evaluation of 2D NoC has been reported in [48], but it does not consider unexpected sources of faults in 3D die-stacked designs. The impact of sub-micron TSVs on future 3D ICs is still unknown [49]. Chip warpage, TSV coupling [50], and thermal stress are known as main causes of TSV failure [33]. To alleviate mechanical reliability issues in 3D ICs, [51] presented an analysis tool as well as a design optimization framework. Reliability evaluation of a specific TSV technology developed by Austria Microsystems AG has been reported in [52]. An analytical reliability analysis for a fault-intolerant 3D NoC under transient TSV failures has been proposed in [53]. While [53] focuses on the evaluation of the fraction of time slots that are affected by transient TSV faults under the assumption of temporally and spatially uniform traffic, here we intend to investigate the effect of permanent faults on inter-level communications.

## 2.3 Routing Algorithms

3D-NoC routing algorithms have been widely studied in the literature. LA-XYZ [54], AFRA [55], $\mathrm{D_y}$XYZ [56] and MAR [57] are the routing algorithms for 3D mesh architectures. Fault-tolerant routing algorithms for 3D mesh NoCs have been presented in HamFa [58], 4NP-First [59], LAFT [60] and HLAFT [61]. However, there are few works that consider partial connectivity.

TDAR [62] proposes an adaptive routing algorithm for the cases in which the vertical bandwidth is less than the horizontal bandwidth. This routing algorithm works for 3D mesh NoCs with limited vertical bandwidth.

A fully adaptive routing algorithm with congestion consideration is presented in [56]. $\mathrm{D_y}$XYZ works on fully connected 3D meshes and it is proven to be deadlock free by using 4, 4, and 2 virtual channels along the $X$, $Y$ and $Z$ dimensions, respectively.

Limited bandwidth in the vertical dimension has been discussed in [63] which is a congestion-aware routing algorithm for the 3D mesh network. In this algorithm, when a router wants to determine the output port, it considers the congestion information of the neighboring nodes along with the distance from the current node to the destination node. So, different weights are assigned to the router outputs and then routing is done based on the congestion information and the assigned weights. This algorithm allows using a non-minimal and adaptive routing algorithm to distribute traffic load over the network.

4NP-First [59] introduces a fault-tolerant routing algorithm for 3D-NoC. In this routing algorithm, when the fault rate is above a threshold value, two redundant packets are transmitted to the destination: one using the 4N-First turn model and the other using 4P-First.

The 3D-FAR algorithm in [64] is another fully adaptive routing algorithm which uses two, two and four virtual channels along the $X$, $Y$ and $Z$ dimensions respectively. In this algorithm, the network is divided into four disjoint virtual subnetworks and packets can use any shortest paths between the source and destination nodes. Non-minimal routes are used in the case of faults.

Elevator-first [65] is a distributed routing algorithm for partially connected 3D-NoCs which requires two virtual channels along $X$ and $Y$ dimensions. Elevator-first is a deterministic routing algorithm with no limitation in choosing elevators to transfer the packets to the destination layer. In this algorithm, two virtual channels per physical link in $X$ and $Y$ dimensions are employed while there is no additional virtual channel in the $Z$ dimension.

A modification on the Elevator-first algorithm has been made in Redelf [66] which requires no virtual channels to ensure deadlock-freedom. In Redelf, certain rules are applied for choosing an elevator. To make distinguishable differences between Elevator-first and Redelf, it is necessary to mention that in the Elevator-first routing algorithm, there is no limitation on choosing an elevator when a packet traverses between layers. In other words, the packet is free to take any elevator in order to reach the destination layer. However, it is at the cost of using two virtual channels in both $X$ and $Y$ dimensions to ensure deadlock-freedom. Redelf on the other hand omits using virtual channels, but in order to guarantee deadlock-freedom, certain rules are applied which are limitative. Both of the routing algorithms are deterministic and are not able to distribute packets in congested networks.

## 2.4 Analytical Verification

Analytical latency models for NoC are formulated for specific topology and traffic patterns in [67] and [68]. Queuing theory is used in [69] to determine individual buffer depths for the given target application and available buffering space. However, the approach relies on many simplistic assumptions such as packet size distribution and deterministic routing. The allocation of link capacities in NoCs is addressed in [70] through an analytical latency model where network contention and queuing delays have been ignored.

Other works have tried to tailor the analysis to the characteristic of the wormhole-switching network. Authors in [71] propose an analytical latency and throughput analysis under Poisson packet arrival rates in low traffic scenarios. The formulation may not be accurate enough under realistic traffic arrival rates and also near the saturation point. In [72], an NoC latency model has been proposed for the priority-based router architecture which takes into account the random processes that accommodate bursty traffic. However, the framework is only applicable to deterministic routing algorithms and is valid under the limiting assumption that packet inter-arrival processes over different channels are identical. More complex and realistic models have been proposed in [73] and [74] for deterministic routing algorithms, where comprehensive information can be extracted from latency distribution, rather than average latency.

# Chapter 3

# ETW Routing Algorithm

As explained in Chapter 2, a light weight adaptive routing algorithm for partially connected 3D-NoCs lacks the literature. These issues motivated us to develop an efficient routing algorithm for partially connected 3D-NoCs, called East-Then-West (ETW) [75]. This algorithm is reliable as long as there is at least one TSV at the eastmost column while the performance can be improved by increasing the number of TSVs. The ETW algorithm is extremely light-weight. That is, it only requires one virtual channel along the Y dimension. This algorithm provides adaptivity to deliver packets, preferably using the shortest paths. Besides, this adaptivity can be applied to avoid congestion in the network. Whenever there are more than one valid output channels available to deliver a packet, the utilization of the input buffer of the neighboring routers is used to prioritize one output channel over the other.

In comparison with the Elevator-first algorithm, ETW uses one less virtual channel. Elevator-first is a deterministic algorithm offering adaptivity with no significant limitation as is the case with dimension order routing. In ETW, a group of eligible TSVs is selected to support the communication between a source and destination. Then, a single TSV is selected from these eligible TSVs. The ETW fault tolerance is based on runtime elevator selection for every single node upon packet arrival. On the contrary, the Elevator-first algorithm assigns a fixed elevator to a packet which results in packet being blocked if the elevator is faulty. In the Elevator-first algorithm, a new header is added to the packet containing the address of the elevator leading to both hardware and timing overhead. There is no such a header update in ETW. Besides, the main difference between the ETW routing algorithm and the Elevator-first routing algorithm is that adaptivity in the former enables different paths for the same source and destination pair depending on the network condition.

## 3.1 Overview of ETW

ETW is proposed for vertically partially connected 3D-NoCs. In ETW, every router is statically informed about the location of the vertical links. This information is stored locally at router registers. The vertical links are considered to be pillars. That is, the TSV in the first layer connects to all the other layers. In vertically partially connected 3D-NoCs, in order to deliver a packet to the destination, the packet needs to be delivered to the destination layer through a vertical link (elevator), and then routed toward the destination.

In the ETW algorithm, two virtual channels along the $Y$ dimension is needed while there is no need to have any further virtual channel along the $X$ and $Z$ dimensions. To prove freedom from deadlock, the network can be virtually partitioned into two disjoint subnetworks including different channels: Subnetwork1 $(X^+, Y0^*, Z^+)$ and Subnetwork2 $(X^-, Y1^*, Z^-)$ where +, - represent channels along the positive and negative directions respectively, while * stands for both positive and negative directions (bidirectional channels) as shown in Figure 3.1.

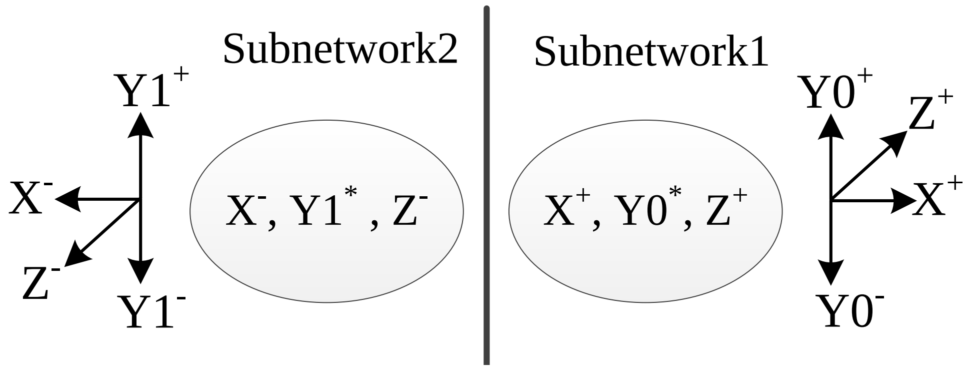


Figure 3.1: Two different regions

Packets in Subnetwork1 have the flexibility to move along the following directions in any order: (1) Eastward ($X^+$), (2) Northward using the virtual channel number zero ($Y0^+$), (3) Southward using the virtual channel number zero ($Y0^-$), or (4) upward ($Z^+$). Similarly, valid movements in Subnetwork2 are as follows: Westward ($X^-$), moving Northward or Southward using the virtual channel number one ($Y1^*$), or moving downward ($Z^-$). Packets in each subnetwork can switch between the directions dynamically and do not necessarily follow the dimension order routing.

The basic idea of this routing algorithm is that packets are allowed to use any channels either in Subnetwork1 or Subnetwork2 or move from Subnetwork1 to Subnetwork2 and then use any channels of Subnetwork2 (no transfer from Subnetwork2 to Subnetwork1 is allowed). Thereby, if any Eastward movement is needed, a channel of Subnetwork1 should be used before using any channels of Subnetwork2. At the worst case, packets should reach the East-most column with the flexibility to take $Y0^*$, deliver to the desired layer and then to the destination node. In other words, having at least one TSV in the East-most column guarantees delivery of packets between each pair of source and destination nodes.

| Subnetworks | Pair$(X^+, X^-)$ | Pair$(Y^+, Y^-)$ | Pair$(Z^+, Z^-)$ | CompletePair |
|---|---|---|---|---|
| $(X^+)(Y0^*)(Z^+)$ | $X^-$ is missing | Pair exists | $Z^-$ is missing | Y |
| $(X^-)(Y1^*)(Z^-)$ | $X^+$ is missing | Pair exists | $Z^+$ is missing | Y |

Table 3.1: Completed pairs within each subnetwork

## 3.2 Proof of Deadlock-freedom

A sufficient condition for a routing algorithm to be deadlock-free is the exclusion of cycles [64]. A cycle occurs if both positive and negative directions along at least two dimensions can be adopted by a packet. As an example, to form a cycle in the $XY$ plane, it is necessary to take the $X^+, X^-, Y^+$ and $Y^-$ directions. The same trend is true for $XZ$ and $YZ$ as well. No U-turn (360-degree turn) is allowed in the algorithm. As can be obtained from the subnetwork definition in Table 3.1, only the $Y$ dimension is completed (i.e. both positive and negative directions of $Y$ can be taken by packets) in each subnetwork, and thus there is no possibility of forming a cycle. In order to prove the deadlock-freedom between subnetworks, it suffices to show that two subnetworks are disjoint. A pairwise comparison in Table 3.2 between the two subnetworks reveals that these two subnetworks are different in $X$ and $Z$ direction and the virtual channel number along $Y$. That is, Subnetwork1 only covers positive direction of $X$ and $Z$ while Subnetwork2 covers the negative parts. The two subnetworks are disjoint in virtual channel number along $Y$. Packets are allowed to use any channels either in Subnetwork1 or Subnetwork2 or move from Subnetwork1 to Subnetwork2 and then use any channels of Subnetwork2. Since no transfer from Subnetwork2 to Subnetwork1 is allowed, a cycle can never be formed. Therefore, moving toward $X^+$ and $Z^+$ will not be made after moving toward $X^-$ and $Z^-$ and the freedom from deadlock is proved.

|  |  |
|---|---|
| Subnetwork1 | $(X^+)(Y0^*)(Z^+)$ |
| Subnetwork2 | $(X^-)(Y1^*)(Z^-)$ |
| $X$ Dimension | Different in direction |
| $Y$ Dimension | Different in VC number |
| $Z$ Dimension | Different in direction |

Table 3.2: Disjoint subnetworks

## 3.3 Routing Algorithm Procedure

If the destination is to the East of the source, Subnetwork1 will be used to deliver the packet to the destination; otherwise, Subnetwork2 will be applied.

### 3.3.1 Destination is on the upper layer

Subnetwork1 should be used first since moving upward is allowed only in Subnetwork1. When the packet reaches the destination layer, depending on the position of the destination router, the packet either continues routing in Subnetwork1 (destination is to the East of the current node) or switches to Subnetwork2 (destination is to the West of the current node). In more details, the destination region can be in East-Up or West-Up of the source. When the destination is in East-Up of the source, only Subnetwork1 will be used to deliver the packet to the destination. In the other case (i.e. the destination is on West-Up of the source), the channels of Subnetwork2 will be used when the packet reaches the destination layer. In order to illustrate the two scenarios, a $4 \times 3 \times 2$ network is shown in Figure 3.2 having four TSVs connecting the nodes 0 to 12 (0-12), 8-20, 10-22, and 7-19. The TSVs are bidirectional. Based on the introduced algorithm, if the source node 17 targets the node 1 or node 4 as the destination, two elevators (i.e. 10-22 and 7-19, bolded in the figure) can be taken to transmit the packet to the destination layer and finally Subnetwork2 is used for delivering the packet to the destination node. Moreover, when the source node 17

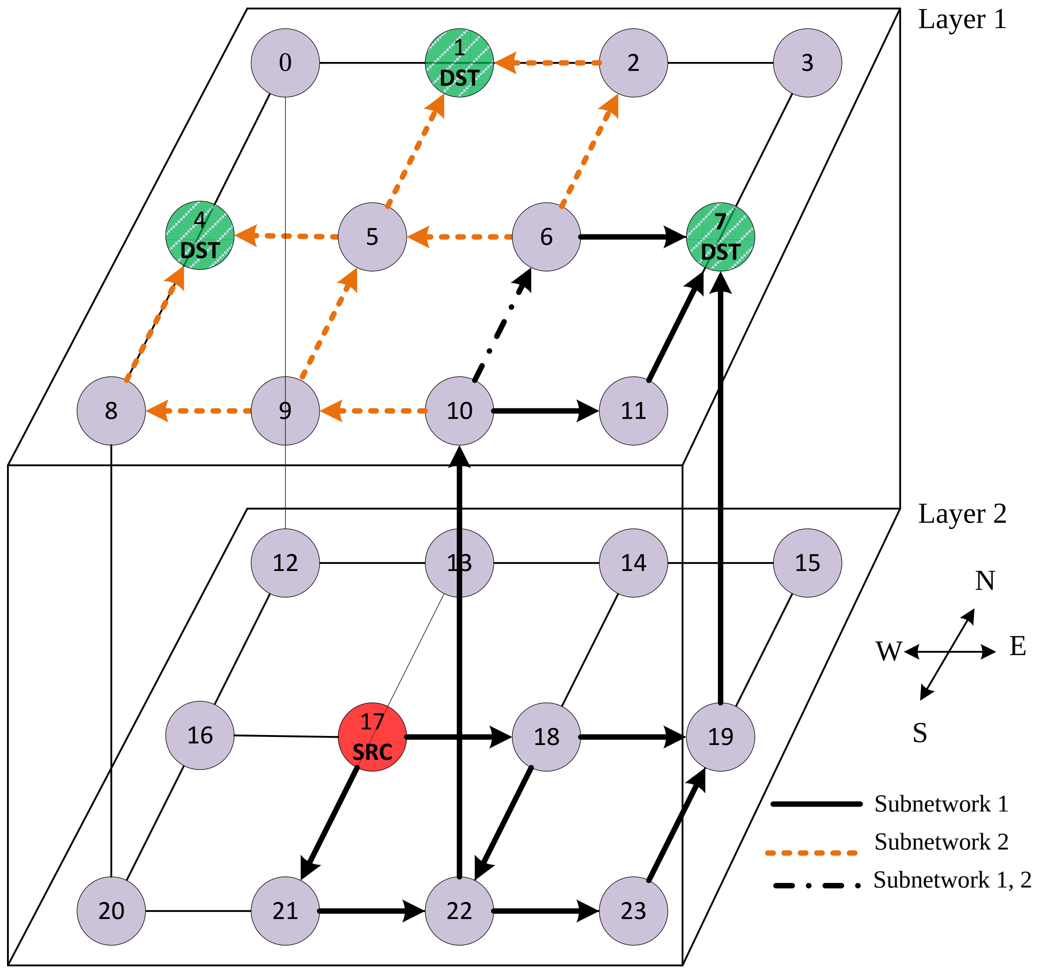


Figure 3.2: An example of destination in the upper layer

wants to send a packet to the destination node 7, again both bolded elevators are eligible and Subnetwork1 will be sufficient to deliver the packet to the destination.

### 3.3.2 Destination is on the lower layer

Packets should be delivered to the destination layer through a TSV which is located to the east side of the destination. The reason is that once the downward channel is used ($Z^-$ from Subnetwork2), no further movement to the East direction is possible. So, the packet has to move toward East sufficiently before moving downward. The destination can be in East-Down or West-Down of the source. In both cases, the packet is first forwarded to an elevator in the east side of the destination (using Subnetwork1). Then Subnetwork2 will be utilized to deliver the packet to the destination layer and finally to the destination node. Let us consider two examples shown in Figure 3.3. First, the source node 6 sends a packet to the destination 19. In this case, the elevator 10-22 should not be used as the packet has to take the East direction after delivering to the destination layer and it is not possible when the packet is in Subnetwork2. The elevator 7-19 is the only eligible elevator in this example. Second, for sending a packet from the source node 6 to the destination 17, the elevator 10-22 is between the source and destination nodes, and thus it can be used. The elevator 7-19 is also valid and can be used. It is the same condition as the case when the source node 1 wants to send a packet to the destination 17. Since there is no elevator between the source and the destination, the elevators on the East side of the destination are eligible which are the elevator 10-22 and elevator 7-19.

## 3.4 Algorithms for Elevator Assignment

Since ETW suggests different routing options, one of the most important steps in ETW is choosing an elevator among eligible options. The way in which elevators are assigned to each pair of source and destination has a considerable impact on the performance of the routing algorithm. In this section, we introduce two algorithms which can be used on top of the proposed mechanism for the selection of an elevator among the eligible options.

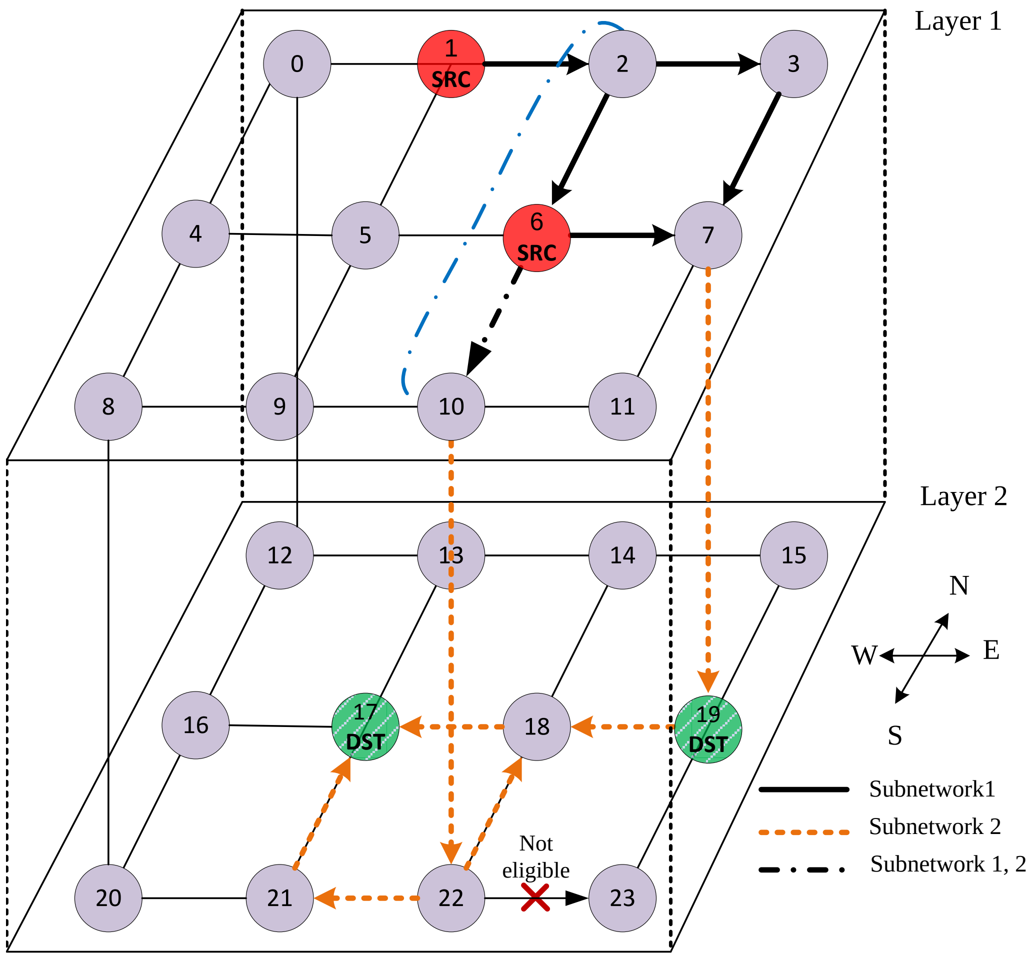


Figure 3.3: An example of destination in the lower layer

### 3.4.1 Static Elevator Assignment (SEA)

The basic idea of this method is that elevators are assigned to the nodes statically according to the region of the destination. The important consideration in assigning elevators to nodes is the destination region, regardless of how far the destination is from the source node. In this method, each router stores the location of three elevators which will be used for the destinations located Up, East-Down and West-Down of their source according to Table 3.3. In other words, each router registers the location of the nearest eastern and western elevators relative to its location as well as the east-most elevator in the network. The first one *(elevator_east)* will be used for all the destinations located on upper layer for that specific router without considering how far they are from their source node. Moreover, this elevator will be used for the destinations located on west down side of that router, if there is no elevator between the source router and the destination. The second elevator *(elevator_west)* will be used for all destinations in west down of the source. That is, when a router wants to send packets to a destination located on its west down region, first it tries to find an elevator between the source and destination that is the nearest elevator in west side of the node. If not, then the nearest elevator in the east side of the node will be used. Finally, the third elevator *(elevator_east_down)* is for all the destinations on east down of the current router. Since this elevator assignment is done offline, for all the destinations on east down of the source the east most elevator will be used because no further east transmission is allowed after taking down direction. Therefore, it is necessary to move toward east as much as possible.

The pseudo code for assigning elevators to the nodes is shown in Algorithm 1. In order to assign an elevator to a node, three conditions must be satisfied. The first condition forces the $X$ coordinate of the candidate elevators to be either greater or less than the $X$ coordinate of the current node, depending on the destination location.

Among these candidate elevators, the second condition chooses the elevators with the least Manhattan distance from the elevator to the current node. Lines 10 and 13 of Algorithm 1 summarize the first two conditions. The third condition further narrows down the selection by choosing the elevators with minimum $X$ distance (lines 11 and 14). At this point, there could be at most two candidate elevators on the same column, one of which is selected arbitrarily as the target elevator (lines 12 and 15) and the corresponding elevator ID is stored before runtime in local registers for all routers. Then, the routing will be done according to Algorithm 2. As an example in Figure 3.2, *elevator_east* registered in the node 16 is 12. Therefore, the node 16 will use the elevator at node 12 for all the destinations on east side (i.e. the nodes 0, 1, 2, 3, 4, 5, ... , 11).

Another example is that, according to Figure 3.2, the source node 1 generates a packet destined for the node 18. Since the destination is in the east down side of the source, the elevator at node 7 will be used while there is a closer elevator at node 10. This suboptimal selection is an inherent requirement of the offline/static elevator assignment that should work for any east-down destination. In other words, whenever the destination is on the east-down of the source, the statically assigned elevator should be such that the routing constraints hold regardless of the exact location of source and destination. The only solution that would work for all such situations is the east-most elevator named *elevator_east_down.* Since no further east movement is allowed after taking the down direction, the packet has to be forwarded toward east sufficiently in the source layer. Therefore, for all the destinations located at east down of their source, the packet will use the elevator located at the east most column. The main drawback of this technique is that if the elevator is faulty, no further rerouting will be possible. The following technique mitigates this problem by enabling elevator assignment during runtime.

**Algorithm 1** SEA Pseudo code

```
1: $E = \{e_i\}$ (set of elevator indices)
2: $X_c$, $Y_c$, $Z_c \leftarrow X$, $Y$, $Z$ coordinates of current router
3: $X_e$, $Y_e$, $Z_e \leftarrow X$, $Y$, $Z$ coordinates of elevator $e$
4: $MD(e,c) = (|X_e - X_c| + |Y_e - Y_c|)$
5:
6: if (current node is an elevator) then
7:    address_elev_east ← current node
8:    address_elev_west ← current node
9: else
10:   S1 ← $\{e \in E : (X_e \geq X_c)$ and $MD(e,c) = \min_i(MD(e_i,c))\}$
11:   S1 ← $\arg\min_{e \in S1} X_e$
12:   address_elev_east ← $S1(1)$
13:   S2 ← $\{e \in E : (X_e \leq X_c)$ and $MD(e,c) = \min_i(MD(e_i,c))\}$
14:   S2 ← $\arg\max_{e \in S1} X_e$
15:   address_elev_west ← $S2(1)$
16: end if
17: address_elev_east_down ← $\{e \in E : X_e = \max_i(X_{e_i})$ and $MD(e,c) = \min_i(MD(e_i,c))\}$
```

**Algorithm 2** ETW-SEA routing algorithm

```
1: $X_c$, $Y_c$, $Z_c \leftarrow X$, $Y$, $Z$ coordinates of current router
2: $X_d$, $Y_d$, $Z_d \leftarrow X$, $Y$, $Z$ coordinates of destination router
3: ETW($e$) (function for routing through elevator $e$)
4:
5: if ($Z_d > Z_c$) then
6:    ETW(address_elev_east)
7: else
8:    if ($X_d < X_c$) then
9:       if ($X_d \leq X_{\text{address\_elev\_west}} \leq X_c$) then
10:         ETW(address_elev_west)
11:      else
12:         ETW(address_elev_east)
13:      end if
14:   else if ($X_d > X_c$) then
15:      ETW(address_elev_east_down)
16:   else
17:      ETW(address_elev_east)
18:   end if
19: end if
```

|  | Destination Region |  |  |  |
|---|---|---|---|---|
| Used Elevator | East Up | West Up | East Down | West Down |
|  | Elevator-right | Elevator-right | Elevator-right-down | Elevator-right or elevator-left |

Table 3.3: Elevator assignment in ETW-SEA

### 3.4.2 Dynamic Elevator Assignment (DEA)

In this method, elevators are assigned to routers at runtime. This technique is proposed in order to enhance the fault tolerance of the routing algorithm. When the packet reaches a faulty elevator, the current node has the capability to choose a new elevator and reroute the packet toward a new elevator. It is considered that a faulty elevator is considered as a node which has no vertical link. Therefore, by changing a node status, Algorithm 3 is called to assign a new elevator for the current node. The algorithm assigns an elevator in the valid region according to the location of destination. Figure 3.4 shows valid regions for different locations of destination as compared to the current node.

Figure 3.4 (a) shows that if the current node receives an eastbound packet from its western input port (packet has already been forwarded toward East) or the case when the current node is the source, no matter where the destination is (upper or lower layer compared to the current node), the valid region for selecting an elevator is at east side of the current node. Figure 3.4 (b) represents the valid region when the input direction to the current node is North (packet has already been forwarded toward South). According to the figure, the red part of the figure will not be considered because the input direction of the node is North and forwarding the packet toward North might make a loop. Figure 3.4 (c) illustrates the case in which the input port is South. Again, the red part is forbidden in order to avoid deadlock.

At the next step, the algorithm attempts to find a unique elevator in the valid region according to Algorithm 3. In the algorithm, $T$ is the set of eligible TSVs in the valid region. $MD(i, j)$ is the Manhattan Distance between the two nodes $i$ and $j$. $OMD(s, d, t)$ is the Overall Manhattan Distance from the source node $s$ to the destination $d$ using the TSV at node $t$. $PMD(s, t)$ is the Partial Manhattan Distance from the source node $s$ to the TSV. The algorithm calculates, for all the elevators located in the valid region, the overall Manhattan distance, which is the Manhattan distance from the source to TSV plus Manhattan distance from the TSV to the destination. If this calculation does not result in identifying a unique elevator, Manhattan distance from the source to the TSV will be considered. If a unique elevator cannot be found by these two steps, in the third step, the elevator leading to the shortest $X$ separation ($|X_{elev} - X_{source}|$) will be chosen in order to avoid transferring toward east prematurely. Finally, if the above criteria do not lead to selecting a unique TSV, the source node selects the TSV in the lower (upper) Y-half-plane, if the node is in the upper (lower) Y-half plane. If the size of Y dimension of the network is $N_y$, a node is said to be in the upper (lower) Y-half plane, if its $y$ coordinate is less than or equal to (greater than) $\lfloor N_y/2 \rfloor$. By selecting the Y-half-plane that is not the same as Y-half-plane of the source, a larger number of TSVs can be used to carry out the communication, leading to higher resilience to the TSV failure.

An example is illustrated in Figure 3.5. Let us consider a case where the source node 4 targets the node 23 as its destination. For this example, all the elevators (i.e. 1, 2, 3, 9, 10, 11, and 14) are in the valid region. Step 1 determines that six elevators (i.e. 1, 2, 3, 9, 10, and 11) have the same overall Manhattan distance from the source to destination. Step 2 limits the list to two elevator (i.e. 1 and 9) since they have the least Manhattan distance from the source. Step 3 has no effect on the list since both elevators have the same X separation from the source. Finally, Step 4 chooses node 9 as its elevator because it is not in the same Y-half-plane with the source (source is

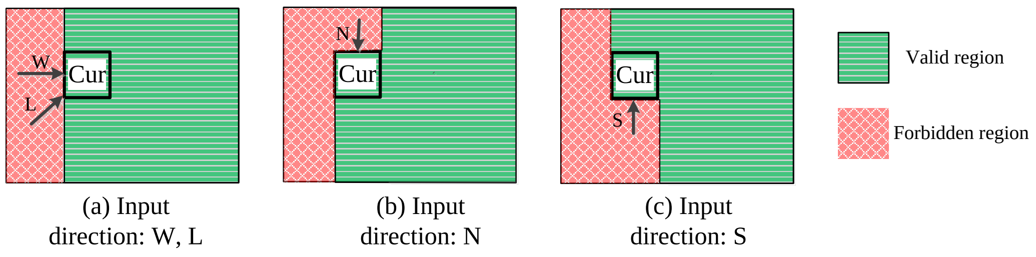


Figure 3.4: Valid regions based on the input port to current node

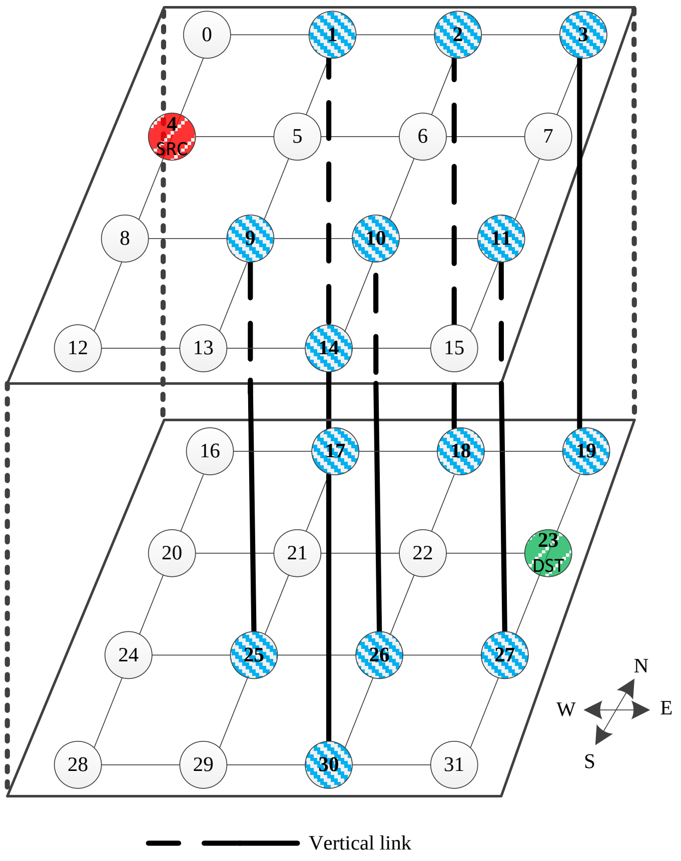


Figure 3.5: An example of the DEA algorithm

in the upper Y-half-plane).

As it was discussed earlier, DEA can handle faulty elevators by selecting new ones during runtime and thus enhances the fault tolerance of the 3D NoC.

**Algorithm 3** DEA Pseudo code

```
1: Input: $T = \{t_1, \cdots, t_n\}, s, d$
2: Output: $t \in T$
3: Auxiliary functions:
4: $MD(i, j) = (|X_i - X_j| + |Y_i - Y_j|)$
5: $OMD(s, d, t) = MD(s, t) + MD(t, d)$
6: $PMD(s, t) = MD(s, t)$
7:
8: if $(|T| = 1)$ then
9:    return $t_1$
10: else
11:    $T \leftarrow \{t \in T :$
12:    $OMD(s, d, t) = \min_{i=1 \cdots n}(OMD(s, d, t_i))\}$
13:    $n \leftarrow |T|$
14:    if $(|T| = 1)$ then
15:       return $t_1$
16:    else
17:       $T \leftarrow \{t \in T :$
18:       $PMD(s, t) = \min_{i=1 \cdots n}(PMD(s, t_i))\}$
19:       $n \leftarrow |T|$
20:       if $(|T| = 1)$ then
21:          return $t_1$
22:       else
23:          $T \leftarrow \{t \in T :$
24:          $|X_t - X_s| = \min_{i=1 \cdots n}(|X_{t_i} - X_s|)\}$
25:          $n \leftarrow |T|$
26:          if $(|T| = 1)$ then
27:             return $t_1$
28:          else
29:             if $Y_s < \lfloor N_y/2 \rfloor$ then
30:                return $t \in T : Y_t \geq \lfloor N_y/2 \rfloor$
31:             else
32:                return $t \in T : Y_t < \lfloor N_y/2 \rfloor$
33:             end if
34:          end if
35:       end if
36:    end if
37: end if
```

## 3.5 Formal Model of the Network

Consider an $N_x \times N_y \times N_z$ network. The nodes of the network are linearly indexed from 1 to $N_x \times N_y \times N_z$. Similarly, TSVs are linearly indexed by a set of numbers $\mathbf{T} = \{t_1, \cdots, t_{|\mathbf{T}|}\} \subseteq \{1, \cdots, N_xN_y\}$. If $i \in T$, there is a TSV whose base is at node $i$. As an example, consider the planar view of a network with two layers as shown in Figure 3.6. The location of TSVs are highlighted in black. Each cell contains two numbers, corresponding to the node index in each layer. In this example, the nodes are indexed from 1 to $8 \times 8 \times 2 = 128$. Also, the total number of TSVs, $|\mathbf{T}|$, is equal to 10 and we have $\mathbf{T} = \{6, 8, 10, 13, 22, 41, 54, 55, 60, 64\} = \{t_1, \cdots, t_{10}\}$.

Furthermore, denote each source-destination pair by $(s, d)$, where $s$ is the source ID and $d$ is the destination ID. As we are concerned with vertical transmissions, we focus on pairs in which the source and destination are in two different layers. In addition, it suffices to focus on 2 layer networks because TSVs are pillars and cross through all layers. Then, the total number of source destination pairs, with the source and destination located at two different layers, is equal to $2(N_xN_y)^2$. For notational convenience in what follows, we map each $(s, d)$ to a scalar $k$ as follows:

$$\begin{aligned}
&(1, N_xN_y + 1) \rightarrow 1 \\
&(1, N_xN_y + 2) \rightarrow 2 \\
&\qquad\qquad\vdots \\
&(2, N_xN_y + 1) \rightarrow N_xN_y + 1 \\
&\qquad\qquad\vdots
\end{aligned} \tag{3.1}$$

or $k = N_xN_y(s - 1) + ((d - 1) \bmod N_xN_y) + 1$. With this notation, the pair $(s, d)$ can be symbolically represented by $\alpha_k$.

To model the dynamic routing, suppose that network topology $\mathcal{N}$ and routing algorithm $\mathcal{A}$ are known. The communication corresponding to $\alpha_k$ attempts to use some TSV $t_{k,1} \in \mathbf{T}$. If TSV $t_{k,1}$ fails, the algorithm attempts to use TSV $t_{k,2}$ and so on. Denote the set of prioritized TSVs for $\alpha_k$ by $\mathbf{t}_k = (t_{k,1}, \cdots, t_{k,n_k}) \subseteq \mathbf{T}$. As a sidenote, we require that the routing algorithm is such that each $\alpha_k$ corresponds to exactly one $\mathbf{t}_k$. Also note that if $\mathbf{t}_k$ has only one member for all $k$, this model reduces to a static routing algorithm. Symbolically, we can define a function $F_{\mathcal{N},\mathcal{A}}$ that maps $\alpha_k$ to $\mathbf{t}_k$:

$$\mathbf{t}_k = F_{\mathcal{N},\mathcal{A}}(\alpha_k) \tag{3.2}$$

As an example, suppose that the source 1 and the destination 87 (circled in Figure 3.6) are denoted by $\alpha_{k=23}$. Also suppose that the routing algorithm has the possibility to use TSVs 10, 13, and 22 to carry out the transmission from the source 1 to the destination 87. Thereby, we have $\mathbf{t}_{23} = \{10, 13, 22\} = \{t_{23,1}, t_{23,2}, t_{23,3}\}$.

It is possible to illustrate the mapping $\alpha_k \to \mathbf{t}_k$ by constructing a $|\mathbf{T}| \times 2(N_x N_y)^2$ (row$\times$column) binary matrix $C$, where a 1 in $C(i,j)$ represents the fact that TSV $t_i$ can be used to support the communication of pair $\alpha_j$, or $t_i \in \mathbf{t}_j$. As an example, the 23rd column of $C$, denoted by $C(:,23)$, of the aforementioned example looks like this:

$$C(:,23) = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \tag{3.3}$$

Figure 3.7 shows the entire $C$ matrix for the network of Figure 3.6. The number of columns is equal to the number of source-destination pairs and the number of rows is equal to the number of TSVs (10 in this example). The $x$ axis is labeled by $k$

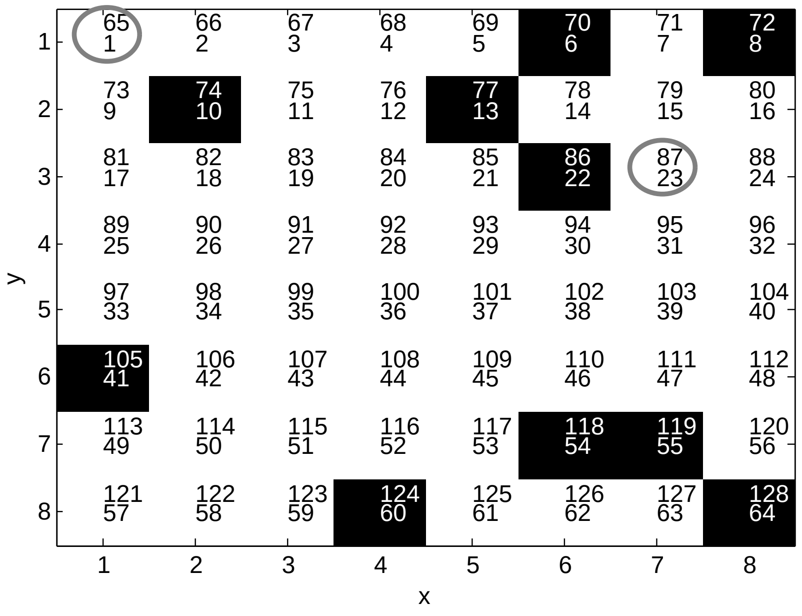


Figure 3.6: An example of an $8 \times 8 \times 2$ network with 10 TSVs

and the $y$ axis is labeled with the ID of TSVs $t_1, \cdots, t_{10}$. The black (white) bars correspond to 1 (0) in the $C$ matrix. By looking at a row of this figure, it is possible to estimate the fraction of source-destination pairs that use the TSV corresponding to that row. For example, it is observed that the 2nd and 10th row of $C$ (shown by arrow on Figure 3.7) are the darkest rows, signifying the fact that a large number of source-destination pairs use the corresponding TSVs (8 and 64). In general, the tendency of our routing algorithm to send the packets to the east leads to more usage of TSVs on the right hand side of Figure 3.6.

## 3.6 Reliability Analysis

Traditionally, the reliability of an 'object' at time $t$, $R(t)$, is defined as the probability of observing a 'fault' in the 'object' after time $t$. We take the 'object' to be a source-destination pair $\alpha_k$, and the 'fault' corresponds to an unsuccessful packet delivery from the source to the destination through TSVs. Considering the mechanism of

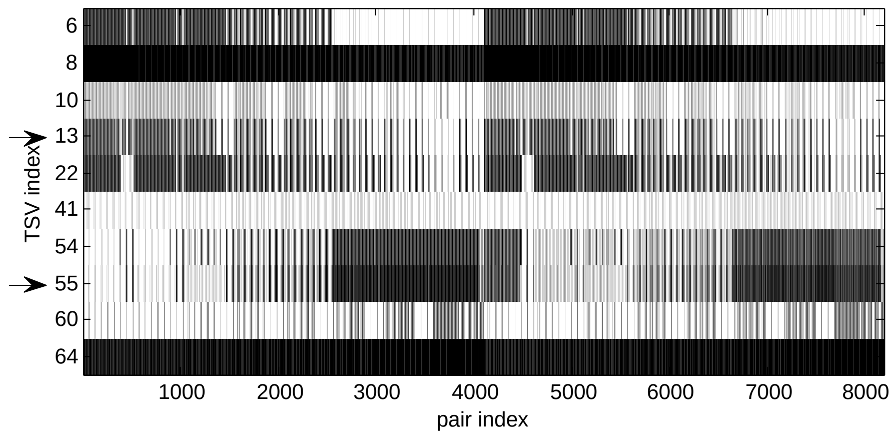


Figure 3.7: $C$ matrix representation of the $8 \times 8 \times 2$ network

routing, a communication fails if none of TSVs in $\mathbf{t}_k$ are healthy. This is equivalent to the concept of 'parallel systems' in reliability theory. Following the same course of deduction, we can write the reliability of $\alpha_k$ as:

$$R_{\alpha_k}(t) = 1 - (1 - R_{\text{TSV}}(t))^{n_k} \tag{3.4}$$

where it is assumed that all TSVs follow the same reliability model $R_{\text{TSV}}(t)$ and fail independently. The reliability can be readily derived from a life distribution. Fortunately, many different life distributions have been examined in the literature and shown to either empirically fit the failure behavior of electrical components or comply with the underlying processes that generate the failure. Examples of such life distributions include exponential, Weibull, Bayesian Weibull, normal, lognormal, mixed Weibull, Gamma and generalized Gamma, logistic, loglogistic, and Gumbel among others.

In practice, the reliability of the communication of a specific $\alpha_k$ is not of interest. Rather, it is of interest to evaluate a measure of 'overall' reliability of the system. Such a measure is defined with the specific concept of reliability in mind. In this thesis, we focus on the average fraction of source destination pairs that communicate successfully at time $t$, henceforth denoted by $f(t)$.

To calculate $f(t)$, we first focus on the fraction of $\alpha_k$ that does not fail if certain TSVs are faulty at time $t$. Let us denote the failure status of TSVs at time $t$ by a binary vector $\ell(t)$ of size $|\mathbf{T}|$. A zero in location $i$ of $\ell(t)$ means that TSV $t_i$ is faulty at time $t$. As an instance, $\ell(t) = [1\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0]$ for the example of Section 3.5 means that TSVs 22, 41, 54, 55, 60, and 64 are faulty at time $t$ .To see what fraction of source-destination pairs are still connected at time $t$, the rows of $C$ indexed by zeros of $\ell(t)$ are set to zero, resulting in a new matrix $C_{\ell(t)}$. The matrix $C_{\ell(t)}$ is of the same size as $C$, but with certain rows of $C$ set to 0. For example, $\ell(t) = [1\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0]$ leads to a $C_{\ell(t)}$ matrix identical to Figure 3.7 except for the 5th through 10th row set to 0. Then, the number of non-zero columns in $C_{\ell(t)}$ represents the number of pairs that can communicate at time $t$. This is because a zero column in $C_{\ell(t)}$ means that no TSV exists to support the communication between the source-destination pair corresponding to that column. Dividing this number by $2(N_xN_y)^2$ returns the fraction of connected pairs given $\ell(t)$. Denote this fraction by $f_{\ell(t)}$. For example, if the 5th through 10th row of Figure 3.7 are set to zero, only 7808 out of 8192 pairs can communicate, which is a fraction of about 95% ($f_{[1111000000]} = .95$).

In order to relate $f_{\ell(t)}$ to $f(t)$, we note that each $\ell$ occurs with the time dependent probability:

$$p_\ell(t) = (1 - R_{\text{TSV}}(t))^{|\mathbf{T}|-\text{sum}(\ell)} R_{\text{TSV}}^{\text{sum}(\ell)}(t) \tag{3.5}$$

where $\text{sum}(\ell(t))$ returns the sum of elements of $\ell(t)$ (i.e. the number of healthy TSVs). Denote the set of all possible binary vectors of length $|\mathbf{T}|$ by $\mathbf{B}_{|\mathbf{T}|}$ (for example, $\mathbf{B}_3 = \{[000], [001], [010], [011], [100], [101], [110], [111]\}$). In other words, $\mathbf{B}_{|\mathbf{T}|}$ represents all different combinations of faulty TSVs. Then, $f(t)$ is a weighted sum of $f_{\ell(t)}$, where the weights are probabilities of individual $\ell$s:

$$\begin{aligned} f(t) &= \sum_{\ell \in \mathbf{B}_{|\mathbf{T}|}} p_\ell(t) f_\ell \\ &= \sum_{\ell \in \mathbf{B}_{|\mathbf{T}|}} (1 - R_{\text{TSV}}(t))^{|\mathbf{T}|-\text{sum}(\ell)} R_{\text{TSV}}^{\text{sum}(\ell)}(t) f_\ell \end{aligned} \tag{3.6}$$

To further simplify this identity, note that $p_\ell(t)$ is the same for all binary vectors $\ell$ with equal number of ones (same sum$(\ell)$). The total number of $\ell$s with $n$ 1s is equal to $\binom{|\mathbf{T}|}{n}$. Denote the set of $\ell$s with $n$ 1s by $\mathbf{L}_n = \{\ell_{n,1}, \cdots, \ell_{n,\binom{|\mathbf{T}|}{n}}\}$. For example, with total number of TSVs ($|\mathbf{T}|$) equal to 3, we have $\mathbf{L}_1 = \{[001], [010], [100]\} = \{\ell_{1,1}, \ell_{1,2}, \ell_{1,3}\}$. Then, the previous equation can be rewritten as a summation over the number of healthy TSVs ($n = sum(\ell)$):

$$\begin{aligned} f(t) =& \sum_{n=0}^{|\mathbf{T}|} (1 - R_{\text{TSV}}(t))^{|\mathbf{T}|-n} R_{\text{TSV}}^{n}(t) \sum_{m=1}^{\binom{|\mathbf{T}|}{n}} f_{\ell_{n,m}} \\ =& \sum_{n=0}^{|\mathbf{T}|} (1 - R_{\text{TSV}}(t))^{|\mathbf{T}|-n} R_{\text{TSV}}^{n}(t) \binom{|\mathbf{T}|}{n} \frac{\sum_{m=1}^{\binom{|\mathbf{T}|}{n}} f_{\ell_{n,m}}}{\binom{|\mathbf{T}|}{n}} \\ =& \sum_{n=0}^{|\mathbf{T}|} (1 - R_{\text{TSV}}(t))^{|\mathbf{T}|-n} R_{\text{TSV}}^{n}(t) \binom{|\mathbf{T}|}{n} \bar{f}_{\ell_{n,*}} \end{aligned} \tag{3.7}$$

where $\bar{f}_{\ell_{n,*}}$ is the average fraction of connected pairs when $n$ TSVs are healthy. Given $C$, $\bar{f}_{\ell_{n,*}}$ can be calculated by setting to zero different combinations of $|\mathbf{T}| - n$ rows of $C$ and counting the number of non-zero columns of $C$.

The following example illustrates the concept. Suppose that the total number of TSVs is equal to 3. For notational convenience, temporarily replace $R_{\text{TSV}}(t)$ with $R$.

Then we have:

$$\begin{aligned} f(t) = &(1-R)^{3-0}R^0 f_{[000]}+ \\ &(1-R)^{3-1}R^1(f_{[001]}+f_{[010]}+f_{[100]})+ \\ &(1-R)^{3-2}R^2(f_{[011]}+f_{[110]}+f_{[101]})+ \\ &(1-R)^{3-3}R^3 f_{[111]} \end{aligned} \tag{3.8}$$

where the terms with same sum ($\ell$) have been factored together.

## 3.7 Results and Discussion

In order to perform a complete set of tests including different traffic scenarios, the AccessNoxim simulator is used [76]. AccessNoxim is an integration of Noxim (i.e. a cycle-accurate SystemC NoC simulator) [77] and HotSpot (i.e. providing the architecture-level thermal model) [78]. This co-simulator combines the network model, power model and thermal model of the 3D NoC.

The experiments are carried out for a $4 \times 4 \times 4$ 3D-NoC. All the routers have 5-flit FIFOs and the packet size is 8 flits. Out of the 11000 cycles, the first 1000 cycles were excluded to allow the transient faults to fade away.

### 3.7.1 Traffic Scenarios

In order to make a meaningful comparison, the Elevator-first routing algorithm was also implemented as a baseline along with the ETW routing algorithm. Therefore, the Elevator-First routing algorithm and ETW algorithm with static and dynamic elevator assignments are compared versus each other in terms of latency and reliability by using both synthetic and real traffic scenarios. In uniform traffic, each node has the same probability to be chosen as a destination for the other node. In transpose traffic, a node $(i, j)$ only sends packets to a node $(N-1-j, N-1-i)$, where N is the total number of nodes in the network. Assuming a $4 \times 4 \times 4$ network, in shuffle traffic the first half of nodes (0 to 31) target destinations which have IDs that are twice of the source node IDs. As an example, node with ID equals 10 has node 20 as its destination. Besides, sources in the second half (32 to 63) target destinations whose IDs equal twice of the source node minus 63. In a hotspot traffic scenario, certain nodes receive hotspot traffic in addition to the regular uniform traffic. Given a hotspot percentage $h$, a newly generated packet is directed to each hotspot node with an additional $h$ percent probability. Finally two real traffic scenarios named barnes [79] and streamcluster [80] are considered.

### 3.7.2 Latency Analysis

In order to evaluate the efficiency of the proposed routing algorithms, two architectures have been tested. In the first architecture, five elevators are used located at nodes 0, 2, 7, 8 and 10. Second, an architecture with eight elevators located at nodes 0, 2, 5, 7, 8, 10, 13 and 15 is tested.

The following line graphs compare the efficiency of the Elevator-first routing algorithm and the ETW algorithm under SEA and DEA. It is necessary to mention that to make a fair comparison, elevator assignment in Elevator-first is based on the nearest elevator assignment which provides the least number of hop counts from the source to the elevator and from the elevator to the destination. In the line graphs, the horizontal line represents the packet injection rate of every router (packet/cycle/node) and the vertical line reports latency. Latency is measured in cycles.

Figures 3.8, 3.9, 3.10 and 3.11 compare the latency results under random, hotspot, transpose and shuffle traffic for the first scenario where five elevators are employed. According to Figures 3.8 and 3.9, the Elevator-first routing algorithm enhances latency for random and hotspot traffic. The reason is one more virtual channel compared to the ETW. ETW-based approaches (SEA and DEA) follow the same trend in random and hotspot traffic.

Figure 3.10 compares the latency results for the routing algorithms under the transpose traffic. This traffic is based on vertical link transmission for every pair of source and destination. This traffic pattern is the one having the most vertical transmission compared to any other traffic since every pair of source to destination are on different layers. According to Figure 3.10 for the transpose traffic, ETW-SEA outperforms the other two algorithms.

Figure 3.12 illustrates the number of times a specific elevator is being used in that traffic. So, the horizontal line represents the index of the elevators in the first layer. As it has already been mentioned, the elevators are pillars, meaning that the elevator at node 0 will be connected to node 48. According to this figure, traffic on the elevators located at node 2 and 8 are extremely high in the Elevator-first routing algorithm. Therefore, this technique suffers from long latency. A comparison between SEA and DEA implies that the usage of every elevator in SEA is less than DEA except the elevator at node 7. Traffic on this elevator is relatively high because SEA always forwards the packets toward the east most elevators especially for destinations located at east down of their source. Due to the variety of the paths from source to the east most elevator in SEA, traffic on horizontal link will be distributed. Therefore, SEA performs relatively better compared to DEA. Latency comparison in Figure 3.11 for shuffle traffic reveals that DEA outperforms the other two algorithms. According to Figure 3.13, the elevator usage in SEA is higher than DEA and Elevator-first

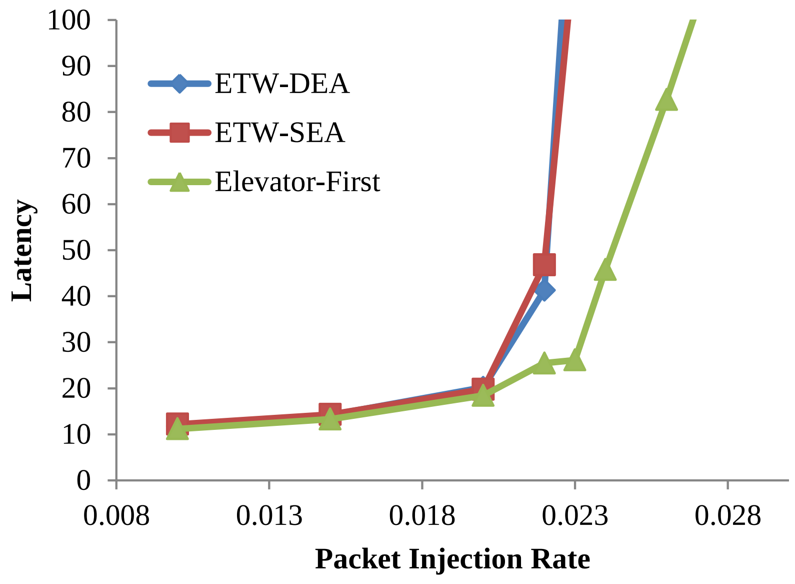


Figure 3.8: Performance under random traffic for 5 TSVs

routing algorithm. Therefore, traffic increases on the elevators and latency increases accordingly. Moreover, elevator usage in DEA is better than the Elevator-first routing algorithm.

Latency comparisons for architecture with eight TSVs under different traffic patterns are shown in Figures 3.14, 3.15, 3.16 and 3.17. According to the figures, as the number of elevators increases, the Elevator-first routing algorithm boasts a better latency performance compared to the other routing algorithm. This is because the load is distributed among a larger number of elevators, and each elevator receives a smaller portion of traffic. Moreover, the network will be saturated in higher packet injection rates. The observations for the five elevators case also hold for eight elevators. According to the results, ETW performs better under the lower number of TSVs.

In order to demonstrate the performance of the proposed routing algorithm versus the Elevator-first routing algorithm, a set of application benchmarks from standard suits including PARSEC [80] and SPLASH2 [79] have been employed. Figures 3.18 and 3.19 illustrate the latency comparison for the routing algorithms for different

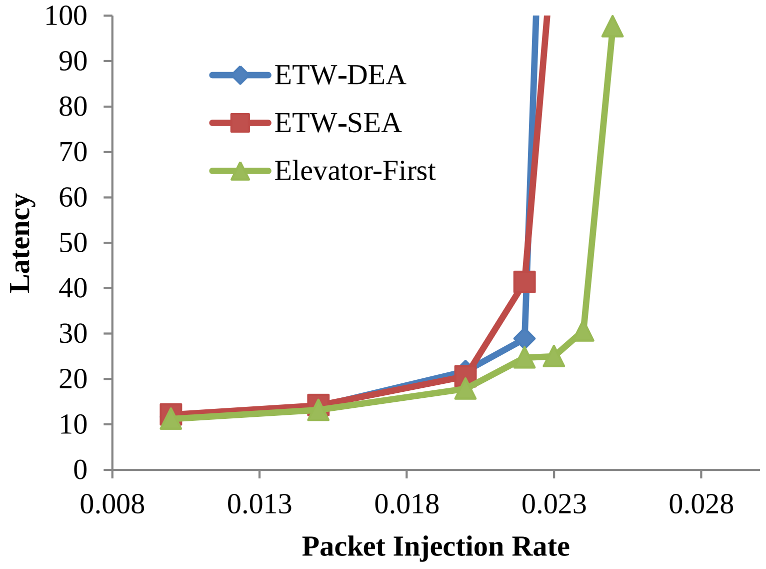


Figure 3.9: Performance under hotspot traffic for 5 TSVs

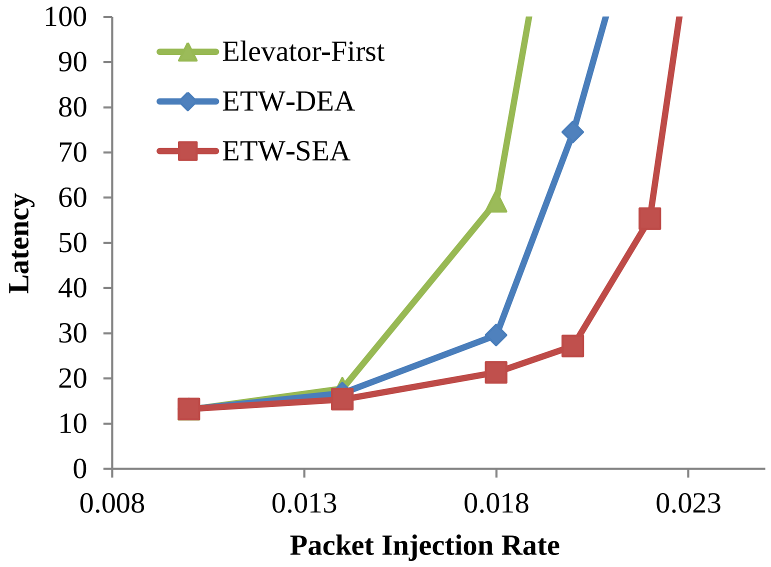


Figure 3.10: Performance under transpose traffic for 5 TSVs

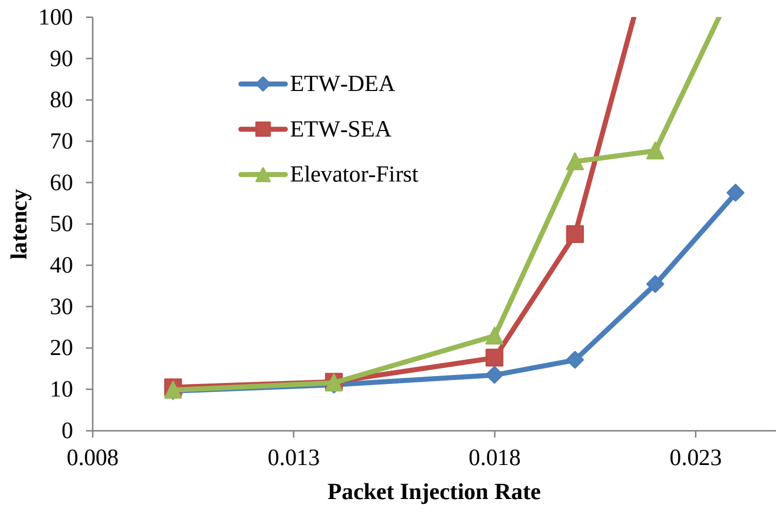


Figure 3.11: Performance under shuffle traffic for 5 TSVs

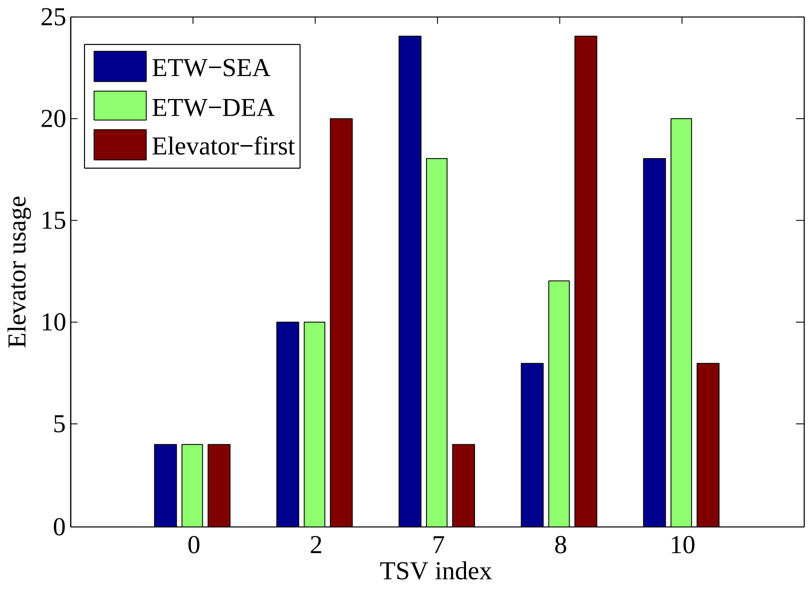


Figure 3.12: Transpose traffic elevator usage for 5 TSVs

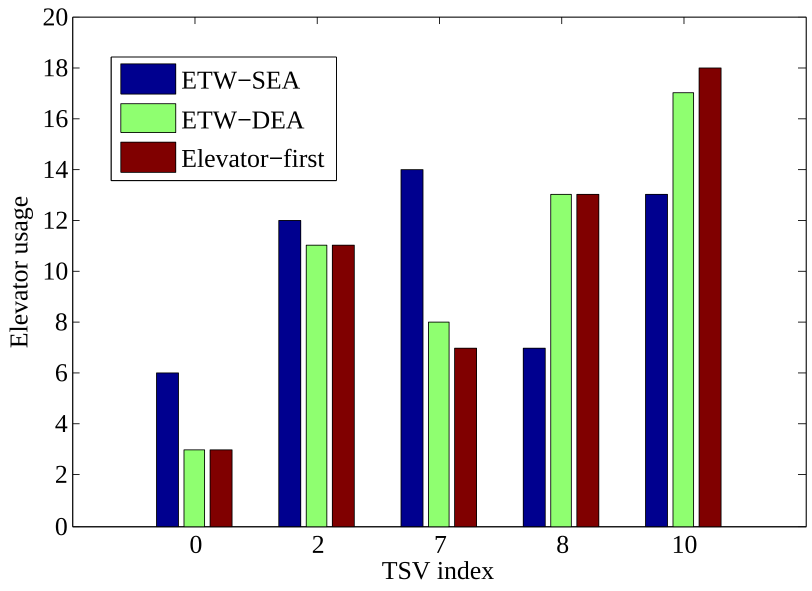


Figure 3.13: Shuffle traffic elevator usage for 5 TSVs

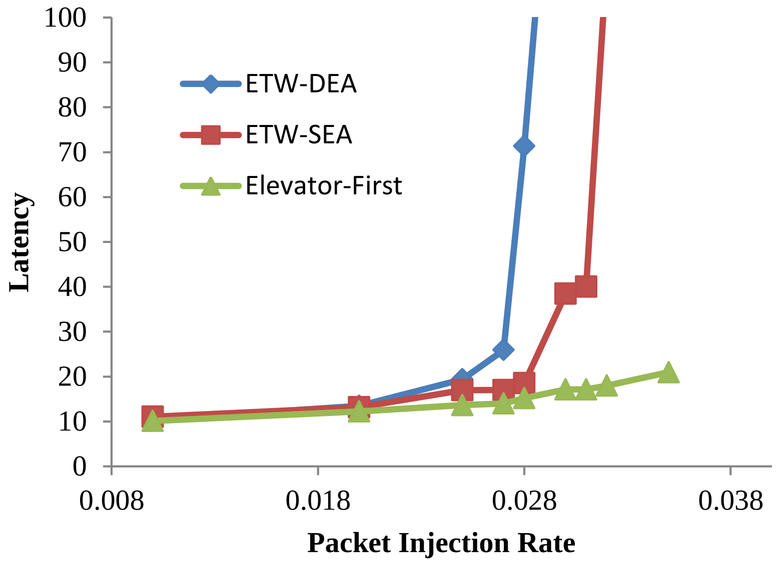


Figure 3.14: Performance under random traffic for 8 TSVs

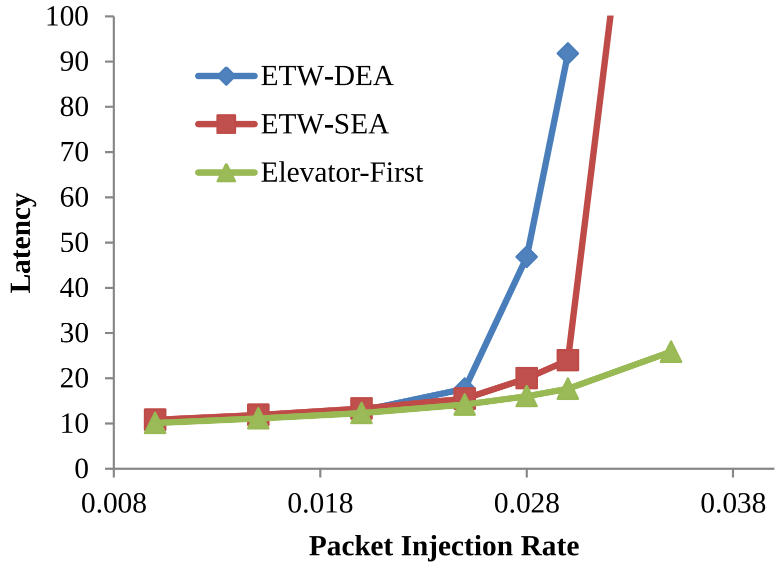


Figure 3.15: Performance under hotspot traffic for 8 TSVs

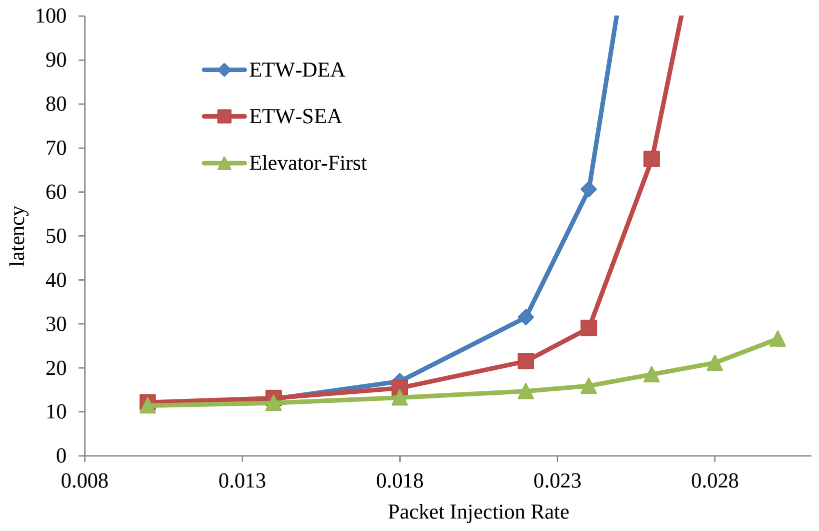


Figure 3.16: Performance under transpose traffic for 8 TSVs

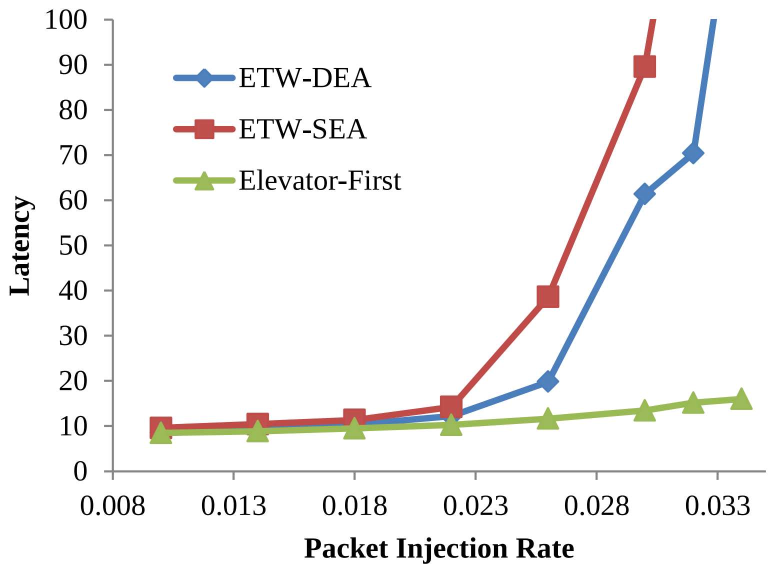


Figure 3.17: Performance under shuffle traffic for 8 TSVs

applications for two architectures. The former has only five TSVs and the latter is for the architecture with eight TSVs. The results are based on a $4 \times 4 \times 4$ network in which buffer depth is five and the packet size is randomly chosen between five to eight flits. According to these figures, as the number of TSVs increases, the performance gap between the Elevator-first and ETW routing algorithm decreases as well. As Figure 3.18 illustrates, the Elevator-first algorithm provides better latency than ETW with each of DEA and SEA elevator assignment. However, according to Figure 3.19, the routing algorithms perform relatively the same for different benchmarks. The results reveal that as the number of elevators increases, especially the centrally located elevators, a better traffic distribution among vertical links is obtained. So, ETW provides nearly the same performance as the Elevator-first routing algorithm although it has one less virtual channel.

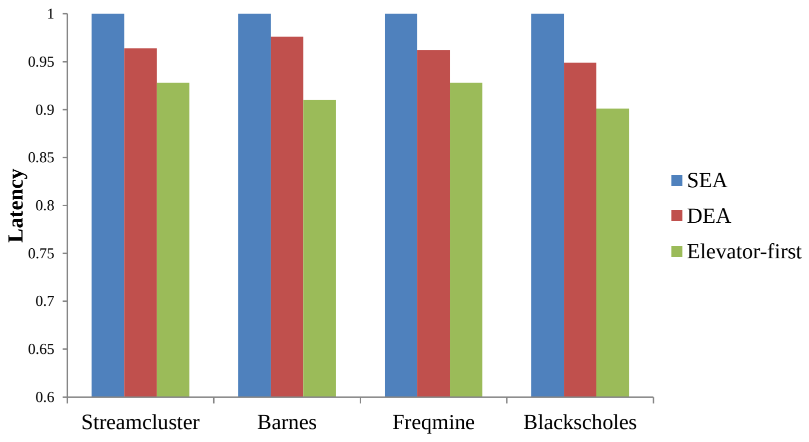


Figure 3.18: Performance under table-based traffic with 5 TSVs

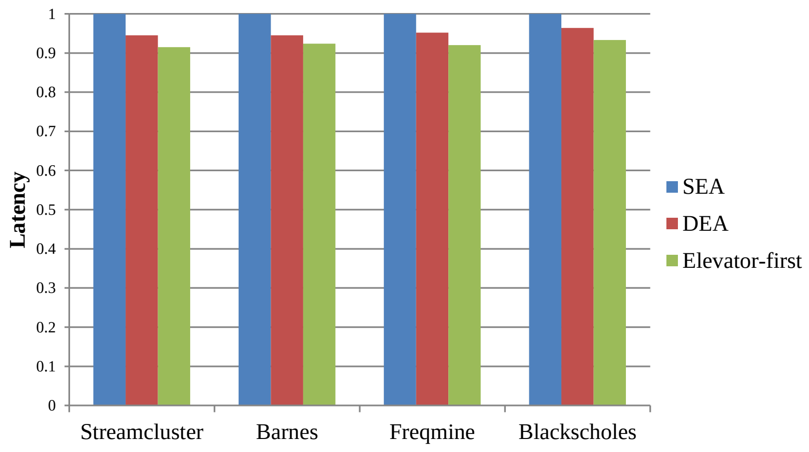


Figure 3.19: Performance under table-based traffic with 8 TSVs

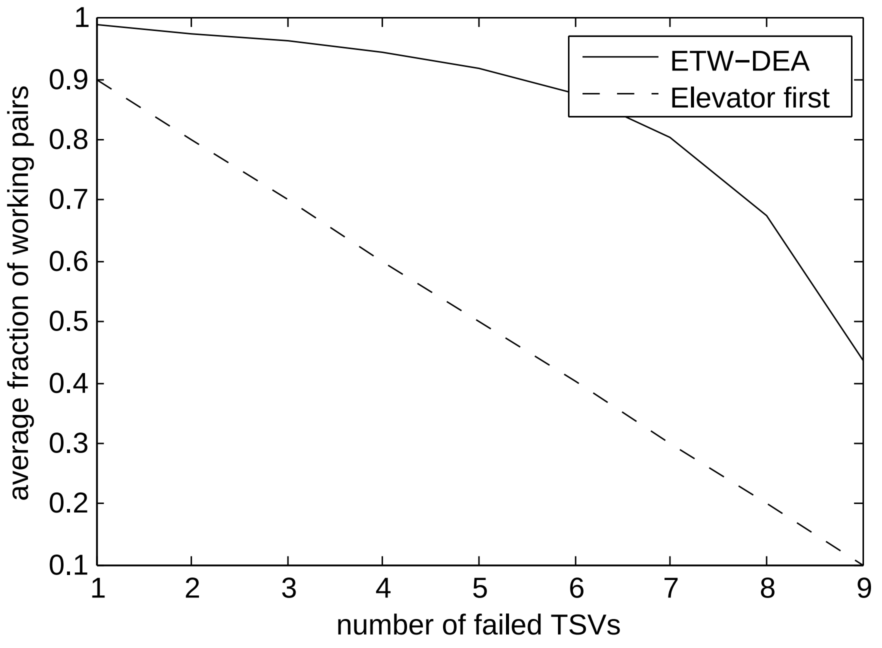


Figure 3.20: Average fraction of working pairs vs number of failed TSVs

### 3.7.3 Reliability Evaluation

Figure 3.20 compares $\bar{f}_{\ell_{n,*}}$ vs $|\mathbf{T}| - n$ (number of failed TSVs) for ETW-DEA and Elevator-first. Compared to the dashed line which corresponds to the Elevator-first algorithm, it is observed that the bold line corresponding to ETW-DEA resists falling when the number of failed TSVs increases. This signifies the higher resilience of ETW-DEA to loss of TSVs as compared to the Elevator-first algorithm. As an example, with 2 failed TSVs, ETW-DEA results in 98% healthy pairs, while Elevator-first retains only 80% of the pairs. With 5 failed TSVs, 90% of pairs have successful packet delivery while this value is less than 50% in the Elevator-first algorithm. Similarly, under 9 faulty TSVs, the average fraction of healthy pairs of ETW-DEA is 4 times larger than that of the Elevator-first algorithm (40% vs. 10%).

Next, we use $\bar{f}_{\ell_{n,*}}$ to calculate $f(t)$ and obtain the overall reliability of the system over time. The specific temporal pattern of $f(t)$ depends on the individual TSV reliability model $R_{\text{TSV}}(t)$. While there are many choices for this, we choose the Weibull distribution due to its versatility [81] in modeling component failure. The 3-parameter Weibull distribution is:

$$p_{\text{Weibull}}(t;\beta,\gamma,\eta) = \frac{\beta}{\eta}\left(\frac{t-\gamma}{\eta}\right)^{\beta-1} e^{-\left(\frac{t-\gamma}{\eta}\right)^{\beta}} \tag{3.9}$$

Here, we consider $(t-\gamma)/\eta$ as 'normalized' time, and focus on the single parameter Weibull distribution:

$$p_{\text{Weibull}}(t;\beta) = c_1 t^{\beta-1} e^{-t^{\beta}} \tag{3.10}$$

where $c_1$ is a normalization factor such that $\int_{-\infty}^{\infty} p^2_{\text{Weibull}}(t)dt = 1$. It is a well-known fact [81] that with $\beta = 1$, the Weibull distribution reduces to exponential distribution and can be used to model component life during its 'working life'. Moreover, $0 < \beta < 1$ and $\beta > 1$ can model component life during the 'infant mortality' and 'wear out' phases of the component's life. The reliability function corresponding to the Weibull distribution is given by:

$$R_{\text{Weibull}}(t) = e^{-t^{\beta}} \tag{3.11}$$

Once plugged into $f(t)$:

$$f(t) = \sum_{n=0}^{|\mathbf{T}|} (1-e^{-t^{\beta}})^n e^{-t^{\beta}(|\mathbf{T}|-n)} \binom{|\mathbf{T}|}{n} \bar{f}_{\ell_{n,*}} \tag{3.12}$$

Figure 3.21 shows $f(t)$ for different choices of $\beta$. The figure illustrates the superior resilience of ETW-DEA compared to the Elevator-first algorithm for all phases of the component's life. For instance, by comparing the blue dashed and solid line, it is clear that the fraction of working pairs falls much more rapidly in time when Elevator-first is used. The same observation holds for the working life and wear-out phases of TSVs.

In practice, the life distribution of a component should be calculated by taking the transition between life phases (infancy, working, and wear out) into account. Unfortunately, equating $R_{\text{TSV}}(t)$ to a single Weibull distribution does not accomplish this goal. To model the transition, we start with the traditional belief that the failure rate $\lambda(t)$ of a component with the three life phases follows a bath-tub curve over time. Specifically, assume that the infancy and working life of a component are of duration $\Delta T_i$ and $\Delta T_w$ respectively, and that the failure rate during the working life is $\lambda$. Also, take the two functions $\lambda_1(t)$ and $\lambda_2(t)$ to be monotonically decreasing and increasing with time, and with $\lambda_1(\Delta T_i) = \lambda_2(0) = \lambda$. Then, the failure rate of the component is given by:

$$\lambda(t) = \begin{cases} \lambda_1(t) & 0 \leq t < \Delta T_i \\ \lambda & \Delta T_i \leq t < \Delta T_i + \Delta T_w \\ \lambda_2(t - \Delta T_i - \Delta T_w) & t \geq \Delta T_i + \Delta T_w \end{cases}$$

Next, $R(t)$ can be calculated from $\lambda(t)$ by solving the following differential equation:

$$\frac{-\frac{dR(t)}{dt}}{R(t)} = \lambda(t), t > 0 \text{ subject to } R(0) = 1 \tag{3.13}$$

and plugged into $f(t)$. Interestingly, if $R_1(t)$ and $R_2(t)$ corresponding to $\lambda_1(t)$ and $\lambda_2(t)$ are known, $R(t)$ can be calculated readily as follows. Clearly, solving the differential equation in $[0\ \Delta T_i]$ results in $R(t) = R_1(t)$. The differential equation in

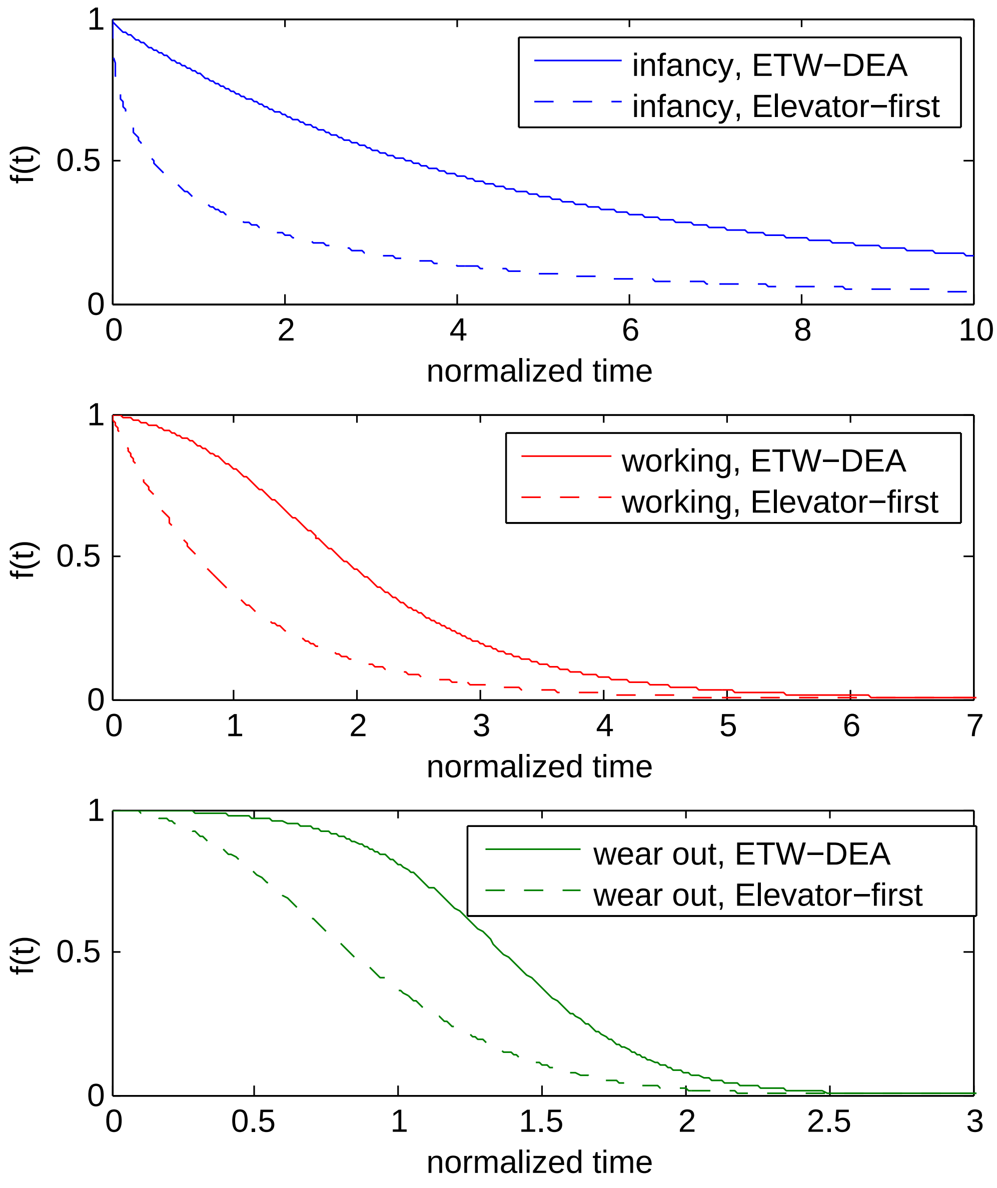


Figure 3.21: Temporal trend of fraction of working pairs for different Weibull parameters

$[\Delta T_i \ \Delta T_i + \Delta T_w]$ is now:

$$\frac{-\frac{dR(t)}{dt}}{R(t)} = \lambda \tag{3.14}$$

$$, t \in [\Delta T_i \ \Delta T_i + \Delta T_w] \text{ subject to } R(\Delta T_i) = R_1(\Delta T_i)$$

which means:

$$R(t) = R_1(\Delta T_i)e^{-\lambda(t-\Delta T_i)}, t \in [\Delta T_i \ \Delta T_i + \Delta T_w] \tag{3.15}$$

Similarly, for $t > \Delta T_i \ \Delta T_i + \Delta T_w$:

$$R(t) = R_1(\Delta T_i)e^{-\lambda \Delta T_w} R_2(t - \Delta T_i - \Delta T_w), t > \Delta T_i + \Delta T_w \tag{3.16}$$

In a nutshell:

$$\begin{aligned} R(t) =& R_1(t), \ t < \Delta T_i \\ R(t) =& R_1(\Delta T_i)e^{-\lambda(t-\Delta T_i)}, \ \Delta T_i \le t < \Delta T_i + \Delta t_w \\ R(t) =& R_1(\Delta T_i)e^{-\lambda \Delta T_w} R_2(t - \Delta T_i - \Delta T_w) \\ & , \ t > \Delta T_i + \Delta T_w \end{aligned} \tag{3.17}$$

Figure 3.22 shows an example of $\lambda(t)$ and $R(t)$ where $\lambda_1(t)$ is Weibull failure rate with parameters $\gamma_1 = -.25, \eta_1 = 1$, and $\beta_1 = .5$. Also, $\Delta T_i = 1$ and $\Delta T_w = 5$. $\lambda_2(t)$ is Weibull failure rate with parameters $\gamma_2 = \Delta T_i + \Delta T_w, \eta_2 = 1$, and $\beta_2 = 2.5$. Using $R(t)$, $f(t)$ is calculated for both Elevator-first and the proposed algorithm. Figure 3.23 compares $f(t)$ for the two algorithms for the entire life of the system, including infancy, working life, and wear out. As can be seen from Figure 3.23, there is a large gap between the two algorithms in terms of fraction of healthy connections especially

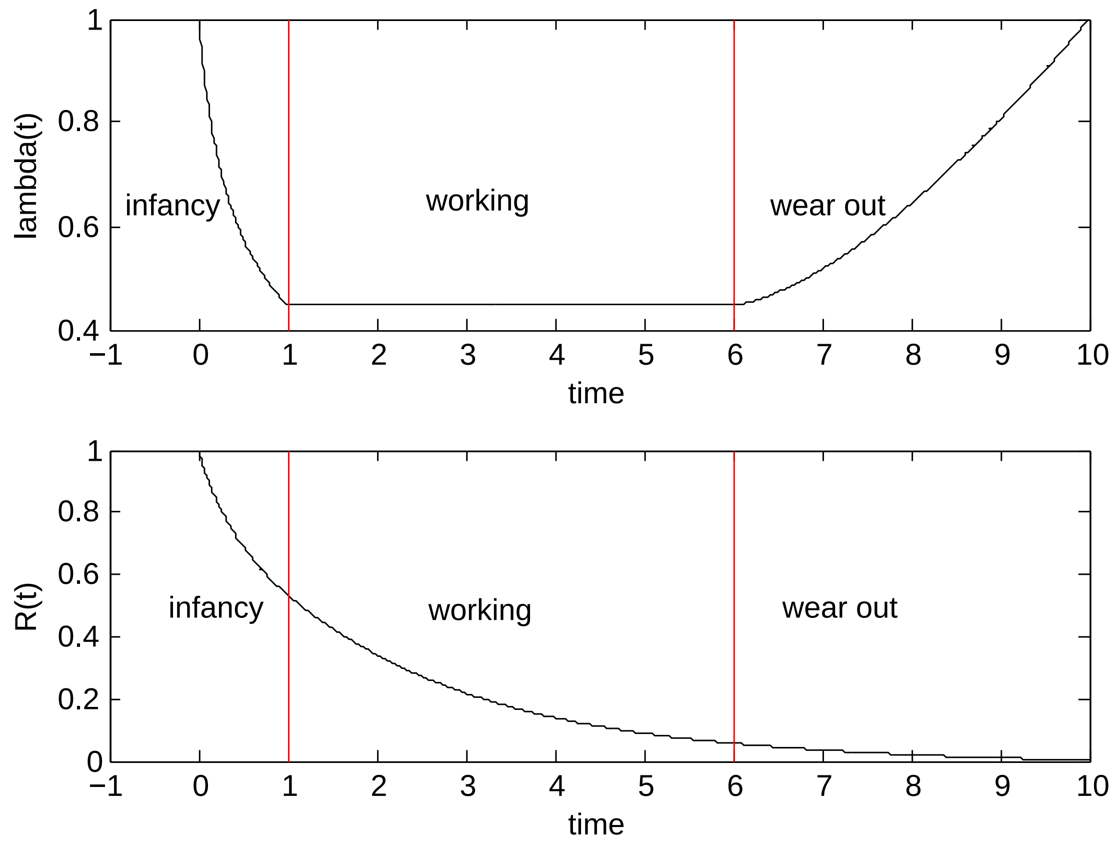


Figure 3.22: Variation of failure rate and reliability

during the working life of the system.

### 3.7.4 Power and Area Analysis

In this section power results extracted from AccessNoxim are reported. Table 3.4 compares power consumption of the Elevator-first routing algorithm with the ETW algorithm under SEA and DEA. The power results are extracted for the architecture with 8 TSVs. Two elevator assignment mechanisms have been used for Elevator-first. According to Table 3.4, the power consumption of Elevator-first with nearest elevator assignment and ETW-DEA is nearly the same under random and shuffle traffic while Elevator-first with random elevator assignment and ETW-SEA consume more power

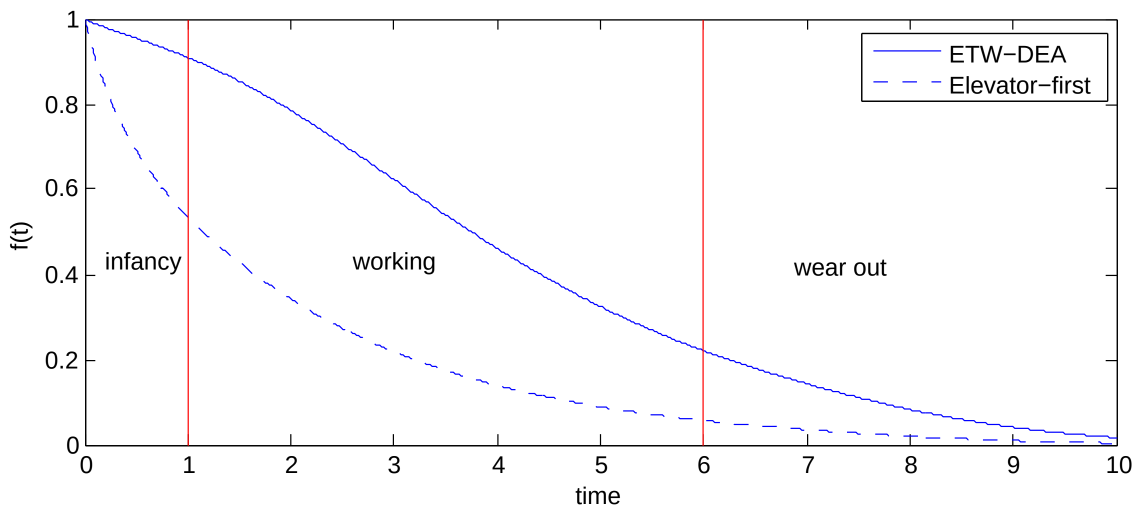


Figure 3.23: Life-time variation of fraction of working pairs

| | Random 8 TSV | | Shuffle 8 TSV | |
|---|---|---|---|---|
| Routing algorithm | Avg power ($\mu W$) | Avg power per router ($nW$) | Avg power ($\mu W$) | Avg power per router ($nW$) |
| Elevator-first(SMD) | 4.29 | 67 | 3.54 | 55.3 |
| Elevator-first(random) | 4.63 | 72.3 | 3.91 | 61.1 |
| ETW-SEA | 4.64 | 72.5 | 3.96 | 61.9 |
| ETW-DEA | 4.30 | 67.2 | 3.55 | 55.5 |

Table 3.4: Power consumption evaluation

since the traversed paths are longer. The tables report the average power and average power per router.

Regarding the area, the routing unit is a light-weight unit and the area consumption of Elevator-first and ETW are nearly the same. Buffers are the most area-hungry part of a router design. In this respect, Elevator-first uses one more virtual channel than ETW, occupying a relatively larger area.

# Chapter 4

# CoBRA Routing Algorithm

In this chapter, some modifications have been made in ETW to enhance the fault tolerance of the network. According to Chapter 3, ETW with dynamic elevator assignment reroutes the packet to a new elevator located on the east side of the network imposed by the routing rules. CoBRA tries to postpone moving toward east in order to take advantage of using the elevators located in the same column as the current node.

## 4.1 Overview of CoBRA

In this algorithm, routers do not have a global knowledge about the location of TSVs as it is the case in the Elevator-first routing algorithm. The only information that routers maintain is the presence of any healthy TSV in the same column (with the same $X$ value) as the current router. The details on the propagation of TSV statuses are elaborated in Section 4.2.

CoBRA uses two virtual channels along the $Y$ dimension to guarantee the freedom from deadlock. For this purpose, the network is virtually partitioned into two disjoint subnetworks: Subnetwork1 $(X^+, Y0^*, Z^+)$ and Subnetwork2 $(X^-, Y1^*, Z^-)$ where +, - represent channels along positive and negative directions respectively, while * stands for both directions. Based on this partitioning, packets can use the channels of either Subnetwork1 or Subnetwork2. In addition packets can move from Subnetwork1 to Subnetwork2 or vice versa but switching is allowed only in one direction at a time to avoid deadlock. The deadlock-freedom is elaborated in Section 4.3.

In CoBRA, the default transition occurs from Subnetwork1 to Subnetwork2. If all TSVs fail in the eastmost column, the algorithm is reconfigured to route packets first in the Subnetwork2 and then in Subnetwork1. This reverses all the conditions and thus TSV failures are tolerated again as long as there is one healthy TSV in the westmost column.

In more details, CoBRA routing algorithm can be described as follows:

### 4.1.1 Current and destination are on the same layer:

If the destination is on the east side of the current node, Subnetwork1 is used to deliver the packet (i.e. channels $X^+$ and $Y0^*$). On the other hand, if the destination is on the west side of the current node, the channels of Subnetwork2 are used (i.e. channels $X^-$ and $Y1^*$).

### 4.1.2 Current and destination are not on the same layer:

As it was mentioned, the routers are aware of TSV statuses in their columns. If no elevator is found in the column, the packet is forwarded one hop to the east and the process is repeated until a healthy TSV is found. In the worst case, if no TSV is found at the eastmost column, the packet has to be dropped. This implies that a healthy elevator in the eastmost column guarantees the delivery of packets to all destinations no matter how many elevators are available in the network or disabled at runtime. Upon the loss of all TSVs in the eastmost column, CoBRA routing algorithm is reconfigured to deliver the packets toward the west direction. After this reconfiguration, every router forwards the packet one hop to the west if there is no healthy elevator in the same column.

Figure 4.1 shows a $4 \times 4 \times 2$ network where the source node 20 sends a packet to the destination node 6. Since there is no TSV at the source column, packets are forwarded toward east through Subnetwork1. In the next column (with $X = 1$), there are two available TSVs, located at node 17 and 25. Since the node 6 is in the upper $Y$-half plane, the elevator located at the node 17 is a better choice to deliver the packet. The routing path is as follows: $20 \rightarrow 21 \rightarrow 17 \rightarrow 1 \rightarrow 2 \rightarrow 6$, or alternatively the packet can take the path $20 \rightarrow 21 \rightarrow 17 \rightarrow 1 \rightarrow 5 \rightarrow 6$. When the source node 10 targets the destination node 21, the elevator at the node 2 is used since the destination is in the upper $Y$-half plane.

## 4.2 Providing Partial Knowledge

Propagation of TSV statuses locally enhances the reliability of 3D-NoC significantly. Providing global information about the location of healthy and faulty elevators in a network may improve the performance but in turn it consumes more resources. In CoBRA, routers in the same column share the TSV statuses with each other. For this purpose, a router is equipped with two signals as it is shown in Figure 4.2, one transferring the TSV status from north to south (i.e. called signal $A$) and another one from south to north (i.e. called signal $B$). Figure 4.2 represents how these signals

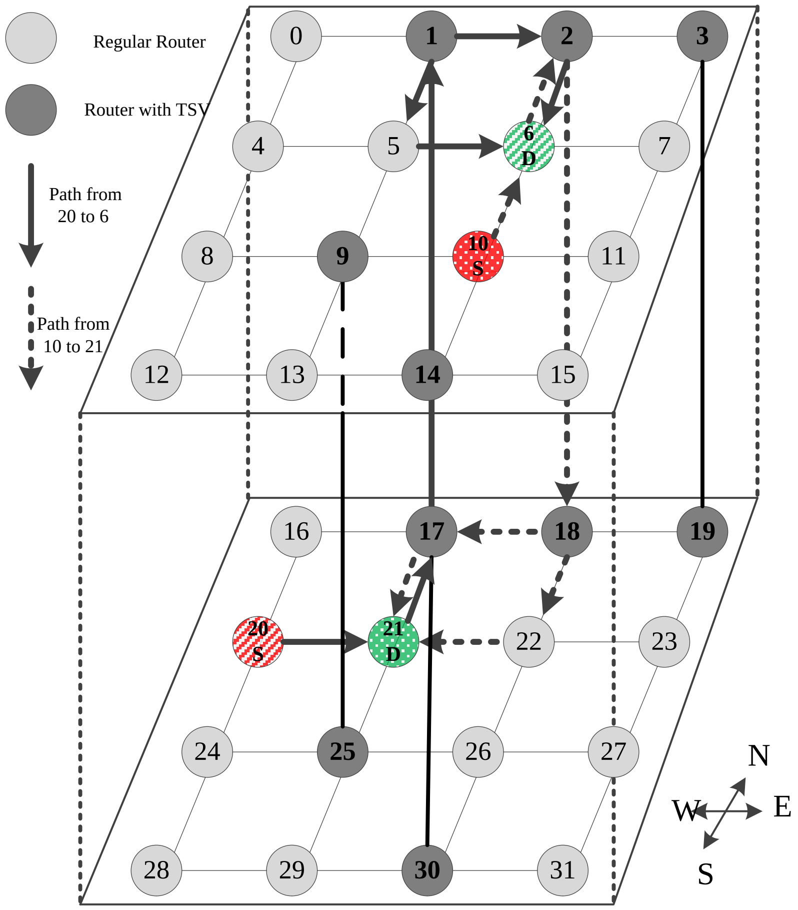


Figure 4.1: An example of a 3D-NoC

have been connected among four routers located in the same column. According to the figure, the signal $A$ reflects the fault information on the north neighbors of a router. If this signal value is one, it means that there is at least one healthy elevator on the north direction of the node. Similarly, the signal $B$ propagates the fault information in the south direction. The signal $A$ and the TSV information of the current router are ORed together to form the signal $A$ that should be sent to the next router. Therefore, if $A = 1$ or the current node has an elevator, the signal $A$ of the next router gets the value of one representing the existence of a healthy elevator in the north direction of a node. The same trend is applied to the signal $B$.

Figure 4.2 (a) shows the value of signals when there are two TSVs at the routers 4 and 12. As it is clear, the router at the node 4 does not have any elevator at the north direction while there is one at the south direction which is indicated by the value of the signals $A = 0$ and $B = 1$, respectively. The router at the node 4 does not exactly know where the healthy elevator is located or how many healthy elevators there are on the south direction. The router just knows that there are healthy elevators in the south direction that can be used for vertical transmission.

If a fault disconnects one of the TSVs during runtime, the new status is propagated in the column through the wires. The ORed signals will be updated and routers will adapt themselves to TSV failures. Figure 4.2 (b) shows the value changes on the signal $A$ when the TSV at the router 4 becomes faulty.

## 4.3 Discussion of Deadlock Freedom

We have already mentioned that the network will be deadlock free if packets use either the channels of Subnetwork1 or Subnetwork2 in addition to the possibility

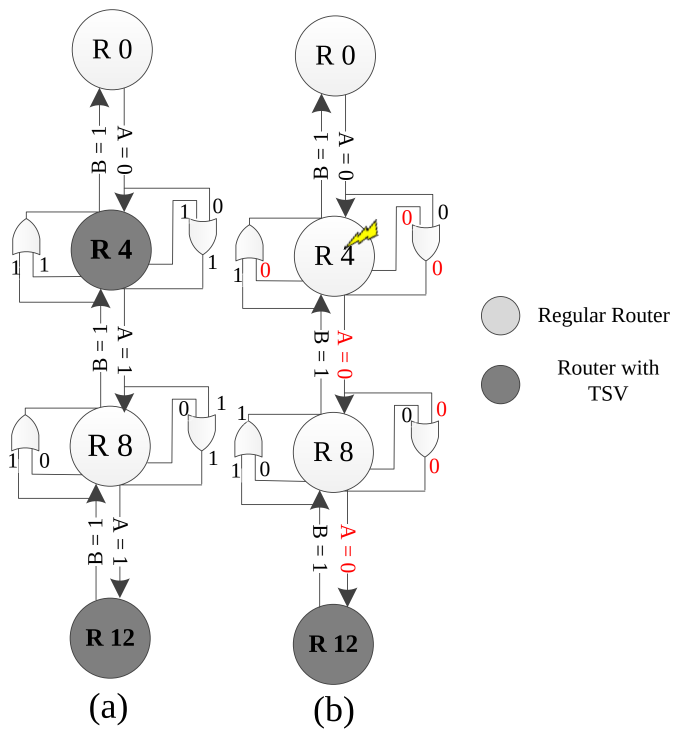


Figure 4.2: Implementation of TSV status propagation.

of switching from Subnetwork1 to Subnetwork2. Based on subnetwork definition, there are no circular dependencies in each subnetwork. Moreover, when packets are switched from Subnetwork1 to Subnetwork2, there cannot exist any circular path since the direction of moving along $X$ and $Z$, as well as the virtual channel index along $Y$, change upon subnetwork switch. In this section, we use formal methods to verify three properties. The routing logic has to ensure *deadlock-* and *livelock-freedom*. Additionally, even in the presence of faulty TSVs, the routing logic should always be *connected*. In other words, for any pair of source and destination, there must be at least one possible route. All these properties depend on the assumption that between each pair of layers, there is at least one non-faulty TSV.

Let a *configuration* be an assignment of TSVs (faulty or not) to nodes. Each configuration induces a new channel dependency graph as each configuration causes the routing logic to make different choices. Let $x$, $y$ and $z$ be the dimension of the mesh and let $t$ be the number of TSVs. The total number of configurations is:

$$\sum_{f \leq t} \binom{xyz}{e} \cdot \binom{e}{f} \tag{4.1}$$

For example, in a $4 \times 4 \times 4$ mesh with 6 TSVs there are $512,512$ different configurations. It is infeasible to run simulations for all these configurations or perform a manual proof.

To address this issue, we have used DCI2 to formally verify CoBRA for all of the above properties [82]. DCI2 takes as input a model of the routing logic in the form of a function $R :: N \times N \mapsto P$, i.e., a function $R$ that takes as input the current node and the destination node and produces as output the port to which the packet is routed. DCI2 enumerates all configurations and generates the corresponding channel dependency graphs. Based on these graphs, it checks a necessary and sufficient condition for deadlock-free adaptive routing.

We integrated DCI2 and AccessNoxim and instead of analyzing a separate model, DCI2 has been applied to the exact same routing code as was used for AccessNoxim.

When given a $4 \times 4 \times 4$ mesh with 6 TSVs, all $512,512$ configurations are generated. Among all, $208,252$ configurations satisfy the assumption that there is at least one healthy elevator at the eastmost column. All these configurations are formally proven to be deadlock- and livelock-free and to be connected. Other configurations are not considered. We have verified CoBRA for any number of elevators from 0 up to and including 6. The total verification time is about 90 minutes on a 4 core 2 GHz Intel Core i7 machine. Table 4.1 reports the required verification time for 1, 2, 4, and 6 elevators. A configuration is an assignment of locations to elevators.

| Number of elevators | Number of faults | Total number of config | Number of eastmost config | Verification time (Sec) |
|---|---|---|---|---|
| 1 | 0 | 16 | 4 | 2 |
| 2 | 0 | 120 | 54 | 3 |
|  | 1 | 240 | 60 | 3 |
| 4 | 0 | 1820 | 1325 | 28 |
|  | 1 | 7280 | 4420 | 93 |
|  | 2 | 10920 | 4914 | 113 |
|  | 3 | 7280 | 1280 | 46 |
| 6 | 0 | 8008 | 7084 | 127 |
|  | 1 | 48048 | 39336 | 640 |
|  | 2 | 120120 | 87450 | 1355 |
|  | 3 | 160160 | 97240 | 1655 |
|  | 4 | 120120 | 54054 | 980 |
|  | 5 | 48048 | 12012 | 248 |

Table 4.1: Verification results

## 4.4 Results and Discussion

The efficiency of the proposed routing algorithm under different number of faults has been evaluated using AccessNoxim simulator [76].

A $4 \times 4 \times 4$ mesh NoC has been considered for experiments. All the routers have 5-flit FIFO and the packet size is 8 flits. The simulator is warmed up for 1000 cycles and then the reliability is evaluated over another $20,000$ cycles. The defective TSV is modelled as an open fault. Therefore, if a TSV or a bundle of TSVs are faulty, the entire vertical connection is considered broken.

To evaluate the reliability of the proposed routing algorithm against available routing algorithms, Elevator-first is implemented in AccessNoxim alongside CoBRA. It is necessary to mention that there are few algorithms in literature tolerating faults in partially connected 3D-NoCs. For this reason, the performance of CoBRA cannot be compared with the commonly used routing algorithms, such as $XYZ$ which is proposed for the fully connected 3D-NoCs. The measure of reliability defined in this

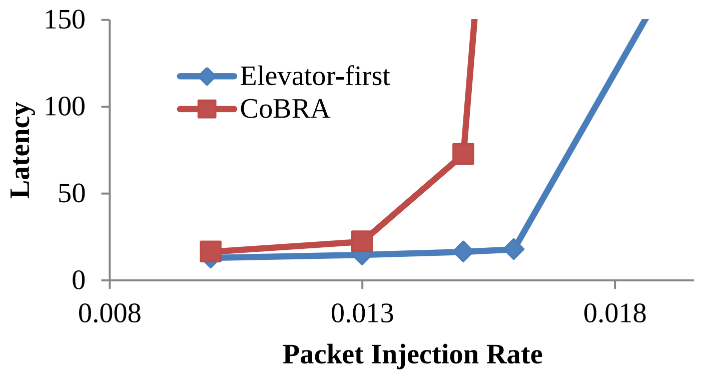


Figure 4.3: Performance under random traffic for 4 TSVs

dissertation is the percentage of flits successfully delivered to the target destinations.

In order to model run-time TSV failures, faults are injected at every 5000 cycles. This value is selected to ensure that the network is stabilized before injecting a new fault. Moreover, results for different traffic patterns including synthetic and real traffic scenarios are reported [79] [80].

Three architectures have been used to evaluate the efficiency of the CoBRA routing algorithm. The first one has four elevators at four corners located at nodes 0, 3, 12 and 15 based on the numbering given in Figure 4.1. The second one has eight TSVs at nodes 0, 2, 5, 7, 8, 10, 13 and 15. Finally, the third one has five TSVs located at nodes 0, 2, 7, 8 and 10. Figures 4.3, 4.4 and 4.5 illustrate the latency comparison for the fault-free Elevator-first and CoBRA routing algorithms for the three architectures under random, real traffic, and shuffle, respectively.

According to Figure 4.3, under the uniform random traffic pattern and by the availability of four TSVs, Elevator-first outperforms CoBRA. According to Figure 4.4, CoBRA and Elevator-first perform relatively close under the real traffic. As it is clear in Figure 4.5, CoBRA outperforms Elevator-first under shuffle traffic if there are five elevators in the network at nodes 0, 2, 7, 8 and 10. Therefore, the number and the location of elevators affect the performance of CoBRA and Elevator-first.

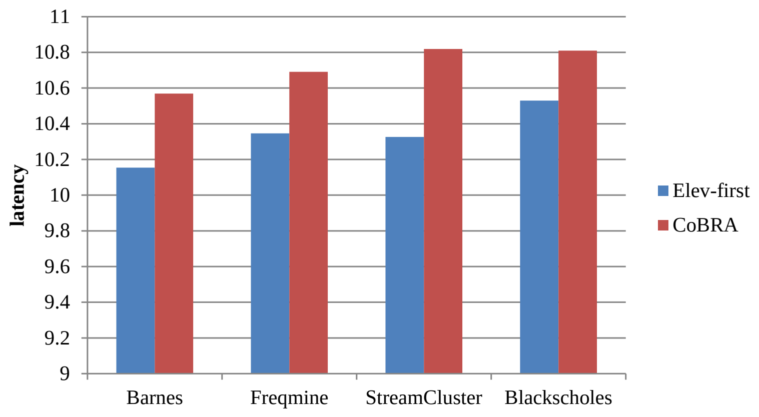


Figure 4.4: Latency comparison under real traffic for 8 TSVs

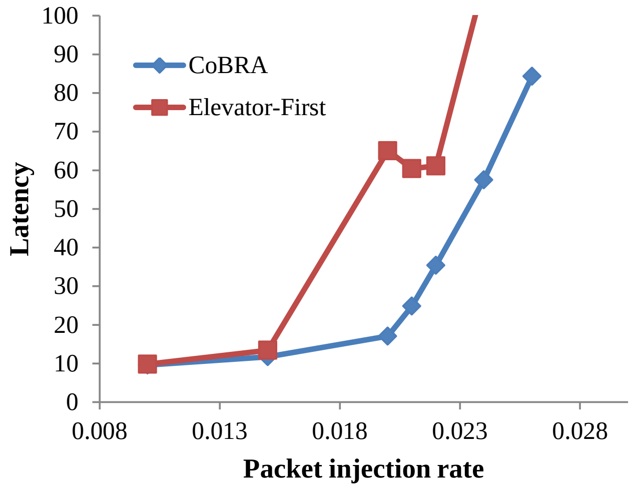


Figure 4.5: Performance under shuffle traffic for 5 TSVs

### 4.4.1 Reliability Comparison under Synthetic Traffic

The reliability comparison for the architecture with four TSVs at four corners have been represented in this section. For this architecture, the effects of single, double and triple faults have been assessed. According to the results (Figure 4.6), CoBRA provides full reliability in the presence of a single fault. The reliability of Elevator-first drops to nearly 85% and 70% under random and transpose traffic, respectively. Since the transpose traffic is based on vertical transmission for every pair of source and destination, a single fault has more severe effect on this traffic. As it is clear, Elevator-first can not adapt itself to faults at runtime.

The effect of changing the location of double faults is presented in Figure 4.7. CoBRA is fully fault-tolerant as long as there exists one healthy TSV at the eastmost column. When both TSVs in the eastmost column fail, CoBRA is reconfigured to switch from Subnetwork2 to Subnetwork1. By routing packets to the west, CoBRA will be able to tolerate faults as long as there exists at least one healthy TSV in the westmost column. For all of the presentend fault scenarioa, the reliability of Elevator-first falls within 47% to 78%.

Three failure scenarios have been considered in the Figure 4.7, where in each scenario two TSVs have been disconnected:

1. Faulty TSVs at nodes 0 and 12: under this scenario CoBRA supports full reliability since it dynamically seeks for the healthy elevators by forwarding the packet to the east direction.
2. Faulty TSVs at nodes 3 and 15: In this case the reliability of CoBRA decreases considerably as no elevator can be found in the east direction. The reconfiguration, to the contrary, provides full reliability due to the existence of healthy

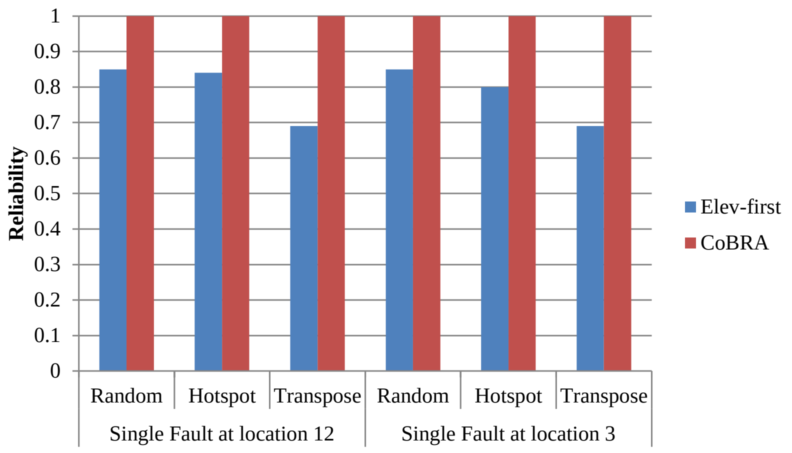


Figure 4.6: Reliability under single faults for 4 TSVs

elevators in the westmost column. It should be noted that some packets are dropped in the reconfiguration phase until the network backs to its stable condition again. Elevator-first drops 30% of flits under this condition.

3. Faulty TSVs at nodes 0 and 15: CoBRA provides full reliability because of one healthy elevator at the node 3.

Figure 4.8 illustrates the effect of triple faults. According to this figure, only one healthy elevator located at the node 15 guarantees the delivery of all packets to destinations in CoBRA while Elevator-first delivers only 45% and 15% of packets to destinations under random and transpose traffic respectively. Moreover, triple faults at locations 0, 3 and 15 provide a reliability of 50% for the Elevator-first. On the other hand, no healthy elevator at the eastmost column drops the reliability by 55% and 70% for the random and transpose traffic in CoBRA, respectively compared to full reliability support in the first triple fault scenarios. Again, reconfiguration solves the problem.

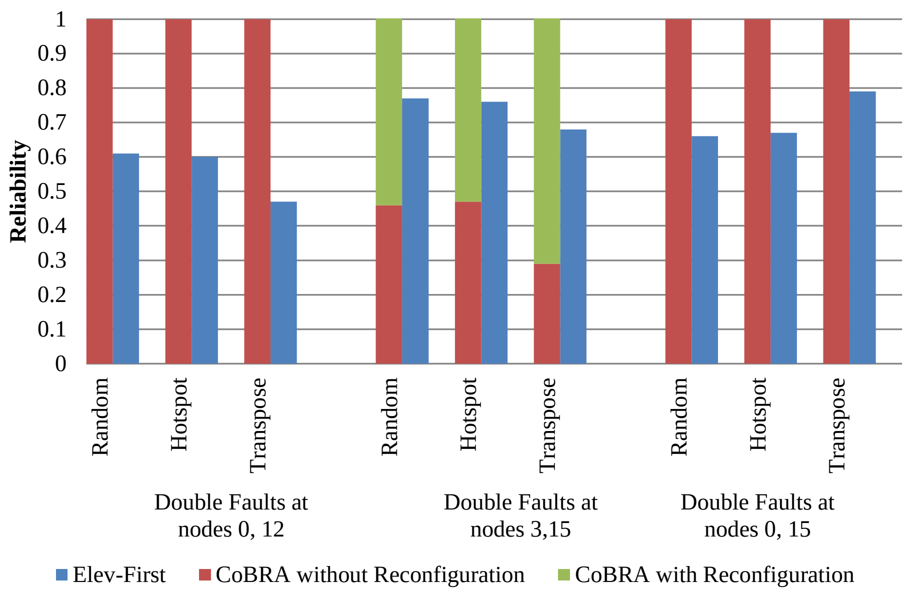


Figure 4.7: Reliability under double faults for 4 TSVs

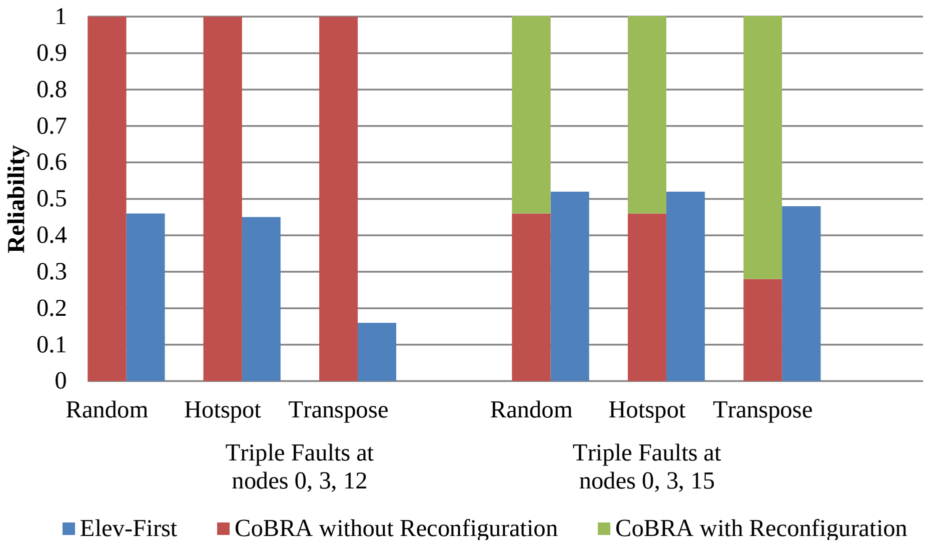


Figure 4.8: Reliability under triple faults for 4 TSVs

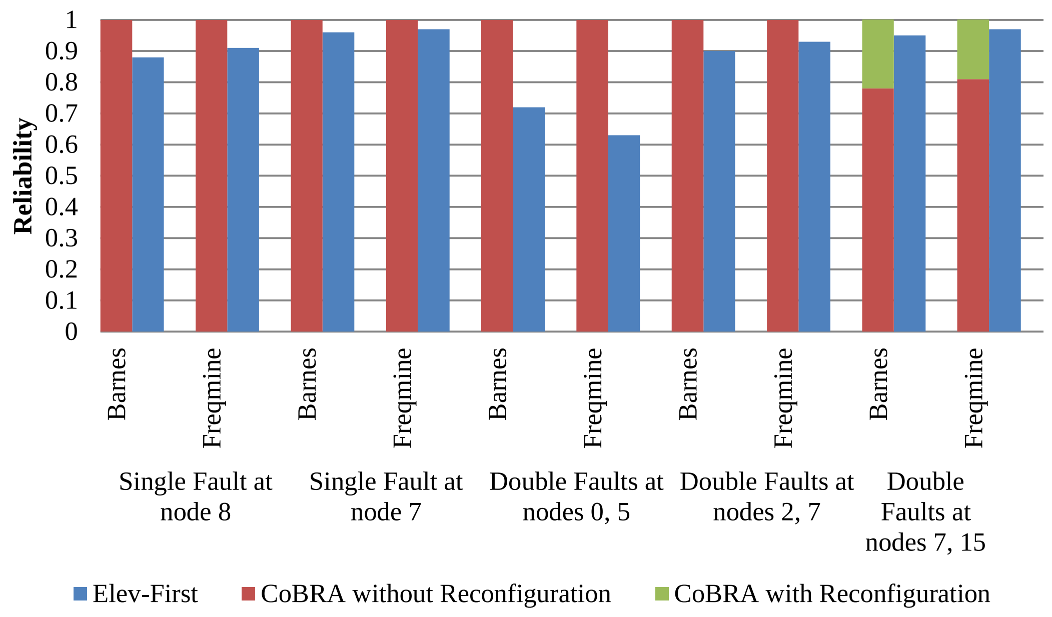


Figure 4.9: Single and double faults comparison for 8 TSVs

### 4.4.2 Reliability Comparison under Real Traffic

The fault tolerance of CoBRA versus Elevator-first for single, double and four faults have been evaluated for the real traffic Barnes and Freqmine. The Streamcluster and Blackscholes perform relatively close to Freqmine so they have been omitted due to the lack of space. Based on Figure 4.9, CoBRA delivers all the packets to destinations unless double faults are located at nodes 7 and 15. Figure 4.10 illustrates the reliability comparison under 4 faults.

### 4.4.3 Power and Area Comparison

In this section, the power consumption of CoBRA and Elevator-first are compared under different fault scenarios for the random traffic. The power reports are extracted from AccessNoxim which accumulates energy upon flit reception/transmission at a router.

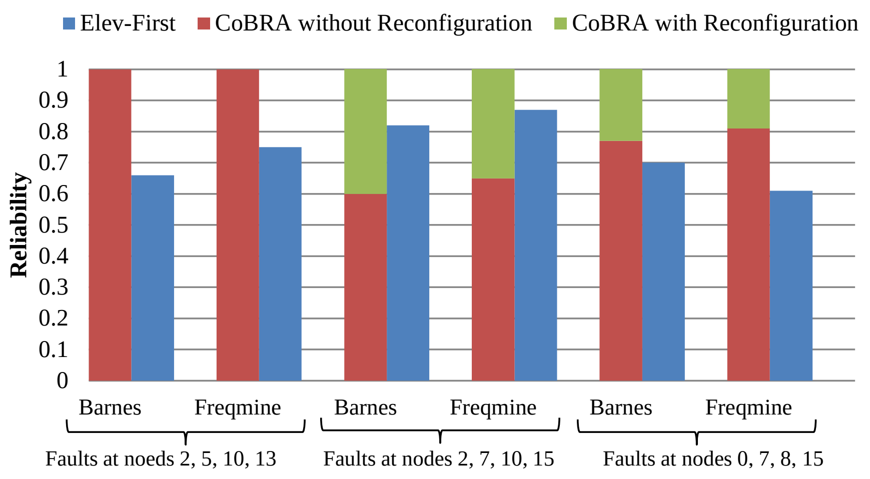


Figure 4.10: Reliability comparison for 8 TSVs

Table 4.2 reveals that the power consumption of CoBRA is higher than Elevator-first in most cases. This is because Elevator-first drops packets when a TSV is faulty. Therefore, network congestion reduces and the power consumption decreases as well. It is in contrast with CoBRA where packets are rerouted when they encounter a faulty TSV. In a fault-free network, the energy consumption of CoBRA might be higher or lower than Elevator-first depending on the number and location of TSVs. It is worth noting that Elevator-first consumes more static power due to the extra buffers in the north and south input ports. However, static power consumption is not considered in the results as it is independent of the routing algorithm.

The packet injection rates for the fault scenarios are adjusted to be on the saturation threshold. It can be observed from the table that these PIRs are different for each fault scenario. Specifically, the PIRs of the first (faults at 0 and 15) and the third fault scenario (faults at 0, 3 and 12) are noticeably lower that the PIR of the second scenario (faults at 3 and 15). Also, the PIR of the third faulty scenario (faults at 0, 3 and 12) is slightly lower than the PIR of the first case (faults at 0 and 15). This observation is based on the fact that for example when faults are presented at 3 and 15, the drop rate is higher compared to the other two cases, which allows the nodes

| Routing algorithm | Random Traffic<br>Average power<br>(Whole network)<br>($\mu J/cycle$)<br>Double Faults<br>0, 15<br>PIR = 0.007 | <br><br><br><br>Double Faults<br>3, 15<br>PIR = 0.017 | <br><br><br><br>Triple Faults<br>0, 3, 12<br>PIR = 0.006 |
|---|---|---|---|
| CoBRA without Reconfiguration | 1.68 | 2.22 | 1.52 |
| Elevator-first | 1.15 | 2.78 | 0.9 |

| Routing algorithm | Random Traffic<br>Average power<br>per router<br>($nJ/cycle$)<br>Double Faults<br>0, 15<br>PIR = 0.007 | <br><br><br><br>Double Faults<br>3, 15<br>PIR = 0.017 | <br><br><br><br>Triple Faults<br>0, 3, 12<br>PIR = 0.006 |
|---|---|---|---|
| CoBRA without Reconfiguration | 26.23 | 34.6 | 23.8 |
| Elevator-first | 18 | 43.5 | 14.12 |

Table 4.2: Power consumption comparison

to inject more packets into the network before reaching the saturation point.

Regarding the area, Elevator-first occupies relatively larger area compared to CoBRA since it uses two virtual channels along the $X$ and $Y$ dimensions while CoBRA has just two virtual channels along the $Y$ dimension.

# Chapter 5

# LEAD Routing Algorithm

In this chapter, LEAD routing algorithm is proposed which relaxes the condition for moving toward east in ETW routing algorithm. LEAD has no specific condition in choosing the elevator for vertical transmission. Besides, the analytical models are applied to compare the performance with the simulation results.

## 5.1 Overview of LEAD

The LEAD algorithm takes into account the vertical partial connectivity in 3D-NoCs where the traditional routing algorithms (e.g. $XYZ$) are not applicable. In this routing algorithm, a packet is free to take any elevator without limitation.

LEAD needs two, two and one virtual channels along the $X$ ($X0, X1$), $Y$ ($Y0, Y1$)and $Z$ dimensions, respectively. $X0$ and $Y0$ refer to virtual channel 0, while $X1$ and $Y1$ refer to virtual channel 1. The algorithm takes advantage of adaptivity for transmitting the packets in source and destination layers. To prove the deadlock-freedom, we assume that the network is virtually partitioned into five disjoint subnetworks: Subnetwork1 ($X0^+$, $Y0^*$), Subnetwork2 ($X0^-$), Subnetwork3 ($Z^*$), Subnetwork4 ($X1^+$) and Subnetwork5 ($X1^-$, $Y1^*$). The +, - and * symbols in the subnetwork definition represent positive, negative and bi-directional channels respectively. In Subnetwork1 and Subnetwork5, packets have the flexibility to take $X$ and $Y$ dimensions in any order. In other words, moving along the $X$ and $Y$ dimensions is random and does not necessarily follow the dimension order routing.

In addition to subnetwork definitions, a rule for switching between subnetworks is provided to completely characterize the routing algorithm. The switching rule is very simple: the only allowed switching from subnetwork $i$ is $i \rightarrow j$, for $j > i$.

The subnetwork definitions and the switching rule imply that in the source layer packets move adaptively toward east and deterministically toward west. Next, Subnetwork3 is applied to deliver packets to the destination layer. Finally, packets follow the reverse pattern in the destination layer (i.e. adaptively toward west and deterministically toward east). Overall, the adaptivity leads to a more balanced load across the network.

## 5.2 Proof for Deadlock-freedom

A routing algorithm is deadlock-free if no cycle forms in the network. A deadlock is a situation in which packets are waiting for each other to release the reserved resources. In other words, if a waiting activity never finishes, deadlock lasts forever.

To prove the routing algorithm is deadlock-free, it is necessary to show that each subnetwork is deadlock-free and also transitions between subnetworks do not form cycles. We argue that to form a cycle, positive and negative directions along at least two dimensions have to be taken. Based on this definition, all subnetworks

| Subnetworks | CompletedPair | Missed Dimension | MissedDirection |
|---|---|---|---|
| ($X0^+$, $Y0^*$) | Y | Z is missing | $X0^-$ is missing |
| ($X0^-$) | - | Y and Z are missing | $X0^+$ is missing |
| ($Z^*$) | Z | X and Y are missing | - |
| ($X1^+$) | - | Y and Z | $X1^-$ is missing |
| ($X1^-$, $Y1^*$) | Y | Z is missing | $X1^+$ is missing |

Table 5.1: Completed pairs within each subnetwork

are deadlock free. Table 5.1 illustrates how the subnetworks are deadlock-free in this algorithm. As an example, subnetwork1 is deadlock-free as packets can use only three channels and with these three channels no cycles can be formed. We should note that 180-degree turns are not allowed. Transitions between deadlock-free subnetworks in an ascending order (or descending order) can not lead to a deadlock as it forms a spiral rather than a closed cycle.

## 5.3 LEAD Algorithm Procedure

The basic goal of every routing algorithm is to find a path from a specific source to a specific destination. Routing algorithms proposed for partially connected 3D-NoCs are responsible for delivering the packets to the elevator in the source layer and determine a path from the elevator to the destination in the destination layer. The proposed routing algorithm based on the subnetwork definition is described as follows:

### 5.3.1 Source and destination are on the same layer

The virtual channel number is randomly selected from $\{0, 1\}$, which corresponds to $(X0, Y0)$ and $(X1, Y1)$ respectively. The randomness in selecting the channel distributes the traffic more evenly compared to deterministic assignment of one fixed channel to source and destinations located in the same layer. The algorithm behaves as follows:

1) Virtual channel 0 is selected: Two cases can happen depending on the relative position of source and destination:

1a) If the destination is to the east of the source, Subnetwork1 ($X0^+$, $Y0^*$) is used to deliver the packet to the destination adaptively, not necessarily following the dimension-ordered routing.

1b) If the destination is to the west of the source, first, Subnetwork1 is applied and the packet is forwarded through $Y0$ channel. Then, the packet is delivered to the destination by switching to Subnetwork2 and taking $X0^-$ direction. The routing algorithm is deterministic if the destination is to the west of the source.

2) Virtual channel 1 is applied:

2a) If the destination is to the east of the source, the packet is delivered to the destination deterministically. That is, the packet takes Subnetwork4 ($X1^+$) first since moving eastward is not allowed in Subnetwork5. Second, the packet is delivered to the destination by switching to Subnetwork5 and taking the $Y1^*$ dimension.

2b) If the destination is on the west side of the source, the packet is delivered to the destination adaptively by moving in Subnetwork5 $X1^-, Y1^*$.

Figure 5.1 illustrates an example in which the destination is located on the east or west side of the source. First, suppose that source node 9 generates a packet for node 7 at the east side of the source. Since source and destination are on the same layer, either of the virtual channels 0 or 1 might be selected randomly. If virtual channel 0 is chosen, the channels of subnetwork1 are utilized which allows the packet to be delivered to the destination through one of the possible paths: {9, 10, 11, 7}, {9, 10,

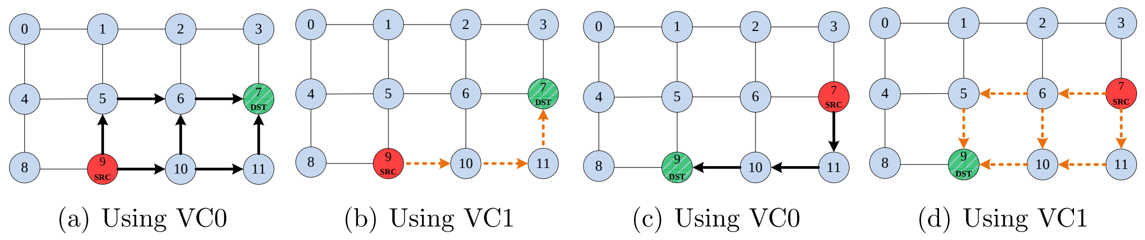


Figure 5.1: Routing example

6, 7} or {9, 5, 6, 7}, as shown in Figure 5.1(a). If the virtual channel 1 is selected as in Figure 5.1(b) , the packet is delivered to the destination by taking the channels of Subnetwork4 and then Subnetwork5 which enables the packet to take $X1^+$ before taking $Y1^+$, referring to the path {9, 10, 11, 7}.

Now lets us assume that the source node is node 7 and destination is at node 9 to cover the case where the destination is to the west of the source. In this example, taking $(X0, Y0)$ virtual channels result in deterministic routing as in Figure 5.1(c) while applying $(X1, Y1)$ virtual channels takes advantage of adaptive routing as shown in Figure 5.1(d) .

### 5.3.2 Source and destination are not on the same layer

A vertical transmission is necessary to deliver a packet to a destination that is not located on the same layer as the source. Therefore, the packet needs to be forwarded to the elevator in the source layer, transferred to the destination layer, and delivered from the elevator to the destination in the destination layer.

A random elevator is assigned to each packet upon its creation if the packet needs to move vertically. With this strategy, the traffic between any source-destination pair is distributed uniformly over the network. Compared to fixed elevator assignment, the random method improves the overall performance by not overwhelming any specific elevator. The advantage is much more significant if there are hot-spots in the network, which could cause bottlenecks if traffic is not distributed uniformly. Moreover, random elevator assignment improves the fault tolerance of the network. Although a smart deterministic selection can outperform random selection, such a scheme requires availability of global network knowledge at the cost of more complicated hardware. The algorithm delivers the packets to elevators using Subnetwork1 or/and Subnetwork2; then utilizes Subnetwork3 to forward the packet to the destination layer, and finally switches to Subnetwork4 or/and Subnetwork5 to deliver the packet to the destination.

Figure 5.2 illustrates an example in which the source node 0 delivers packets to the destination at node 21. Suppose that the elevator at node 6 is randomly selected for this transmission among the elevators located at nodes 6, 7 and 8. Since the elevator is to the east side of the source, the channels of Subnetwork1 are used to forward the packet to the elevator adaptively. Then, the channels of Subnetwork3 are applied to deliver the packet to the destination layer. Packets are ultimately delivered to the destination by using the channels of Subnetwork5 because the destination is located at the west of the elevator.

**Algorithm 4** Routing algorithm procedure

```
1: $X_s$, $Y_s$, $Z_s \leftarrow X$, $Y$, $Z$ coordinates of source router
2: $X_c$, $Y_c$, $Z_c \leftarrow X$, $Y$, $Z$ coordinates of current router
3: $X_d$, $Y_d$, $Z_d \leftarrow X$, $Y$, $Z$ coordinates of destination router
4: Sub1 $\leftarrow$ Subnetwork1$\{X0^+, Y0^*\}$
5: Sub2 $\leftarrow$ Subnetwork2$\{X0^-\}$
6: Sub3 $\leftarrow$ Subnetwork3$\{Z^*\}$
7: Sub4 $\leftarrow$ Subnetwork4$\{X1^+\}$
8: Sub5 $\leftarrow$ Subnetwork5$\{X1^-, Y1^*\}$
9: Route_VC0(a,b) (function for routing from a to b through VC0)
10: Route_VC1(a,b) (function for routing from a to b through VC1)
11:
12:
13: if ($Z_s = Z_d$) then
14:     Randomly chosen VC {0,1}
15:     if ($VC = 0$) then
16:         Route_VC0(S, D)
17:     else
18:         Route_VC1(S, D)
19:     end if
20: else
21:     Route_VC0(S, E)
22:     Vertical transmission to destination layer through Sub3
23:     Route_VC1(E, D)
24: end if
25:
```

**Algorithm 5** Routing function through specific virtual channels

```
function Route_VC0(a, b)
{
if ($X_a < X_b$) then
    Randomly choose channels from Sub1
else if ($X_a > X_b$) then
    $Y0^*$ submission from Sub1 then $X0^-$ from Sub2
else
    $Y0^*$ submission from Sub1
end if

return
}

function Route_VC1(a, b)
{
if ($X_a < X_b$) then
    $X1^+$ Submission from Sub4 then $Y1^*$ from Sub5
else if ($X_a > X_b$) then
    Randomly choose channels from Sub5
else
    $Y1^*$ submission from Sub5
end if

return
}
```

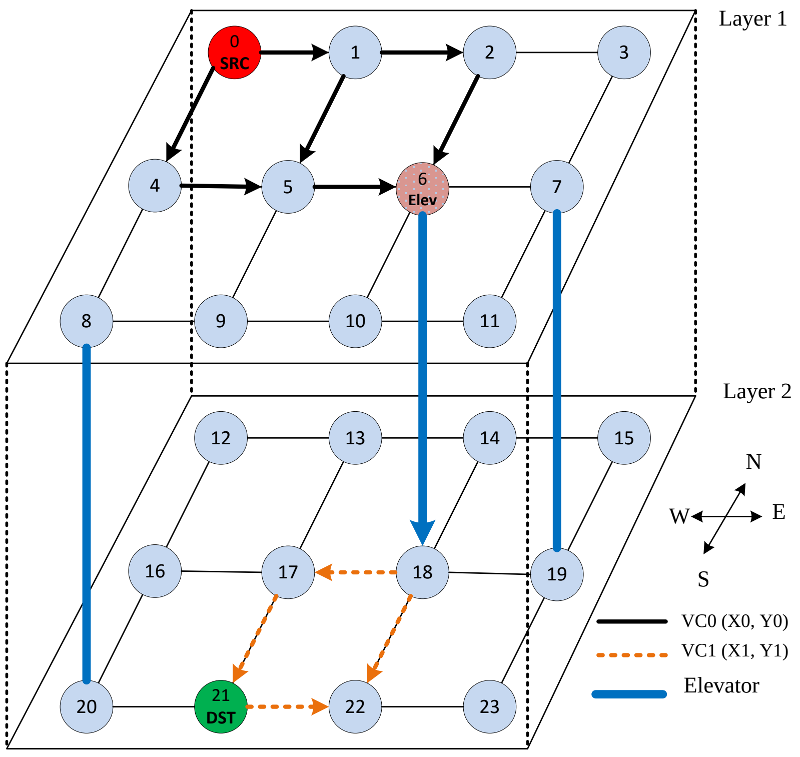


Figure 5.2: An example of destination in the other layer

## 5.4 Results and Discussion

### 5.4.1 Latency analysis

AccessNoxim simulator [76] is used for the simulation-based results. Noxim [77] (a cycle accurate simulator) and HotSpot [78] (providing architecture-level thermal model) are integrated in AccessNoxim. This co-simulator combines the network model, power model and thermal model of 3D-NoC.

A $4 \times 4 \times 4$ 3D-NoC is considered as the baseline architecture. The routers have two pipeline stages and the buffer depth equals 4-flit FIFOs in all the routers. Also, the packet is assigned a size between a minimum and maximum value of two and six flits, respectively. The latency results are reported for 100,000 cycles simulations with 10,000 cycles for warm-up.

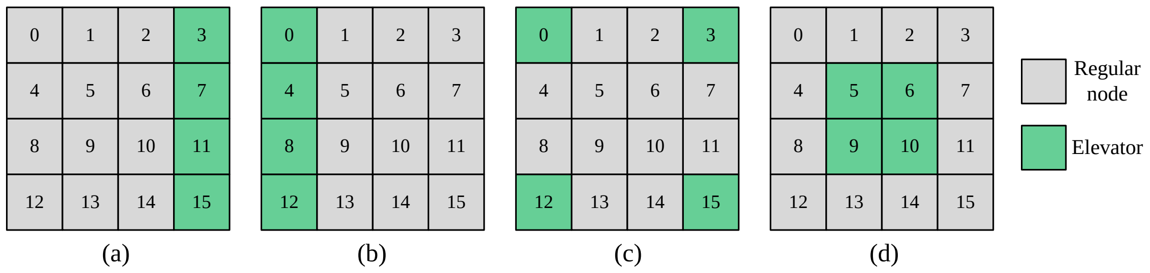


Figure 5.3: Different elevator configurations

We compare the efficiency of LEAD routing algorithm with three recently proposed routing algorithms named Elevator-first [65], Redelf [83] and ETW [84]. While Redelf has no virtual channels, ETW has only one extra virtual channel along the $Y$ dimension. Moreover, both Elevator-first and LEAD have two virtual channels along both $X$ and $Y$ dimensions. A variation of Redelf so called Redelfv2 [83] with two added virtual channels is considered in order to increase performance and make a fair comparison with LEAD and Elevator-first which employ two extra virtual channels. The routing algorithms are compared under various traffic patterns, number of elevators and elevator assignment mechanisms. Different TSV configurations are illustrated in Figure 5.3. The applied traffic patterns span both synthetic and real traffics.

Horizontal lines in the line graphs represent the packet injection rate in every router (packet/cycle/node) and the vertical lines reports latency (cycles). While this section provides intuitive reasons justifying relative performance of LEAD versus other algorithms, more solid mathematical verification based on queuing theory is presented in the next section.

### Performance under synthetic traffics

Figures 5.4(a), 5.4(b) and 5.4(c) illustrate the latency comparisons under random, transpose and shuffle traffic for the elevators located at the east-most column, i.e.

the pattern of Figure 5.3(a). Under random traffic (Figure 5.4(a)), Redelfv2 outperforms LEAD by a small margin, while both LEAD and Redelfv2 perform relatively better than Elevator-first and ETW. Under Transpose traffic (Figure 5.4(b)), Redelfv2 performs better than all other algorithms. This can be attributed to the traffic being distributed more evenly as a result of the rules imposed by Redelfv2 under this specific TSV configuration and traffic. However, the applicability of Redelfv2 is limited to TSV configurations with a south-east corner elevator. This limitation also threatens the fault tolerance of the system when there is no healthy elevator at south-east corner of the network. Under shuffle traffic (Figure 5.4(c)), LEAD provides the best performance compared to the other three routing algorithms. Also, Elevator-first and Redelfv2 provide nearly the same performance. The performance improvement of LEAD is due to taking advantage of adaptivity in moving toward east and distributing the traffic more uniformly compared to the other algorithms.

Figures 5.5 through 5.7 compare the performance of LEAD and Elevator-first. ETW and Redelfv2 have the limiting requirement of the presence of an east most column TSV and a south-east corner TSV respectively. Both ETW and Redelfv2 create these rules as a byproduct of removing virtual channels and providing lower communication cost.

Figures 5.5(a), 5.5(b) and 5.5(c) represent the latency results for the aforementioned traffics where elevators are located similar to Figure 5.3(b). Moreover, Figure 5.5(a) includes the performance comparison for the hot-spot traffic. For this TSV configuration, the same trend as the random traffic is observed but the network saturates at a lower rate since specific nodes receive more load compared to uniform traffic. The superiority of LEAD over Elevator-first is due to Elevator-first taking the $X$ dimension first by applying the $XY$ algorithm and thus increasing the traffic on the west-most elevators. Therefore, these elevators are applied both as intermediate node to forward the packet to the proper elevator and as elevator to the other layers. LEAD, on the other hand, performs a deterministic routing in this configuration by using the $Y$ dimension first and elevators are only responsible for transmitting packets to the other layers.

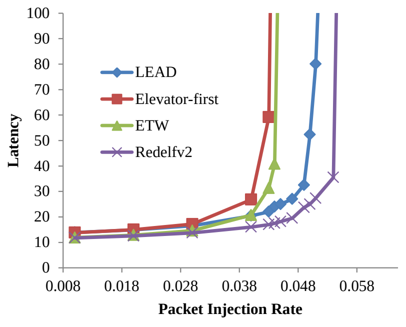
Figures 5.6(a), 5.6(b) and 5.6(c) report the performance with the TSV configuration of Figure 5.3(c). LEAD and Elevator-first perform relatively similar when elevators are located at four corners of the network. Based on Figure 5.6(b), Elevator-first performs slightly better than LEAD because in this configuration the impact of TSV selection on the performance is minimized.

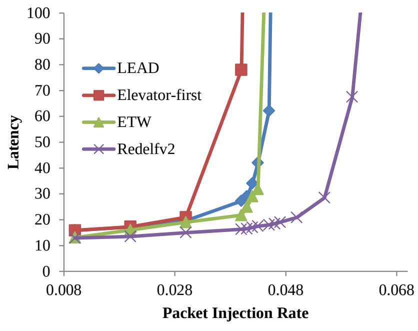
Figures 5.7(a), 5.7(b) and 5.7(c) illustrate the latency result under the TSV configuration of Figure 5.3(d). LEAD marginally outperforms Elevator-first for the same reasons discussed in justification of Figures 5.4 and 5.5, where same column elevators lead to better performance.

Figure 5.8 illustrates the latency comparison for two more traffic patterns named bit-reversal and butterfly for elevators located at the east-most column. LEAD outperforms Elevator-first for two cases. Figure 5.8(b) shows that the network is saturated at higher injection rates for butterfly traffic, since many sources target themselves as their destinations according to this traffic pattern.

### Performance under real traffics

Besides synthetic traffic patterns, a set of application benchmarks including PARSEC [80] and SPLASH2 [79] are used for performance evaluation. Figures 5.9(a) and 5.9(b) compare the latency of the west-most and cornered elevators illustrated in Figure 5.3. The reported results are based on a $4 \times 4 \times 4$ network in which buffer depth equals four and the packet size is randomly chosen between two to six flits. LEAD provides


(a) Random traffic


(b) Transpose traffic

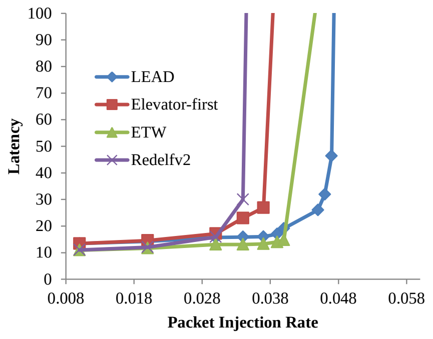
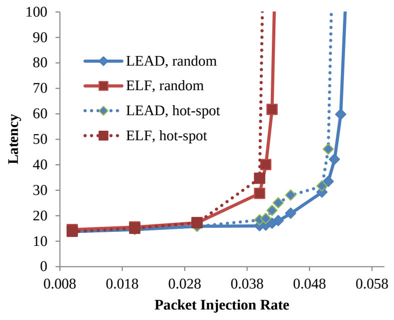
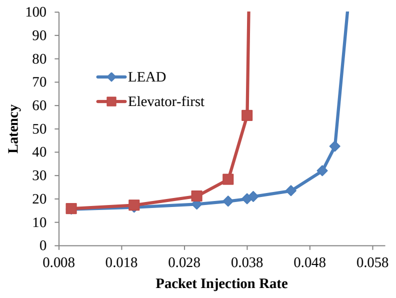

(c) Shuffle traffic

Figure 5.4: Performance comparison for east-most elevators


(a) Random and hot-spot traffic


(b) Transpose traffic

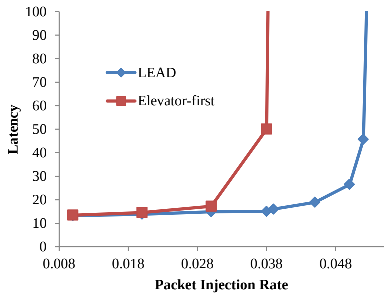
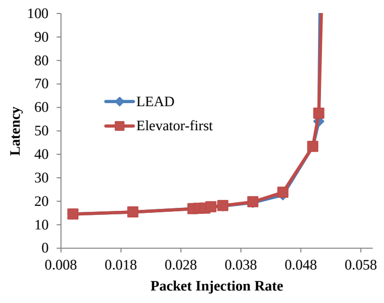
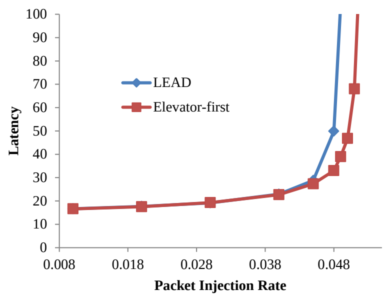

(c) Shuffle traffic

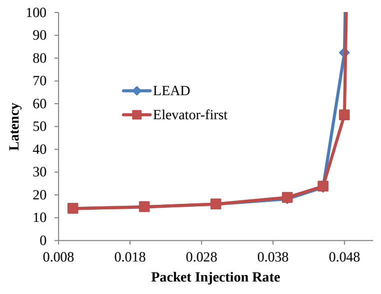
Figure 5.5: Performance comparison for west-most elevators



(a) Random traffic

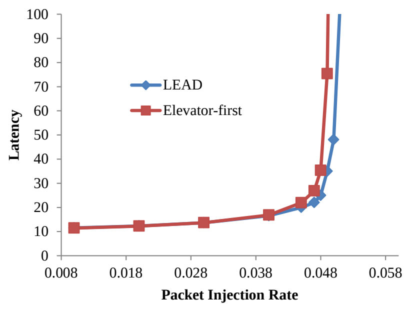


(b) Transpose traffic

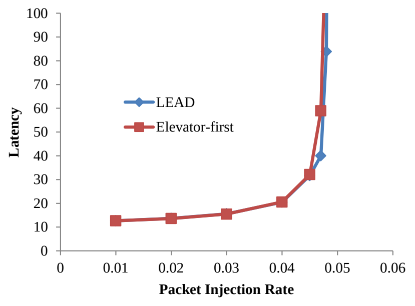


(c) Shuffle traffic

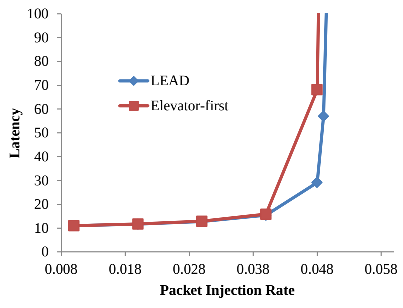
Figure 5.6: Performance comparison for cornered elevators



(a) Random traffic

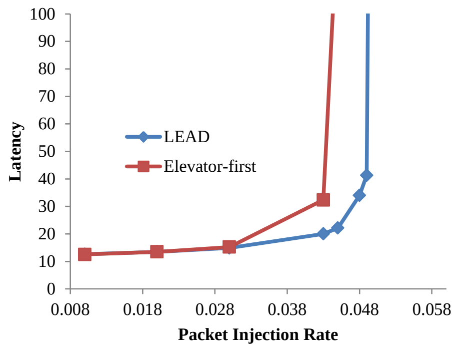


(b) Transpose traffic

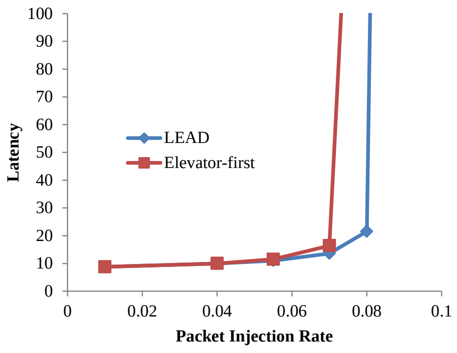


(c) Shuffle traffic

Figure 5.7: Performance comparison for centered elevators

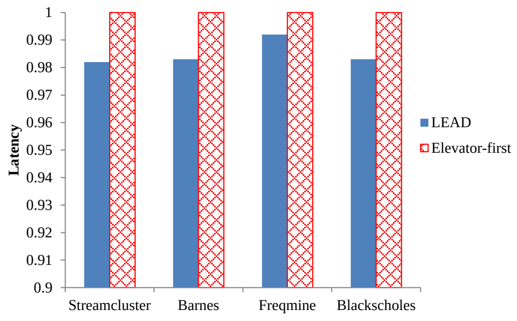


(a) Bit-reversal traffic

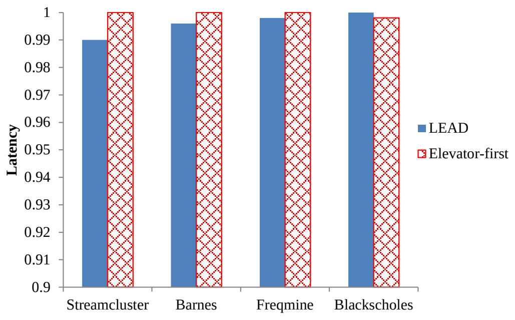


(b) Butterfly traffic

Figure 5.8: Performance comparison for east-most elevators under bit-reversal and butterfly traffic



(a) Westmost elevators



(b) Cornered elevators

Figure 5.9: Performance comparison under real traffic

noticeable performance improvement especially when elevators are located at the west-most column. For corner located elevators, LEAD performs slightly better than Elevator-first.

### Performance under different number of elevators

To illustrate the effect of the number of elevators on performance, Figure 5.10 provides the results for a fully connected 3D-NoC and 3D-NoC with 50% and 75% reduction in the number of TSVs under transpose traffic. The fully connected 3D-NoC provides the best performance as expected. By decreasing the number of TSVs, the performance degrades accordingly, since more traffic is directed toward TSVs. In a fully connected

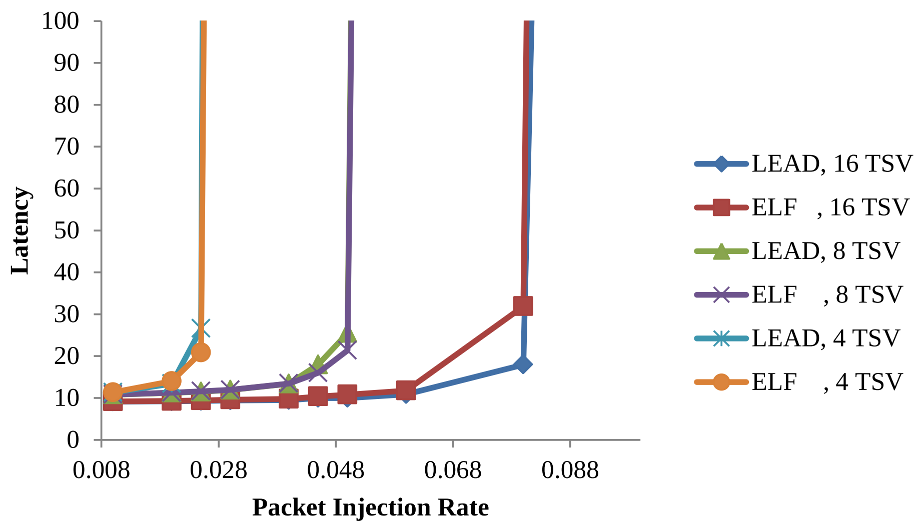


Figure 5.10: Performance under different number of elevators

3D-NoC, every router has an elevator to forward the packet to the destination layer. In a 3D-NoC with 8 TSVs, every other router has an elevator. In a 3D-NoC with 4 TSVs, TSVs are located at the center of the network. To make a fair comparison, elevators are chosen using the minimum hop count scheme.

### Performance under different elevator assignments

To investigate how elevator assignment affects the performance, Figure 5.11 compares three mechanisms of elevator assignment under random traffic in a $4 \times 4 \times 4$ 3D-NoC. The mechanisms are as follows: random elevator, the closest elevator to the source and the elevator that minimizes the hop count between the source and destination called *(MHpCnt)*. All previous results are based on random elevator assignment for every source-destination pair. Figure 5.11(a) compares the performance of LEAD versus Elevator-first under random and *(MHpCnt)* elevator assignments for the centered elevators. According to Figure 5.11(a), LEAD and Elevator-first perform considerably better compared to *(MHpCnt)* under the random elevator assignment, since random elevator assignment distributes traffic on the elevators and the network,

| | Percentage of elevator usage | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Random assignment | | | | Minimum hop count | | | |
| Routing algorithm | Node 5 | Node 6 | Node 9 | Node 10 | Node 5 | Node 6 | Node 9 | Node 10 |
| LEAD | 25.1 | 25 | 25 | 24.9 | 80 | 13 | 5.76 | 1.24 |
| Elevator-first | 25.1 | 24.9 | 25 | 25 | 76.35 | 4.37 | 17.8 | 1.48 |

Table 5.2: Distribution of elevator usage for centered elevators

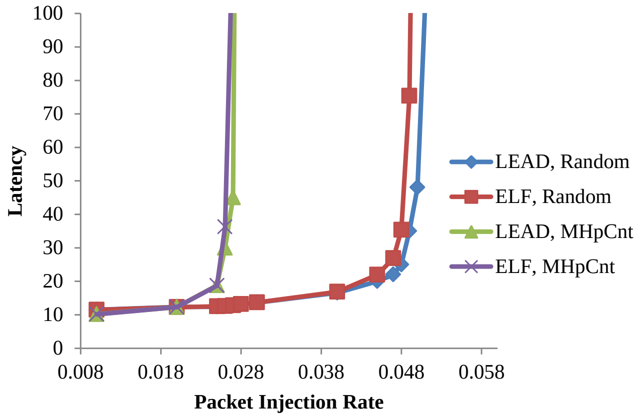
| | Percentage of elevator usage | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Random assignment | | | | Minimum hop count | | | |
| Routing algorithm | Node 0 | Node 3 | Node 12 | Node 15 | Node 0 | Node 3 | Node 12 | Node 15 |
| LEAD | 25.11 | 24.91 | 25.02 | 24.96 | 71.8 | 13.2 | 10.6 | 4.4 |

| | Percentage of elevator usage | | | |
|---|---|---|---|---|
| | Closest assignment | | | |
| Routing algorithm | Node 0 | Node 3 | Node 12 | Node 15 |
| LEAD | 24.6 | 25.43 | 24.29 | 25.68 |

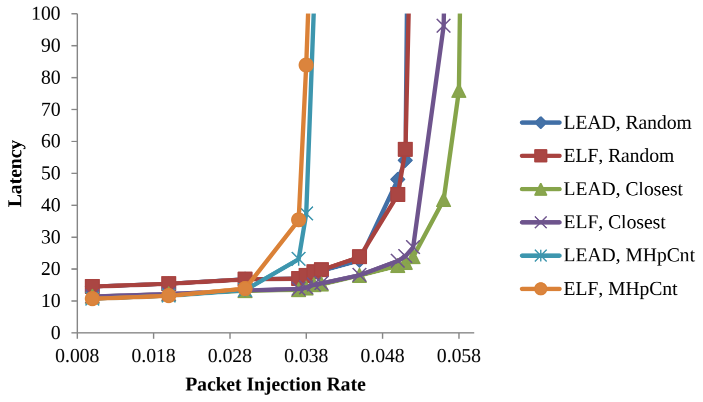
Table 5.3: Distribution of elevator usage for cornered elevators

thus the saturation point extends. Table 5.2 lists the percentage of times that different elevators are used for different mechanisms. Random assignment distributes the traffic symmetrically in the network while the minimum hop count forwards the majority of the traffic to a specific elevator.

Figure 5.11(b) compares the performance of the three different elevator assignments for the cornered elevators. According to this figure, when the closest elevator to the source is chosen to transfer packets to the destination layer, LEAD and Elevator-first outperform both the random assignment and elevator with minimum hop count. Table 5.3 summarizes the percentage of elevator assignments for the three cases. As it is clear, minimum hop count assignment targets specific elevators, and thus decreases the performance dramatically.



(a) Centered elevators



(b) Cornered elevators

Figure 5.11: Performance comparison for different elevator assignment mechanisms

### 5.4.2 Temperature Distribution

Thermal distribution of LEAD and Elevator-first are compared using a traffic-thermal and mutual coupling co-simulation platform [85]. The physical floor-plans and power traces based on Intel 80-core chip are used as the inputs of the thermal simulation.

To compare the thermal distribution of the two routing algorithms, a $4 \times 4 \times 4$ NoC under two routing algorithms are simulated for 3 million cycles with 10000 cycles for warm-up. Figure 5.12 illustrates the temperature distribution for the west-most elevator configuration and the packet injection rate of 0.042 and under random traffic for LEAD and Elevator-first. According to the figure, there are eight nodes in Elevator-first hotter than $110°C$ while these temperature are not found in LEAD. As expected, the TSV-located nodes are among these very hot nodes which can increase the likelihood of fault on TSVs. It is noteworthy that the same behavior is observed under other simulation configurations.

Furthermore, Table 5.4 reports the power consumption of LEAD and Elevator-first routing algorithms under random traffic for the west-most TSV configuration. AccessNoxim accumulates energy upon flit reception and transmission at every router. LEAD and Elevator-first consume nearly the same amount of energy for these cases. The two algorithms have the same number of virtual channels along the $X$ and $Y$ dimensions.

## 5.5 Queuing Theory and Analytical Model

The overall latency of a routing algorithm is a coarse performance metric used to gauge the merit of the algorithm. The overall latency is typically calculated as the average of end-to-end latencies experienced by all packets in the network. Although

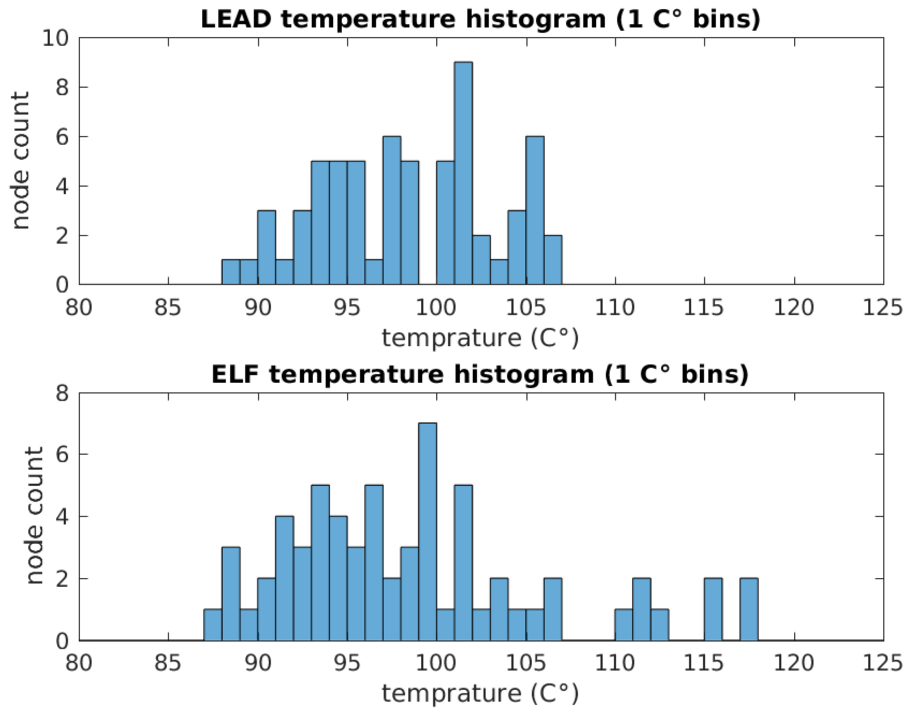


Figure 5.12: Temperature distribution in LEAD vs Elf

| | Random Traffic | |
|---|---|---|
| | Average power (Whole network) ($\mu J/cycle$) | Average power (per router) ($nJ/cycle$) |
| Routing algorithm | Westmost elevators | Westmost elevators |
| LEAD | 5.0 | 78.1 |
| Elevator-first | 5.0 | 78.1 |

Table 5.4: Average power consumption

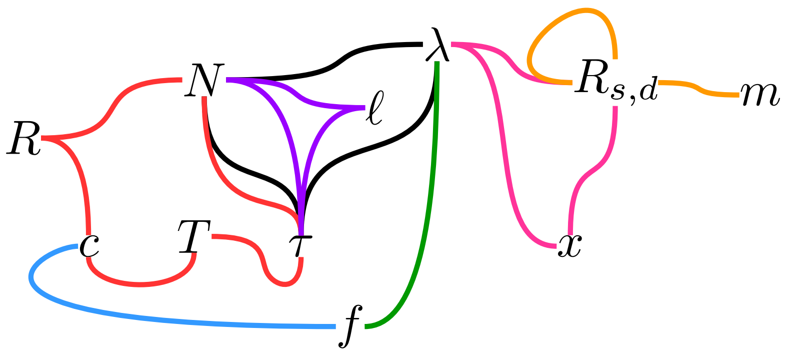


Figure 5.13: Outline of analytical framework and the coupling of variables.

the average case analysis is not sufficient for some purposes (for example, calculating the fraction of times a specific application is serviced properly), it is much simpler to handle, provides useful insight into design bottlenecks, and can be used to compare different algorithms/architectures.

The end-to-end latency experienced by packets between a specific source-destination pair depends on the latency imposed by the individual links of the path traversed. Consequently, the individual buffer latencies should be estimated first. In wormhole routing, the latency experienced by a header flit consists of buffer waiting time (from the time flit enters the buffer until it reaches the router) plus residual service time, the latter of which depends on router architecture. The buffer waiting time is naturally calculated with queuing theory.

The relationship between involved variables is outlined in Figure 5.13. Each set of lines with a common color represents a mathematical relationship exploited in one of the subsections that follow. Importantly, we develop a tractable method to calculate the required variables for *adaptive* algorithms from the knowledge of network topology and routing algorithm. Finally, the results of the analytic model are compared with simulation under various scenarios to verify their accuracy and examine any possible shortcomings.

### 5.5.1 Analysis Framework

#### Service Time

Under wormhole-switching network with low packet injection rates the service time of a packet at the buffer head (i.e. after waiting in the buffer) is given by:

$$T = H_S + \frac{S}{W} \tag{5.1}$$

in unit of cycles, where $H_S$ is the router service time (in cycles) seen by a header flit at the buffer head (which depends on the router architecture), $S$ is the packet size in bits, and $W$ is the channel bandwidth in bit per cycle. Note that in wormhole-switching network, once the header flit is serviced, the trailing flits follow in a pipelined fashion. If there is no head-of-line blocking in the following routers, the trailing flits follow at a maximum uniform rate. This condition holds under low traffic rate, and consequently there is no need to consider the delay added by the head-of-line blocking.

#### Average Number of Packets in Buffer

For an input buffered router $r$ with $P$ channels, denote the average number of packets in buffer $i$ by $N_i^r$. Let us assume that the header flit arrival rate on the channel $i$ follows a Poisson distribution with the mean $\lambda_i^r$. Then the following equilibrium equation relates $\tau_i^r$, the average waiting time in a queue for an incoming packet, and the average number of packets in buffer:

$$\lambda_i^r = \frac{N_i^r}{\tau_i^r} \tag{5.2}$$

The average waiting time $\tau_i^r$ is composed of 1) service time of packets already waiting in the same buffer 2) packet waiting time in other buffers of the same router that are served before the target packet, and 3) residual service time [86], $R$, seen by the target packet. Mathematically, these components are written as:

$$\tau_i^r = TN_i^r + T \sum_{k=1,k\neq i}^{P} c_{i,k}^r N_k^r + R \tag{5.3}$$

where $c_{i,k}^r$ terms, called 'contention probabilities', represent the probability that a header flit at buffer $k$ of router $r$ is serviced before a header flit at buffer $i$ of the same router, assuming that both headers are present at the buffer head during the decision cycle. The contention probabilities can be calculated for different scheduling policies (priority, round robin, etc.).

The last two equations can be combined to remove variable $\tau_i^r$:

$$\frac{N_i^r}{\lambda_i^r} = TN_i^r + T \sum_{k=1,k\neq i}^{P} c_{i,k}^r N_k^r + R \tag{5.4}$$

which is *one* equation for $P$ unknowns $N_1^r$, $N_2^r$, $\cdots$, $N_P^r$. A similar equation can be derived for all $P$ input ports, leading to a linear system of $P$ equations for $P$ unknowns. After some algebraic manipulation and vectorization , the following equation is obtained:

$$(\boldsymbol{I} - T\boldsymbol{\Lambda}^r\boldsymbol{C}^r)\boldsymbol{N}^r = \boldsymbol{\Lambda}^r\boldsymbol{R} \tag{5.5}$$

where

$$\boldsymbol{N}^r = \begin{bmatrix} N_1^r & N_2^r & \cdots & N_P^r \end{bmatrix}^T \tag{5.6}$$

$$\boldsymbol{\Lambda} = \mathrm{diag}(\lambda_1^r, \cdots, \lambda_P^r) \tag{5.7}$$

$$(\boldsymbol{C}^r)^T = \begin{bmatrix} (\boldsymbol{C}_1^r)^T & (\boldsymbol{C}_2^r)^T & \cdots & (\boldsymbol{C}_P^r)^T \end{bmatrix} \tag{5.8}$$

$$\boldsymbol{C}_i^r = \begin{bmatrix} c_{i,1}^r & c_{i,2}^r & \cdots & c_{i,P}^r \end{bmatrix} \tag{5.9}$$

$$\boldsymbol{R}^T = \begin{bmatrix} R & R & \cdots & R \end{bmatrix} \tag{5.10}$$

with the solution:

$$\boldsymbol{N}^r = (\boldsymbol{I} - T\boldsymbol{\Lambda}^r\boldsymbol{C}^r)^{-1}\boldsymbol{\Lambda}^r\boldsymbol{R} \tag{5.11}$$

In summary, given traffic arrival rates $\boldsymbol{\Lambda}$ on all ports of a router, the contention probabilities $\boldsymbol{C}$, and the residual service time $\boldsymbol{R}$, the average number of packets $\boldsymbol{N}$ in each input buffer of the router can be calculated by solving the linear system of equations in Eq. 5.11.

## Contention Probabilities

To calculate the contention probabilities under round robin policy, let $f_{j|i}^r$ represents the probability that a packet exits port $j$ (of router $r$) given that it is entering through port $i$. A contention happens if two packets from different input channels need to use the same output channel. Assuming statistical independence, the contention probability $c_{i,j}^r$ $(i \neq j)$ is estimated by:

$$c_{i,j}^r = \sum_{k=1}^{P} f_{k|i}^r f_{k|j}^r \tag{5.12}$$

The $f$ terms are calculated from the knowledge of packet arrival rates on different ports. Specifically, if $\lambda^r_{i,j}$ denotes the arrival rate of packets that enter through the port $i$ and exit through the port $j$ of the router $r$ , then:

$$f^r_{j|i} = \frac{\lambda^r_{i,j}}{\lambda^r_i} \tag{5.13}$$

### Injection Rate and Port Utilization Probability

The $\lambda^r_{i,j}$ parameters can be calculated from the routing algorithm and the source-destination traffic pattern. Denote the packet generation rate from source $s$ to destination $d$ by $x_{s,d}$. Each packet generated at source $s$ and destined to destination $d$ can potentially enter through any input port $i$ and exit from any output port $j$ of any router $r$ with some probability. Let $R^r_{s,d}(i,j)$ denote this probability. Then, $\mathrm{T}\lambda^r_{i,j}$ is given by:

$$\lambda^r_{i,j} = \sum_{s,d} x_{s,d} R^r_{s,d}(i,j) \tag{5.14}$$

To calculate $R$, let's consider a specific source-destination pair $(s,d)$. First, the behavior of router $r$ is modeled by a $P \times P$ relaying matrix $\boldsymbol{M}^r_{s,d}$. The $(i,j)$ element of the matrix, $m^r_{s,d}(i,j)$, is the probability that a packet generated by $s$ toward $d$ and entering on port $i$ of router $r$ exits through port $j$. This transition matrix can be explicitly derived from the the routing algorithm.

In the first scenario, assume that an input port $i$ of the router $r$ is connected to exactly one output port of another router, but not to the local port that is connected to PE. Let's denote this neighbor router by $N(r,i)$ and the corresponding output port by $O(r,i)$. Next, the $R$ variable of neighbors can be related to each other recursively by:

$$R^r_{s,d}(i,j) = m^r_{s,d}(i,j) \times \sum_{k=1}^{P} R^{N(r,i)}_{s,d}(k, O(r,i)) \tag{5.15}$$

Simply stated, the target input port $i$ of router $r$ is connected to the port $O(r,i)$ of router $N(r,i)$. The probability of a packet passing from port $i$ to port $j$ is the corresponding forwarding probability (first term) times the probability of a packet exiting through $O(r,i)$ (second term).

In the second scenario, assume that the input port $i$ of router $r$ is connected to the local output port, called $\mathcal{P}_s$. In this simple case, the utilization probability of the input port is equal to 1:

$$R^r_{s,d}(i,j) = m^r_{s,d}(i,j) \tag{5.16}$$

The last two equations provide a set of $P \times n_r$ linear equations for the same number of unknowns, where $n_r$ is the total number of routers. Although any general methods can be used to solve the system, the dependence of each variable on only $P$ other variables (rather than all $P \times n_r - 1$ other variables) significantly simplifies the solution. On top of that, for simple enough routing algorithms, one could start from the source node, and calculate the $R$ values for all immediately connected nodes. The same procedure is applied to each new node until all $R$ values are calculated. In other words, the probability is calculated as *flowing* and distributing away from source to destination.

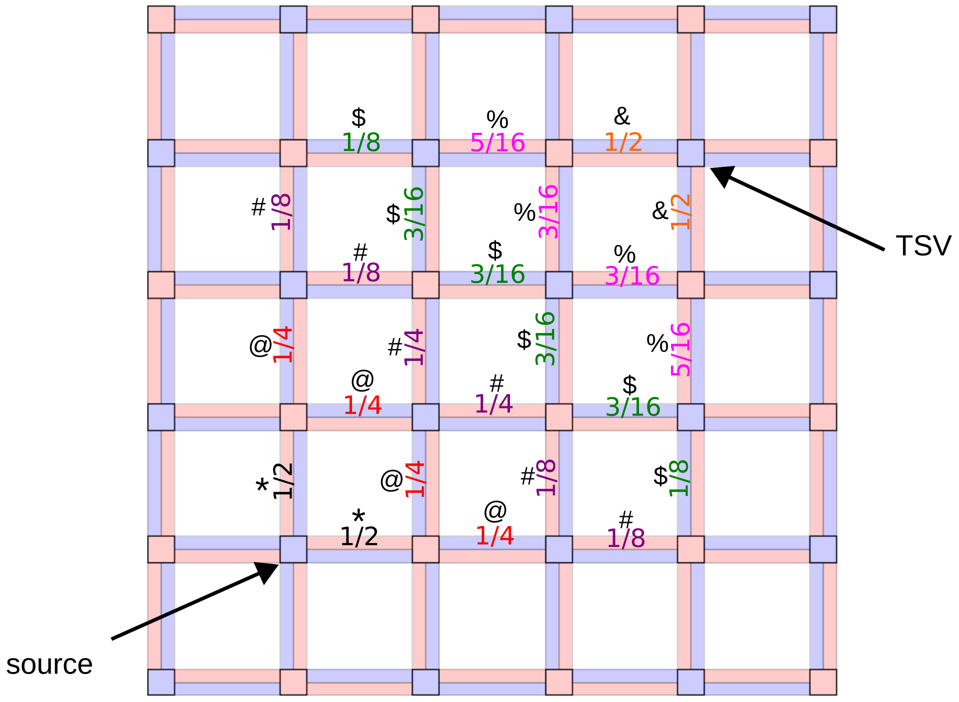


Figure 5.14: The flow method for calculating port usage probabilities

To illustrate the flow method, consider the $6 \times 6$ network of Figure 5.14. Each central square represents a router, and the same-color immediate long hands represent the input buffers of that router. The numbers on input buffers represent the probability of that buffer being used. Suppose that node (2,2) decides to transmit a packet to TSV at (5,5). Based on the proposed routing algorithm, the network will use $VC0$ and subnetwork1. The input buffer of router (2,2) connected to the PE (the PE and the corresponding input buffer of router not shown here) is used with a probability of 1. Next, the $*$ probabilities are calculated, @ calculated from $*$, # from @, \$ from #, and so on.

In general, $R$ values are completely specified from the knowledge of routing algorithm and network geometry. The definition of $R$ values are by no means bound to the specific geometry and routing algorithms presented in this dissertation. An independent module could calculate these R values and provide them as inputs to the queuing analysis framework. Once the probabilities are calculated, the traffic pattern $x_{s,d}$ can be added to Eq. 5.14 to calculate the port-to-port injection rates. Once $\lambda_{i,j}^r$ values are calculated, all other queuing parameters are found through the discussed equations.

As an example, consider a $6 \times 6 \times 4$ network with 4 TSVs at the corners carrying a random traffic using the proposed routing algorithm. Figure 5.15 illustrates the $\lambda_{i,j}^r$ values (in log scale) for the second layer of the network. In this $11 \times 11$ representation, each square represents a $6 \times 6$ heat-map of the traffic rate injected from port specified by the horizontal index into the port specified by the vertical index. A black color represents no traffic, while a white color represents maximum traffic among all.

### Buffer Waiting Time

To relate the overall latency to the calculated loads, we start by using Little's theorem to relate waiting time, injection rate, and number of packets in queue:

$$\tau_i^r = N_i^r / \lambda_i^r \tag{5.17}$$

$\tau_i^r$ is the average waiting time of packets in an input buffer $i$ of a router $r$. Figure 5.16 shows an example of the calculated waiting times (in log scale) for the same $6 \times 6 \times 4$ network using LEAD under a uniform random traffic pattern with the waiting times ranging from 1 cycle to 10 cycles for service time $T = 10$ and residual time $R = 1$.

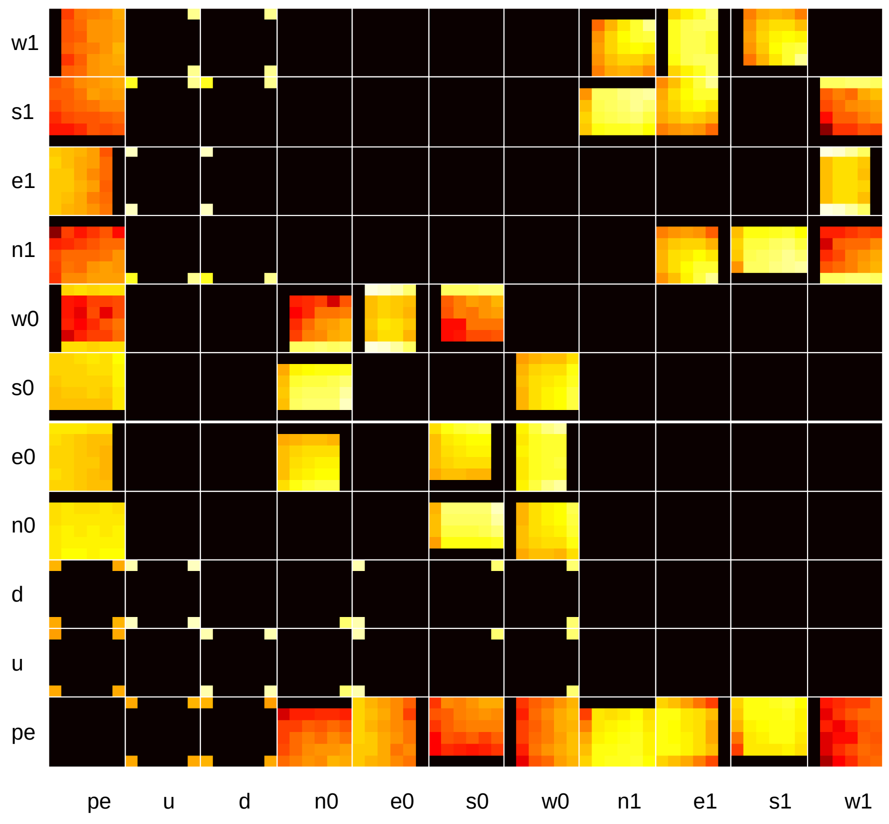


Figure 5.15: $\lambda^r_{i,j}$ for all ports of 2nd layer of a $6 \times 6 \times 4$ network with corner TSVs under random traffic. Input and output port indices $i$ and $j \in$ {pe, u, d. n0, e0, s0, w0, n1, e1, s1, w1}

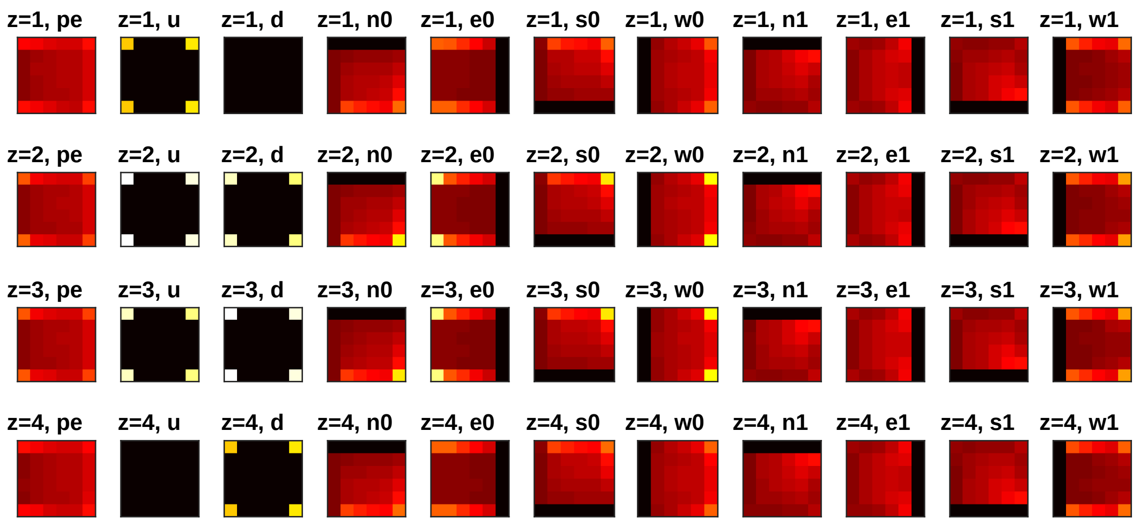


Figure 5.16: $\tau^r_i$ values (buffer waiting time). Each rectangle is $6 \times 6$, and the corresponding level is indicated above each rectangle with letter $z$. The port index is also included at the top of each box.

**End-to-End Latency**

Once individual buffer latencies (waiting times) are calculated, the end-to-end latency of a packet generated at router $s$ and destined to the router $d$ is found by adding the individual latencies of the buffers traversed by the packet. In adaptive routing, however, the packet may take different paths with different probabilities. Denote the set of paths traversed by packets generated from $s$ to $d$ by $\Pi_{s,d}$. Also, let $B_{s,d}(k)$ represents the set of input buffers traversed by the $k^{th}$ member path $\pi_{s,d}(k) \in \Pi_{s,d}$. If $p_{s,d}(k)$ represents the probability of path $\pi_{s,d}(k)$ being used, the end-to-end latency for a single source-destination pair $(s, d)$ is found by:

$$L_{s,d} = W_s + \frac{S}{W} + \sum_k p_{s,d}(k) \sum_{(r,b)\in B_{s,d}(k)} (H_s + \tau_b^r) \tag{5.18}$$

where $L_{s,d}$ is the average latency between source $s$ and destination $d$, $W_s$ is the header service time at source, $S$ is average packet length, $W$ is the bandwidth, $H_s$ is header service time at each buffer, and $\tau_b^r$ is the average waiting time at buffer $b$ of router $r$. Unfortunately, enumeration of all possible paths between all possible source and destination pairs is computationally inefficient. To resolve this issue, consider Figure 5.14 again, and assume that the bottom left node (1,1) is sending packets to top right node (6,6). From the point of view of (6,6), a fraction of the $N$ packets arrives from west ($N_w$), and the rest from south ($N_s = N - N_w$). The $N_w$ packets arriving from west have experienced a random delay right before entering the west input buffer of (6,6), where the randomness comes from both the path taken and the system status during the packet traversal. Denote the average of these delays by $\ell^{(5,6)}(e)$ (average delay experienced by packets (from (1,1) to (6,6)) at the moment of being ejected from the east port of (5,6)). Then, the average added delay until each such packet header is serviced by (6,6) is:

$$\ell^{(6,6)}(pe) = \ell(5,6)(e) + \tau_w^{(6,6)} \tag{5.19}$$

where buffer waiting times $\tau$ can be calculated using Equation 5.17. This equation only holds for the packets arriving from west. If we consider all arriving packets, from both south and west, a correct averaging results in:

$$\ell^{(6,6)}(pe) = \frac{N_w}{N}[\ell^{(5,6)}(e) + \tau_w^{(6,6)}] + \frac{N_s}{N}[\ell^{(6,5)}(n) + \tau_s^{(6,6)}] \tag{5.20}$$

This suggests a recursive relationship between variables $\ell$. Similar to the flow method used to calculate the probabilities, $\ell$ values can be calculated by starting from the source and flowing toward the destination. Also, note that the relative occurrence terms such as $N_w/N$ are calculated directly from the previously calculated probabilities $R$.

In general, if $\ell^r_{s,d}(j)$ represents the average delay experienced by packets from $s$ to $d$ at the moment of being ejected into output port $j$ of router $r$, we can write the following recursive equation:

$$\ell^r_{s,d}(j) = \sum_{i=1}^{P} \frac{N_i^r}{\sum_k N_k^r} \left[\tau_i^r + \ell_{s,d}^{N(r,i)}(O(r,i))\right] \tag{5.21}$$

It is observed that, similar to $R^r_{s,d}(i,j)$, $\ell^r_{s,d}(j)$ of neighbor ports are related to each other recursively. Consequently, the same flow method used to calculate $R$ values can be used to calculate $\ell$ values.

### Overall Network Latency

Finally, once $L_{s,d}$ values are calculated, the overall latency of the network (considering all source destination pairs) is found by the following waited averaging:

$$L = \sum_{s,d} \frac{x_{s,d}}{\sum_{s',d'} x_{s',d'}} L_{s,d} \tag{5.22}$$

### Overview of the Procedure

To summarize, the following sequence of calculations is performed to find the network overall latency:

1. Using the flow method, port utilization probabilities, $R^r_{s,d}(i,j)$, are calculated from the knowledge of network topology and routing algorithm (Section 5.5.1).
2. $R$ values and traffic injection rates, $x_{s,d}$, are combined to calculated buffer injection rates $\lambda^r_{i,j}$ (Section 5.5.1).
3. Contention probabilities are calculated from the knowledge of injection rate into and out of router ports. (Section 5.5.1).
4. Contention probabilities and injection rates are combined to find the average number of packets in buffers (Section 5.5.1).
5. Buffer waiting times are calculated from average number of packets and injection rates (Section 5.5.1).
6. End-to-end latencies for each source destination pair are calculated using the flow method (Section 5.5.1).
7. Overall network latency is found by averaging single source destination pair

end-to-end latencies, with a weighting determined by traffic injection rates $x_{s,d}$ (Section 5.5.1).

### Applicability to Generic Topologies

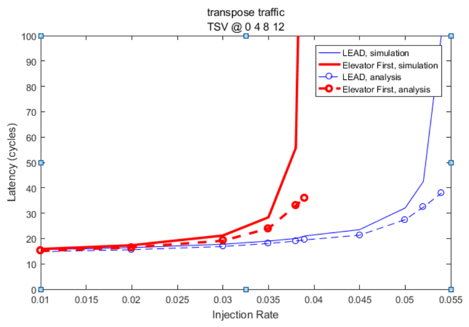
One of the advantages of our formulation is applicability to general topologies. To see why, note that all derivations before Section 5.5.1 are local equations holding for a single router, and thus will not be affected by the 'global' picture of the network, including topology. On the other hand, in Section 5.1.4, we have introduced the quantity $R^r_{s,d}(i,j)$ which quantifies the probability that for an $s \to d$ communication, port $i$ and $j$ of router $r$ will be used as input and output ports, respectively. As mentioned previously, $R$ will depend on network *geometry* and routing algorithm. Given the endless variety in the design of routing algorithms and geometries, it is prohibitive to formulate $R$ for all combinations. Consequently, we assume that $R$ is available as an input. Designers of other geometries/routing algorithms can use the definition of $R$ to calculate it and then feed it into the proposed queuing based framework.

## 5.5.2 Analysis vs. Simulation

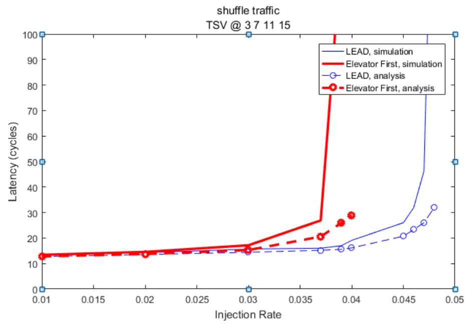
In this section, the simulation and analytical results are compared to verify the analytical model and cross check simulation results. Figure 5.17 shows a comparison of analytical and simulation results under four different scenarios using different routing algorithms. In Figure 5.17(a), the latencies of LEAD and Elevator-first, under random traffic with corner-located TSVs as in Figure 5.3(c), is repeated using the analytical framework and then compared with simulation results. The remaining three figures follow the same goal for other configurations explained in the captions. The following observations are consistent in all figures:

- For a given routing algorithm, the latencies reported by simulation and analysis agree very well within an error margin of 5% for the low injection rate zone.
- Although only significant under higher injection rates, the analytical latencies generally underestimate the actual latency reported by simulation. This can be contributed to the model ignoring head of line blocking.
- The accuracy of analytical latency drops as injection rate increases. This is an expected result since the model is formulated under the assumption of low injection rates.
- Even though the accuracy drops with increasing injection rate, the analytical model reports valid *relative results* at all injection rates. For example, in Figure 5.17(c), it is observed that analytical results verify the same saturation point superiority of LEAD over Elevator-first as reported by the simulation. Consequently, the analytical framework can still be used to compare the performance of different routing algorithms for all injection rates.

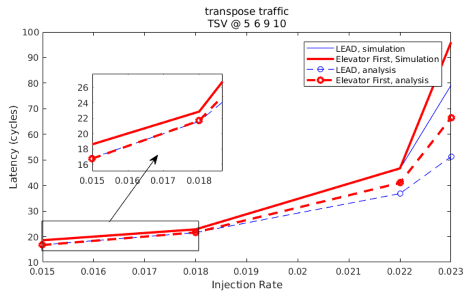
In summary, analytical results show very good accuracy at low injection rates and also provide meaningful relative performance measures for all injection rates. The saturation points, however, cannot be accurately estimated since the analysis is built upon the assumption of low injection rates. If the final goal is not to estimate saturation points accurately, the analytical framework can be used to calculate performance measures, either relatively or individually, in a fraction of the time consumed by simulation. Finally, although accurate estimation of saturation point is not possible with the analytical model, it is possible to find the vicinity of true saturation point and then fine tune with simulation.



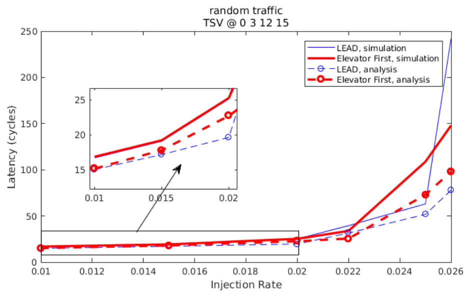
(a) LEAD and Elevator-first, transpose traffic, Westmost TSVs



(b) LEAD and Elevator-first, shuffle traffic, Eastmost TSVs



(c) LEAD and Elevator-first, transpose traffic, Centered TSVs



(d) LEAD and Elevator-first, random traffic, Cornered TSVs

Figure 5.17

# Chapter 6

# Future Work

3D-NoC architectures are considered necessary for future systems following the trend of increasing the number of cores . The increase in the number of cores on 2D planes and the ensuing increase in the latency and power consumption encourage architectures in which layers of 2D-NoC are stacked on top of each other and communicating through TSVs. However, the large bonding area of TSVs as compared to horizontal wires along with the low yield of TSV fabrication process promote the idea of partially connected 3D-NoCs. The proposed routing algorithms in this dissertation distribute the traffic over the network, thus enhancing the performance and improving the fault tolerance of the overall network.

One extension to the proposed ideas is to determine the optimum number and location of TSVs for different routing algorithms. The routing algorithms impose certain limitations on the routing paths according to their number of virtual channels to avoid deadlock in the network. Therefore, the number and location of TSVs will have an impact on the performance of the routing algorithm under study. In other words, for every routing algorithm there is an optimum TSV configuration that provides much better performance compared to a random TSV placement.

Moreover, for a given number and location of TSVs, further modifications in the routing algorithms can enhance the fault tolerance of the network . In general, when a TSV fails, the load on the other TSVs in the network increases. If not taken care of probably, this increased load will lower the network performance. However, if the routing algorithms design takes the possibility of TSV failure into account, it may be possible to obtain higher performance compared to a failure-agnostic algorithm. In other words, a good routing algorithm should not only guarantee good performance for a specific TSV configuration, but it should also be fault-tolerant and guarantee a certain level of performance by properly distributing the traffic over the network when faults are present.

Further extensions are possible to take into account the widely studies high temperature problem of TSVs. It is possible to design a routing algorithm capable of distributing the traffic over the network and TSVs under some upper-bound constraints on TSVs temperatures. One enabling mechanism is providing information about TSV temperature during run-time. When TSV temperature reaches a threshold, the routing algorithm can be reconfigured to detour the high temperature regions. Another interesting approach is to deploy machine learning algorithms in the routers to enable them make routing decisions based on the information provided by TSVs regarding their current temperature.

The analytical methods presented in this thesis may also be extended to more general models. Current analytical models predict performance metrics in an average sense. For instance, it is possible to obtain the average network latency, the average buffer occupancy, the average temperature of a certain TSV and so on. Unfortunately, these average measures may not be sufficient for certain applications. As an example, consider a NoC running a streaming application alongside other possible applications.

Streaming applications typically specify a minimum QoS in terms of the tolerable distribution of delays. Therefore, an ideal analytical model should provide complete distributional information, rather than just an average, about the requested performance metric. A reasonable first step toward this extension is to predict the variance of the performance metric along with the average if a unimodal and approximately Gaussian distribution on the performance metric can be assumed.

# Chapter 7

# Conclusion

Planar 2D NoC has limitations for floor planning. Long global wires causes increased latency and therefore limit the performance improvements resulting from Network-on-Chip paradigm. This challenge comes basically from the high network diameter that NoC suffers from. Thus the need to optimize the 2D-NoC system arises and one possible solution was porting the 2D-NoC architecture to the third dimension to enhance the performance of NoC systems and alleviate their limitations. A 3D-NoC is a composite of multiple device layers of 2D NoC with direct vertical interconnects using TSVs. Although there are various alternatives to build the vertical interconnects such as wire bonding and micro bumping, TSVs are the most popular choice since they enhance the overall performance. On the other hand, employing a large number of TSVs has negative effects on reliability issues and the consumed area. Partially connected 3D-NoC emerges to take advantage of 3D-NoC with respect to the limiting number of TSVs.

A partially connected 3D-NoC achieves a compromise between the scalability of NoC and the considerable footprint of fully connected 3D-NoC. The first part of the dissertation proposes an adaptive routing algorithm named ETW for partially connected 3D-NoC. The routing algorithm tolerates faults on the vertical links by enabling the intermediate routers to opt for other TSVs whenever the vertical link at the router's location is faulty. Our simulations show that the proposed algorithm is slightly inferior to the non-adaptive fault-intolerant Elevator-first algorithm in terms of latency. However, ETW has the advantage of using fewer virtual channels as compared to Elevator-first. Moreover, our analytical results show that the proposed algorithm is significantly more resilient to permanent TSV faults.

The vulnerability of TSVs during manufacturing process makes these interconnections susceptible to faults which is discussed in the second part of the dissertation. TSV failures might occur during or after manufacturing process. Either the TSV yield should be increased or chips with faulty TSVs should be discarded but both solutions are costly. We proposed a reconfigurable routing algorithm, called CoBRA, to tolerate TSV failures during runtime and manufacturing process. First, the routing algorithm dynamically searches for a healthy elevator in the same column. If no TSV is found, the packet moves to east and and if the packet reaches the eastmost column and fails to find a healthy TSV, the network is reconfigured to find an elevator at the west direction. Simulation results indicate that CoBRA enhances reliability considerably compared to Elevator-first.

The last part of the dissertation is dedicated to relaxing the moving toward east condition in the already proposed routing algorithms. LEAD provides lower latency, higher saturation point and better temperature distribution under a variety of traffic patterns and TSV configurations. Also, a queuing theory based model targeting adaptive routing algorithms is developed. In low injection rate regime, simulation and analytical results agree very well within a 10% margin, proving that the analytical model can be reliably used to estimate performance in a fraction of the time consumed by simulation. Even though analytical results deviate from simulation under the high injection rate regime, it is observed that the analytical model can reliably provide a valid relative saturation point comparison between two different algorithms.

# Bibliography

[1] Shashi Kumar, Axel Jantsch, J-P Soininen, Martti Forsell, Mikael Millberg, Johny Oberg, Kari Tiensyrja, and Ahmed Hemani. A network on chip architecture and design methodology. In *VLSI, 2002. Proceedings. IEEE Computer Society Annual Symposium on*, pages 117–124. IEEE, 2002.

[2] Nejib Mediouni, Samir Ben Abid, Oussama Kallel, and Salem Hasnaoui. Modeling and performance evaluation of 2d and 3d nocs using discrete event simulation. *International Journal of Computer Applications*, 137(12), 2016.

[3] Kanchan Manna, Shivam Swami, Santanu Chattopadhyay, and Indranil Sengupta. Integrated through-silicon via placement and application mapping for 3d mesh-based noc design. *ACM Transactions on Embedded Computing Systems (TECS)*, 16(1):24, 2016.

[4] Axel Jantsch, Hannu Tenhunen, et al. *Networks on chip*, volume 396. Springer, 2003.

[5] K Emerson. Asynchronous design-an interesting alternative. In *VLSI Design, 1997. Proceedings., Tenth International Conference on*, pages 318–320. IEEE, 1997.

[6] Wen-Hsiang Hu, Seung Eun Lee, and Nader Bagherzadeh. Dmesh: a diagonally-linked mesh network-on-chip architecture. *Network on Chip Architectures*, page 14, 2008.

[7] Xiaohang Wang, T. Mak, Mei Yang, Yingtao Jiang, M. Daneshtalab, and M. Palesi. On self-tuning networks-on-chip for dynamic network-flow dominance adaptation. In *Networks on Chip (NoCS), 2013 Seventh IEEE/ACM International Symposium on*, pages 1–8, April 2013.

[8] Tobias Bjerregaard and Shankar Mahadevan. A survey of research and practices of network-on-chip. *ACM Computing Surveys (CSUR)*, 38(1):1, 2006.

[9] Philip Garrou, Christopher Bower, and Peter Ramm. *Handbook of 3d integration: volume 1-technology and applications of 3D integrated circuits*. John Wiley & Sons, 2011.

[10] Anna W Topol, DC La Tulipe, Leathen Shi, David J Frank, Kerry Bernstein, Steven E Steen, Arvind Kumar, Gilbert U Singco, Albert M Young, Kathryn W Guarini, et al. Three-dimensional integrated circuits. *IBM Journal of Research and Development*, 50(4.5):491–506, 2006.

[11] B. Black, M. Annavaram, N. Brekelbaum, J. DeVale, Lei Jiang, G.H. Loh, D. McCauley, P. Morrow, D.W. Nelson, D. Pantuso, P. Reed, J. Rupley, Sadasivan Shankar, J. Shen, and C. Webb. Die stacking (3d) microarchitecture. In *Microarchitecture, 2006. MICRO-39. 39th Annual IEEE/ACM International Symposium on*, pages 469–479, Dec 2006.

[12] B.S. Feero and P.P. Pande. Networks-on-chip in a three-dimensional environment: A performance evaluation. *Computers, IEEE Transactions on*, 58(1):32–45, Jan 2009.

[13] Luca P Carloni, Partha Pande, and Yuan Xie. Networks-on-chip in emerging interconnect paradigms: Advantages and challenges. In *Proceedings of the 2009 3rd ACM/IEEE International Symposium on Networks-on-Chip*, pages 93–102. IEEE Computer Society, 2009.

[14] Bart Swinnen, W Ruythooren, P De Moor, L Bogaerts, L Carbonell, K De Munck, B Eyckens, S Stoukatch, D Sabuncuoglu Tezcan, Z Tokei, et al. 3d integration by cu-cu thermo-compression bonding of extremely thinned bulk-si die containing 10 $\mu$m pitch through-si vias. In *2006 International Electron Devices Meeting*, pages 1–4. IEEE, 2006.

[15] J. Van Olmen, A. Mercha, G. Katti, C. Huyghebaert, J. Van Aelst, E. Seppala, Z. Chao, S. Armini, J. Vaes, R. C. Teixeira, M. Van Cauwenberghe, P. Verdonck, K. Verhemeldonck, A. Jourdain, W. Ruythooren, M. de Potter de ten Broeck, A. Opdebeeck, T. Chiarella, B. Parvais, I. Debusschere, T. Y. Hoffmann, B. De Wachter, W. Dehaene, M. Stucchi, M. Rakowski, P. Soussan, R. Cartuyvels, E. Beyne, S. Biesemans, and B. Swinnen. 3d stacked ic demonstration using a through silicon via first approach. In *2008 IEEE International Electron Devices Meeting*, pages 1–4, Dec 2008.

[16] K. Banerjee, S. J. Souri, P. Kapur, and K. C. Saraswat. 3-d ics: a novel chip design for improving deep-submicrometer interconnect performance and systems-on-chip integration. *Proceedings of the IEEE*, 89(5):602–633, May 2001.

[17] W.R. Davis, J. Wilson, S. Mick, J. Xu, H. Hua, C. Mineo, A.M. Sule, M. Steer, and P.D. Franzon. Demystifying 3d ics: the pros and cons of going vertical. *Design Test of Computers, IEEE*, 22(6):498–510, Nov 2005.

[18] G. H. Loh, Y. Xie, and B. Black. Processor design in 3d die-stacking technologies. *IEEE Micro*, 27(3):31–48, May 2007.

[19] C. C. Liu, I. Ganusov, M. Burtscher, and Sandip Tiwari. Bridging the processor-memory performance gap with 3d ic technology. *IEEE Design Test of Computers*, 22(6):556–564, Nov 2005.

[20] K. T. Park, D. Kim, S. Hwang, M. Kang, H. Cho, Y. Jeong, Y. I. Seo, J. Jang, H. S. Kim, S. M. Jung, Y. T. Lee, C. Kim, and W. S. Lee. A 45nm 4gb 3-dimensional double-stacked multi-level nand flash memory with shared bitline structure. In *2008 IEEE International Solid-State Circuits Conference - Digest of Technical Papers*, pages 510–632, Feb 2008.

[21] P. Batude, M. A. Jaud, O. Thomas, L. Clavelier, A. Pouydebasque, M. Vinet, S. Deleonibus, and A. Amara. 3d cmos integration: Introduction of dynamic coupling and application to compact and robust 4t sram. In *2008 IEEE International Conference on Integrated Circuit Design and Technology and Tutorial*, pages 281–284, June 2008.

[22] Uksong Kang, Hoe-Ju Chung, Seongmoo Heo, Soon-Hong Ahn, Hoon Lee, Soo-Ho Cha, Jaesung Ahn, DukMin Kwon, Jin Ho Kim, Jae-Wook Lee, Han-Sung Joo, Woo-Seop Kim, Hyun-Kyung Kim, Eun-Mi Lee, So-Ra Kim, Keum-Hee Ma, Dong-Hyun Jang, Nam-Seog Kim, Man-Sik Choi, Sae-Jang Oh, Jung-Bae Lee, Tae-Kyung Jung, Jei-Hwan Yoo, and Changhyun Kim. 8gb 3d ddr3 dram using through-silicon-via technology. In *2009 IEEE International Solid-State Circuits Conference - Digest of Technical Papers*, pages 130–131,131a, Feb 2009.

[23] B. Kim, C. Sharbono, T. Ritzdorf, and D. Schmauch. Factors affecting copper filling process within high aspect ratio deep vias for 3d chip stacking. In *56th Electronic Components and Technology Conference 2006*, pages 6 pp.–, 2006.

[24] P. Benkart, A. Kaiser, A. Munding, M. Bschorr, H. J. Pfleiderer, E. Kohn, A. Heittmann, H. Huebner, and U. Ramacher. 3d chip stack technology using through-chip interconnects. *IEEE Design Test of Computers*, 22(6):512–518, Nov 2005.

[25] Silke Spiesshoefer, Leonard Schaper, Susan Burkett, Gowtham Vangara, Ziaur Rahman, and Parthiban Arunasalam. Z-axis interconnects using fine pitch, nanoscale through-silicon vias: Process development. In *Electronic Components and Technology Conference, 2004. Proceedings. 54th*, volume 1, pages 466–471. IEEE, 2004.

[26] Igor Loi, Federico Angiolini, and Luca Benini. Supporting vertical links for 3d networks-on-chip: toward an automated design and analysis flow. In *Proceedings of the 2nd international conference on Nano-Networks*, page 15. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2007.

[27] James Burns, Lisa McIlrath, Craig Keast, Craig Lewis, Andrew Loomis, Keith Warner, and Peter Wyatt. Three-dimensional integrated circuits for low-power,

high-bandwidth systems on a chip. In *Solid-State Circuits Conference, 2001. Digest of Technical Papers. ISSCC. 2001 IEEE International*, pages 268–269. IEEE, 2001.

[28] Sudeep Pasricha. Exploring serial vertical interconnects for 3d ics. In *Proceedings of the 46th Annual Design Automation Conference*, pages 581–586. ACM, 2009.

[29] Shamik Das, Andy Fan, Kuan-Neng Chen, Chuan Seng Tan, Nisha Checka, and Rafael Reif. Technology, performance, and computer-aided design of three-dimensional integrated circuits. In *Proceedings of the 2004 international symposium on Physical design*, pages 108–115. ACM, 2004.

[30] Robert S Patti. Three-dimensional integrated circuits and the future of system-on-chip designs. *Proceedings of the IEEE*, 94(6):1214–1224, 2006.

[31] Igor Loi, Subhasish Mitra, Thomas H Lee, Shinobu Fujita, and Luca Benini. A low-overhead fault tolerance scheme for tsv-based 3d network on chip links. In *Proceedings of the 2008 IEEE/ACM International Conference on Computer-Aided Design*, pages 598–602. IEEE Press, 2008.

[32] K. Chakrabarty, S. Deutsch, H. Thapliyal, and Fangming Ye. Tsv defects and tsv-induced circuit failures: The third dimension in test and design-for-test. In *Reliability Physics Symposium (IRPS), 2012 IEEE International*, pages 5F.1.1–5F.1.12, April 2012.

[33] KN Tu. Reliability challenges in 3d ic packaging technology. *Microelectronics Reliability*, 51(3):517–523, 2011.

[34] A. P. Karmarkar, X. Xu, and V. Moroz. Performanace and reliability analysis of 3d-integration structures employing through silicon via (tsv). In *2009 IEEE International Reliability Physics Symposium*, pages 682–687, April 2009.

[35] G. Katti, M. Stucchi, K. De Meyer, and W. Dehaene. Electrical modeling and characterization of through silicon via for three-dimensional ics. *IEEE Transactions on Electron Devices*, 57(1):256–262, Jan 2010.

[36] Aditya P Karmarkar, Xiaopeng Xu, Sesh Ramaswami, John Dukovic, Kedar Sapre, and Ajay Bhatnagar. Material, process and geometry effects on through-silicon via reliability and isolation. In *MRS Proceedings*, volume 1249, pages 1249–F09. Cambridge Univ Press, 2010.

[37] Jose Duato, Sudhakar Yalamanchili, and Lionel M Ni. *Interconnection networks: an engineering approach*. Morgan Kaufmann, 2003.

[38] Prasant Mohapatra. Wormhole routing techniques for directly connected multicomputer systems. *ACM Computing Surveys (CSUR)*, 30(3):374–410, 1998.

[39] *A Touchstone Delta System Description*, 1991.

[40] Masoumeh Ebrahimi and Masoud Daneshtalab. A light-weight fault-tolerant routing algorithm tolerating faulty links and routers. *Computing*, 97(6):631–648, 2013.

[41] Davide Bertozzi, Antoine Jalabert, Srinivasan Murali, Rutuparna Tamhankar, Stergios Stergiou, Luca Benini, and Giovanni De Micheli. Noc synthesis flow for customized domain specific multiprocessor systems-on-chip. *IEEE transactions on parallel and distributed systems*, 16(2):113–129, 2005.

[42] Partha Pratim Pande, Cristian Grecu, Michael Jones, Andre Ivanov, and Resve Saleh. Performance evaluation and design trade-offs for network-on-chip interconnect architectures. *IEEE transactions on Computers*, 54(8):1025–1040, 2005.

[43] Hsien-Hsin S Lee and Krishnendu Chakrabarty. Test challenges for 3d integrated circuits. *IEEE Design & Test of Computers*, 26(5), 2009.

[44] Guruprasad Katti, Michele Stucchi, Kristin De Meyer, and Wim Dehaene. Electrical modeling and characterization of through silicon via for three-dimensional ics. *Electron Devices, IEEE Transactions on*, 57(1):256–262, 2010.

[45] M. Motoyoshi. Through-silicon via (tsv). *Proceedings of the IEEE*, 97(1):43–48, Jan 2009.

[46] Abbas Sheibanyrad, Frédéric Pétrot, and Axel Jantsch. *3D integration for NoC-based SoC Architectures.* Springer, 2011.

[47] H. Matsutani, M. Koibuchi, and H. Amano. Tightly-coupled multi-layer topologies for 3-d nocs. In *Parallel Processing, 2007. ICPP 2007. International Conference on*, pages 75–75, Sept 2007.

[48] A. Dalirsani, M. Hosseinabady, and Z. Navabi. An analytical model for reliability evaluation of noc architectures. In *On-Line Testing Symposium, 2007. IOLTS 07. 13th IEEE International*, pages 49–56, July 2007.

[49] Dae Hyun Kim and Sung Kyu Lim. Design quality trade-off studies for 3-d ics built with sub-micron tsvs and future devices. *Emerging and Selected Topics in Circuits and Systems, IEEE Journal on*, 2(2):240–248, June 2012.

[50] Pooria M Yaghini, Ashkan Eghbal, Misagh Khayambashi, and Nader Bagherzadeh. Coupling mitigation in 3-d multiple-stacked devices. 2014.

[51] Moongon Jung, J. Mitra, D.Z. Pan, and Sung Kyu Lim. Tsv stress-aware full-chip mechanical reliability analysis and optimization for 3-d ic. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 31(8):1194–1207, Aug 2012.

[52] C. Cassidy, J. Kraft, S. Carniello, F. Roger, H. Ceric, A.P. Singulani, E. Langer, and F. Schrank. Through silicon via reliability. *Device and Materials Reliability, IEEE Transactions on*, 12(2):285–295, June 2012.

[53] Misagh Khayambashi, Pooria M Yaghini, Ashkan Eghbal, and Nader Bagherzadeh. Analytical reliability analysis of 3d noc under tsv failure. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 11(4):43, 2015.

[54] A Ben Ahmed and A Ben Abdallah. La-xyz: low latency, high throughput look-ahead routing algorithm for 3d network-on-chip (3d-noc) architecture. In *The 6th IEEE international symposium on embedded multicore SoCs*, pages 167–174, 2012.

[55] Sara Akbari, Ali Shafiee, Mahmoud Fathy, and Reza Berangi. Afra: A low cost high performance reliable routing for 3d mesh nocs. In *2012 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 332–337. IEEE, 2012.

[56] Mojtaba Ebrahimi, Xin Chang, Masoud Daneshtalab, Juha Plosila, Pasi Liljeberg, and Hannu Tenhunen. Dyxyz: Fully adaptive routing algorithm for 3d nocs. In *Parallel, Distributed and Network-Based Processing (PDP), 2013 21st Euromicro International Conference on*, pages 499–503. IEEE, 2013.

[57] Masoumeh Ebrahimi, Masoud Daneshtalab, Pasi Liljeberg, Juha Plosila, Jose Flich, and Hannu Tenhunen. Path-based partitioning methods for 3d networks-on-chip with minimal adaptive routing. *IEEE Transactions on Computers*, 63(3):718–733, 2014.

[58] M. Ebrahimi, M. Daneshtalab, and J. Plosila. Fault-tolerant routing algorithm for 3d noc using hamiltonian path strategy. In *2013 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1601–1604, March 2013.

[59] S. Pasricha and Yong Zou. A low overhead fault tolerant routing scheme for 3d networks-on-chip. In *Quality Electronic Design (ISQED), 2011 12th International Symposium on*, pages 1–8, March 2011.

[60] Akram Ben Ahmed and Abderazek Ben Abdallah. Architecture and design of high-throughput, low-latency, and fault-tolerant routing algorithm for 3d-network-on-chip (3d-noc). *The Journal of Supercomputing*, 66(3):1507–1532, 2013.

[61] Akram Ben Ahmed and Abderazek Ben Abdallah. Graceful deadlock-free fault-tolerant routing algorithm for 3d network-on-chip architectures. *Journal of Parallel and Distributed Computing*, 74(4):2229–2240, 2014.

[62] M. Zhu, J. Lee, and K. Choi. An adaptive routing algorithm for 3d mesh noc with limited vertical bandwidth. In *VLSI and System-on-Chip, 2012 (VLSI-SoC), IEEE/IFIP 20th International Conference on*, pages 18–23, Oct 2012.

[63] Mingyang Zhu, Jinho Lee, and Kiyoung Choi. An adaptive routing algorithm for 3d mesh noc with limited vertical bandwidth. In *VLSI and System-on-Chip (VLSI-SoC), 2012 IEEE/IFIP 20th International Conference on*, pages 18–23, Oct 2012.

[64] M. Ebrahimi, M. Daneshtalab, P. Liljeberg, and H. Tenhunen. Fault-tolerant method with distributed monitoring and management technique for 3d stacked meshes. In *Computer Architecture and Digital Systems (CADS), 2013 17th CSI International Symposium on*, pages 93–98, Oct 2013.

[65] Florentine Dubois, Abbas Sheibanyrad, Frederic Petrot, and Maryam Bahmani. Elevator-first: A deadlock-free distributed routing algorithm for vertically partially connected 3d-nocs. *IEEE Transactions on Computers*, 62(3):609–615, 2013.

[66] Jinho Lee and Kiyoung Choi. A deadlock-free routing algorithm requiring no virtual channel on 3d-nocs with partial vertical connections. In *Networks on Chip (NoCS), 2013 Seventh IEEE/ACM International Symposium on*, pages 1–2, April 2013.

[67] Abbas Eslami Kiasari, Hamid Sarbazi-Azad, and Mohamed Ould-Khaoua. An accurate mathematical performance model of adaptive routing in the star graph. *Future Generation Computer Systems*, 24(6):461–474, 2008.

[68] Jong Kim and Chita R. Das. Hypercube communication delay with wormhole routing. *IEEE Transactions on Computers*, 43(7):806–814, 1994.

[69] J. Hu, U. Y. Ogras, and R. Marculescu. System-level buffer allocation for application-specific networks-on-chip router design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25(12):2919–2933, Dec 2006.

[70] Zvika Guz, Isask'har Walter, Evgeny Bolotin, Israel Cidon, Ran Ginosar, and Avinoam Kolodny. Network delays and link capacities in application-specific wormhole nocs. *VLSI Design*, 2007, 2007.

[71] Umit Y Ogras, Paul Bogdan, and Radu Marculescu. An analytical approach for network-on-chip performance analysis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 29(12):2001–2013, 2010.

[72] A. E. Kiasari, Z. Lu, and A. Jantsch. An analytical latency model for networks-on-chip. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 21(1):113–123, Jan 2013.

[73] Jeffrey T Draper and Joydeep Ghosh. A comprehensive analytical model for wormhole routing in multicomputer systems. *Journal of Parallel and Distributed Computing*, 23(2):202–214, 1994.

[74] Po-Chi Hu and Leonard Kleinrock. An analytical model for wormhole routing with finite size input buffers. In *Teletraffic Science and Engineering*, volume 2, pages 549–560. Elsevier, 1997.

[75] R. Salamat, M. Ebrahimi, and N. Bagherzadeh. An adaptive, low restrictive and fault resilient routing algorithm for 3d network-on-chip. In *Parallel, Distributed*

*and Network-Based Processing (PDP), Euromicro International Conference on*, pages 392–395, March 2015.

[76] `http://access.ee.ntu.edu.tw/noxim/index.html`.

[77] V. Catania, A. Mineo, S. Monteleone, M. Palesi, and D. Patti. Noxim: An open, extensible and cycle-accurate network on chip simulator. In *Application-specific Systems, Architectures and Processors (ASAP), 2015 IEEE 26th International Conference on*, pages 162–163, July 2015.

[78] Wei Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron, and M.R. Stan. Hotspot: a compact thermal modeling methodology for early-stage vlsi design. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 14(5):501–513, May 2006.

[79] Jaswinder Pal Singh, Wolf-Dietrich Weber, and Anoop Gupta. Splash: Stanford parallel applications for shared-memory. *ACM SIGARCH Computer Architecture News*, 20(1):5–44, 1992.

[80] Christian Bienia, Sanjeev Kumar, Jaswinder Pal Singh, and Kai Li. The parsec benchmark suite: Characterization and architectural implications. In *Proceedings of the 17th international conference on Parallel architectures and compilation techniques*, pages 72–81. ACM, 2008.

[81] Georgia-Ann Klutke, Peter C Kiessler, and MA Wortman. A critical look at the bathtub curve. *IEEE Transactions on Reliability*, 52(1):125–129, 2003.

[82] Freek Verbeek and Julien Schmaltz. A decision procedure for deadlock-free routing in wormhole networks. *IEEE Transactions on Parallel and Distributed Systems*, 25(8):1935–1944, 2014.

[83] Jinho Lee, Kyungsu Kang, and Kiyoung Choi. Redelf: An energy-efficient deadlock-free routing for 3d nocs with partial vertical connections. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 12(3):26, 2015.

[84] Ronak Salamat, Misagh Khayambashi, Masoumeh Ebrahimi, and Nader Bagherzadeh. A resilient routing algorithm with formal reliability analysis for partially connected 3d-nocs. *IEEE Transactions on Computers*, 65(11):3265–3279, 2016.

[85] Kai-Yuan Jheng, Chih-Hao Chao, Hao-Yu Wang, and An-Yeu Wu. Traffic-thermal mutual-coupling co-simulation platform for three-dimensional network-on-chip. In *VLSI Design Automation and Test (VLSI-DAT), 2010 International Symposium on*, pages 135–138. IEEE, 2010.

[86] Dimitri P Bertsekas, Robert G Gallager, and Pierre Humblet. *Data networks*, volume 2. Prentice-Hall International New Jersey, 1992.