LEBANESE AMERICAN UNIVERSITY

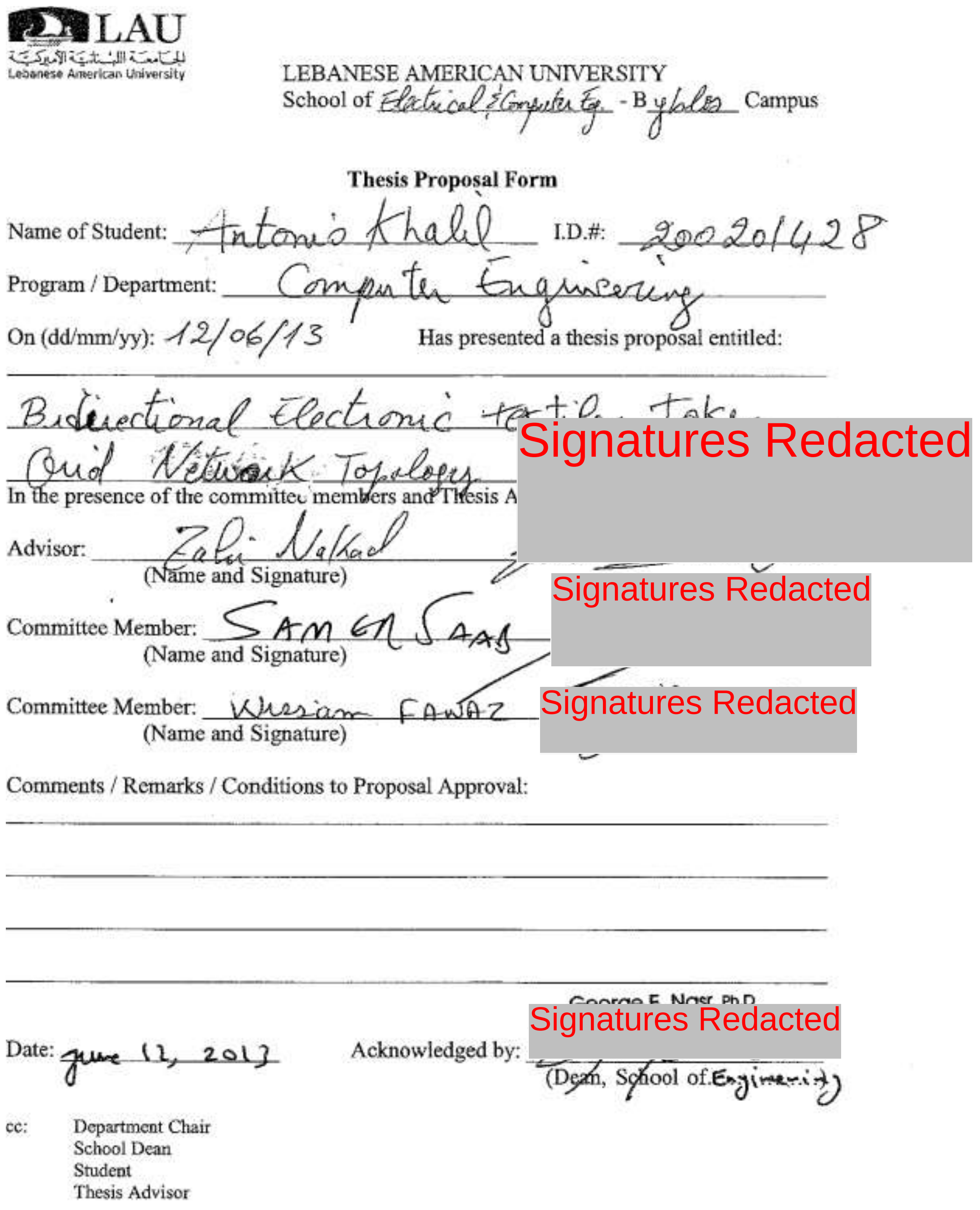
BIDIRECTIONAL ELECTRONIC TEXTILE TOKEN GRID NETWORK TOPOLOGY

By

ANTONIO F. KHALIL

A thesis
Submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Engineering

School of Electrical and Computer Engineering
June 2013



LAU
الجامعة اللبنانية الأميركية
Lebanese American University

LEBANESE AMERICAN UNIVERSITY
School of Electrical & Computer Eng. - Byblos Campus

Thesis Proposal Form

Name of Student: Antonio Khalil I.D.#: 200201428

Program / Department: Computer Engineering

On (dd/mm/yy): 12/06/13 Has presented a thesis proposal entitled:

Bidirectional Electronic textile Toke

Grid Network Topology

Signatures Redacted

In the presence of the committee members and Thesis A

Advisor: Zahi Nakad
(Name and Signature)

Signatures Redacted

Committee Member: SAMER SAAB
(Name and Signature)

Signatures Redacted

Committee Member: Wissam FAWAZ
(Name and Signature)

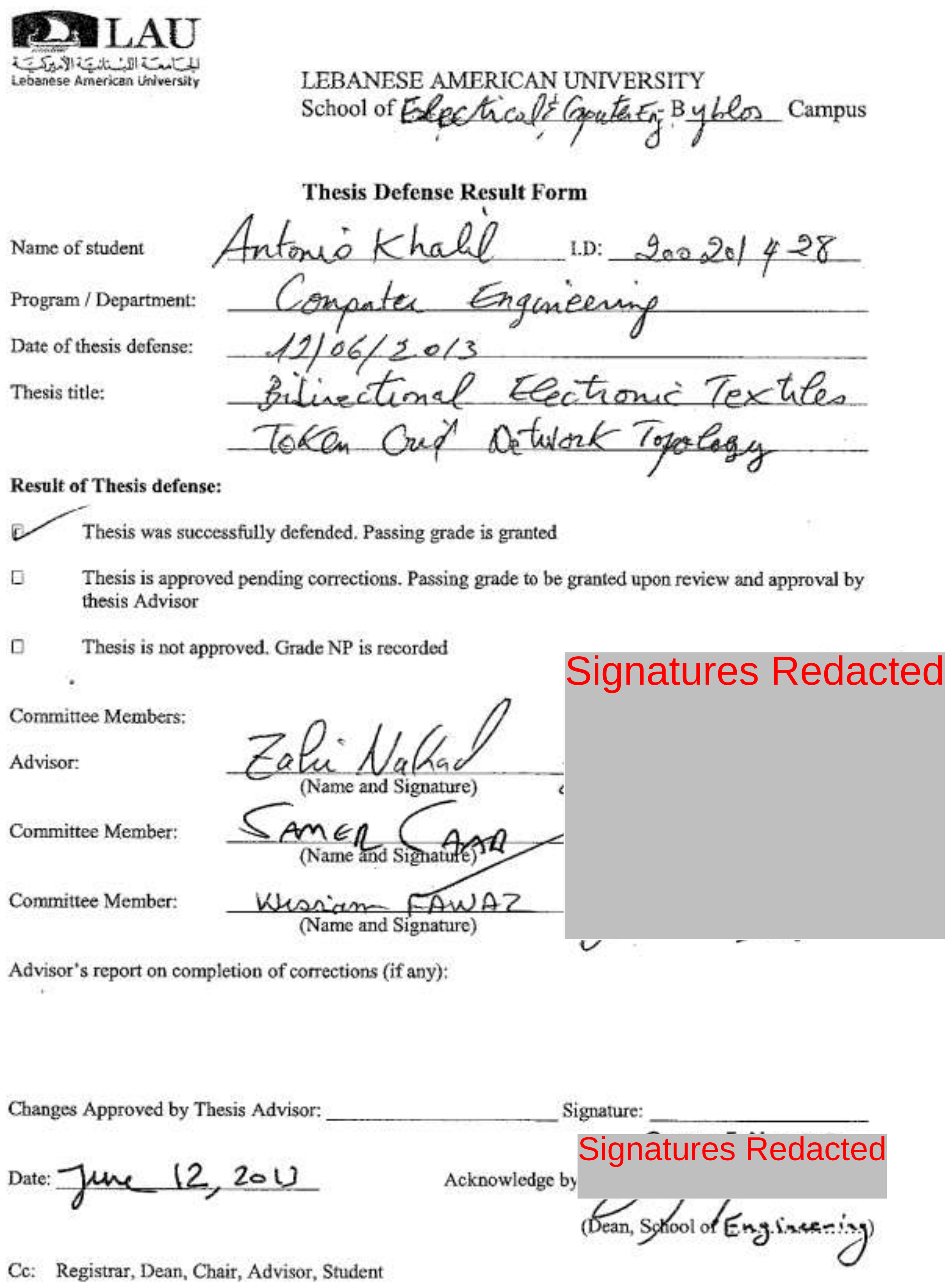
Comments / Remarks / Conditions to Proposal Approval:

George F. Nasr, Ph.D

Signatures Redacted

Date: June 12, 2013 Acknowledged by:
(Dean, School of Engineering)

cc: Department Chair
School Dean
Student
Thesis Advisor



LAU
Lebanese American University

LEBANESE AMERICAN UNIVERSITY
School of Electrical & Computer Eng. Byblos Campus

Thesis Defense Result Form

Name of student	Antonio Khalil	I.D:	200201428
Program / Department:	Computer Engineering		
Date of thesis defense:	12/06/2013		
Thesis title:	Bidirectional Electronic Textiles Token Ring Network Topology		

Result of Thesis defense:

- [✓] Thesis was successfully defended. Passing grade is granted
- ☐ Thesis is approved pending corrections. Passing grade to be granted upon review and approval by thesis Advisor
- ☐ Thesis is not approved. Grade NP is recorded

Signatures Redacted

Committee Members:

Advisor:	Zahi Nakad (Name and Signature)
Committee Member:	Samer Saab (Name and Signature)
Committee Member:	Wissam Fawaz (Name and Signature)

Advisor's report on completion of corrections (if any):

Changes Approved by Thesis Advisor: ____________________ Signature: ____________________

Signatures Redacted

Date: June 12, 2013 Acknowledge by (Dean, School of Engineering)

Cc: Registrar, Dean, Chair, Advisor, Student

LEBANESE AMERICAN UNIVERSITY

Thesis approval Form (Annex III)

Student Name: Antonio Khalil I.D. #: 200201428

Thesis Title: Bidirectional Electronic Textile Token Grid Network Topology.

Program : Masters of Science in Computer Engineering

Division/Dept: Computer Engineering

School: Electrical & Computer Engineering

Approved by:

Signatures Redacted

Zahi Nakad, [illegible]

Thesis Advisor-name, Ph.D. (Advisor)

Associate Professor of ECE

LEBANESE AMERICAN UNIVERSITY
Department of
Electrical and Computer
Engineering

SAMIR SAAB

Member One-name, Ph.D.

Professor of ECE

Date: June 12, 2013

THESIS COPYRIGHT RELEASE FORM

Name: Antonio F. Khalil

Signatures Redacted

Signature

Date: 13/06/2013

PLAGIARISM POLICY COMPLIANCE STATEMENT

I certify that I have read and understood LAU's Plagiarism Policy. I understand that failure to comply with this Policy can lead to academic and disciplinary actions against me.
This work is substantially my own, and to the extent that any part of this work is not my own I have indicated that by acknowledging its sources.

Name: Antonio F. Khalil

Signatures Redacted

Signa

Date: 13/06/2013

ACKNOWLEDGMENTS

This research would not have been possible without the help and assistance of many persons. First I would like to express my gratitude to my supervisor Dr. Zahi Nakad. I am also deeply grateful to the committee members Dr. Samer Saab and Dr. Wissam Fawaz. Finally, special thanks go also to my family for their continuous support.

To my great family

Bidirectional Electronic Textile Token Grid Network Topology

Antonio F. Khalil

Abstract

Electronic Textile (e-textile) applications, seem to obtain more and more importance in our daily lives; the huge amount of fabrics around us help electronic textile to mingle and integrate seamlessly with our daily life without interfering or changing our habits and lifestyles. Reliability is a very important factor for an electronic textile application especially in the fields of medicine, security and other fields where system failure is not tolerable and promptness of response is very delicate. The contributions of this study focus on defining a token grid networking scheme that enhances fault tolerance for electronic textile systems and increases the communication speed between the communicating components vis-à-vis other e-textile token grid topologies, it also presents a simulation environment used to examine and validate the defined topology.

Keywords: Electronic Textiles, Token Grid, Network, Fault Tolerance, Minimization

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER ONE
INTRODUCTION

Electronic textiles, also known as e-textiles, smart fabrics, smart textiles … are a growing field of applications. Electronic textiles have electronics and interconnections merged into the fabric. The E-Textile Research Group in [Hwang, 1993] defines the electronic textiles as follows: "*Electronic textiles (e-textiles) are fabrics that have electronics and interconnections woven into them, with physical flexibility and size that cannot be achieved with existing electronic manufacturing techniques. Components and interconnections are intrinsic to the fabric and thus are less visible and not susceptible to becoming tangled together or snagged by the surroundings. An e-textile can be worn in everyday situations where currently available wearable computers would hinder the user. E-textiles can also more easily adapt to changes in the computational and sensing requirements of an application, a useful feature for power management and context awareness.*"

Clothes, carpets, curtains, furniture … are all every day materials that are totally or partially made from textiles. The presence of non-intrusive electronic systems in our daily usage is becoming more and more important. The mix of textiles and interconnected electronic devices blending together to make a system operate invisibly among everyday objects is needed. Electronic textiles are mostly emerging in fields like sports, health monitoring, emergency response and protective equipment [Martin May 2012]. A reliable electronic textile system is a must; textiles are susceptible to tears and ruptures, interconnection between devices is prone to disconnection, maintaining this connection along with an efficient and reliable networking topology is essential.

The main objective of this thesis is to present the design and simulation of a network topology, the Bidirectional E-Textile Token Grid (BTTG). The BTTG is used as a network topology for electronic textiles systems, it assures minimal communication cost between electronic devices along with high fault tolerance. The usage of such a network topology expresses its prominence in systems where promptness in response and intolerability of faults are delicate. For instance, burglar alarm systems as explained in [National Park Service 2002] are very hard to design and setup, they are bounded by many difficulties that cannot be avoided easily (large areas should be covered by detectors, detector should be nonintrusive, furniture should not block detectors, and many other physical and aesthetic considerations), the use of electronic textiles simplifies most of the hard conditions, and the use of reliable communication topologies such as the BTTG makes the system reliable and fault tolerant, it also makes thief tracking possible due to faster connection between detection devices.

Below are the major contributions of this thesis:

- The first contribution of this thesis is the design of a bidirectional communication scheme, which enhances the communication between nodes of the e-textile token grid, enables the communication over a minimal path, and minimizes the time cost in case of high communication traffic.
- The second contribution is the design of a fault tolerant scheme that enhances the fault tolerance capability of the token grid while maintaining a relatively small wiring volume through the use of I^2C bus in multi-master mode.

The thesis is organized as follows: Chapter 2 discusses background information necessary for understanding the work performed in this research. Chapter 3 presents the Bidirectional E-Textile Token Grid and discusses in detail its network topology and operation schemes. Chapter 4 presents the simulation process of both the original e-textile token grid and the bidirectional e-textile token grid in normal and faulty situations and discusses simulation results along with theoretical and simulated transmission costs of the bidirectional e-textile token grid. Chapter 5 presents conclusions and summarizes the contributions of this thesis.

CHAPTER TWO
RELATED WORK

This chapter reviews multiple electronic textile communication schemes: it focuses on the e-textile token grid network communication scheme [Nakad 2003] then examines two variations of the e-textile token grid; the E-Textile Token Grid with Dual Rings [Zheng, Wu, Chen, and Zhou 2006] and the E-Textile Token Grid with No Merge. Finally an overview of the OMNet++ simulation framework [OMNet++ community 2009] and the I^2C bus topology [I^2C Bus. 2012] are provided.

If we consider an e-textile network as communicating nodes, many network schemes can be adopted for the communication, but not all are suitable and optimal for the requirements of an e-textile network. Important features of e-textiles network schemes are low power consumption, fault tolerance, and scalability [Nakad, Jones, Martin and Fawaz 2009]. Ease of implementation in textile and fast response to events are additional important features.

In “Architectures for e-Textiles” in [Nakad 2003], two networking scheme for e-textile were discussed and proved unreliable. The first networking scheme is the hypercube network scheme. In this scheme, nodes connections are linearly dependent on the hypercube dimensions that make scalability extremely hard to accomplish. Thus hypercube topology is not suitable for e-textile. The second networking scheme that was considered is the tree network topology. However, the communication between the nodes is highly dependent on the parent nodes in the tree and any failure affecting the main nodes is not recoverable and fault tolerance is not efficient, thus this scheme cannot be considered.

On the other hand, one of the most suitable network topologies for e-textile is the token grid network scheme, it is scalable and fault tolerant [Nakad 2003]. The e-textile token grid networking scheme is discussed in the following section.

2.1 E-Textile Token Grid

2.1.1 Network Topology

The e-textile token grid (referred as TTG in what follows) is a two-dimensional network architecture based on the token grid networking scheme presented in [Todd 1992] and [Todd 1994]. It is composed of nodes connected with each other through token rings in an X-Y layout which facilitates its implementation on a fabric backplane [Nakad, Jones, Martin and Fawaz 2009]. Figure 2.1 (a) is an example of a 4x4 TTG network.

Each node in the TTG is connected to two token rings one on the row level and the other on the column level. A token that circulate on a token ring keeps information about the network topology and state of the nodes, it is also used by the nodes to acquire the “right to send” of data over the ring. Z. Nakad in his TTG, presented a transverse dimension, this dimension was used to join processing nodes on different fabric pieces, it also helped in improving the communication speed, by decreasing the number of hops a token has to traverse on a ring (the number of hops is directly related to the number of nodes on a ring) [Nakad, Jones, Martin and Fawaz 2009]. Figure 2.1 (b) gives an illustration about the transverse dimension.

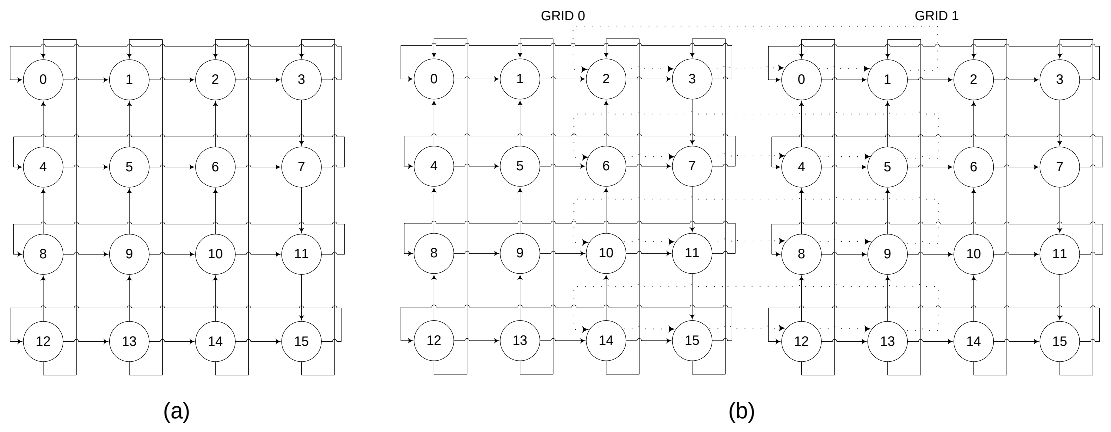


Figure 2.1: (a) is a 4x4 example of an E-Textile token grid TTG (based on Figure 6.20 in [Nakad 2003]). (b) Two TTG grids connected with a transverse ring dimension (based on Figure 6.1 in [Nakad 2003])

A fault tolerant scheme was introduced that overcomes link and node failures in the grid. Power saving is ensured through the handling of routing processes around sleeping nodes without the need for waking them unless needed.

In the following sections, the operation of the TTG in normal mode (without faults in the grid) is explained, as well as its operation when faults are introduced.

2.1.2 Normal Operation Mode

In a normal operation mode, two types of communication between nodes can be initiated. The first type of communication is when a node desires to send a data packet to another node on the same row ring or same column ring. In this case, the source node will have to wait for the corresponding ring token and once the token is acquired, the node sends the data packet on the ring and then relieves the token for the next node to acquire. The other nodes on the ring, once they receive the data packet they check if they are the intended destination, then they will consume the packet, otherwise the packet is forwarded to the next node along the ring. Figure 2.2 is an example where Node (0, 1) sends a packet to Node (0, 3), a node on the same row ring.

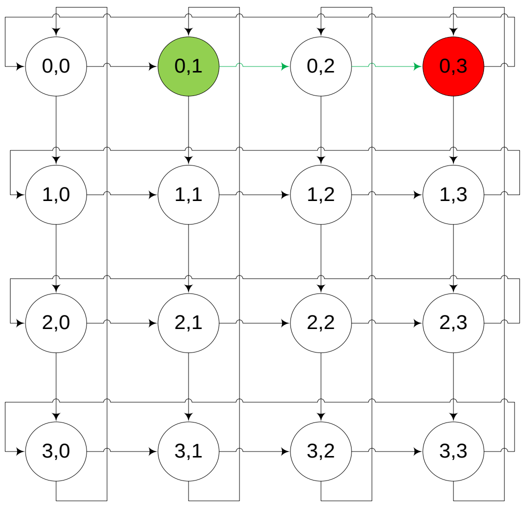


Figure 2.2: A communication along row ring 0 from Node (0, 1) to Node (0, 3) passing by Node (0, 2)

The second type of communication is when a node desires to send a data packet to another node that is neither on its row ring nor on its column ring. In this case, the source node needs to send a merge request to a node on the same row ring of the source node and the same column ring of the destination node, or to a node on the same column ring of the source node and the same row ring of the destination node. When the intended node receives the merge request circulated within the token, it will wait to acquire the token of the other ring then it enters a merge state and it sends a row/column token to its row ring that gives the nodes on the row ring the right to send packet to a destination on the merged node's column, and once the row/column token is received by the merged node, it will send a column/row token to its column ring nodes and they can send a packet to a destination on the merged node's row ring. After both tokens are captured by the initiating node it exits the merged state and a regular token operation is reinstated on the related row and column rings. Figure 2.3 is an example where Node (0, 1) sends a packet to Node (3, 3) and we can see Node (0, 3) in merge state.

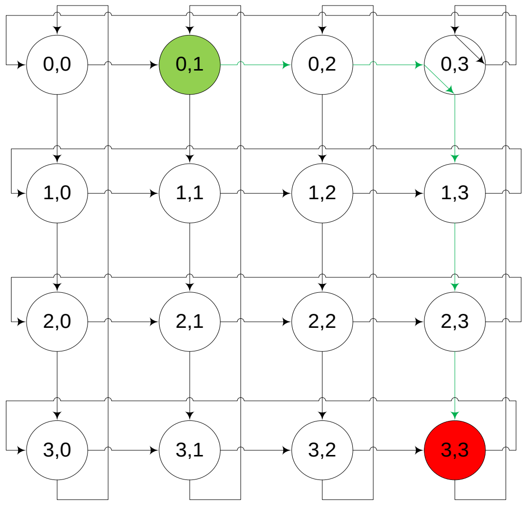


Figure 2.3: A communication from Node (0, 1) to Node (3, 3) we can see the Node (0, 3) in merge state.

In the following section we will discuss the TTG networking schemes in case of faults or sleeping nodes; we will present the error detection and information propagation in case of error, then we will explain the routing schemes.

2.1.3 Fault Tolerance and Sleeping Node Operation

As a convention, the routing process around a sleeping node is treated the same way as the case of a failing node, for this reason in what follows we will explain the fault tolerant networking scheme.

Faults can occur along a ring (row or column). A link failure disables the corresponding token ring connecting the failing link. However, a node failure will disable both column and row rings connected to the node. The failure of a ring will stop the communication between the nodes connected on the ring however the remaining grid token rings will continue to operate normally. Faults can either decay the performance of the system or they can even destroy the grid communication depending on their location and the number of their occurrence on the token grid.

2.1.3.1 Error Detection and Information Propagation

In the TTG case, each node has three separate timeout counters each of which refers to a link on a ring (row, column and transverse rings). Each one of the counters is reset to zero at the moment the node receives a token, a data packet, or error packet from the corresponding ring. In case the counter is not reset to zero and it reaches a preset timeout period, the corresponding node creates an error packet and propagates it along the affected ring. Every node that receives this error packet will update a flag indicating that the corresponding ring is in error.

In order to propagate the error information to the remaining nodes in the grid, the nodes on the affected ring, will propagate this information to the remaining rings they are connected to, and by this action the other nodes on the grid will be notified about the error. It is important to note that each node holds three different variables for errors occurring on rows, columns and transverse rings in the grid network. Figure 2.4 shows a TTG with errors along different locations in the grid.

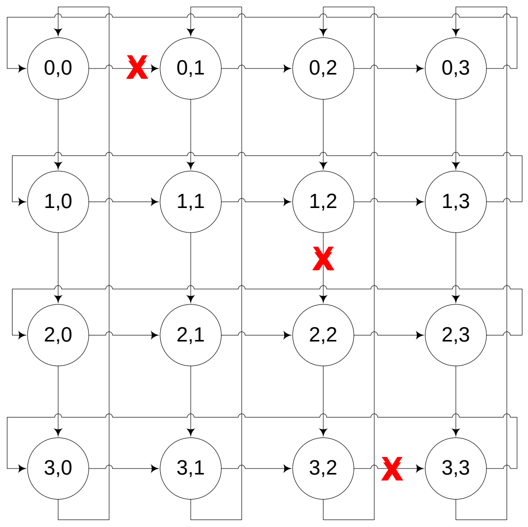


Figure 2.4: TTG 4x4 grid with failure along row ring 0, row ring 3 and column ring 2

2.1.3.2 Fault Tolerant Routing Scheme

In order to explain the routing scheme in case of faults, let us consider a simple case where Node (0, 0) wants to send a data packet to Node (1, 1) and considering that a link fault is present on the destination row. Figure 2.5 illustrates the above case. After propagation of error information; Node (0, 0) requests Node (0, 1) to go into a merged state. Node (0, 1) will receive the request for the merge through the row token. Once it receives the row/column token, Node (0, 0) releases the data packet that is forwarded by Node (0, 1) to Node (1, 1).

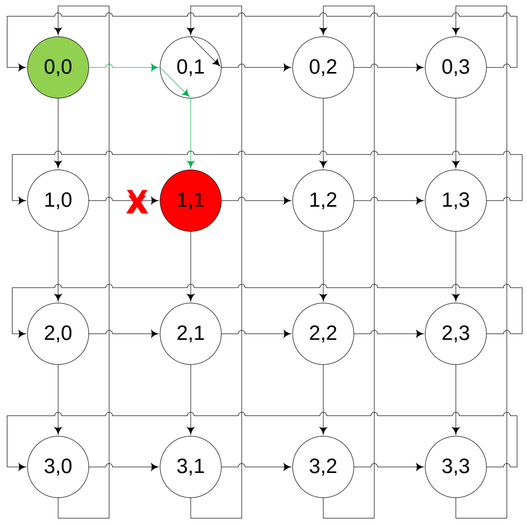


Figure 2.5: TTG with a link error at the destination row

Another and more complex case of fault occurrences, is when there exist an error on both row rings or both column rings of the source and destination nodes. Figure 2.6 (a, b) presents examples where Node (0, 0) is the source node and Node (1, 1) is the destination node. In this case, the nodes needed for merge for the transmission to occur are both isolated, for this reason, the TTG forwards the message along a “Wrong Route" (the column in the case of Figure 2.6(a) and the row in case of Figure 2.6(b)). Once a node receives the data packet, it identifies that this packet should be saved in its data queue then processed when the right token is received. If we consider the example of figure 2.6 (a), the data packet will first reach Node (1, 0). At that node, the destination Node (1, 1) is not reachable because of the link error and the data packet is forwarded on the “Wrong Route" again, down the column. Node (2, 0) requests Node (2, 1) for a merge; this final merge enables the data packet to reach Node (1, 1) through the column ring.

Additional detailed information about the fault tolerance scheme in the same grid and along a transverse link fault can be observed in [Nakad 2003].

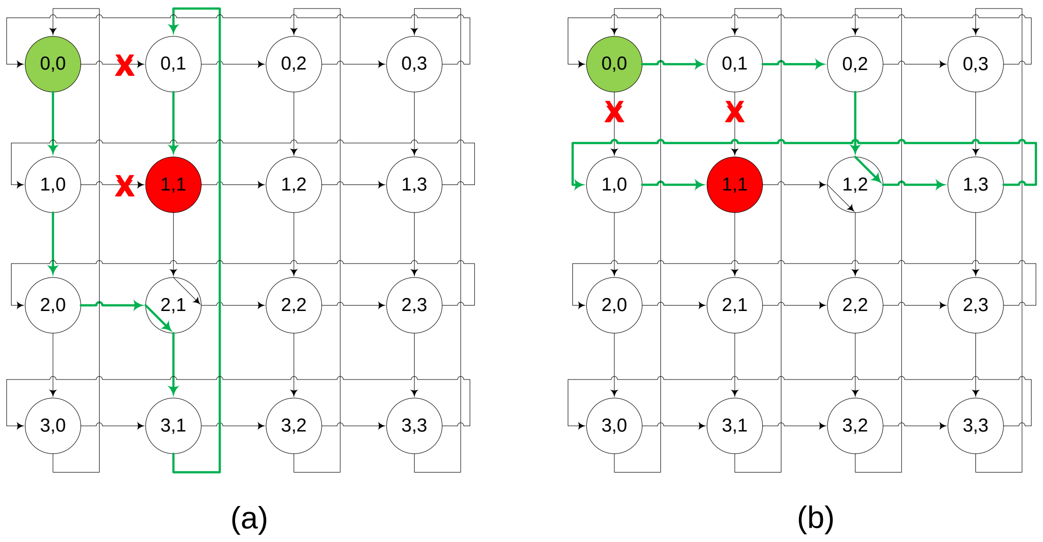


Figure 2.6: (a) TTG with links error along row ring of source Node(0, 0) and row ring of destination Node(1, 1). (b) TTG with links error along column ring of source Node(0, 0) and column ring of destination Node(1, 1).

In the following section, the E-Textile Token Grid with Dual Rings is explained, a variation of the TTG with higher fault tolerance capability.

2.2 E-textile Token Grid with Dual Rings

A variant to the e-textile token grid is the e-textiles token grid with dual rings (TGDR) [Zheng, Wu, Chen, and Zhou 2006]. This grid network topology operates the same way as the TTG in case of normal mode operation. However, it enhances the fault tolerance capability through the introduction of additional rings parallel to every exiting ring with opposite direction of circulation.

2.2.1 Network Topology

The e-textile token grid with dual rings is a two dimensional grid composed out of M rows and N columns. A node in the TGDR grid has a row ID and a column ID which indicates its position in the token grid. Each node is connected to two rings on the row level and two other rings on the column level. The routes of two row rings or two column rings on the same node are connected in reverse directions. An example of a 4x4 TGDR grid is shown in Figure 2.7.

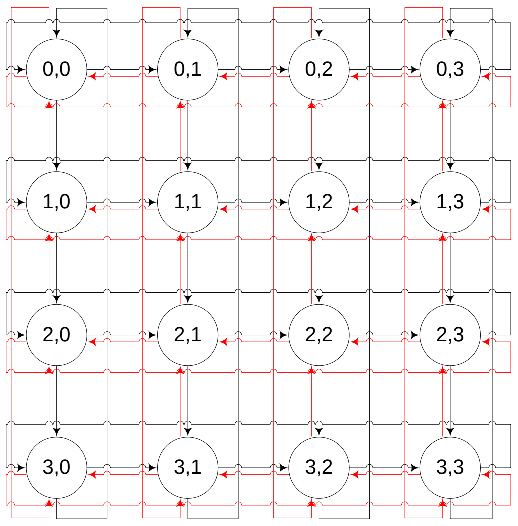


Figure 2.7: 4x4 TGDR grid where primary rings are in black and the parallel rings with opposite routing direction are in red.

In addition to the above, each node in the grid has 4 transmitters and 4 receivers [Zheng, Wu, Chen, and Zhou 2006]. Each ring in the TGDR has a token circulating on it, keeping the information of the network and giving the nodes the access right to send data over the communication links. Each node has the chance of acquiring four tokens, two on the row level and two on the column level.

2.2.2 Normal Operation Mode

The normal operation mode of the TGDR is similar to the TTG grid operation explained in subsection 2.2.1.1 above. If a node wants to send a packet to another node on the same row or the same column ring, the node should wait for the corresponding ring token then release the packet on the ring for the destination node to grab it. In case the destination of the data packet is on a different row and column ring a merge request is sent to a node that intersects the row ring of the source and the column ring of the destination node, or a node that intersects the column node of the source and the row ring of the destination node in order to connect the source node with the destination node into one merged ring. More information regarding the normal operation mode can be found in details in [Todd 1992].

In what follows we discuss the fault tolerance and sleeping node operations of the TGDR.

2.2.3 Fault Tolerance Operation

The TGDR altered the original TTG networking scheme through the introduction of dual rings. The following subsections explains how the introduction of the dual rings is supposed to enhance the fault tolerance ability of the TGDR.

The TDGR follows the same error detection and information propagation scheme as the TTG explained in subsection 2.2.3.1. The use of delay counters is adopted to check a link or a node failure.

As already stated, each node in the TGDR is connected to a dual ring on the row level and dual ring on the column level. If a failure occurs on one of the dual rings at the same level, the other ring will take charge of the packets transmission. For instance in the example presented in Figure 2.8, if one of the rings at the row level breaks, the other one is still operative and is able to communicate the nodes normally.

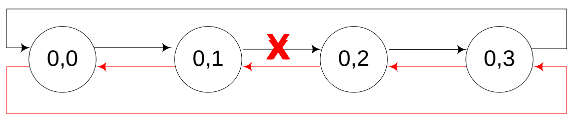


Figure 2.8: When an error causes failure of one ring (black) the other ring (red) will take charge of the operation.

On the other hand, if both dual rings at the same level are broken and the failure affects a single node in the dual rings, the other nodes will maintain connection with each other by bypassing the failing node. Also in case of several faults on the dual rings, each group of connected nodes are able to form their own rings and continue their operation. The variations in the grid are broadcasted to the remaining nodes on the TGDR network. Figure 2.9 illustrates such a case.

With the dual rings and the fault-tolerant operations, even when a textile is worn and split into numerous fragments, the connectivity of the nodes residing on the same fragment is conserved for the remaining parts of the textile. Additional details about the fault tolerance scheme of the TDGR are presented in [Zheng, Wu, Chen, and Zhou 2006].

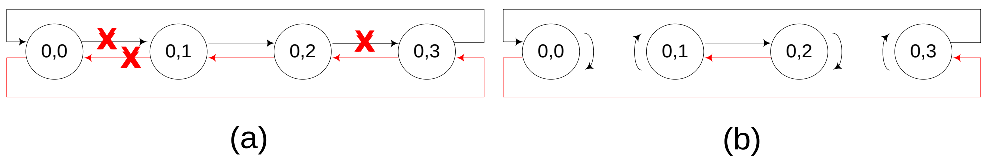


Figure 2.9: (a) Several faults on the dual ring (b) result of (a)

2.2.4 Advantages and Disadvantages of the TDGR

The main advantage of the TDGR over the TTG is that it enhances fault tolerance operation capability compared to the TTG network scheme. The TDGR also claims maintaining full ring connectivity in case one of the rings fails by mean of the dual ring. However, in real application, tears and ruptures will affect both dual rings and a case where only one of the rings is only destroyed is nearly impossible. The TDGR network scheme is hard to implement in real world textile applications because it introduces additional hardware (i.e. 4 transmitters and 4 receivers) at each node and needs at least double the wiring volume of the TTG, which makes the integration with the fabric very difficult. Thus further study on this scheme will not be considered.

2.3 E-textile Token Grid with No Merge (TTGNM)

Another approach to the e-textile token grid is the e-textiles token grid with No merge (TTGNM). This grid network topology has the same operation capability of the original gird in [Nakad 2003], but it enhanced the communication time cost when connecting two nodes lying on different row and column rings.

The e-textile token grid without merge has the same structure as the original textile token grid, it is formed out of M row rings and N column rings a depicted in Figure 2.1 (a).

2.3.1 Normal Operation Mode

In normal operation, the communication between the different nodes of the TTGNM token grid are governed based on the below rules:

Rule 1: If the destination node is on the same row ring or column ring of the source node, the source node will wait for the token ring. When the token ring is captured, the source node will release the data packet to the destination node through the ring route and once the packet is received by the destination node it will be consumed. Figure 2.2 represents a case where the source node and destination node are on the same ring.

Rule 2: If the source and destination nodes are on a different row and column rings, the source node waits to capture the row or column token ring and then releases the data packet to a node on the same row or same column ring as the destination node, based on the captured token. Once the data packet is received by the node intersecting the row and column rings of the source and destination nodes, it will be queued for transmission, this node waits for the other ring token and then it will send the data packet to the destination node, once the data is received at the destination node, it will consume it. Figure 2.10 illustrate the above case.

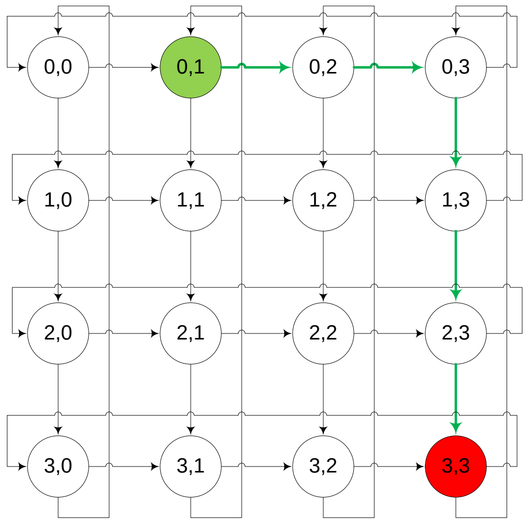


Figure 2.10: Node(0, 1) sends data to Node(3, 3), the data is first forwarded along the row ring once received by Node (0, 3) it is queued then it is forwarded to Node (3, 3) along the column ring of Node (0, 3)

2.3.2 Advantages and Disadvantages of the TTGNM

The TTGNM networking topology, decreased the waiting latency of the TTG by removing the merge operation of the nodes when sending data between two nodes on two different row and column rings. The TTGNM was not studied with a fault tolerance ability. Simulation and comparison results of the TTGNM are discussed in chapter 4.

2.4 Simulation using OMNet++

OMNet++ is a component-based, modular and open-architecture discrete event simulation framework [OMNet++ community 2009] whose application area is the simulation of communication networks. OMNet++ generic and flexible architecture, made it successful in other areas like the simulation of complex IT systems, queuing networks and hardware architectures as well. Network components are programmed in C++, and then assembled into larger models using a high-level language (NED). OMNet++ has extensive GUI support, and due to its modular architecture, the simulation kernel (and models) can be embedded easily into applications.

The simulation in OMNet++ is based on basic components such as: modules, gates, messages, and channels. A module can contain sub-modules it is then called a compound module, otherwise it is a simple module. The module is implemented in C++ supported by an OMNet++ simulation library, it contains the algorithm of the model. Modules communicate by means of messages. Messages are sent using output gates and received over input gates. The gates are connected using channels that can be configured with data rates, propagation delays, and other parameters.

2.5 I^{2}C Bus

The I^{2}C bus also known as Inter IC bus, is a 2 wire hardware interface developed by Philips and is used to communicate electronic devices. It supports slave, master and multi-master configurations. The bus has two bidirectional wires that are used one for serial data and the other for a serial clock. Each device connected to the bus has a unique address and can function as both a transmitter and receiver, depending on its purpose. In addition to transmitters and receivers, devices can also be considered as masters or slaves when performing data transfers. A master is the device which initiates a data transfer on the bus and generates the clock signals to permit the transfer. The I^{2}C bus can operate in a multi-master mode. In this case, more than one master can attempt to initiate a communication over the bus. Arbitration exist in case of simultaneous attempt to control the bus and the first master to gain control is the winner.

Figure 2.11 below shows the send operation of the I^{2}C bus. Device **A** wants to send information to device **B**; **A** controls the bus, it sends data to **B** and then terminates the transmission.

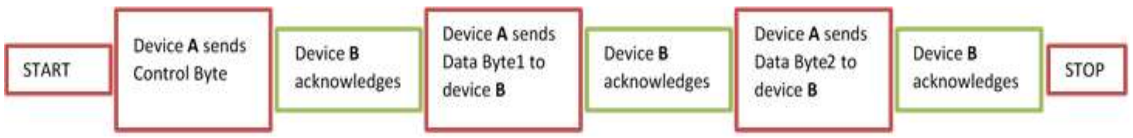

START
Device A sends Control Byte
Device B acknowledges
Device A sends Data Byte1 to device B
Device B acknowledges
Device A sends Data Byte2 to device B
Device B acknowledges
STOP

Figure 2.11: write operation on the I2C bus.

For the I^2C bus in the case of a token grid, all nodes should be considered as master devices, each bus connects only two nodes. Once a master transmits data on the bus, the other master device plays the role of a slave. Two nodes cannot initiate communication at the same time as only one of them possesses the token ring at a given time, and it is the one allowed to send data. Thus collision is not going to arise and arbitration is not needed. For detailed information about the I^2C Bus specifications you can refer to [10].

2.6 Conclusion

This chapter introduced the E-Textile Token Grid networking scheme and two variants the Dual Ring Textile Token Grid and the E-Textile Token Grid with no Merge. It also introduced the OMNet++ simulation framework and I^2C bus. In the following chapter, the Bidirectional E-Textile Token Grid Topology is discussed and its operation is explained.

CHAPTER THREE

BIDIRECTIONAL E-TEXTILE TOKEN GRID (BTTG) NETWORK ARCHITECTURE

The present chapter discusses the Bidirectional E-Textile Token Grid network architecture that is based on the E-Textile Token Grid Architecture discussed in Section 2.1. The BTTG enhances the communication speed and latency between the nodes of the token grid, it also enhances the fault tolerance capability of the textile token grid in case of rings and/or nodes failures.

3.1 Proposal

The BTTG (Bidirectional E-Textile Token Grid) networking architecture scheme is an extension to the E-Textile Token Grid architecture scheme discussed in section 2.1. This novel scheme introduces basic changes that contribute in minimizing the communication cost between nodes and maximizing the fault tolerance capabilities of the grid.

As already presented in chapter 2, the TGDR (section 2.2) claimed (no proof) that it enhanced the fault tolerance capability of the original E-Textile Token Grid by introducing dual rings on the row and column levels of the token grid. Each ring has its own token and the dual ring will play a role of backup once the original ring is in failure. The above case is not always true, especially that in a real case situation both links, the original and the dual link, will be broken in case of a fabric split, thus the backup ring will also be lost. Also the introduction of additional transmitters and receivers to support the dual ring will burden the node. However, with the BTTG, fault tolerance is enhanced and optimal wiring and hardware usage is preserved by the use of I^2C bus in multi-master mode (refer to section 2.5), thus no need for additional wires, transmitters and receivers. Furthermore, in the TTGNM (section 2.4) the removal of the merge process enhanced the communication time cost by eliminating the merging latency factor; however, with the BTTG, cost is minimized by minimizing hops that a packet should traverse on a row ring and on a column ring, presented later in section 3.2.3.

In the following section, the network topology and communication scheme of the BTTG are discussed.

3.2 Network Topology

The BTTG is a grid network structure with N x M interconnected nodes, where N and M being respectively the number of rows and columns in the BTTG grid. Each node in the grid is characterized by a unique Row ID and Column ID (Col ID) combination composed of its row position and its column position respectively in the token grid. For instance, the node intersecting at row 1 and column 3 will have a Row ID = 1 and a Col ID = 3, it will be identified as Node (1, 3). A 4x4 BTTG grid is shown in Figure 3.1(a).

Each node in the BTTG is connected to two token rings one on the row level and the other on the column level. A token ring is characterized by its ability to pass information (token, data packet …) in both directions (Right or Left direction for a row ring and Down or Up direction for a column ring). The BTTG uses the I^{2}C bus in multi-master mode to ensure the full duplex communication on its links. Figure 3.1(b) illustrates the example of two rings intersecting at Node (1, 2).

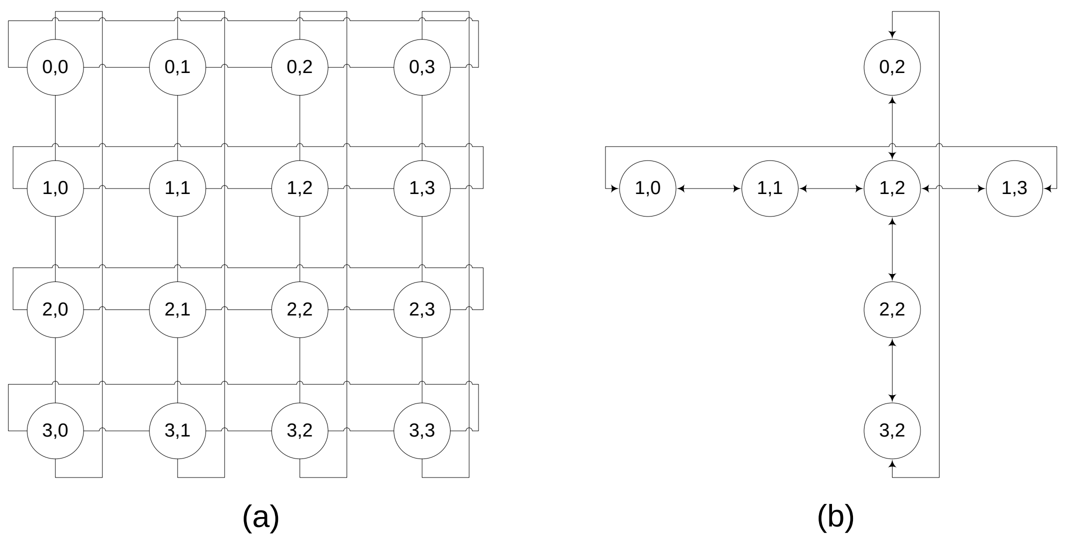


Figure 3.1: (a) is a 4x4 BTTG Token Grid (4 Rows and 4 Columns), (b) represents token rings intersecting at Node (1, 2).

Each ring has a token circulating between nodes, in a normal situation the token circulates in the right direction for a row ring from Node (A, B) to Node (A, B+1), and in the down direction for a column ring from Node (A, B) to Node(A+1, B). Once a node acquires a token it has the right to send data packets to other nodes in the grid and once it is done sending the packets it releases the token to the next node in the ring. A token that circulates on a token ring holds information about its type, it can be either a Normal token in a normal ring configuration or a U-Shaped token when the ring is in fault (explained in details in the following sections). The data packet can be sent to a destination node that lies on the same row or column ring of the sending node, otherwise, the destination node can be on different row and column rings.

The following subsections explain the BTTG protocol for sending a data packet to a destination node that exists on the same row or same column ring as the source node. Next, details about sending a data packet to a node that falls on a different row and column rings than the source node are presented. The last subsection discusses finding minimal path in the BTTG and shows examples of both nodes on the same and on different rings.

3.2.1 Sending a Data Packet on the Same Row or Column Ring

As explained above one of the scenarios of sending a data packet is from a source node to a destination node on the same ring, row or column. In this case when the source node acquires the corresponding ring token it forwards the data packet to the destination node. An advantage in the BTTG networking topology is the fact that the data packet is sent along a minimal path (explained in subsection 3.2.3) from source to destination.

A node in the grid holds information regarding its Row ID, Col ID, and the max number of nodes in its row ring and in its column ring. From this available data, the source node forwards the data packet to its neighbor, the receiving node checks if the received data packet destination ID is the same as its own ID, if this is the case then it consumes the data packet, otherwise, it forwards the data packet to the following node in the ring.

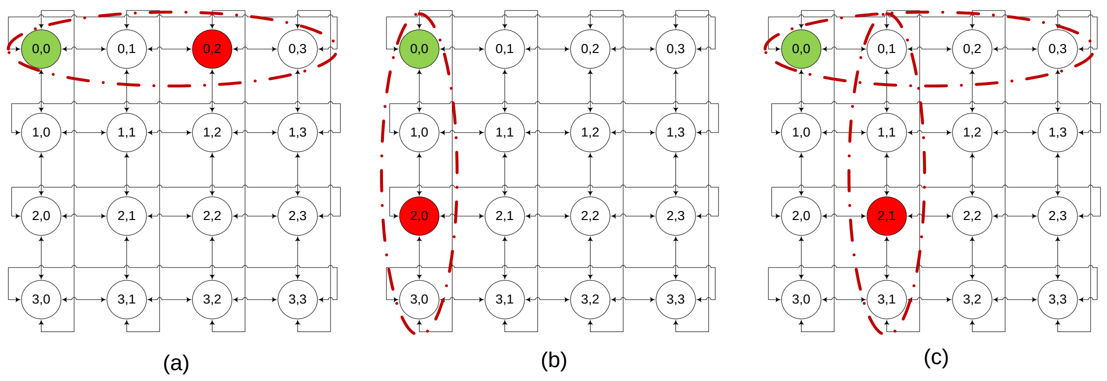


Figure 3.2: An illustration of a 4x4 gird where (a) shows two nodes on the same row ring, (b) shows two nodes on the same column ring, and (c) shows two node on different row and column rings.

Figure 3.2(a) represents the case where both source node and destination node are on the same Row ring; Node (0, 0) source node and Node (0, 2) destination node. In this case, Node (0, 0) waits for the row ring token, once picked up, it sends the data packet to the next node in the row ring, Node (0, 1), then it releases the token. Node (0, 1) is not the intended destination for the packet, it forwards it to the next node in the ring, once the packet reaches Node (0, 2) it is consumed. After forwarding the data packet, Node (0, 0) releases the token to the next node in the chain. (Figure 3.2(b) represents the case where both source node and destination node are on the same Column ring; Node (0, 0) source node and Node (2, 0) destination node).

3.2.2 Sending a Data Packet on a Different Row and Column Ring

As a second scenario, the source node requests to send a data packet to a destination node that is not on the same row ring and column ring. In this case, after acquiring a ring (row or column) token, the source node forwards the data packet on its row ring or column ring (taking into consideration the minimal path explained in subsection 3.2.3), after the packet is fully transmitted the token is released. Once the packet is received by a node on the communication ring, the node checks the packet to identify if it is the intended destination node, if not, the node checks if the destination RowID (ColID case of column ring) of the data packet is the same as its RowID (ColID if column ring), if this is the situation, the data packet is forwarded to the next node in the ring. Otherwise, the capturing node checks if the data packet destination has the same ColID (RowID if column ring), if this is the case, then the node adds the data packet to its data queue and waits to acquire the other ring token to forward the packet to the intended destination node. Once the other ring token is acquired, the node releases the packet to the next node in the ring, the packet circulates until it is captured by the destination node then it is consumed. Figure 3.3 shows the case of a 4x4 BTTG grid where the source node Node (0, 0) and destination node Node (2, 1) are on different Row and Column rings. The example assumes that Node (0, 0) acquires the row ring token first, and then it releases the data packet to the next node on the row ring. Node (0, 1) captures the data packet; Node (0, 1) is not the intended destination, but the Col ID of the data packet's destination node is equal to the Col ID of Node (0, 1), thus, Node (0, 1) adds the data to its column ring data queue and waits for the column ring token. Once the column token is captured, Node (0, 1) releases the data packet to the next node in the column ring, the packet reaches Node (1, 1), this node forwards the packet along the ring, then it reaches Node (2, 1), the intended destination, this node consumes the data packet.

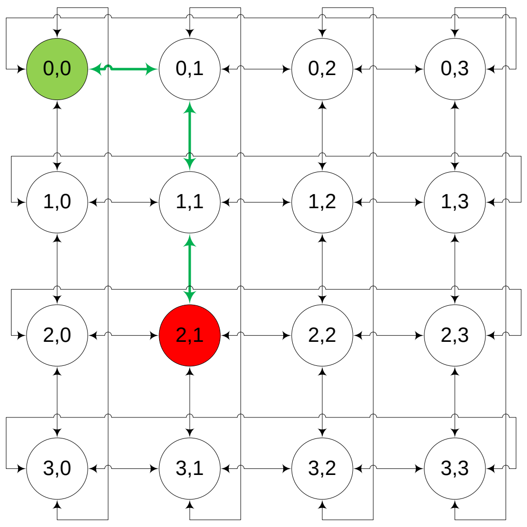


Figure 3.3: 4x4 BTTG grid with source Node (0, 0) and destinatoin Node (2, 1)

3.2.3 Sending a Data Packet on the Minimal Path

As already stated, the rings in the BTTG have bidirectional communication links thus the node has the ability to send and receive information along the same link. On the other hand, the above capability of the BTTG makes it possible to forward data packets along the path that minimizes the cost of transmission.

In normal mode operation, for the previous networking topologies based on the token grid explained in chapter 2, a data packet is always forwarded in one path direction, the path direction of the ring Token circulation (right for a row ring and down for a column ring). However, in the BTTG a node has the ability to forward a data packet right/left on a row ring and down/up on a column ring thus enabling the data packet to be delivered over a minimal path.

In the following subsections we will explain how the minimal (shortest) path is discovered for a communication between two nodes laying on the same row or column ring and on two different row and column rings. Finally, a couple of illustrative examples are presented.

3.2.3.1 Minimal Path

The ring of BTTG grid can be observed as an undirected graph with nodes as vertices and links between nodes as edges, with the graph having equal edge weights. Figure 3.4 represents two illustrations of undirected graphs.

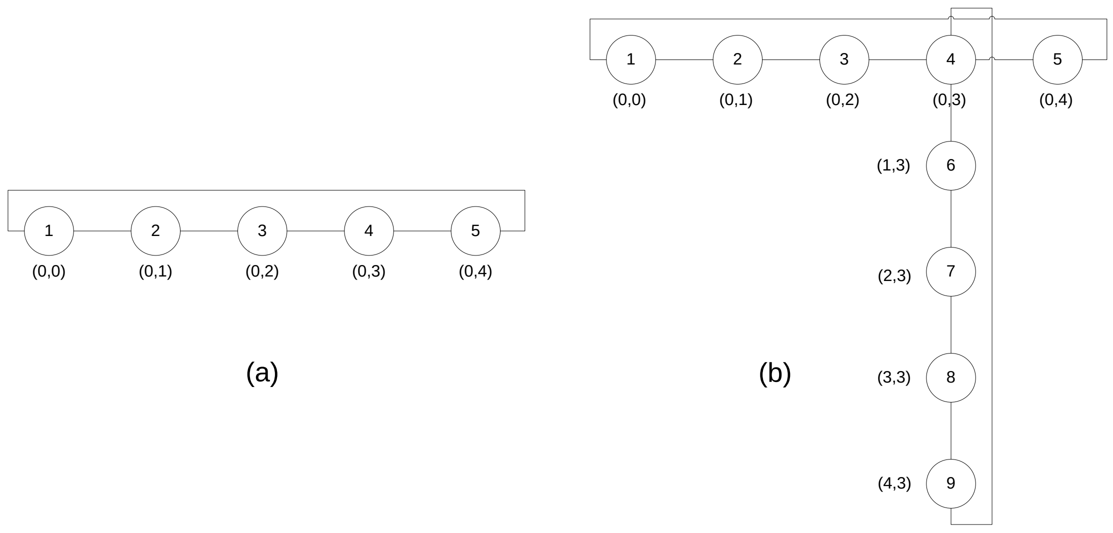


Figure 3.4: (a) A graph with 5 vertices representing a 5 nodes row or column token ring in the BTTG, (b) a graph with 9 vertices representing two 5 nodes rings row and column intersecting at a node in a BTTG.

As a first situation; let us find the shortest path from Vertex (4) to Vertex (3) in Figure 3.4(a). The paths to consider are {4-5-1-2-3} and {4-3}. The total weight of path {4-5-1-2-3} is 4 and the total weight of path {4-3} is 1, thus, path {4-3} is the shortest path. Translating this fact to the BTTG ring, the paths from Node (0, 3) to Node (0, 2) can be either {(0, 3) - (0, 4) - (0, 0) - (0, 1) - (0, 2)} with cost 4 or {(0, 3) - (0, 2)} with cost 1, the second path is the minimal path having the lower cost.

Next, let us find the shortest path from Vertex (1) to Vertex (8) in Figure 3.4(b). The paths to consider are {1-2-3-4-6-7-8} with cost 6, {1-2-3-4-9-8} with cost 5, {1-5-4-6-7-8} with cost 5, and path {1-5-4-9-8} with cost 4. In this case path {1-5-4-9-8} has the lowest cost and is considered the shortest path. Translating the above example to the BTTG grid, the minimal path form Node (0, 0) to Node (3, 3) will be {(0, 0) - (0, 4) - (0, 3) - (4, 3) - (3, 3)} (denoted as *Path (0)*) with cost 4. As Node (0, 3) is the intersecting node between the row and column ring and the communication between Node (0, 0) and Node (3, 3) should pass by Node (0, 3) for this reason the minimal path cost between Node (0, 0) and Node (3, 3) should be equal to the summation of the minimal paths costs between Node (0, 0) to Node (0, 3) and Node (0, 3) to Node (3, 3). As a proof, the minimal path to send a data packet from Node (0, 0) to Node (0, 3) is path {(0, 0) - (0, 4) - (0, 3)} (denoted as *Path (1)*) with cost 2 and the minimal path to send a data packet from Node (0, 3) to Node (3, 3) is path {(0, 3) - (4, 3) - (3, 3)} (denoted as *Path (2)*) with cost 2. Resulting in the following:

Cost Path (0) = Cost Path (1) + Cost Path (2) = 2+2 = 4

In what follows, the minimal path discovery protocol used by the nodes of the BTTG is explained.

3.2.4 Discovering the Minimal Path by the Node of BTTG

Initially, the node to the top left of the token grid is configured manually with Row ID = 0 and Col ID = 0. At system start up, two pre-initiation tokens are released from Node (0, 0) on the row and column level respectively. The pre-initiation tokens holds information about the initiating node and a counter that is incremented at each node before being forwarded. Each of the nodes that receive the token at the row level, set their Row ID = Token Initiating Node Row ID and their Col ID = Token Initiating Node Col ID + Counter. On the column level the node that receive the token sets its Row ID = Token Initiating Node Row ID + Counter and their Col ID = Token Initiating Node Col ID. The Nodes at the row level once they set their ID they initiate a token to their column nodes for them to set their Row and Col IDs in the same way.

Every node in the BTTG holds information regarding its Row ID, Col ID, the number of nodes in its row ring, the number of nodes on its column ring and if the link on a ring is related to a node with lower or higher index. For instance Node (1, 2) in Figure 3.5 holds the following information: Row ID = 1, Col ID =2, Number

Row Ring Nodes = 5, Number Col Ring Nodes = 5, Right and Down link are connected to a higher node index, and Left and Up link are connected to a lower node index.

The above information is enough to allow the node to take the decision regarding the path direction the node needs to take. The pseudo code of the algorithm used by a sending node to determine which path is minimal is presented below:

```
// case of column ring communication
If (Destination Row ID > Row ID)
        Down Path Cost = Destination Row ID - Row ID;
        Up Path Cost = Row ID + Total Number of Node on Row - Destination Row ID;
Else
        Down Path Cost = Destination Row ID + Total Number of Node on Row - Row ID;
        Up Path Cost = Row ID - Destination Row ID;

// case of row ring communication
If (Destination Col ID > Col ID)
        Right Path Cost = Destination Col ID - Col ID;
        Left Path Cost = Col ID + Total Number of Node on Col - Destination Col ID;
Else
        Right Path Cost = Destination Col ID + Total Number of Node on Col - Col ID;
        Left Path Cost = Col ID - Destination Col ID;
```

If we consider the first example presented in subsection 3.2.3.1, we have Node (0, 3) requesting to send a data packet over its row ring to Node (0, 2). Node (0, 3) should decide whether to send the packet either using left or right ring direction based on the shortest path to the destination. The information that Node (0, 3) can use for the decision are as per below:

1. Row ID: 0
2. Col ID: 3
3. Total number of nodes in row ring: 5
4. Node to the Right has higher Col ID
5. Node to the Left has lower Col ID
6. Destination Row ID: 0
7. Destination Col ID: 2

If we apply the algorithm presented previously for the row ring case we get the following results:

Right Path Cost = Destination Col ID + Total Number of Node on Col - Col ID
= 2 + 5 – 3
= 7 – 3
= 4

Left Path Cost = Col ID - Destination Col ID
= 3 – 2
= 1

The ***Left Path Cost < Right Path Cost*** for this reason Node (0, 3) should forward the data packet to the path at its left; thus the path {(0, 3) – (0, 2)} is adopted.

Considering a bit more complex example, presented in figure 3.5(a), where a node sends a packet to a node that is not its direct neighbor; Node (2, 2) wants to send a data packet to Node (2, 0). In this case the Node (2, 2) needs to decide on the minimal path direction; using the previous algorithm:

Right Path Cost = Destination Col ID + Total Number of Node on Col - Col ID
= 0 + 5 – 2
= 5 – 2
= 3

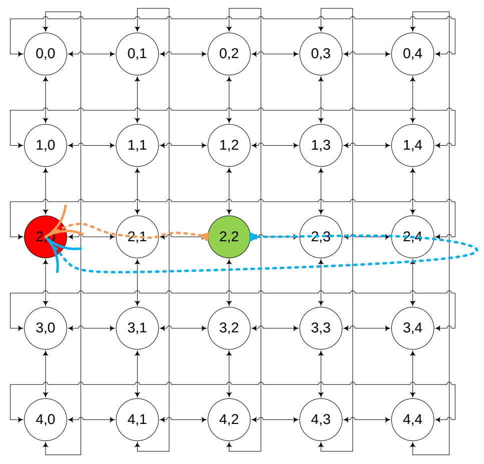
Left Path Cost = Col ID - Destination Col ID
= 2 – 0
= 2

The ***Left Path Cost < Right Path Cost*** for this reason Node (2, 2) forwards the data packet to the path at its left, the packet reaches Node (2, 1) it uses the same way to determine the minimal path direction:

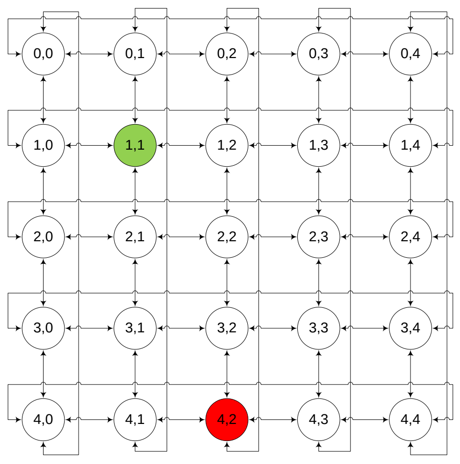
Right Path Cost = Destination Col ID + Total Number of Node on Col - Col ID
= 0 + 5 – 1
= 5 – 1
= 4

Left Path Cost = Col ID - Destination Col ID
= 1 – 0
= 1

The ***Left Path Cost < Right Path Cost*** for this reason Node (2, 1) forwards the data packet to the path at its left, the packet reaches Node (2, 0) it is the destination node and it consumes the packet.



(a)



(b)

Figure 3.5: (a) A 5x5 grid with source Node (2, 2) (Green) and destination Node (2, 0) (Red). The Blue path is the path taken by the unidirectional token grid, the Orange path is the minimal path followed by the BTTG protocol. (b) is another example with a source node and destination node each on a different row and column rings.

Next, the case where both sending node and destination node fall on different row and column rings is considered. In this case, the minimal path the data packet should follow is the minimal path from the source node to a node that intersects with the ring of the destination node, then the minimal path from the intersecting node to the destination node. For instance in the example shown in Figure 3.5(b), Node (1, 1) desires to send a data packet to Node (4, 2), in this case Node (1, 1) can send the data over the row ring first to Node (1, 2) or over the column ring to Node (4, 1), where both nodes intersect with the rings of Node (4, 2). As both cases are dealt with in the same way, let us consider that Node (1, 1) wants to send the packet over the column ring first. In this case Node (1, 1) computes the cost of sending over the upper path as opposed to the lower path:

Up Path Cost ***= Row ID + Total Number of Node on Row - Destination Row ID***

= 1 + 5 – 4

= 6 – 4

= 2

Down Path Cost ***= Destination Row ID - Row ID***

= 4 – 1

= 3

From the above computation ***Up Path Cost < Down Path Cost*** thus Node (1, 1) sends the packet over the upper path, the data reaches Node (0, 1) that in its turn computes the paths costs:

Up Path Cost	***= Row ID + Total Number of Node on Row - Destination Row ID***
	= 0 + 5 – 4
	= 5 – 4
	= 1

Down Path Cost	***= Destination Row ID - Row ID***
	= 4 – 0
	= 0

Up Path Cost < Down Path Cost the decision of Node (0, 1) is to send the packet over the upper path. The data reaches Node (4, 1), the intersecting node, it adds the packet to its data queue. Once it acquires the row token, it decides whether to forward the data to the left or right path. Using the above, the path costs are as per below:

Right Path Cost	***= Destination Col ID - Col ID***
	= 2 – 1
	= 1

Left Path Cost	***= Col ID + Total Number of Node on Col - Destination Col ID***
	= 1 + 5 – 1
	= 6 – 1
	= 5

Right Path Cost < Left Path Cost the decision of Node (4, 1) is to send the packet over the right path, the data reaches Node (4, 2) the destination node, the data packet is consumed.

Finally, as a conclusion, a node in the BTTG grid doesn’t need to hold information about the full path for communication, it just deduces the minimal path direction from source node to destination node, and forwards the packet along this path. Each node along the ring does the same until the data packet reaches the intended destination.

In the following sections, the BTTG’s fault tolerance routing scheme in case of link and/or node failures is discussed.

3.3 Fault Tolerance

Harsh environments make a textile and any electronic part or wire inside of it susceptible to damages and tears resulting in a communication link failure. Thus, Fault tolerance is of a great importance in electronic textile systems. In what follows the fault tolerant scheme adopted by the BTTG is discussed. As a convention we will consider that a node failure is equivalent to a failure of all links related to the node.

3.3.1 Error Detection

Each node in the BTTG is connected to two different rings (row and column rings). There are four different possibilities for link errors illustrated in Figure 3.6. An error can occur on the upper, lower, left, or right links of a node. The error can affect the node itself, equivalent to four link failures at the same time.

Each node keeps two separate Idle State Counters each related to one of the node rings, and four separate flags that indicate the link's status. Initially the counters are set to zero and they are reset every time the node receives data (token or other type of packets) over the ring. If any of these counters reaches a preset value (empirically set) an idle state is declared at the ring level and an error test is initiated to test the corresponding links.

To check the link connections, the node sends a link test token to each of the gates in the corresponding ring; this token will be received by the neighbor node and then automatically resent back to the original sending node through the same link. Once the original node sends the link test tokens it initiates a time out counter, if the counter reaches a pre-specified value without receiving back a link test token, the corresponding link is flagged to be in failure mode and the node stops forwarding data through it.

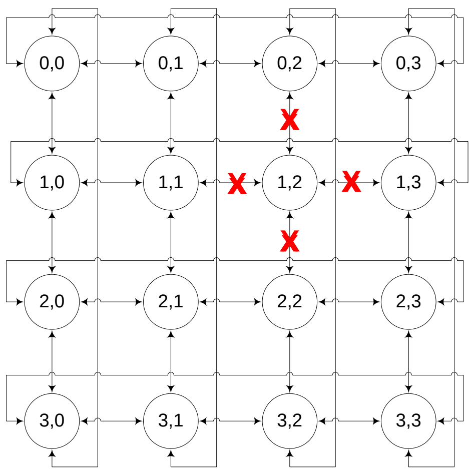


Figure 3.6: four different link error possibilities on a node level, if this is the case the node can no more interact with the network.

In case a link failure is detected, the error information is not propagated and the token and data packet forwarding is handled locally at each node. The sending node in this case has error information regarding the direct link connections. Thus, the node sends the data packet on the links without error.

3.3.2 Token Handling in Case of Fault

A link failure along a ring will destroy the token circulating over the ring. The BTTG ring will not be disabled but instead a new token is reinitiated on the ring. Once a node detects a link failure, if the node's left link (up link in case of a column ring) is damaged and the node's right ring (down link in case of a column ring) is not damaged, the node will re-initiate a token with a U-Shape mode. U-Shape token, holds information regarding the Row ID and Col ID of the initiating node called "start node" and the last node in the partial ring called "end node". A U-Shape token is inactive and a node cannot use it for communication unless both start and end nodes are set. Once initiated, the U-Shape token will have the Start Row ID and Start Col ID set to the Row ID and Col ID of the initiating node then it is forwarded along the ring. When the token reaches a node with a detected link failure in its right link (up link case of column ring) the node will set the End Row ID and End Col ID for the token to be equal to its Row ID and Col ID, the token is now active, its direction flow is reversed and it is forwarded in the opposite direction.

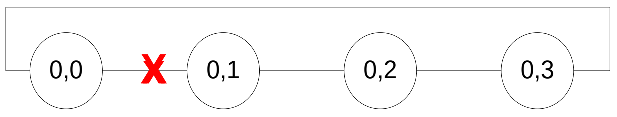


Figure 3.7: a ring with a link failure between Node (0, 0) and Node (0, 1)

Figure 3.7 displays a case where a link failure occurs between Node (0, 0) and Node (0, 1). In this example Node (0, 0) detects a failure at the right link level, and Node (0, 1) detects a failure at the left link level. As Node (0, 1) is the node with left link damaged and right link intact, this node will initiate a U-Shape token, it will set its Start Row ID = 0 and Start Col ID = 1 and the token is forwarded through the right link, it reaches Node (0, 2) then Node (0, 3), finally the token reaches Node (0, 0), this node already detected a failure at its right link, it sets the End Row ID and End Col ID of the token, changes its direction, and it forwards the token in the opposite direction back to Node (0, 3) through the left link. It is important to note that a node will forward a token along the preset direction and is not allowed to change this direction unless it has a detected link failure.

From the above, it can be deduced that in the case where many failures occur at the level of a BTTG ring, multiple partial rings are formed and nodes in a partial ring can communicate with each other and with the remaining part of the grid through the connections they hold with other column or row rings. For instance, in Figure 3.8, multiple link failures occurred at the level of the BTTG grid. At the level of row 0, we have 2 new partial rings, at the level of columns 1 and 2 we have two new partial rings. Node (0, 2) has only one intact link with Node (0, 1) using this link, Node (0, 2) can communicate with all the nodes reachable by Node (0, 1) in the BTTG.

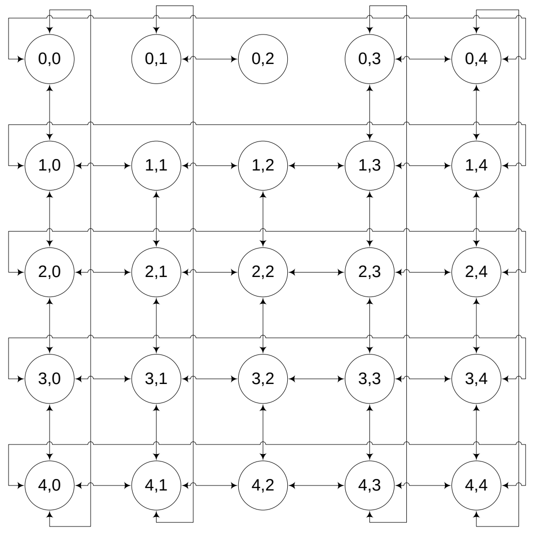


Figure 3.8: a 5x5 BTTG grid with multiple faults and new formed partial rings

For instance if Node (0, 2) wants to send a data packet to Node (0, 0), it waits to acquire the U-Shaped token and forwards the packet to Node (0, 1) in its turn Node (0, 1) forwards the token to Node (4, 1) on its column ring. Node (4, 1) forwards the packet to Node (4, 0) and it is finally delivered to Node (0, 0) the destination node.

In the following subsection, we will discuss the data packet handling process in case of faults in the BTTG.

3.3.3 Data Packet Handling in Case of Faults

A node with link failure is able to send packets to other nodes on the BTTG grid unless it is totally isolated from the network and all the links connecting the node to the grid are disabled or the node itself is disabled. After detecting the errors and re-initiating tokens , a node on a partial ring that desires to send a data packet, waits as usual to receive a token (active token if U-Shape mode); once the token is captured the node will send the packet along the appropriate path.

The following subsections discuss the data packets forwarding protocol in case of faults. The discussed cases take into consideration a destination node at the same ring level of the source node and at a different row and column rings levels.

3.3.3.1 Sending a Data Packet on the Same Row or Column Ring

A node in the BTTG grid connected to a partial ring needs to send a data packet at the same ring level. The node has to wait for the U-Shape active token. To explain the protocol of forwarding the data packet, the row ring case is explained in what follows (same protocol is applied for the column ring with replacement of Column IDs with Row IDs check). When the node captures the U-Shaped token it follows the below steps to decide on the right path to send the node:

1. Flags the data packet as *Fixed Direction*[1] to be handled correctly in the next node and not be sent back in case the minimal path is in error.
2. If the Token Start Column ID is smaller than the Token End Column ID
   a. In case the Destination Column ID of the data packet is equal to the Token Start Column ID, then the packet is forwarded on the left path.
   b. In case the Destination Column ID of the data packet is equal to the Token End Column ID, then the packet is forwarded on the right path.
   c. In case the Destination Column ID of the data packet is between the Token Start Column ID and Token End Column ID, if the Source Column ID of the data packet is smaller than

[1] A node that receives a data packets with a Fixed Direction flag, should forward the flag in the same direction. It does not handle the packet based on the minimal path forwarding anymore.

its Destination Column ID the packet is forwarded on the right path else it is forwarded on the left path.

   d. Otherwise the destination node is not reachable through the partial ring. The data packet is flagged as *Cross Ring*[2] and it is dealt with as being needed to be forwarded into the column ring.

3. If the Token Start Column ID is greater than the Token End Column ID
   a. In case the Destination Column ID of the data packet is equal to the Token Start Column ID, the packet is forwarded on the left path.
   b. In case the Destination Column ID of the data packet is equal to the Token End Column ID, the packet is forwarded on the right path.
   c. In case the Destination Column ID is smaller than both Token Start and Token End Column IDs, the data packet is forwarded left if its Source Colum ID is greater than its Destination Column ID and smaller than or equal to its Token End Col ID, else it is forwarded right.
   d. In case the Destination Column ID is greater than both Token Start and Token End Column IDs, the data packet is forwarded right if its Source Colum ID is smaller than its Destination Column ID and greater than or equal to its Token End Col ID, else it is forwarded left.
   e. Otherwise the destination node is not reachable through the partial ring. The data packet is flagged as *Cross Ring* and it is dealt with as being needed to be forwarded into the column ring.

[2] A node that receives a data packet that is flagged as Cross Ring, it insert the packet in its data queue to be handled later, and does not return it to its source as it is known to be in error and unable to handle it.

A detailed algorithm that shows both row and column ring cases is presented below:

```
//Case of a Row Ring
Flag the data packet as fixed direction
{
    if (Token Start Col ID < Token End Col ID)
    {
        if(Destination Col ID == Token Start Col ID)
        {
            Send on Left Path
        }
        else if(Destination Col ID == Token End Col ID)
        {
            Send on Right Path
        }
        else if(Destination Col ID > Token Start Col ID && Destination Col ID < Token End Col ID)
        {
            if (Source Col ID < Destination Col ID)
            {
                Send on Right Path
            }
            else
            {
                Send on Left Path
            }
        }
        else
        {
              Flag the data packet as Cross Ring
             Wait for Column Token and Forward over Column down path if not disabled else over up path
             (in this case the first node to capture the data packet queues it and waits for the row token to forward
             the token as explained in sections 3.2.2 and 3.2.3)
        }
    }
    else if (Token Start Col ID > Token End Col ID)
    {
        if(Destination Col ID == Token Start Col ID)
        {
            Send on Left Path
        }
        else if(Destination Col ID == Token End Col ID)
        {
            Send on Right Path
        }
        else if(Destination Col ID < Token Start Col ID && Destination Col ID < Token End Col ID)
        {
            if (Source Col ID < Destination Col ID)
            {
                Send on Right Path
            }
            else if (Source Col ID > Destination Col ID && Source Col ID <= Token End Col ID)
            {
                Send on Left Path
            }
            else if (Source Col ID > Destination Col ID && Source Col ID >= Token Start Col ID)
            {
                Send on Right Path
            }
        }
        else if(Destination Col ID > Token Start Col ID && Destination Col ID > Token End Col ID)
        {
            if (Source Col ID > Destination Col ID)
            {
                Send on Left Path
            }
            else if (Source Col ID < Destination Col ID && Source Col ID <= Token End Col ID)
            {
                Send on Left Path
            }
            else if (Source Col ID < Destination Col ID && Source Col ID >= Token Start Col ID)
            {
                Send on Right Path
            }
        }
```

```
          else
          {
              Flag the data packet as Cross Ring
              Wait for Column Token and Forward over Column down path if not disabled else over up path
              (in this case the first node to capture the data packet queues it and waits for the row token to forward
              the token as explained in sections 3.2.2 and 3.2.3)
          }
       }
    }



//Case of a Col Ring
Flag the data packet as fixed direction
{
       if (Token Start Row ID < Token End Row ID)
       {
          if(Destination Row ID == Token Start Row ID)
          {
             Send on Upper Path
          }
          else if(Destination Row ID == Token End Row ID)
          {
             Send on Lower Path
          }
          else if(Destination Row ID > Token Start Row ID && Destination Row ID < Token End Row ID)
          {
             if (Source Row ID < Destination Row ID)
             {
                Send on Lower Path
             }
             else
             {
                Send on Upper Path
             }
          }
          else
          {
              Flag the data packet as Cross Ring
              Wait for Row Token and Forward over Row Right Path if not disabled else over left path
              (in this case the first node to capture the data packet queues it and waits for the row token to forward
              the token as explained in sections 3.2.2 and 3.2.3)
          }
       }
       else if (Token Start Row ID > Token End Row ID)
       {
          if(Destination Row ID == Token Start Row ID)
          {
             Send on Upper Path
          }
          else if(Destination Row ID == Token End Row ID)
          {
             Send on Lower Path
          }
          else if(Destination Row ID < Token Start Row ID && Destination Row ID < Token End Row ID)
          {
             if (Source Row ID < Destination Row ID)
             {
                Send on Lower Path
             }
             else if (Source Row ID > Destination Row ID && Source Row ID <= Token End Row ID)
             {
                Send on Upper Path
             }
             else if (Source Row ID > Destination Row ID && Source Row ID >= Token Start Row ID)
             {
                Send on Lower Path
             }
          }
          else if(Destination Row ID > Token Start Row ID && Destination Row ID > Token End Row ID)
          {
             if (Source Row ID > Destination Row ID)
             {
                Send on Upper Path
             }
             else if (Source Row ID < Destination Row ID && Source Row ID <= Token End Row ID)
```

```
            {
               Send on Upper Path
            }
            else if (Source Row ID < Destination Row ID && Source Row ID >= Token Start Row ID)
            {
               Send on Lower Path
            }
         }
         else
         {
            Flag the data packet as Cross Ring
            Wait for Row Token and Forward over Row Right Path if not disabled else over left path
            (in this case the first node to capture the data packet queues it and waits for the row token to forward
            the token as explained in sections 3.2.2 and 3.2.3)
         }
      }
   }
```

For instance Figure 3.9(a) shows an example where Node (2, 2) desires to send a data packet to Node (2, 0) with link failure between Node (2, 0) and Node (2, 1). Assuming that the link error is already detected and the U-Shape token is circulated and active, once Node (2, 2) receives the token it checks using the above protocol to decide whether to send the data over the left or right link path. Destination Col ID = Token End Col ID = 0 for this reason the data packet is flagged as fixed direction and is forwarded along the right path to Node (2, 3). As the data packet direction is fixed the other nodes are obliged to forward it the same direction until it reaches the destination Node (2, 0) where it is consumed.

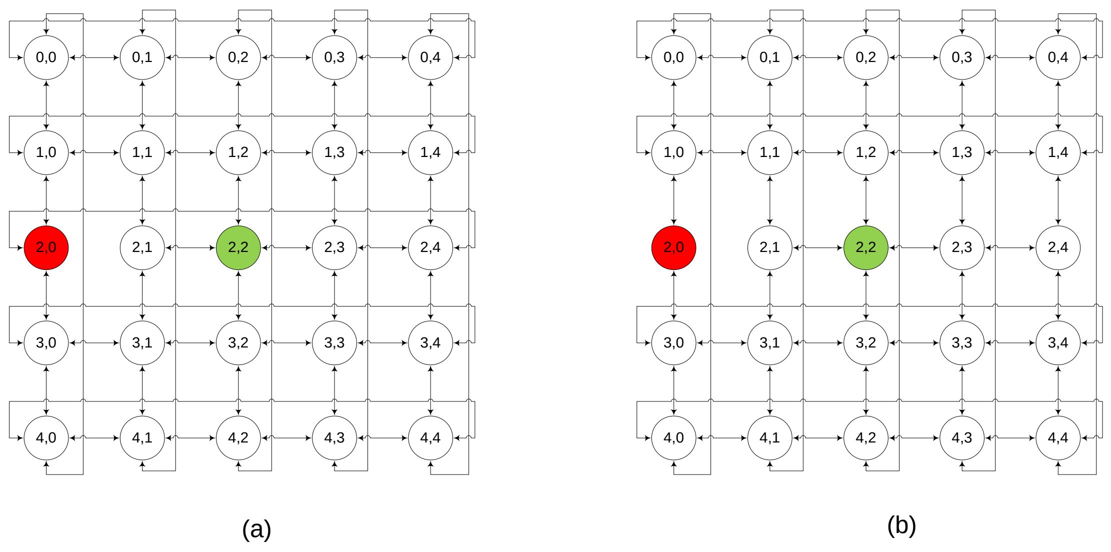


Figure 3.9: (a) Node (2, 2) sending a data packet for Node (2, 0) with link failure between Node (2, 0) and Node (2, 1). (b) Node (2, 2) sending a data packet for Node (2, 0) with two links failures isolating Node (2, 0) from the ring of Node (2, 2)

Another more complex example is shown in Figure 3.9(b) where Node (2, 2) desires to send a data packet to Node (2, 0) that is isolated from the row ring of Node

(2, 2). In this case Node (2, 2) waits to acquire the U-Shape token as in the previous example and checks using the above algorithm regarding the appropriate gate over the row ring to forward the packet, however due to the fact that the Destination Col ID falls outside the partial ring of the source node, Node (2, 2) flags the data packet as Cross Ring, adds it to the data queue, and waits to capture the column token, then it forwards the data packet through the down path over the column ring. Node (3, 2) receives the data packet flagged as Cross Ring, it acts as the sender of the packet and adds it to the data queue and waits for the row ring token and then forwards the data packet to an intersecting node, Node (3, 0), (as explained previously in sections 3.2.2 and 3.2.3 above) then the node is forwarded to the destination node where it is consumed.

3.3.3.2 Sending a Data Packet on a Different Row and Column Ring

Whenever a node tries to connect with a node on a different row and column rings, the source node waits until it acquires a token (row or column) once the token is acquired the data will be forwarded over the appropriate ring until it reaches a node that intersects with the destination node, if the ring is in error the algorithm explained in section 3.3.3.1 is considered, and once the packet reaches an intersecting node it will be handled based on what was previously discussed.

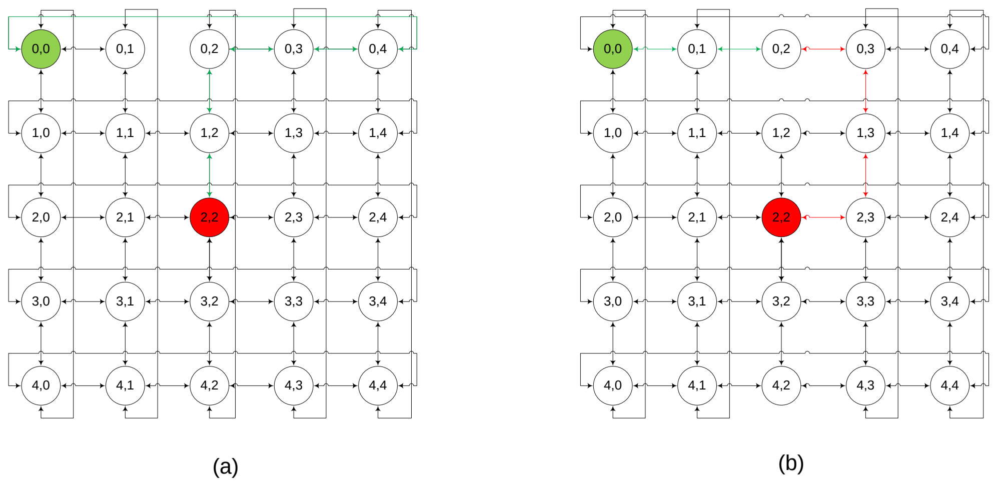


Figure 3.10: (a) Node (0, 0) sending a data packet for Node (2, 2) with link failure between Node (0, 1) and Node (0, 2). (b) Node (0, 0) sending a data packet for Node (2, 2) with two link failures isolating Node (0, 2) from the column ring.

For instance, Figure 3.10(a) depicts an example where Node (0, 0) wants to send a data packet to Node (2, 1). In this example Node (0, 0) captures the row U-shaped token, the link between Node (0, 1) and Node (0, 2) the intersecting node is in error, for this reason, Node (0, 0) sends the data packet to Node (0, 2) through the left gate (not on the minimal path). Once the data packet reaches Node (0, 2) it adds it to the data queue. Node (0, 2) waits to capture the column token then it forwards the data packet to the destination, Node (2, 2).

Another more complex example is shown in Figure 3.10(b) where Node (0, 0) wants to send a data packet to Node (2, 2) with links between Nodes (0, 1), Node (0, 2) and Node (0, 2), Node (1, 2) and Node (0, 2), Node (4, 2) are all in error. In this case if Node (0, 0) acquires the row token first, it doesn't know that the intersecting node is disconnected from its column ring and sends the data packet to Node (0, 2). Once Node (0, 2) receives the data packet, it flags it as forced and sends it to Node (0, 3). When Node (0, 3) receives the data packet with forced option, it adds it to the data queue and acts as a sender of the packet knowing that it cannot be sent to the node on its left gate as it was received in a forced direction mode. Thus, Node (0, 3) waits for the column token and sends the data packet to Node (2, 3) the intersecting node. Once the data packet reaches Node (2, 3) this node adds it to the data queue and waits for the row token to forward it to the destination node, Node (2, 2).

This chapter presented the BTTG token grid network topology, the BTTG operation was discussed in normal fault-free condition along with the minimal path routing scheme. The BTTG operation in a faulty environment was presented.

The next chapter discusses the simulation results of the BTTG networking topology and compares them to the simulation results of the TTG and TTGNM. It demonstrates that the BTTG transmission cost is better than the other topologies' costs, and the fault tolerant scheme of the BTTG is more reliable.

CHAPTER FOUR

SIMULATION

This chapter starts by discussing the transmission costs in the BTTG networking scheme. Afterwards, it compares the costs with the TTG and TTGNM networking schemes costs. Simulation results of the BTTG, TTG, and TTGNM are also presented and the efficiency offered by the BTTG over the other schemes is deduced.

4.1 TTG Costs

In order to demonstrate the efficiency of the BTTG networking scheme, the enumerations of the transmission costs from a source node to a destination node is discussed in the following sections. Two costs will be introduced, the first cost represents the cost of transmission for the source node, and the second represents the number of hops for a data packet in the network to reach the destination.

4.1.1 Fault-Free BTTG Grid

The transmission cost of sending a data packet to a node on the same row or column ring of a fault free BTTG grid is first reflected. The source node initially waits for the ring token, the waiting time is denoted by TW_{token}. Once the token is captured, the node sends the data packet to the destination node then it releases the token. The time needed to send the data packet is denoted by TS_{data}. The operation of the BTTG network requires that a token is always rotated around the ring same as for the TTG in [Nakad 2003]; TS_{data} is considered to be the extra cost of transmitting the data packet. Table 4.1 presents a set of variables that will be used in the discussion that follows. Two assumptions were taken to simplify the discussion:

1. The current node is not holding the token but it released it to the ring previously. Thus, the only parameter to consider in the case of a node waiting to receive a token is the token waiting time, and any other delays are considered to be negligible.

2. Link latencies and node processing time are considered to be insignificant.

Table 4.1: Terms and Variables

Variable		**Value**	**Description**
Grid		-	A BTTG network grid, with nodes connected in row and column rings.
Ring		-	Row or Column
N		-	Number of nodes in a row or column ring
N^2		-	Number of nodes in a grid (case considered is a square grid)
BW		-	Bandwidth of the links communicating nodes
TS_{token}		$Size_{token}$ / BW	The Time needed to transmit a token
TW_{token}			Time waiting to receive a token
No Fault	TW_{token} (Minimum)	TS_{token}	previous node is sending the token
	TW_{token} (Maximum)	$(N – 1)*TS_{token}$	next node is sending the token
	TW_{token} (Average)	$(N/2)*TS_{token}$	Average token waiting time
TW_{token}			Time waiting to receive a token
With Fault	TW_{token} (Minimum)	TS_{token}	previous node is sending the token
	TW_{token} (Maximum)	$[(N – 2) + (N – 1)]*TS_{token}$ = $(2N – 3)*TS_{token}$	next node is sending the token
	TW_{token} (Average)	$(N – 1)*TS_{token}$	Average token waiting time
TS_{data}		$Size_{data}$ / BW	The Time needed to send a data packet

Considering the above stated assumptions, TW_{token} = (# of hops)*TS_{token}. For instance in Figure 4.1, if Node (0, 0) is the sending node, then TW_{token} has a minimum value of TS_{token} when the token is in Node (0, N-1) and a maximum value of (N-1)TS_{token} when the token is at Node (0, 1). There are (N-2) cases for this ring with wait values of 1, 2, 3, 4… N-1 multiples of TS_{token}: the resulting average will be:

$(TS_{token} + (N-1)TS_{token})/2 = (N/2)TS_{token}$.

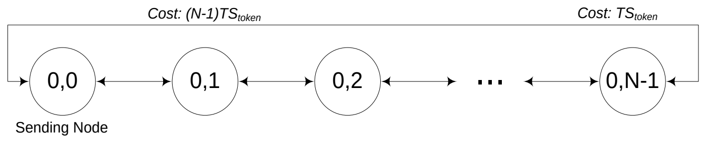


Figure 4.1: A ring with N nodes, with Node (0, 0) as sending node.

Another case to consider is the transmission cost of sending a data packet to a node that lies on a different row and column rings in the fault free BTTG grid. The source node initially waits for the ring token, the waiting time is denoted as TW_{token}. Once the token is captured, the node sends the data packet to a node that intersects with the destination node ring, the time needed is TS_{data}. The intersecting node in its turn waits for the corresponding token ring (TW_{token}). Once the token is captured, the intersecting node sends the data packet to the destination node then the node releases the token. The time needed to send the data packet is another TS_{data}. In this case, the total token waiting time is doubled and the total time to send data is also doubled in comparison the transmission over the same ring. Table 4.1 presents the values and description of variables that will be used in the discussion that follows.

Table 4.2: Sending Node communication timing costs

Variable	**Destination Location**	**Total Time**	**Additional Cost**	**Description**
Ring Token	-	TW_{token} + TS_{token}	-	Time Waiting for Token Time to send the Token
Data Packet	Same Ring	TW_{token} + TS_{data} + TS_{token}	TS_{data}	Time Waiting for Token Time to send data Time to send the Token
	Different Row and Column Rings[3]	$2TW_{token}$ + $2TS_{data}$ + $2TS_{token}$	TW_{token} + $2TS_{data}$ + TS_{token}	Time Waiting for Token (for source node and intersecting node) Time to send data packet for two nodes Time to send the token for two nodes

Table 4.2 offers the cost of communication as perceived by the sending node. When the destination node is on the same ring, the source node waits to capture the ring token, sends the data packet, and then releases the token. This operation is the same if the destination is the next node in the ring or the furthest. The increase of cost in sending to a farther node is observed in the number of hops traversed; this cost is reported in Table 4.3. On the other hand, if the destination node is on a different row and column rings, the source node waits to capture the ring token, sends the packet, and then releases the token. The data packet is captured by a node intersecting with the ring of the destination node. The intersecting node waits to capture the corresponding token ring, send the data to the destination node, and then releases the token. In this case, as previously stated, the operation is the same if the destination is the closest or the furthest node in the ring, the cost will increase depending on the number of hops the packet traverses; this cost is identified in table 4.3.

[3] In this case we have two sending nodes for the data packet: a- the initial source node, b- the node intersecting with the source node ring and destination node ring.

As a last case to consider is the operation of the BTTG in a large network, where the number of nodes increases in the network and affects the network performance. Two costs should be reflected, the cost to the sending node and the cost in number of hops traversed to reach the destination. As already shown in Table 4.2, the cost to the sending node depends on TW_{token}, which in its turn depends on the number of nodes N in a ring (as explained in Table 4.1). Thus, the cost to the sending node is directly affected by the number of nodes in a ring N.

Regarding the cost in number of hops traversed to reach the destination, the connection cost in its turn depends on N, the number of nodes in a ring, illustrated in Table 4.3.

Table 4.3: Network Costs in term of Number of Hops

Variable	**Destination Location**	**N (Even or Odd)**	**Total Cost**		**Description**
Ring Token	-	Even/Odd	1		-
Data Packet No Grid Faults	Same Ring	Even	Minimum	1	Destination is the next node
			Maximum	N/2	Destination is the ring middle node relative to the sending node
			Average	(N+2)/4	
		Odd	Minimum	1	Destination is the next node
			Maximum	(N-1)/2	Destination is the ring middle node relative to the sending node
			Average	(N+1)/4	
	Different Row and Columns Rings	Even	Minimum	2	One Row and One Column off
			Maximum	N	(N/2) Rows and (N/2) Columns off
			Average	(N+2)/2	
		Odd	Minimum	2	One Row and One Column off
			Maximum	N-1	((N-1)/2) Rows and ((N-1)/2) Columns off
			Average	(N+1)/2	
Data Packet With Link Failure	Same Ring	Even/Odd	Minimum	1	Destination is the next node
			Maximum	N-1	Destination is the ring middle node relative to the sending node
			Average	N/2	
	Different Row and Columns Rings (Row or Column Ring	Even	Minimum	2	One Row and One Column off
			Maximum	(3N-2)/2	(N-1) Rows/Columns and (N/2) Columns/Rows off
			Average	(3N+2)/4	

	with one link fault)	Odd	Minimum	2	One Row and One Column off
			Maximum	(3N-3)/2	(N-1) Rows/Columns and ((N-1)/2) Columns/Rows off
			Average	(3N+1)/4	
	Different Row and Columns Rings (Row and Column Rings each with one link fault)	Even/Odd	Minimum	2	One Row and One Column off
			Maximum	2N-2	(N-1) Rows and (N-1) Columns/Rows off
			Average	N	

4.1.2 Faulty BTTG Grid

In case of link and/or node failures, the cost of transmission increases in the BTTG grid when this failure is directly affecting the minimal path from the source node to the destination node and whenever the ring of the sending node is affected by the failure. The same measures, already discussed in the previous sub section, are used to determine the overall cost in a faulty BTTG grid.

Table 4.1 presents the token waiting time for a node in case of a fault in the ring as:
TW_{token} = (# of hops)*TS_{token}. As Table 4.1shows, as average, TW_{token} = (N-1) *TS_{token} in case of a ring fault and this value can be deduced by following the example in Figure 4.2. As defined previously in chapter 3, in case of link failure the token navigates in a U-Shaped mode from Node (0, 1) to Node (0, 0) and vice versa. If Node (0, 1) is the sending node, then TW_{token} has a minimum value of TS_{token} when the token is in Node (0, 2) and Node(0, 1) is the next node in the token chain, and a maximum value of (2N-3)TS_{token} when the token is at Node (0, 2); in case the token is being forwarded toward Node (0, 3) thus it has to go all the way to reach Node(0, 0) then return to Node (0, 1) through the reverse path. The resulting average in this situation will be (TS_{token} + (2N-3)TS_{token})/2 = (N-1)TS_{token}.

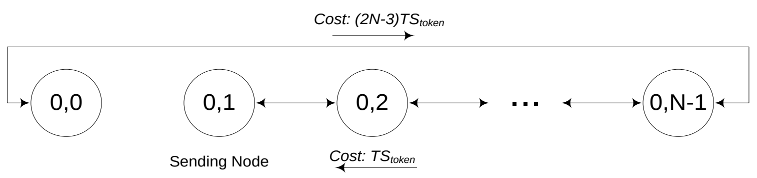


Figure 4.2: A ring with N nodes, with Node (0, 1) as sending node, with link Error between Node (0, 0) and Node (0, 1).

On the other hand, the cost in number of hops a data packet needs to take in order to reach the destination node is also affected by the ring faults. Table 4.3 introduces the costs in number of hops of the three main cases of communication presented in Figure 4.3 below, where part (a) represents the case of a communication with source and destination nodes over the same ring and the ring is in fault. In Part (b) the source node ring is in failure but the destination node ring is fault free. In Part (c) both rings are in fault.

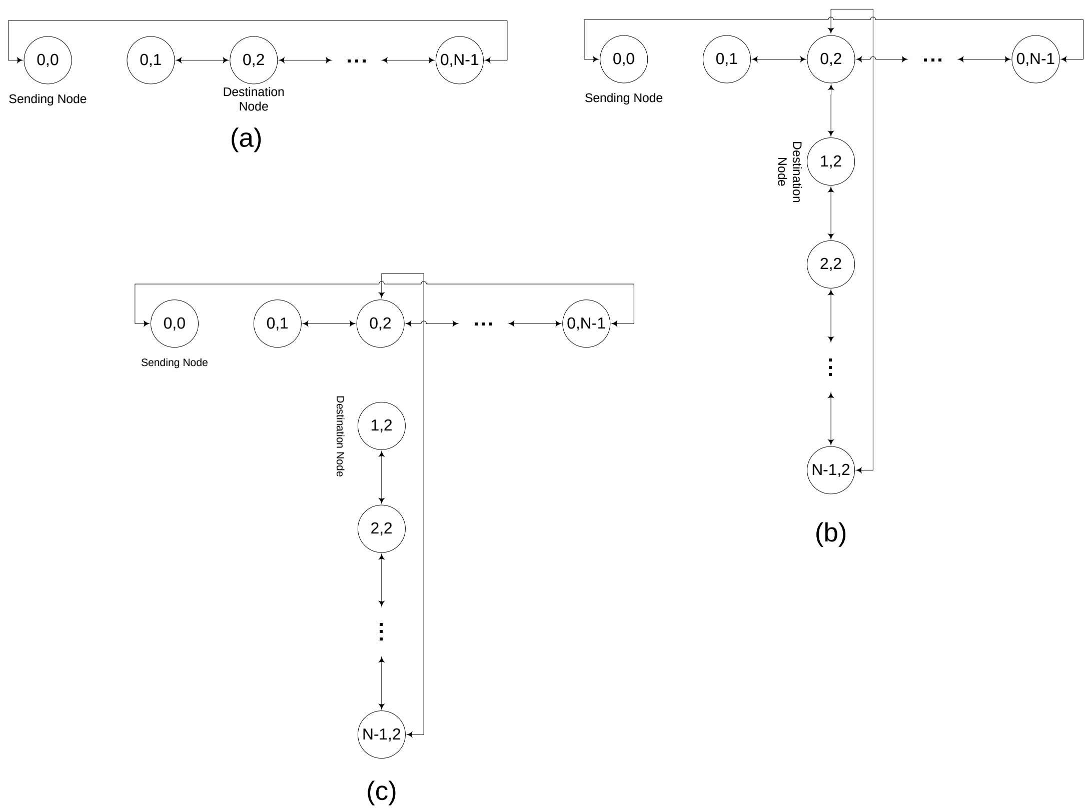


Figure 4.3: Communication between source and destination node where: a- Both on the same ring and the ring is in fault. b- Each node on a different row and column ring and row ring is in fault. c- Each node on a different row and column ring and the two rings are in fault.

As a general case if the data packet needs to cross over more than two rings from source node in order to reach the destination node, the cost in number of hops will be calculated as being the sum of #hops for each ring being in use. For instance, Figure 4.4 presents a case where Node (0, 0) sends a data packet to Node (2, 1), in this case the communication occurs either on the green path, with two rings from three being with a link fault, in this case the cost in number of hops is in average equivalent to (N+2)/4 + (N/2) + (N/2). If the communication occurs on the red path with one ring with link fault, then the average number of hops is equivalent to (N+2)/4 + (N+2)/4 + (N/2).

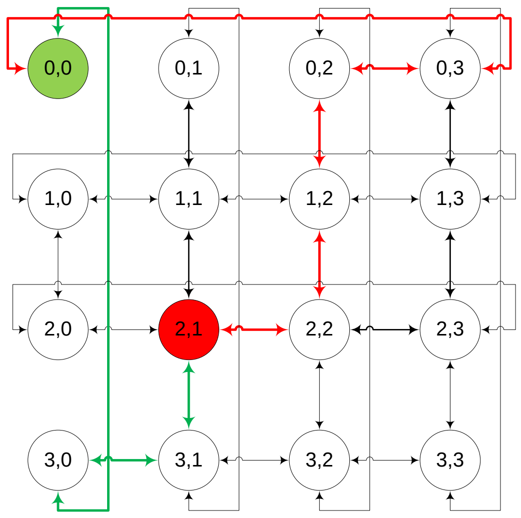


Figure 4.4: Communication from Node (0, 0) to Node (2, 1) should pass by three different rings.

4.2 Analytical vs. Simulated Communication Cost

This section provides an analytical study for the total cost of communication for a N^2 nodes BTTG token grid network with N row and N column rings. The analytical results are then proved by simulation results. The first sub section analyzes the cost of a fault free grid network. The second sub section analyzes the cost of a grid with one or multiple occurring link failures.

In order to quantify the analytical study, an example where Node (0, 0) sends data packets to all the other nodes in the token grid network is used. Once a node receives a data packet from Node (0, 0), it replies back with another packet. Node (0, 0) waits for the reply packet before sending a new data packet. The process ends at the time all the nodes replies back to Node (0, 0). The time spent to finish the communication process is reported afterwards. This time represents an incremental counter that records the cost in terms of token waiting time for a sending node and the number of hops it takes to deliver a data packet to the destination node, which complies with the metrics stated in the previous section. As an assumption, TS_{data} and TS_{token} are considered to be negligible, thus TW_{token} is only dependent on the number of nodes in a ring.

4.2.1 Fault Free Grid

This section, introduces the operation of the TTG, TTGNM and BTTG networking schemes in a fault free environment. The first case to consider is when a node requires to send a data packet to a destination node located on the same ring; in this case the total cost as explained in the previous sections is split into two main parts, the token waiting time for the sending node and the cost in number of hops for the data packet to reach the destination. Considering the values in Table 4.1, TW_{token} = N/2. On the other hand Table 4.3 depicts the value of the cost in number of hops as; #Hops = (N+2)/4 when N is even, and #Hops = (N+1)/4 when N is odd. Thus, the total cost of sending a data packet on the same ring denoted as SR_{BTTG} is the following:

$SR_{BTTG} = TW_{token} + \#Hops$

If N is even:

$SR_{BTTG\ (even\ N)} = (N/2) + (N+2)/4$

$SR_{BTTG\ (even\ N)} = (3N+2)/4$

If N is odd:

$SR_{BTTG\ (odd\ N)} = (N/2) + (N+1)/4$

$SR_{BTTG\ (odd\ N)} = (3N+1)/4$

The second case to consider is when a node requires sending a data packet to a node on a different row and column rings. In this case, the total cost of sending the data packet denoted as DR_{BTTG} is the following:

$DR_{BTTG} = 2TW_{token} + \#Hops$

If N is even:

$DR_{BTTG\ (even\ N)} = 2*(N/2) + (N+2)/2$

$DR_{BTTG\ (even\ N)} = (3N+2)/2$

If N is odd:

$DR_{BTTG\ (odd\ N)} = 2*(N/2) + (N+1)/2$

$DR_{BTTG\ (odd\ N)} = (3N+1)/2$

Finally the total cost, denoted TC_{BTTG}, of a N^2 nodes BTTG grid with no faults should consider the fact that Node (0, 0) has to send data packets for (2N-2) nodes on the same ring; (N-1) nodes on the same row ring and other (N-1) nodes on the same column ring, the other data packets should be sent to the remaining [N^2-(2N-2)-1] = (N^2-2N+1) nodes. Thus the total cost is:

$TC_{BTTG} = 2*[(2N-2)* SR_{BTTG} + (N^2-2N+1)*DR_{BTTG}]$

If N is even:

$TC_{BTTG\ (even\ N)} = 2*[(2N-2)* (3N+2)/4 + (N^2-2N+1)* (3N+2)/2]$

$TC_{BTTG\ (even\ N)} = 3N^3-N^2-2N$ **(A)**

If N is odd:

$TC_{BTTG\ (odd\ N)} = 2*[(2N-2)* (3N+1)/4 + (N^2-2N+1)* (3N+1)/2]$

$TC_{BTTG\ (odd\ N)} = 3N^3-2N^2-N$ **(B)**

The total cost of communication of the TTG (check details in [Nakad 2003]) and the TTGNM in case of no faults, conducting the same calculation steps as above, are as follows:

$SR_{TTG} = TW_{token(TTG)} + \#Hops_{(TTG)}$

($TW_{token(TTG)}$ is the average token waiting time in the TTG grid being equal to N/2 and $\#Hops_{(TTG)}$ is the average number of hops in the TTG grid equal to N/2.)

$SR_{TTG} = N/2 + N/2$

$SR_{TTG} = N$

$DR_{TTG} = TW_{token(TTG)} + TW_{merge\text{-}token(TTG)} + \#Hops_{(TTG)}$

($TW_{merge\text{-}token(TTG)}$ is the average merge token waiting time in the TTG grid being equal to 2.5TWtoken = 2.5N/2)

$DR_{TTG} = N/2 + 2.5TW_{token(TTG)} + N$

$DR_{TTG} = 3.5 * N/2 + N$

$DR_{TTG} = 3.5\ N/2 + N$

$TC_{TTG} = 2*[(2N-2)* SR_{TTG} + (N^2-2N+1)*DR_{TTG}]$

$TC_{TTG} = 2*[(2N-2)* N + (N^2-2N+1)*(3.5N/2 + N)]$

$TC_{TTG} = 5.5N^3-7N^2+1.5N$ **(C)**

And

$SR_{TTGNM} = TW_{token(TTGNM)} + \#Hops_{(TTGNM)}$

($TW_{token(TTGNM)}$ is the average token waiting time in the TTGNM grid being equal to N/2 and $\#Hops_{(TTGNM)}$ is the average number of hops in the TTGNM grid equal to N/2.)

$SR_{TTGNM} = N/2 + N/2$

$SR_{TTGNM} = N$

$DR_{TTGNM} = 2TW_{token(TTGNM)} + \#Hops_{(TTGNM)}$

$DR_{TTGNM} = 2N/2 + N$

$DR_{TTGNM} = 2N$

$TC_{TTGNM} = 2*[(2N-2)* SR_{TTG} + (N^2-2N+1)*DR_{TTG}]$

$TC_{TTGNM} = 2*[(2N-2)* N + (N^2-2N+1)*2N]$

$TC_{TTGNM} = 4N^3-4N^2$ **(D)**

To analyze the efficiency of BTTG vs TTG and TTGNM network schemas, the percentage decrease in total cost between the three schemas is calculated.

The first case to consider is for N being even thus (A) should be compared to both (C) and (D):

Efficiency % BTTG vs TTG:

$\{[(C)-(A)]/(C)\}*100 =$

$\{[(5.5N^3-7N^2+1.5N)-(3N^3-N^2-2N)]/(5.5N^3-7N^2+1.5N)\}*100 =$

$\{(2.5N^3-6N^2+3.5N)/ (5.5N^3-7N^2+1.5N)\}*100 =$

$\{(2.5N^2-6N+3.5)/ (5.5N^2-7N+1.5)\}*100$ **(E)**

Efficiency % BTTG vs TTGNM:

$\{[(D)-(A)]/(D)\}*100 =$

$\{[(4N^3-4N^2)-(3N^3-N^2-2N)]/(4N^3-4N^2)\}*100 =$

$\{(N^3-3N^2+2N)/ (4N^3-4N^2)\}*100 =$

$\{(N^2-3N+2)/ (4N^2-4N)\}*100$ **(F)**

The next case to consider is for N being odd thus (B) should be compared to both (C) and (D):

Efficiency % BTTG vs TTG:

$\{[(C)-(B)]/(C)\}*100 =$

$\{[(5.5N^3-7N^2+1.5N)-(3N^3-2N^2-N)]/(5.5N^3-7N^2+1.5N)\}*100 =$

$\{(2.5N^3-5N^2+2.5N)/ (5.5N^3-7N^2+1.5N)\}*100 =$

$\{(2.5N^2-5N+2.5)/ (5.5N^2-7N+1.5)\}*100$ **(G)**

Efficiency % BTTG vs TTGNM:

$\{[(D)-(B)]/(D)\}*100 =$

$\{[(4N^3-4N^2)-(3N^3-2N^2-N)]/(4N^3-4N^2)\}*100 =$

$\{(N^3-2N^2+N)/ (4N^3-4N^2)\}*100 =$

$\{(N^2-2N+1)/ (4N^2-4N)\}*100$ **(H)**

Table 4.4 depicts the cost efficiency percentage for grids with N=4, 5, 6, and 7. The values are calculated using the equations previously presumed. An average improvement of 37% in communication cost is detected on BTTG over TTG and an average improvement of 18% is detected on BTTG over TTGNM.

Table 4.5 presents the simulated results for the analysis done in Table 4.4. In simulation, BTTG is showing 24% improvement over TTG and 12% over TTGNM.

Table 4.4: Analytical results of BTTG, TTG, and TTGNM networking schemes

N	**TTG Analytical Cost**[4]	**TTGNM Analytical Cost**[5]	**BTTG Analytical Cost**[6]	**BTTG vs TTG Cost efficiency %**[7]	**BTTG vs TTGNM Cost efficiency %**[8]
4	246	192	168	31.707%	12.5%
5	520	400	320	38.462%	20%
6	945	720	600	36.508%	16.667%
7	1554	1176	924	40.541%	21.429%

[4] Using equation (C)
[5] Using equation (D)
[6] Using equation (A) for even N and equation (B) for N odd
[7] Using equation (E) for even N and equation (G) for N odd
[8] Using equation (F) for even N and equation (H) for N odd

Table 4.5: Simulation results of BTTG, TTG, and TTGNM networking schemes

N	TTG Simulation Cost	TTGNM Simulation Cost	BTTG Simulation Cost	BTTG vs TTG Cost efficiency %	BTTG vs TTGNM Cost efficiency %
4	198	183	159	19.697%	13.115%
5	408	353	336	17.647%	4.816%
6	730	616	520	28.767%	15.584%
7	1188	985	839	29.377%	14.822%

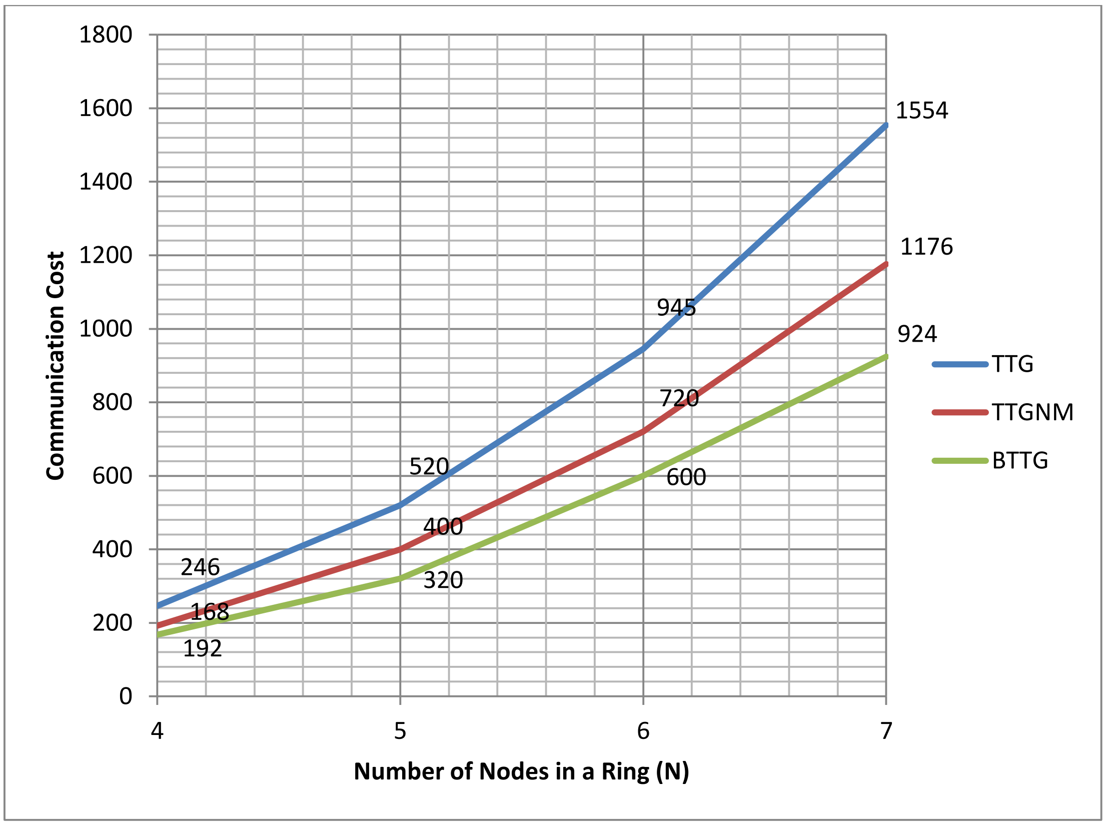


Figure 4.5: Theoretical Results with increasing grid size

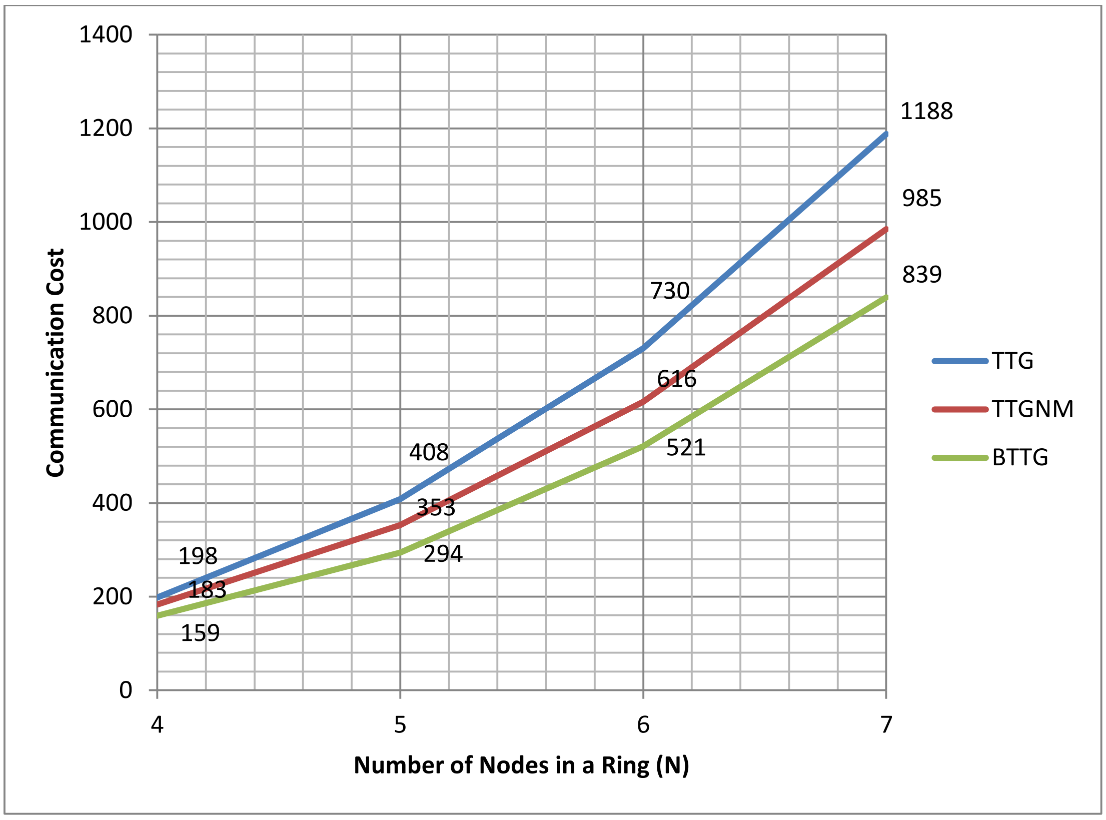


Figure 4.6: Simulated Results with increasing grid size

4.2.2 Fault Introduction

This section introduces links failure and reports the respective communication costs for both the TTG and BTTG networking schemes, the TTGNM scheme is not examined, due to the fact that fault tolerance was not handled in this scheme.

In order to show the efficiency of the BTTG over the TTG, a 4x4 grid with link faults at one link, two link and three link failures are analyzed. The fault analysis study introduced in [Nakad 2003] is considered.

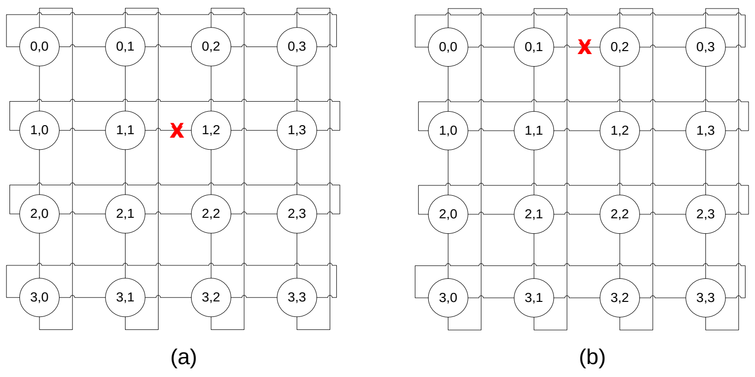


Figure 4.7: (a) Grid with one link on diffrent ring than Node (0, 0), (b) Grid with link error on same ring as Node (0, 0)

Figure 4.7(a) represents the case of a 4x4 TTG grid with one link failure, in this case the link failure affects the ring of the sending Node (0, 0), and the analytical total cost of communication is equal to 275. On the other hand, if the link failure, as show in Figure 4.7 (b), affects a ring not directly connected to the sending node, the analytical total cost of communication is equal to 350.

The same scenario is analyzed for a 4x4 BTTG gird. In the BTTG case, as previously explained in chapter 3, a link failure will not disable the ring, it will affect the communication path direction. Using the results of Tables 4.1 and 4.3 and as previously discussed in subsection 4.1.2 the total cost of communication as presented in Figure 4.7 (a) is affected by the fact that the grid has a link failure on a ring directly connected to the sending node in this case the total cost to a data packet on this ring is as follows:

$SRE_{BTTG} = TW_{token}$ (with fault) + #Hops (with fault)

$SRE_{BTTG} = (N-1) + N/2$

$SRE_{BTTG} = (3N-2)/2$

The Total cost of communication in this case is a below:

$TC_{BTTG} = 2*(3* SRE_{BTTG} + 3* SR_{BTTG\ (even\ N)} + 9* DR_{BTTG\ (even\ N)})$

$TC_{BTTG} = 2*(3* [(3N-2)/2] + 3* [(3N+2)/4] + 9* [(3N+2)/2])$

$TC_{BTTG} = 2*(3*5 + 3*3.5 + 9*7)$ with $N=4$

$TC_{BTTG} = 177$

In case the link failure is not directly affecting the sending node ring, case of Figure 4.7(b), the total cost of sending a data packet to a node with an affected ring used in communication is:

$DRE_{BTTG} = TW_{token} + \#Hops + TW_{token}$ (with fault) + #Hops (with fault)

$DRE_{BTTG} = SR_{BTTG} + SRE_{BTTG}$

$DRE_{BTTG} = (3N+2)/4 + (3N-2)/2$

$DRE_{BTTG} = (9N-2)/4$

The Total cost of communication in this case is a below:

$TC_{BTTG} = 2*(6* SR_{BTTG} + 3* DRE_{BTTG} + 6* DR_{BTTG\ (even\ N)})$

$TC_{BTTG} = 2*(6* [(3N-2)/2] + 3* [(9N-2)/4] + 6* [(3N+2)/2])$

$TC_{BTTG} = 2*(6*5 + 3*8.5 + 6*7)$ with $N=4$

$TC_{BTTG} = 195$

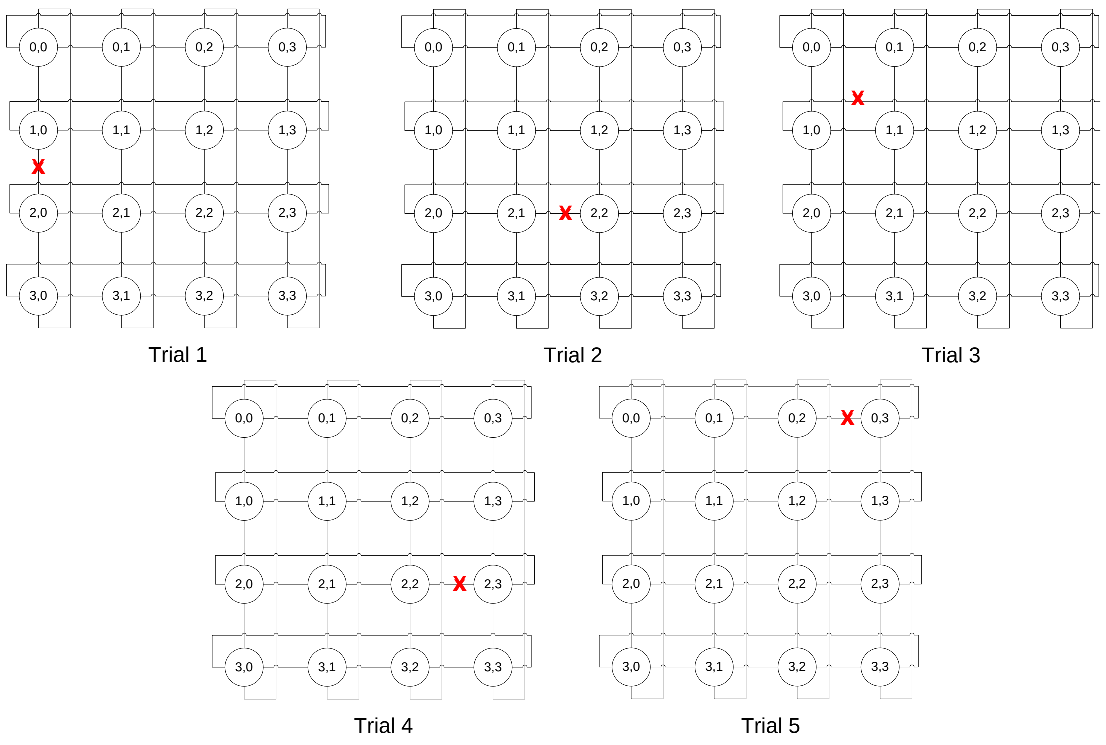


Figure 4.8: Five Trials with one link failure

A simulation was conducted on TTG and BTTG 4x4 grids with one link failure at different locations. Figure 4.8 shows five trials and the simulated counter value for both the TTG and the BTTG networking schemes. Table 4.6 depicts the simulated results for the trial cases presented in figure 4.8

Table 4.6: Five Trials with one link fault

Trial	TTG Simulation Cost	BTTG Simulation Cost	BTTG vs TTG Cost efficiency %
1	293	161	45
2	219	155	29
3	208	167	20
4	220	147	33
5	315	198	37

Table 4.6 shows that the BTTG networking scheme performed better that the TTG networking scheme in case of one link failure. This better performance is due to the fact that a link failure does not disable the whole ring however the minimal path might be affected, preserving the connection over the resulting virtual ring.

Comparing the values of the BTTG in case of one link failure to the BTTG with no link failure, it can be shown that the cost in communication is hardly affected.

The same general communication scheme was tested with two link errors in the network. Figure 4.9 presents 5 trial cases tested for both the TTG and the BTTG. The recorded communication cost values for the two link errors trials are presented in Table 4.7.

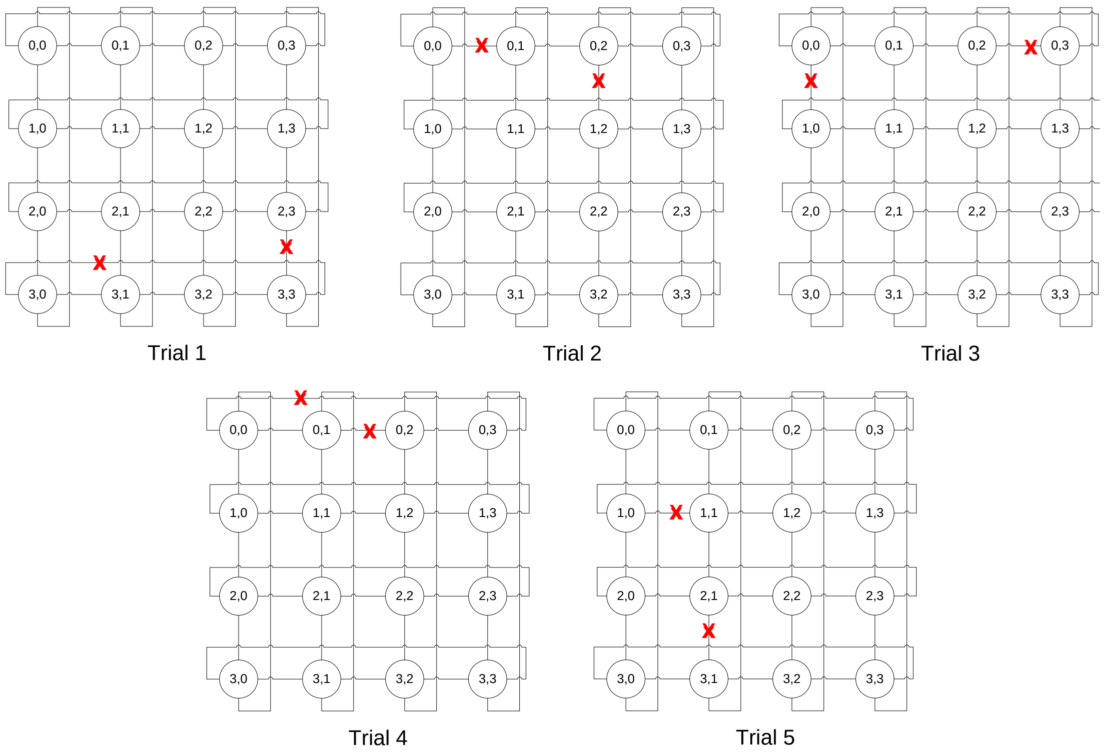


Figure 4.9: Five Trials with two links failure

Table 4.7: Five Trials with two links fault

Trial	**TTG Simulation Cost**	**TTG Nodes not Reached**	**BTTG Simulation Cost**	**BTTG vs TTG Cost efficiency %**
1	201	Node (3, 3)	161	20
2	349	Node (0, 2)	218	38
3	-		192	-
4	315		217	31
5	209	Node (1, 1)	177	15

The values presented in Table 4.7 shows that the BTTG networking scheme is more efficient than the original TTG networking scheme. In the presence of two links errors, the BTTG induces an average of 26% decrease in communication cost vis-à-vis the TTG. Although a closer look to the simulation, shows that in Trials 1, 2, and 5 the TTG networking scheme fails to deliver the data packet for one node in each trial, this is due to the fact that the row and column rings of the corresponding nodes are both with link errors, thus the node is not reachable anymore. Also Trial 3 shows that for TTG case, Node (0, 0) was unable to initiate the communication as it is isolated from the network. The above cases are not encountered in the BTTG as the faced link failures will not disable the ring and connection between nodes is persevered.

The same general communication scheme was tested with three link errors in the network. Figure 4.10 presents 5 trial cases tested for both the TTG and the BTTG. The recorded communication cost values for the above trials are presented in Table 4.8.

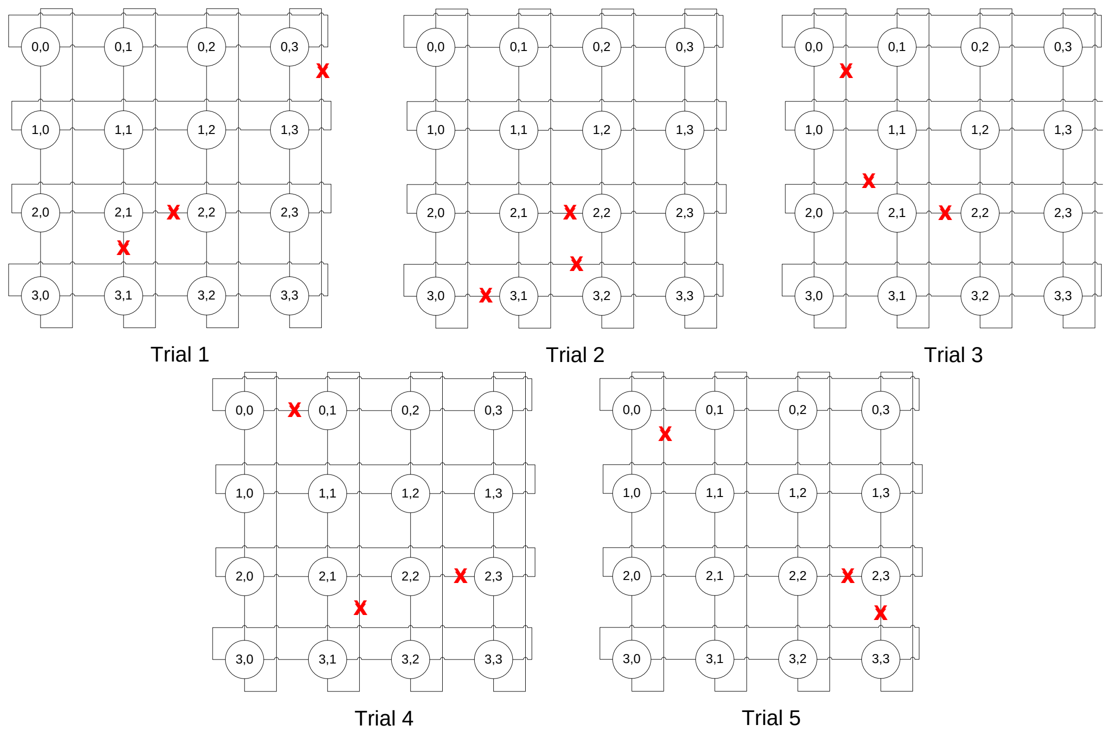


Figure 4.10: Five Trials with three links failure

Table 4.8: Five Trials with three links fault

Trial	TTG Simulation Cost	TTG Nodes not Reached	BTTG Simulation Cost	BTTG vs TTG Cost efficiency %
1	190	Node (2, 1) – Node (2, 3)	177	7
2	219		174	21
3	318	Node (2, 0)	201	37
4	383	Node (0, 1) – Node (2, 1)	159	58
5	377	Node (2, 3)	173	54

The simulation results presented in Table 4.8 shows that with three link failures, the BTTG behaves better than the TTG in two different aspect. As a first point, the BTTG decreases the cost of communication by an average of 35%, although the BTTG is able to deliver the packets to all the nodes in the grid, the case that is not always true for the TTG (Trials 1,3,4,5 all fails to deliver to all nodes).

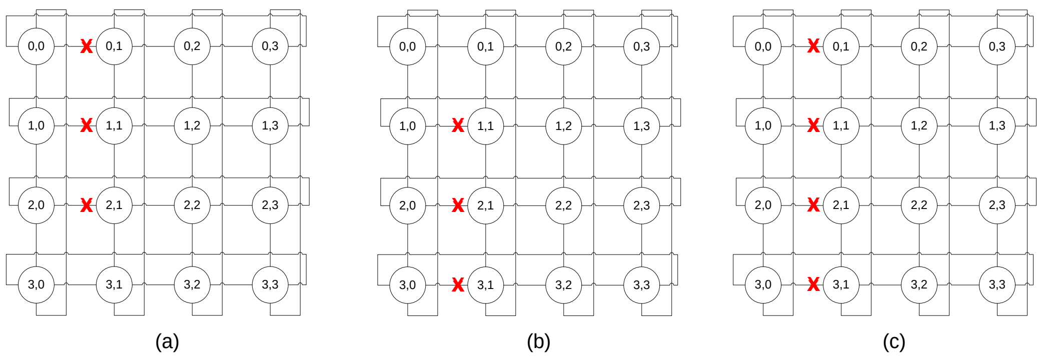


Figure 4.11: (a) and (b) are grids with 3 links failures each on a row, (c) grid with 4 links failures on row rings

Figure 4.11 shows three additional cases of faults, these faults are resulted from tears through the textile that are expected during the operation of e-textiles. Cases in Figures 4.11 (a) and (b) represents three faults affecting 3 of the four rows and Figures 4.11 (c) represents a tear affecting the four rows of the grid. For the TTG the communication in case (a) and (b) is still possible as Node (0, 0) retain one ring connected to the network, although the BTTG communication is tolerated and connection can be operated over the different rings 1 normally and three with fault tolerance in opposite direction. Table 4.9 presents the collected simulation results.

Table 4.9: Simulation Results for cases in Figure 4.11

Trial	**TTG Simulation Cost**	**BTTG Simulation Cost**	**BTTG vs TTG Cost efficiency %**
4.11(a)	459	175	62
4.11(b)	211	175	17
4.11(c)	-	187	100

As results shows in Table 4.9, the BTTG performs better than the TTG with an average of 40% decrease in communication cost (4.11(a) and 4.11(b)). Also when TTG fails the BTTG is able to deliver data to all nodes case of Figure 4.11 (c).

4.3 Increasing the load

This section simulates the performance of the BTTG in case of load increase, where the sample introduced in the previous sections is adopted with a gradual increase in the number of sending nodes from one node to two nodes, three nodes and finally four nodes. This test is adopted on grids with N=4, 5, 6, and 7 nodes in a ring. Table 4.10, 4.11, 4.12, and 4.13 represents the simulated results of BTTG, TTG, and TTGNM grids of different sizes and loads. This particular technique of increasing the sending nodes and using these nodes to communicate with the matching destinations was chosen to elevate conflict and show the decreasing response of the networking schemes.

The simulation results show that when the number of sending nodes increases, the cost increases. This increase is highly noticeable in the TTG networking scheme. According to [Nakad 2003] this increase is due to three major factors:

1- The waiting period increase for data packets waiting in the data queues.
2- Nodes need to send on a same ring but merge token is in action, thus token will be missed and wait time will increase.
3- Need to wait for a merge state to end until another one can occur on the same ring.

The simulation on grids with different sizes shows that the occurrence of these factors increases with increasing the load. The increase in the TTGNM and the BTTG is much lower than the TTG case. This increase in performance is related to the fact that the only increase in waiting time is related to the waiting period increase for data packets in the data queues compared to the TTG where time is consumed in merge operations. Also the BTTG shows the best performance between the three networking schemes due to communication over the minimal path. This makes the BTTG better by an average of 35% from the TTG networking scheme and 11% from the TTGNM networking scheme.

Table 4.10: Simulation results of a 4x4 grid with load increase

Sending Nodes	TTG	TTGNM	BTTG	BTTG vs TTG Cost efficiency %	BTTG vs TTGNM Cost efficiency %
(0, 0)	198	183	159	19.7	13.1
(0, 0), (3, 3)	221	192	173	21.7	9.9
(0, 0), (3, 3), (0, 3)	339	212	204	39.8	3.8
(0, 0), (3, 3), (0, 3), (3, 0)	368	239	218	40.8	8.8

Table 4.11: Simulation results of a 5x5 grid with load increase

Sending Nodes	TTG	TTGNM	BTTG	BTTG vs TTG Cost efficiency %	BTTG vs TTGNM Cost efficiency %
(0, 0)	408	353	294	27.9	20.1
(0, 0), (4, 4)	452	363	336	25.7	8
(0, 0), (4, 4), (0, 4)	640	412	377	41.1	9.3
(0, 0), (4, 4), (0, 4), (4, 0)	751	445	400	46.7	11.2

Table 4.12: Simulation results of a 6x6 grid with load increase

Sending Nodes	TTG	TTGNM	BTTG	BTTG vs TTG Cost efficiency %	BTTG vs TTGNM Cost efficiency %
(0, 0)	730	616	521	28.6	18.2
(0, 0), (5, 5)	801	648	583	27.2	11.1
(0, 0), (5, 5), (0, 5)	1083	700	658	39.2	6.4

(0, 0), (5, 5), (0, 5), (5, 0)	1328	741	676	49.1	9.6

Table 4.13: Simulation results of a 7x7 grid with load increase

Sending Nodes	TTG	TTGNM	BTTG	BTTG vs TTG Cost efficiency %	BTTG vs TTGNM Cost efficiency %
(0, 0)	1188	985	839	29.4	17.4
(0, 0), (6, 6)	1285	1036	911	29.1	13.7
(0, 0), (6, 6), (0, 6)	1920	1098	1035	46.1	6.1
(0, 0), (6, 6), (0, 6), (6, 0)	2117	1172	1081	48.9	8.4

Plotted results of the simulations of the BTTG, TTGNM and the TTG can be visualized in figures 4.12 till 4.15. The figures show clearly that the BTTG is always performing better than the other schemes, followed by the TTGNM. The TTG is with the lowest performance.

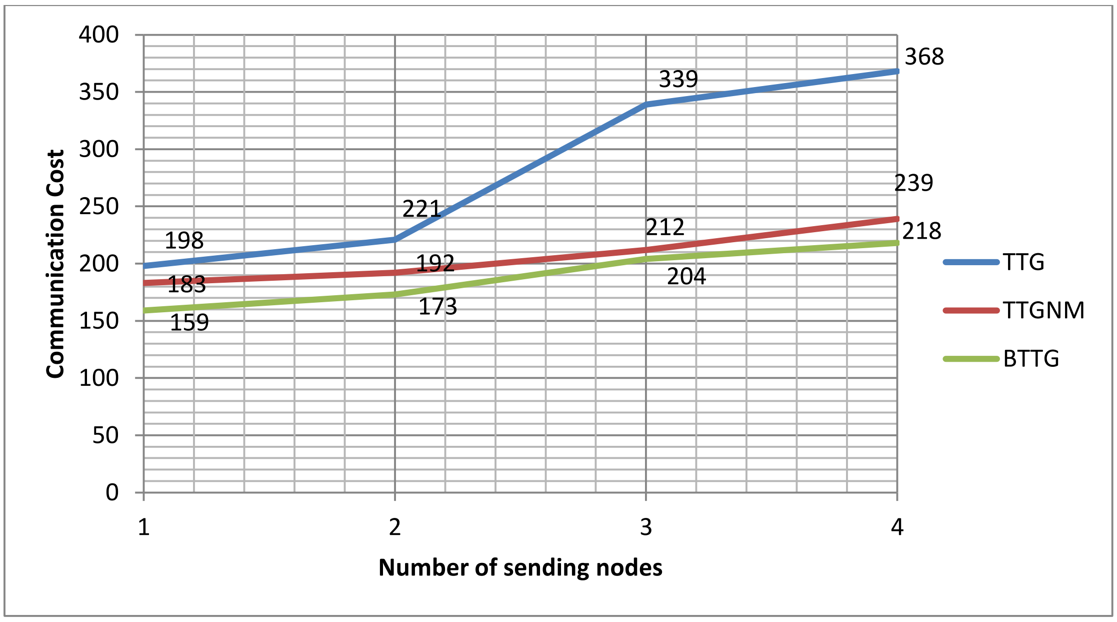


Figure 4.12: Pattern of communication cost increase of a 4x4 grids with load increase

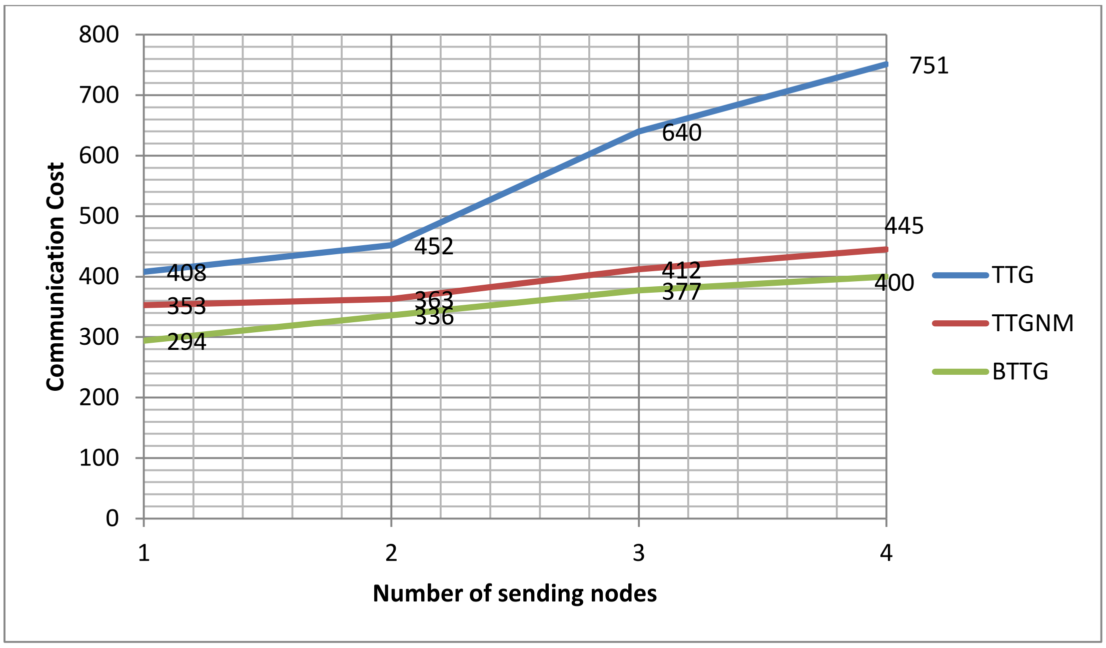


Figure 4.13: Pattern of communication cost increase of a 5x5 grids with load increase

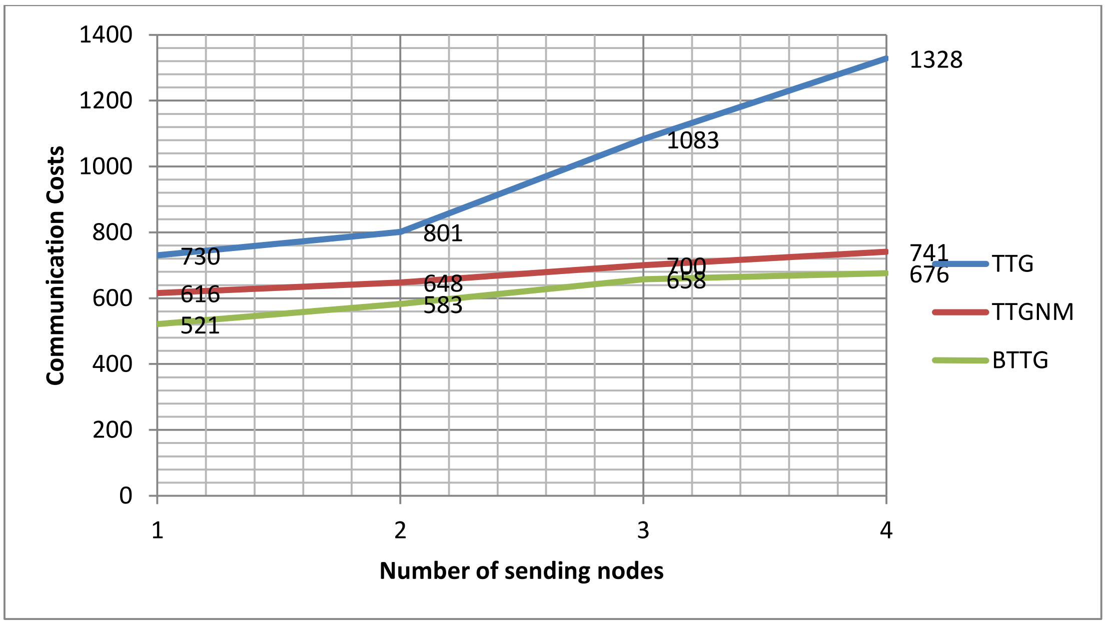


Figure 4.14: Pattern of communication cost increase of a 6x6 grids with load increase

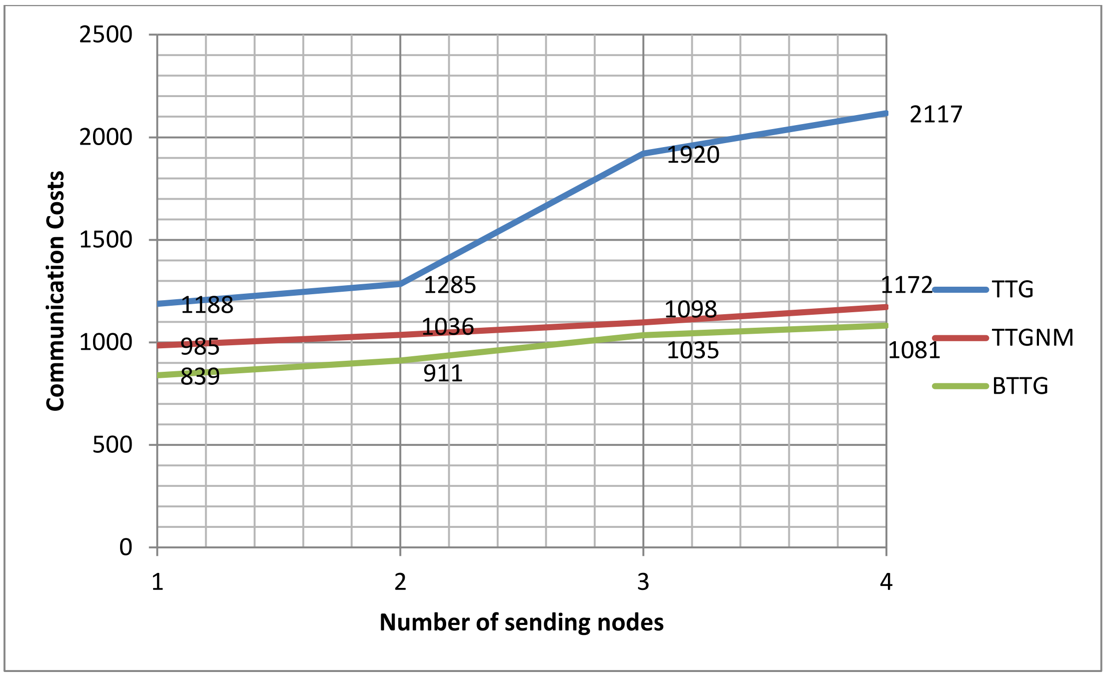


Figure 4.15: Pattern of communication cost increase of a 7x7 grids with load increase

This chapter confirmed that the BTTG networking topology enhances the communication speed between token grid communicating nodes, compared to the TTG and TTGNM. It also increases the fault tolerance capability of the e-textile token grid in case of links and/or nodes failures.

CHAPTER FIVE
CONCLUSION

This research introduced the Bidirectional Electronic Textiles Token Grid networking topology. It aims at improving the cost of communication between the token grid's nodes along with improving the fault tolerance capability of the token grid.

This goal was reached by developing a new communication protocol for the token grid architecture that allows the nodes of the grid to send and receive information in both directions over a link. This capability allowed the communication over a minimal path between the source and destination nodes. The bidirectional feature of the grid links, along with the minimal path routing algorithm that was applied, enhanced the cost of communication between the nodes of the token grid. On the other hand, fault tolerance was also enhanced. The bidirectional communication capability, allowed for partial rings formation in cases of link and/or node errors. A ring that has link errors is transformed into one or multiple partial rings. The communication over partial rings was governed by U-Shaped tokens, minimal path routing was applied for communication over rings as the decision of path routing was governed locally in the sending node (source and intersecting nodes).

A simulation was developed using the OMNet++ simulation environment. This simulation took into consideration the communication in a fault free token grid environment for three different schemes; the TTG, TTGNM, and the BTTG. The BTTG showed the best performance in the cost of communication. This enhancement resulted from two main parameters. The first parameter was the absence of merge operations while communicating between nodes laying on different row and column rings which eliminated the merge waiting time in intersecting nodes. The second parameter was the minimal path routing capability of the BTTG grid which allowed the minimization in number of hops between the source and destination nodes. Another simulation was held in a faulty environment for the TTG and the BTTG networking schemes. The results showed that the BTTG fault tolerance performance is better than the performance of the TTG. This capability was revealed by the fact that the BTTG showed better communication cost in cases where both TTG and BTTG were able to deliver successfully the data packets to all intended nodes in the grid. Moreover, the BTTG was able to deliver packets to all nodes in cases where the TTG failed to communicate with one or more destination nodes due to link failures on the source or destination nodes level, which isolated the node from the token grid in the TTG topology, however, the node was reachable in the BTTG topology.

Finally, the BTTG networking scheme can be used for future applications development and implementation. One of the best applications is the carpet based burglar alarm system that consist of theft detection electronic devices embedded in a carpet textile and communicating by means of the BTTG networking topology. This system will be trustworthy in intrusion detection due to three basic parameters: reliable fault tolerance, speed in communication, and inevitable detection system as the carpet is a textile that can cover a very large area in the premises.

REFERENCES

Hwang, K. (1993). *Advanced computer architecture: Parallelism, scalability, programmability.* West Patel Nagar, New Delhi: Tata McGraw-Hill Higher Education.

Martin, T. (2012). Creating the future of textiles: E-textile technology. *Specialty Fabrics Review*. Retrieved from http://specialtyfabricsreview.com/articles/0512_fot_etextile_technology.html.

National Park Service. (2002). Museum collections security and fire protection. In J.W. Roberts, J. Pasiuk, C. Barton & J. Bacharach (Eds), *NPS Museum Handbook: Museum collections* (pp. 9:1-9:51). Washington, USA: National Park Service

Nakad, Z. (2003). *Architectures for e-textiles* (Ph.D. Dissertation, Virginia Polytechnic Institute and State University, Virginia). Retrieved from http://www.ccm.ece.vt.edu/etextiles/publications/Dissertation_Zahi_Nakad.p df

Nakad, Z., Jones M., Martin, T., & Fawaz, W. (2009, December 10). Networking in e-textiles. *Elsevier Computer Communications*, *33*, 665-666.

Zheng, N., Wu, Z., Chen, L., & Zhou, Y. (2006). The e-textile token grid network with dual rings. In M. L. Gavrilova & V. Kumar (Eds), *Computational science and its applications – ICCSA 2006* (pp. 1149-1158). Glasgow, UK: Springer Berlin Heidelberg

Todd, T. D. (1992). The token grid: Multidimensional media access for local and metropolitan networks. *IEEE Computer and Communications Societies*, *3*, 2415-2424.

Ross, F. E. (1989, September). An overview of FDDI: The fiber distributed data interface. *IEEE J. Select Areas*, *7*(7), 1043-1051.

OMNet++ Community. (2009). What is OMNet++. Retrieved from http://www.omnetpp.org/home/what-is-omnet.

I2C Bus. (2012). I2C bus specification and user manual. Retrieved from http://www.nxp.com/documents/user_manual/UM10204.pdf

Todd, T.D. (1994, June). The token grid network. *IEEE/ACM Transactions on Networking*, *2*(3), 279-287.