Spring 2022

A variable node optimization model for byzantine fault tolerant systems

Ian Robert Fulton

Department:

A VARIABLE NODE OPTIMIZATION MODEL FOR BYZANTINE FAULT TOLERANT SYSTEMS

by

IAN ROBERT FULTON

A THESIS

Presented to the Graduate Faculty of the

MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY

In Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE IN ENGINEERING MANAGEMENT

2022

Approved by:

Dr. Steven Corns, Advisor
Dr. Benjamin Kwasa
Dr. Suzanna Long

ABSTRACT

Byzantine Fault Tolerance (BFT) has been a major subject of study over the last two decades with increasing societal dependance on secure, correct, and reliable computer systems and online services. This research presents a model for high-level optimization of emerging systems that rely on these BFT algorithms and use a variable numbers of decision nodes. The model highlights the relationship between the security of a system and its efficiency. Two experiments were performed to determine system performance by varying the number of compromised nodes, decision nodes, and total nodes. They examine the probability that a transaction will be compromised based on these variables using hypergeometric distribution, a subset of combinatorics. It was found that the compromise probability follows predictable patterns, with certain combinations of decision nodes performing better than others. The results show a trichotomous relationship where one in every three decision nodes results in lower security risk than its neighbors. The purpose of this model is to assist system developers in deciding how to best construct their systems to improve security while minimizing resource usage.

ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. Corns, who was always willing to lend a hand to guide and support my academic journey and provide direction whenever I felt lost. His aid and affirmation throughout this process has been an invaluable asset, without which I would not have achieved the full understanding and appreciation for the field of systems engineering. He has provided every opportunity for my success, for which he has my sincerest gratitude.

I would like to thank the department of Engineering Management for all of their help with funding, scheduling, conferences, and a multitude of other assistances I could not have done without.

I would also like to thank the Boeing company for funding my research project which led me to this study, and for providing the opportunity to pursue it.

TABLE OF CONTENTS

Page

ABSTRACT iii

ACKNOWLEDGMENTS iv

LIST OF ILLUSTRATIONS vii

LIST OF TABLES ix

NOMENCLATURE x

SECTION

1. INTRODUCTION 1

2. METHODS 9

2.1. EXPERIMENT 1 10

2.2. EXPERIMENT 2 12

2.3. ANALYSIS 13

3. RESULTS 14

3.1. EXPERIMENT 1 RESULTS 14

3.2. EXPERIMENT 2 RESULTS 15

4. DISCUSSION 25

4.1. TRENDS IN EXPERIMENT 1 RESULTS 25

4.2. TRENDS IN EXPERIMENT 2 RESULTS 26

4.3. PROPORTIONAL TRENDS 26

4.4. CASE EXAMPLE 27

5. CONCLUSIONS 29

APPENDIX.....31

BIBLIOGRAPHY.....34

VITA.....36

LIST OF ILLUSTRATIONS

Page

Figure 1.1 Byzantine General's Problem 2

Figure 1.2 Example Node Pool with Sample Decision Node Groups 6

Figure 3.1 Comparison of Disruption Chance to Number of Compromised Nodes using 38 Decision Nodes out of 100 Total Nodes 14

Figure 3.2 Comparison of Disruption Chance to Number of Compromised Nodes using 10 Decision Nodes out of 100 Total Nodes 16

Figure 3.3 Comparison of Disruption Chance to Number of Compromised Nodes using 10 Decision Nodes out of 50 Total Nodes 16

Figure 3.4 Comparison of Disruption Chance to Number of Decision Nodes using 10 Compromised Nodes out of 100 Total Nodes 17

Figure 3.5 Comparison of Disruption Chance to Number of Decision Nodes using 25 Compromised Nodes out of 100 Total Nodes 18

Figure 3.6 Comparison of Disruption Chance to Number of Decision Nodes using 33 Compromised Nodes out of 100 Total Nodes 18

Figure 3.7 Comparison of Disruption Chance to Number of Decision Nodes using 32 Compromised Nodes out of 100 Total Nodes 20

Figure 3.8 Comparison of Disruption Chance to Number of Decision Nodes using 34 Compromised Nodes out of 100 Total Nodes 20

Figure 3.9 Comparison of Disruption Chance to Number of Decision Nodes using 50 Compromised Nodes out of 100 Total Nodes 21

Figure 3.10 Comparison of Disruption Chance to Number of Decision Nodes of the Mod3=0 group using 33 Compromised Nodes out of 100 Total Nodes.. 21

Figure 3.11 Comparison of Disruption Chance to Number of Decision Nodes of the Mod3=1 group using 33 Compromised Nodes out of 100 Total Nodes.. 22

Figure 3.12 Comparison of Disruption Chance to Number of Decision Nodes of the Mod3=2 group using 33 Compromised Nodes out of 100 Total Nodes.. 22

Figure 3.13 Comparison of Disruption Chance to Number of Decision Nodes using 66 N_c out of 199 Total Nodes 23

LIST OF TABLES

Page

Table 3.1 Average Modulo Group Values of Proportionately Increasing Node Sets..... 24

NOMENCLATURE

Symbol	Description
k	Consensus Disruption Threshold Number
Nt	Total Nodes in the System
Nd	Decision Nodes
Nc	Compromised Nodes

1. INTRODUCTION

This research explores the optimization of Byzantine Fault Tolerant (BFT) systems to reduce resource usage while maintaining a level of security. Byzantine Fault Tolerance was first proposed in 1978 as a method of achieving a consensus in a computer network even if some of the network nodes were faulty (Lamport et al., 1982). A network achieving consensus means that its nodes have come to an agreement with a high degree of certainty that the information being transferred is correct and has not been duplicated or tampered with. This problem of network consensus in the face of faulty system nodes was initially named the interactive consistency problem before being renamed the Byzantine General's problem by the same authors.

The Byzantine General's problem puts this issue into the context of a general from the byzantine era who is attempting to conquer a city. The general commands several companies surrounding the city, each led by a captain. For the city to be taken, all surrounding companies must attack at the same time. The general must send messages to the surrounding forces to ensure that they will attack at the designated time. In this problem, the captains are the system nodes who must reply to the general so that he may determine if they are all in consensus with the success of the attack being synonymous with the success of the system. Figure 1.1 is an example of when four of the six captains (light colored) are confirmed to attack at the designated time and two captains (dark colored) are unconfirmed.

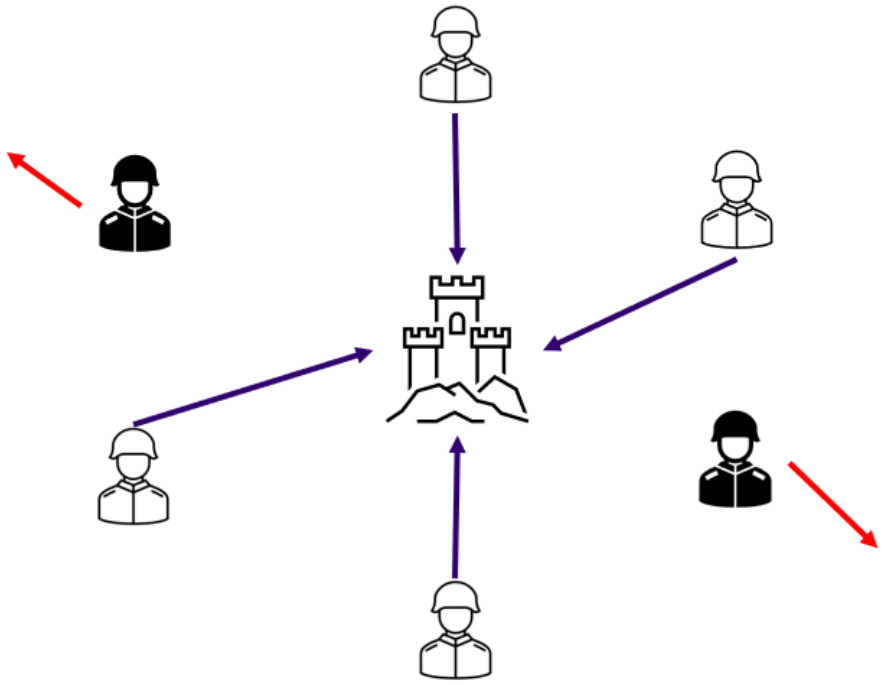

Figure 1.1 Byzantine General's Problem

This problem introduces nuances such as certain captains being traitors and willfully returning false replies or messengers being captured along their routes. In Figure 1.1, these are represented by the dark colored, unconfirmed captains. This translates to network systems as compromised nodes returning faulty decisions or nodes being unable to reply due to issues with maintenance or network communication. Nodes with these problems are said to have "Byzantine Faults". The BFT algorithm was able to overcome these faults if fewer than one third of the nodes were affected.

This method was improved upon by Castro and Lizkov (1999) who introduced Practical Byzantine Fault Tolerance (PBFT) which performed well in asynchronous environments and can be used to build highly available systems (Konnov et al., 1999). This method provides secure consensus and proactive recovery methods to recover faulty

nodes over the system lifetime, reducing the time that any single faulty node can contribute to consensus disruption (Castro & Liskov, 2002). These algorithms were still considered brand-new with little testing to prove that it merited implementation (Nair et al., 2020; Sternberg et al., 2020) in 2008 when Bitcoin and Blockchain technology was introduced. . Bitcoin used BFT to create the first public Blockchain network, reducing energy costs paid by system administrators by allowing anyone to act as a system node. The potential for malicious behavior was mitigated by requiring nodes to provide "Proof of Work", requiring them to compute complex algorithms to even be allowed into a decision-making process with a reward for participating in decisions (Ghosh & Das, 2020).The main draw of these Blockchain systems is their record keeping capabilities. Once a decision has been made to add data into the system, it is encoded as a block in a digital ledger and stored publicly on all nodes so that it can later be analyzed to ensure that no node attempted to change the data in that ledger. Blockchain systems use BFT to obtain a consensus and then write it into an immutable ledger, but not all BFT systems use the same record-keeping methodology (Zhao, 2009). Although Blockchain systems are the most well-known BFT systems, they are not the only. BFT algorithms can be used for any system that requires high levels of security in its decision-making processes. The algorithms are useful for many Web-based systems that require higher levels of security in their messaging processes, and can be implemented to help these systems meet reliable messaging standards (Zhao, 2009).

Some common problems BFT systems encounter are those of excessive energy consumption and high latency times (Nair et al., 2020; Sedlmeir et al., 2020). To achieve the level of security provided by BFT algorithms, a system typically relies on many

nodes running in parallel to achieve a single consensus. Reducing the number of nodes performing calculations creates a tradeoff between security and energy consumption. The second problem is long latency times, caused by the algorithms being run on each node to ensure the security of the messages transmitted in the system. As the number of nodes performing calculations increases, the traffic on the network increases causing congestion and increased latency times to send messages. This increases the amount of time nodes wait for messages to be delivered, contributing to the energy consumption of the system and increasing the time it takes to achieve a consensus.

Power consumption and latency issues have been a focus of research for BFT systems. Different Blockchain types have been developed to optimize the systems for these two issues based on their respective needs (Da Silva et al., 2019; Franke et al., 2020; Vizier & Gramoli, 2020; Y. Wang, 2019). Various internet systems utilizing BFT algorithms have attempted to improve these issues for their purposes, but Blockchains has received a majority of the research focus since the rise of Bitcoin.

The two most prominent types of Blockchain systems are Public and Private. Public Blockchains, like Bitcoin, allow any person to participate in consensus. The added security risk of allowing anyone to participate is mitigated through requiring that person to provide some proof that they will not tamper with the consensus. This is achieved by requiring a person's node/computer to run complex algorithms to achieve a Proof of Work. This is based on the idea that the person now has an investment of energy and time contributed to the consensus and will therefore be motivated to provide a good decision for consensus (Ghosh & Das, 2020). Another motivator in this process is a small reward for their service, in the case of Bitcoin this comes in the form of a small monetary reward

in Bitcoins. This method places the burden of energy consumption on the unaffiliated parties but still pulls a large amount of power from the national grid. One study estimated that, over one year, power consumption from Bitcoin processes in the U.S. alone were equivalent to a small country (Nair et al., 2020). The energy consumption of Proof of Work methods has led others to attempt other methods, such as Proof of X, a template term for any other type of proof which attempts to attain the same results but with the idea of improving energy usage and scalability (Franke et al., 2020). The currently accepted best alternative to Proof of Work is Proof of Stake, which selects nodes from users with stake in the Blockchain and its integrity. This method is considered to be an improvement over Proof of Work, although it has not been thoroughly tested(Franke et al., 2020).

Private Blockchains skirt the energy requirements of Proof of Work by way of only allowing trusted, pre-approved nodes on their networks. This method requires far less energy as the complex algorithms no longer need to be run (F. Wang et al., 2021). Private Blockchains are also used for private companies to secure their transaction records without the public ledger provided by Public Blockchains. This research focuses on Private Decentralized Blockchains as they consist of a set of controlled nodes, each capable of returning a decision.

When first introduced BFT systems have consisted of a set number of nodes capable of returning decisions to reach a secure consensus. These nodes are consistent, and the same nodes are used in each consensus. Recently, systems such as the Redbelly Blockchain have been introduced to utilize a far larger pool of nodes (Concurrent Systems Research Group, University of Sydney, n.d.). In these systems, not all nodes are

used in each consensus. This allows several decision-making processes to be run concurrently, reducing latency times as more decisions can be made at once. This development is only useful if different nodes are used to create pools of nodes each time. These systems constantly switch between which nodes in the pool are selected for each group every time a consensus is reached. Figure 1.2 displays a sample distributed network setup with twelve total nodes and two Byzantine Fault nodes.

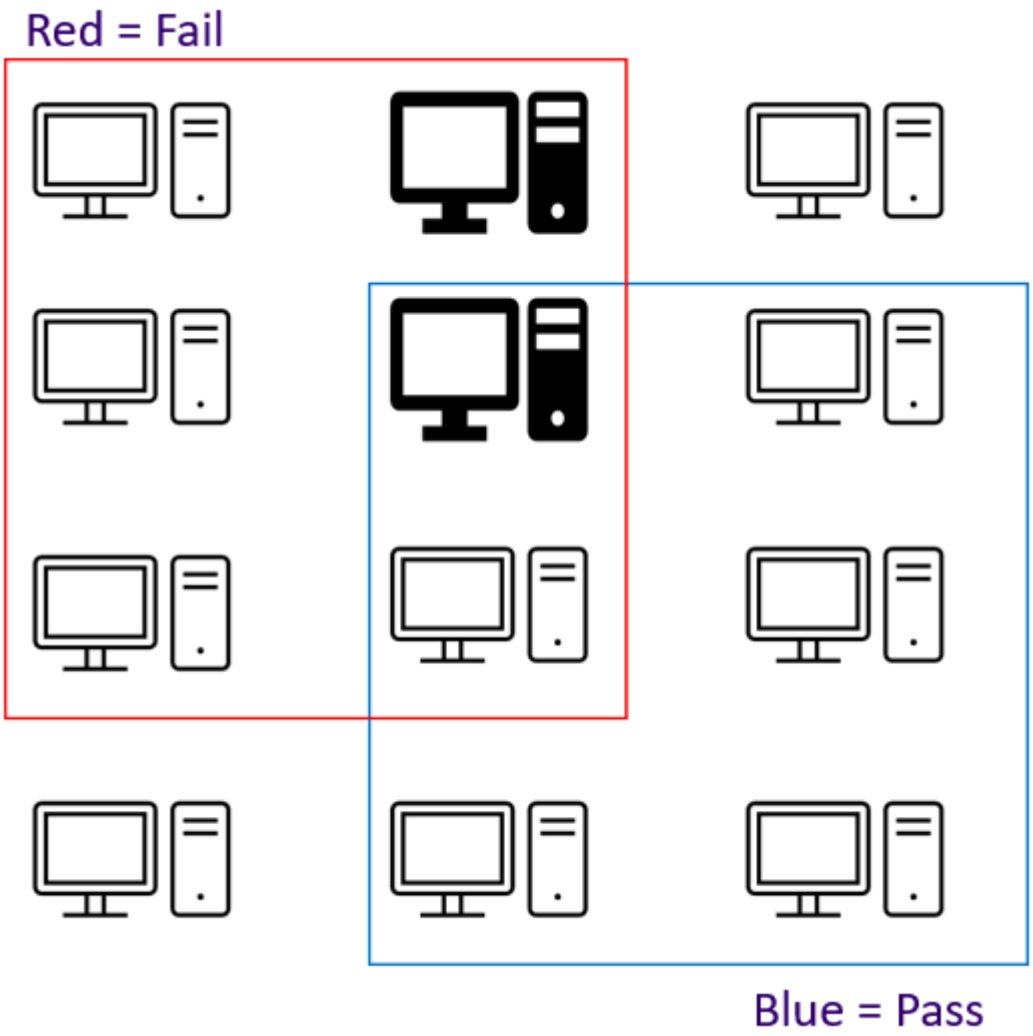

Figure 1.2 Example Node Pool with Sample Decision Node Groups

The nodes in this example are grouped into decision groups of six where two faulty nodes in a decision group are enough to cause the consensus to fail. The blue outlined group shows a selection of nodes where the single faulty node would not prevent the consensus from succeeding, and the red group shows a selection where both faulty nodes are included, and the consensus is disrupted.

The Redbelly Blockchain was not the only attempt this method of varying decision nodes. The ComChain system was introduced to develop a type of Private Blockchain system called a Communal Blockchain, which uses a "Configuration block" to determine which nodes are called upon for the next consensus(Vizier & Gramoli, 2020). The Honey Badger BFT Protocol is another method that focusing on adding system capabilities to run transactions asynchronously, drastically reducing latency times (Miller et al., 2016).

The goal of this research is to analyze different combinations of total nodes (N_t), decision nodes (N_d), and compromised nodes (N_c), to determine combinations that reduce the chance of selecting too many faulty nodes, compromising the system. The R programming language was used to create a model that processes the different combinations of variables to determine the probability that consensus will be compromised for any combination of nodes, while energy consumption is primarily based on the number of decision nodes running per consensus. The model compares the compromise chance to the number of decision nodes to allow system architects to make informed choices on how many N_d to include in their consensus processes to balance security and energy consumption.

This research consists of two parts. The first experiment examines the effect on the security of a distributed network by changing the number of N_c in the network while holding the number of N_d and N_t constant. The second experiment examines the effect of changing the number of N_d in the network while holding the number of compromised and N_t constant.

2. METHODS

The three variables describing this model are N_t, N_d, and N_c. A node is an actor in the network with the power to return a decision to be used in consensus. The number of nodes available to the network for use in consensus is N_t. N_d are those nodes selected for each individual consensus out of the pool of N_t. N_c are those that do not return any decision or incorrect information due to either the node being hacked and malicious, or defective due to maintenance issues with the nodes or the information lines connecting the nodes. For example, consider a distributed network consisting of 100 nodes capable of returning a decision, 10 of those nodes are called upon to provide a decision for a consensus, 4 nodes have been hacked and are malicious, and 3 nodes are down for maintenance issues but have not yet been taken off the active list in the pool. In this scenario, the N_t are the 100, the number of N_d is 10, and the number of N_c is 7 (4 malicious and 3 down).

In both experiments run, the primary statistic observed is the probability of consensus disruption. It is assumed that the system cannot distinguish the N_c from the properly functioning nodes when selecting N_d from the pool. Thus, there exists a chance to select compromised nodes when selecting N_d. The BFT algorithm dictates that a consensus can only be disrupted if greater than (N_d-1)/3 nodes are compromised (Castro & Liskov, 2002; Lamport et al., 1982). In this model, this threshold number required to disrupt a consensus is denoted by the variable *k.* This variable is static in the first experiment and will become dynamic in the second experiment where the number of N_d is varied across each experimental run.

An important distinction is that disruption of a consensus is not the same as compromise of a consensus. A consensus being disrupted means that it cannot reach a secure decision due to not enough responses being uniform. This can happen due to null responses received from nodes down for maintenance who cannot return a decision, or dissenting decisions received from malicious nodes. A consensus compromise can only occur from greater than two thirds of the N_d being malicious and returning dissenting decisions. Null decisions cannot contribute to proper consensus or compromised consensus, only disruption. Disruption of consensus is statistically more likely to occur than compromise, so it will be the focus of these experiments.

The probability of compromised consensus could be modeled in a similar manner to the methods in this research. One method would be calculated by setting the *k* value to one third of the N_d and finding the probability that consensus would be successful or disrupted, and then inverting. This method would give success and disruption probability collectively as it would not distinguish between the system properly functioning and failing but not compromised. Alternatively, it could be done by changing the *k* value to be two thirds of the N_d and inverting the locations of malicious and non-malicious nodes in the combinatorics equations.

2.1. EXPERIMENT 1

The first experiment in this study holds the number of N_d and N_t constant while varying the number of N_c to observe how the probability of disruption changes. A program written in R programming language (Appendix A) was created allowing N_d and N_t to be set to any numbers so long as N_d is less than N_t. The algorithm is as follows:

1. The number of decision (N_d) and N_t are set for the particular experiment.
2. N_c is set to 0.
3. While N_c <=N_t
    a. the program determines the probability density of the hypergeometric distribution using the dhyper function in R.
        i. The probability of selecting a value of N_c less than *k* is calculated, then repeated for all values less than *k*.
        ii. These values are then summed to find the probability of secure consensus, then inverted to obtain the probability of compromise.
    b. Set N_c =N_c +1
4. The resulting data is plotted showing the relationship between Nc and compromise probability and the trends as Nc increases.

The dhyper function in R takes in the number of total items (N_t) and splits it into two types, in our case the two types are $N_d$and N_c (Team,2021). The function then determines the likelihood that a specific value of N_c will be chosen out of the total. However, the likelihood that a specific number is chosen does not indicate how likely it is that a consensus will be compromised. To determine the probability of disruption, the probability density is calculated for each potential number of disrupted nodes less than but not equal to *k.* These probabilities are then summed to give the final probability value that the consensus in question will not be disrupted, then inverted to obtain the probability of compromise. This increases efficiency as the number of iterations required to find the probability of secure consensus is less than would be required to find the

probability of compromise. This process repeats for each combination of N_d, N_t, and N_c to determine how the probability of compromise changes as the number of disrupted nodes increases.

2.2. EXPERIMENT 2

The second experiment varies N_d while N_t and N_c are held constant. The R program also uses the dhyper function to find the probability density of the hypergeometric distribution. In this experiment the variable N_d is changed, but as the *k* value is based solely on N_d its value is expected to impact the results to an observable extent. The algorithm for this second experiment is as follows:

1. The number of N_c and N_t are set. for the particular experiment.
2. N_d is set to 5
3. While N_d <=N_t
    a. the program determines the probability density of the hypergeometric distribution using the dhyper function.
        i. The probability of selecting a number of N_c less than *k* is calculated, then repeated for all values less than *k*.
        ii. These values are then summed to find the probability of secure consensus, then inverted to obtain the probability of compromise.
    b. Set N_d =N_d +1
4. The resulting data is plotted showing the relationship between N_d and compromise probability and the trends as N_d increases.

First, N_t and N_c are entered. Then the iterations begin by taking the number of N_d and calculating the *k* value. N_d is initialized to 5, as using any less is impractical for newer BFT systems (Miller et al., 2016). The three main variables and *k* are used in the dhyper function to obtain the probability of picking between 0 and k-1 disrupted nodes out of the N_d. Picking fewer than *k* compromised nodes results in a secure consensus, so the value is inverted to obtain the probability of disruption. This process is performed in this manner as it requires fewer iterations. The next iteration then begins with the next number of N_d, and this continues until the number of N_d equals the number of N_t. The probability of compromise for each respective number of N_d is then charted so that trends and patterns can be determined.

2.3. ANALYSIS

Once the results are obtained, they will be analyzed for patterns. For experiment 1, the controlled variables are N_d and N_t. N_d is initialized to a low value and increase while N_t is constant to observe pattern changes. Then N_t will be increased to observe pattern changes and analyzed for potential proportional relationships.

For the second experiment, Nc is initialized as a low percentage of N_t and is increased to observe patterns in the results. Once this is complete, any points of interest will be examined. Then the experiment will be run again at higher proportions of Nc to N_t to observe any changes in patterns.

3. RESULTS

3.1. EXPERIMENT 1 RESULTS

The first experiment examined the consensus disruption chance changes with varying N_c while N_d and N_t are held constant. Figure 3.1 shows the results obtained using 38 N_d and 100 N_t.

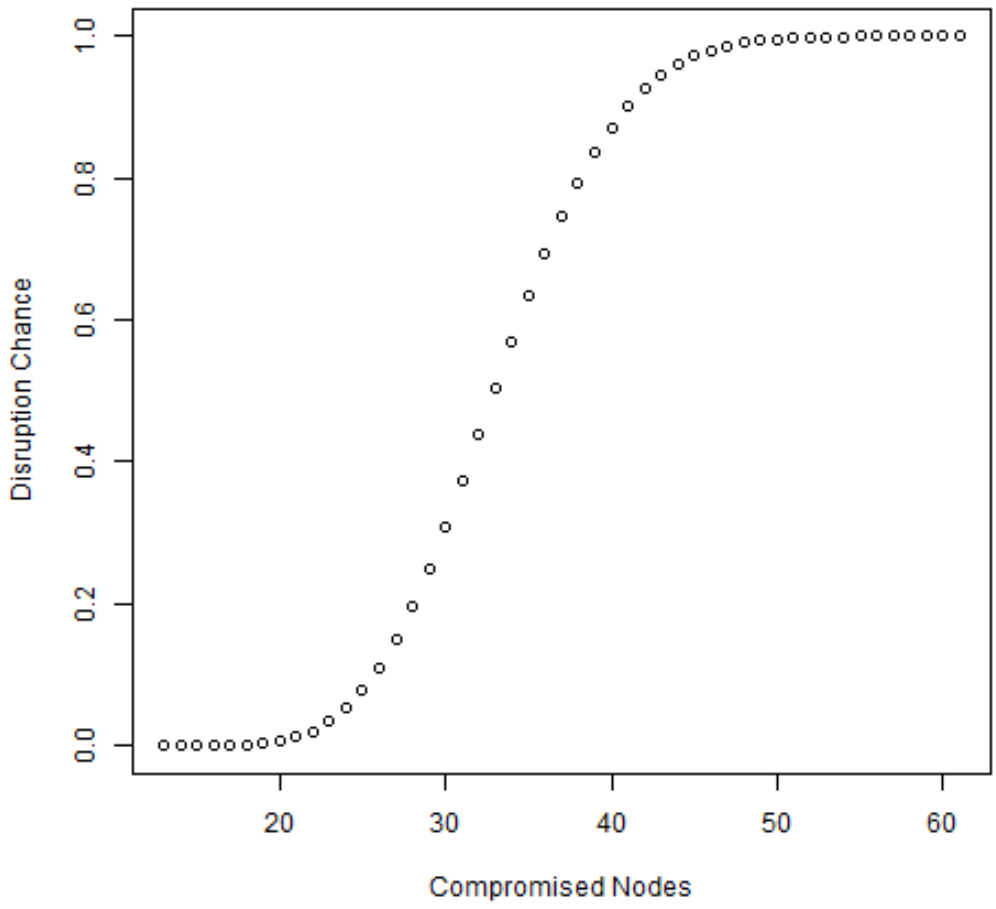

Figure 3.1 Comparison of Disruption Chance to Number of Compromised Nodes using 38 Decision Nodes out of 100 Total Nodes

The results show that the trends follow a logistical curve. The initial probabilities on the lower left remains at a zero percent chance of disruption until N_c reaches 13. This is due to the *k* value being 13 when using 38 N_d, making it impossible to disrupt a consensus with 12 or fewer. The opposite end at the top right reaches a nearly flat pattern when 50 N_c are in the network. From this point, this line slowly approaches a 100% disruption chance until there are 75 N_c, in which case the line reaches 100% and remains there as it is impossible to choose less than 13 N_c out of the 38 when 75 or greater N_t are compromised. To show how changing the number of N_d alters the curve, Figure 3.2 displays the results of using only 10 N_d while still maintaining a total pool of 100 N_t. Altering N_d changes the curve significantly, mainly in the position along the x-axis. Figure 3.3 shows the effect of changing N_t to fifty while keeping the N_d at 10. We see that the curves in Figures 3.2 and 3.3 are similar in shape. This is due to a proportional relationship between the number of N_t and N_c. The number of N_t is halved, but the disruption chance does not change for the same ½ proportioned number of N_c.

3.2. EXPERIMENT 2 RESULTS

In the second experiment the number of disrupted nodes and N_t remain constant, and the disruption chance is calculated while iterating the number of N_d used. This is synonymous with a user taking a given value of N_t and estimate the highest number of down or N_c their system might have before it is noticed and acted upon. Using these results, they could then determine the best number of N_d to use to minimize the chance of transaction disruption.

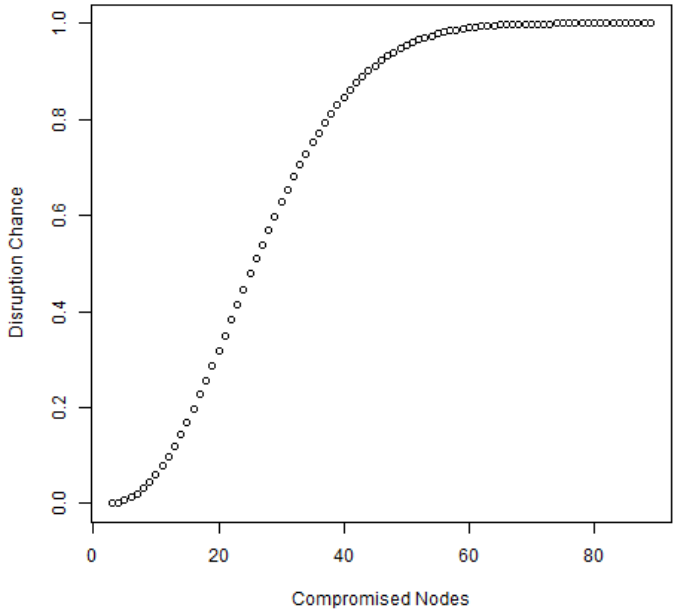

Figure 3.2 Comparison of Disruption Chance to Number of Compromised Nodes using 10 Decision Nodes out of 100 Total Nodes

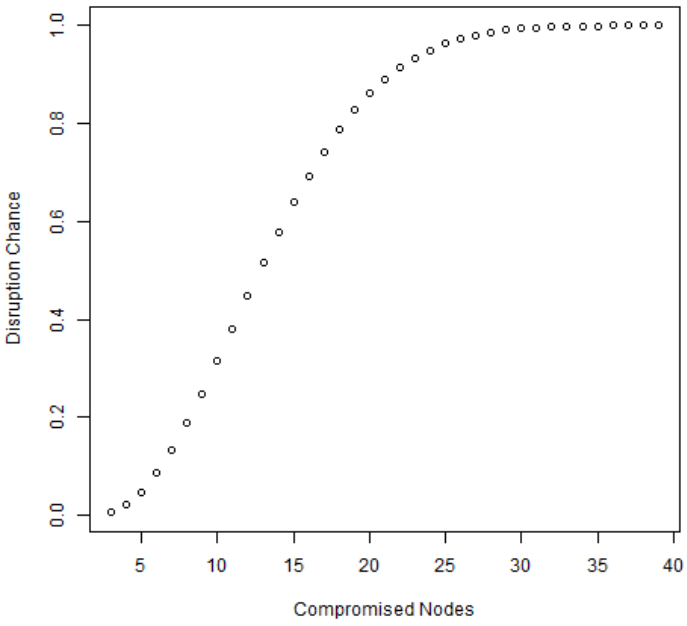

Figure 3.3 Comparison of Disruption Chance to Number of Compromised Nodes using 10 Decision Nodes out of 50 Total Nodes

First, a simulation is run using a low number of N_c. The initial chart is obtained using 100 N_t where 10 of those nodes are compromised. The results are shown in Figure 3.4.

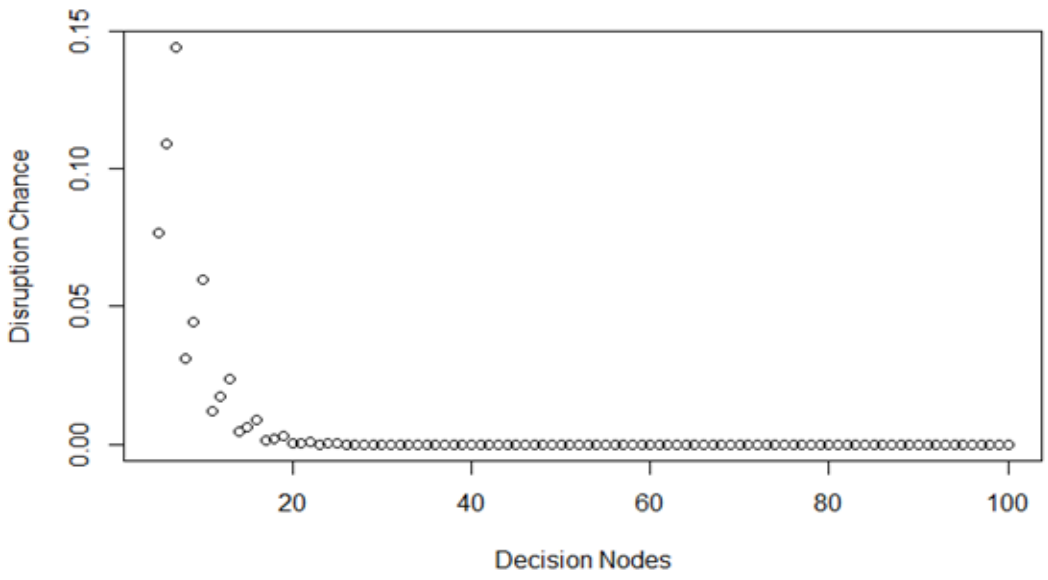

Figure 3.4 Comparison of Disruption Chance to Number of Decision Nodes using 10 Compromised Nodes out of 100 Total Nodes

Figure 3.4 shows a steep decline early in the data. This trend flattens out at around 20 N_d until it reaches 0 at 32, where 10 N_c can no longer have any chance of disrupting consensus. The early data is of particular interest as it shows that the trend is not linear as the charts in the first experiment. Another chart is created using 25 N_c and 100 N_t to observe the changes in the pattern as the N_c are increased and the results shown in Figure 3.5.

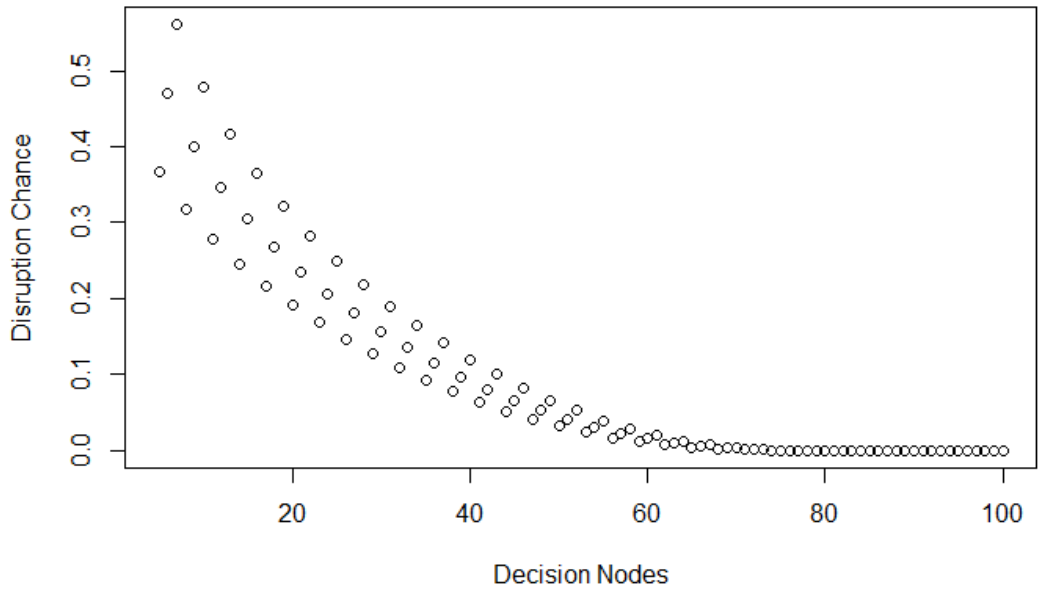

Figure 3.5 Comparison of Disruption Chance to Number of Decision Nodes using 25 Compromised Nodes out of 100 Total Nodes

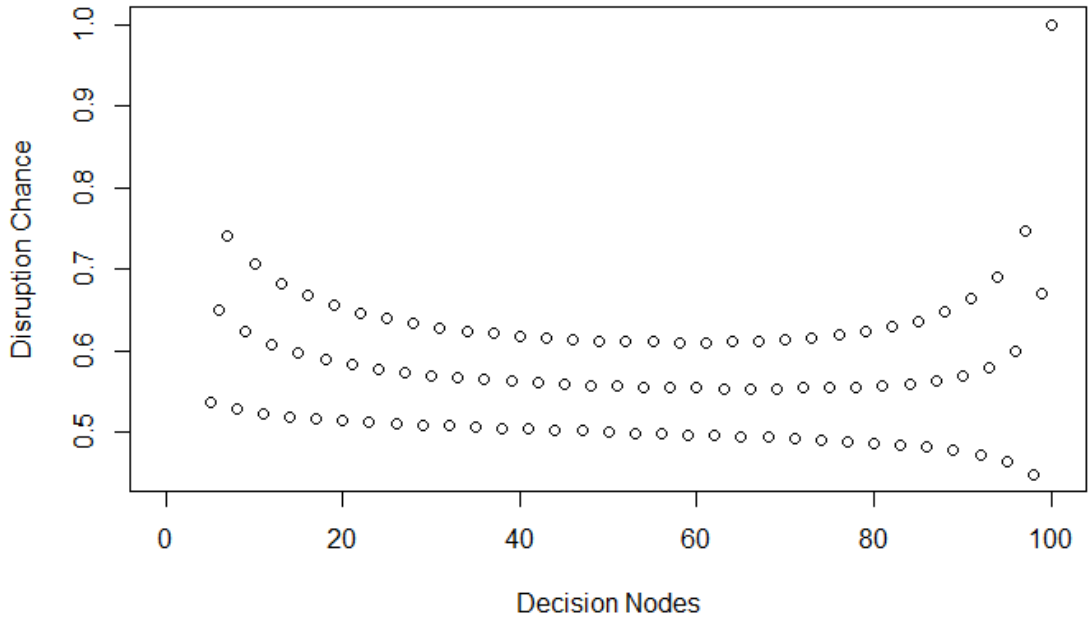

Figure 3.6 Comparison of Disruption Chance to Number of Decision Nodes using 33 Compromised Nodes out of 100 Total Nodes

The results in Figure 3.5 show a pronounced trichotomous pattern. The downward slope has also elongated, not reaching 0 until 76 N_d are examined, where the 25 N_c can no longer disrupt consensus. Next the effect of 33 N_c out of 100 N_t was examined. This number is of particular interest as it is the k value of the number of the N_t. The results of using 33 N_c out of 100 N_t are shown in Figure 3.6.

Figure 3.6 shows a pronounced trichotomous nature in the results. The lines do not converge to a 0% disruption chance but diverge as the value of N_d approaches 100. This likely indicates that this is the point at which the trends in disruption chance change from converging at 0 to converging at 1. To confirm this, we examine the immediate neighbors, 32 and 34 N_c (Figures 3.7 and 3.8 respectively).

It can be seen that just below the 33 compromised node point, the results converge at 0, and above the point they converge at 1. Figure 3.9 shows the results of using 50 N_c out of 100 total. These results show that the trends in Disruption Chance continue to rise more steeply as N_c increases past 33.

The trichotomous nature of the results requires additional explanation. The groups that form these lines are characterized by their results for taking the number of nodes modulo 3. The middle line is N_d modulo 3 = 0 (6,9,12, etc.), the top line is $N_d$modulo 3 = 1 (7,10,13, etc.), and the bottom line is $N_d$modulo 3 = 2 (8,11,14, etc.). The three lines formed in the results are most pronounced in Figure 3.6. That Figure has been broken down and each group charted individually as shown in Figures 3.10, 3.11, and 3.12. These Figures show that the best results are always obtained using the Modulo3=2 group (Figure 3.12) and the worst results are the Modulo3=1 group (Figure 3.10).

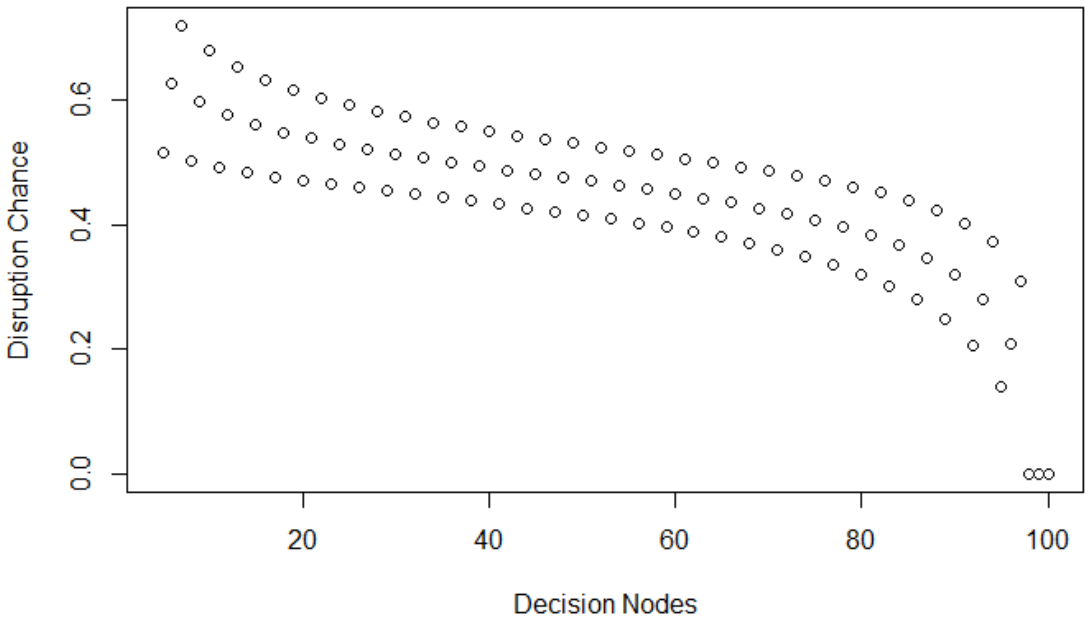

Figure 3.7 Comparison of Disruption Chance to Number of Decision Nodes using 32 Compromised Nodes out of 100 Total Nodes

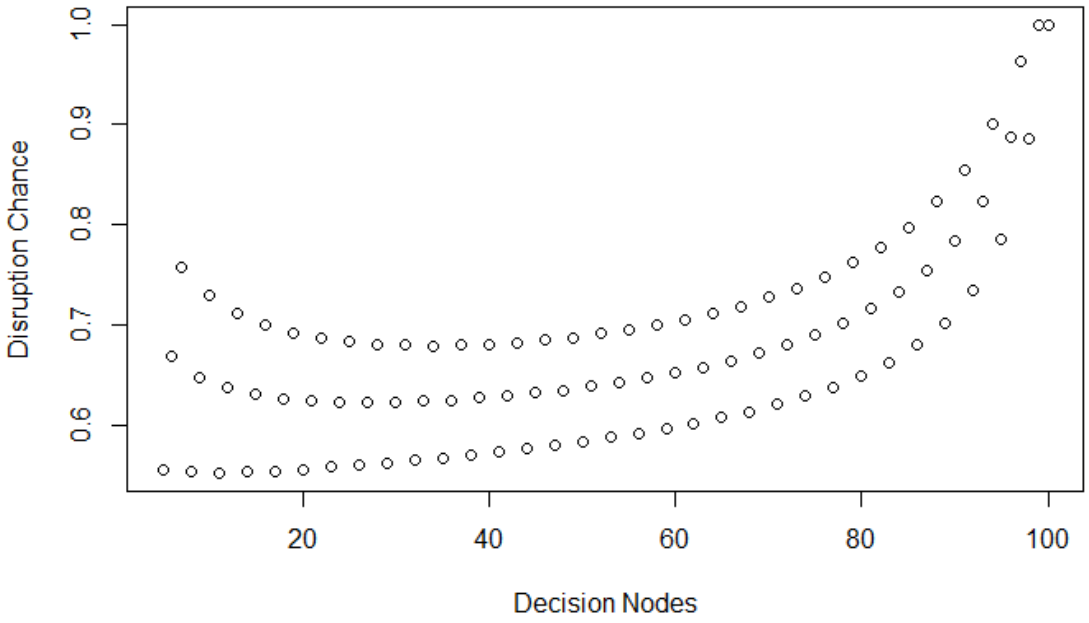

Figure 3.8 Comparison of Disruption Chance to Number of Decision Nodes using 34 Compromised Nodes out of 100 Total Nodes

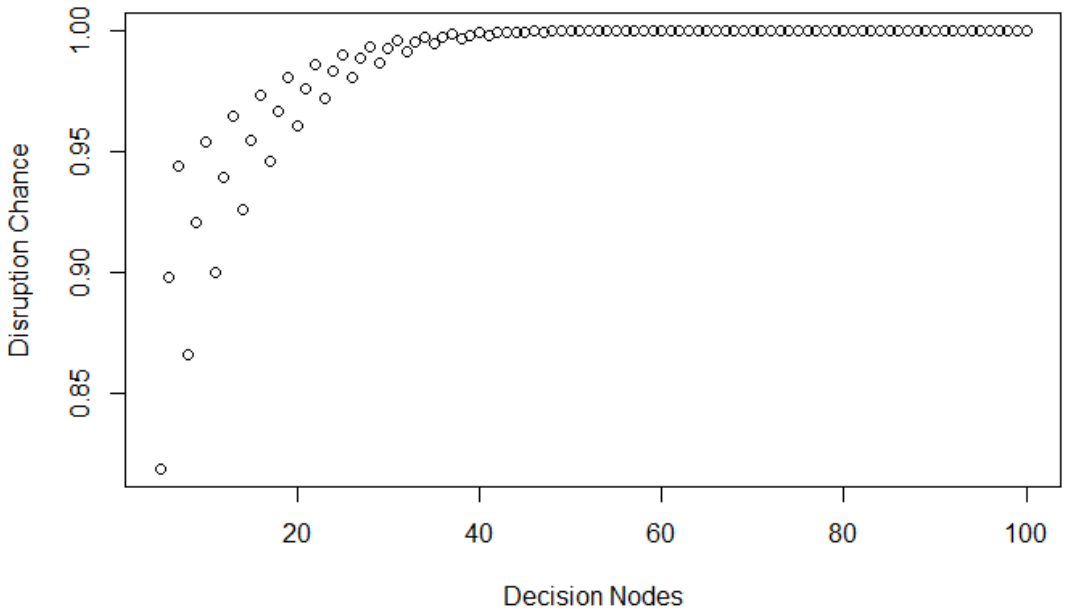

Figure 3.9 Comparison of Disruption Chance to Number of Decision Nodes using 50 Compromised Nodes out of 100 Total Nodes

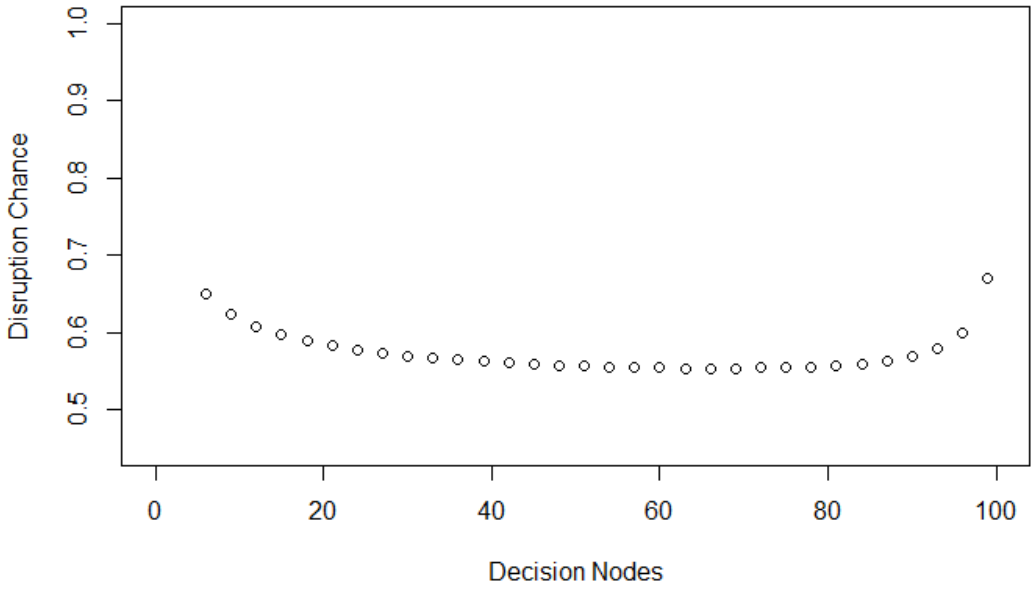

Figure 3.10 Comparison of Disruption Chance to Number of Decision Nodes of the Mod3=0 group using 33 Compromised Nodes out of 100 Total Nodes

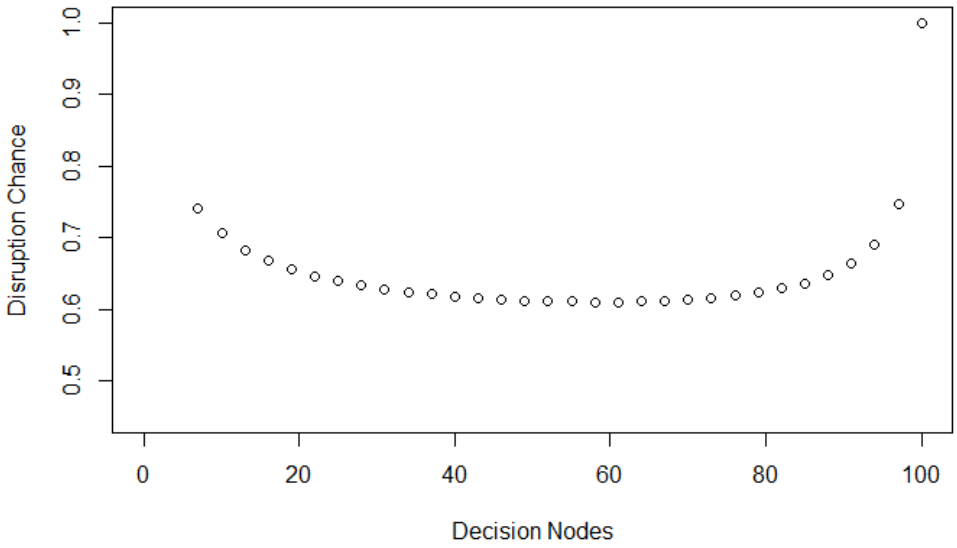

Figure 3.11 Comparison of Disruption Chance to Number of Decision Nodes of the Mod3=1 group using 33 Compromised Nodes out of 100 Total Nodes

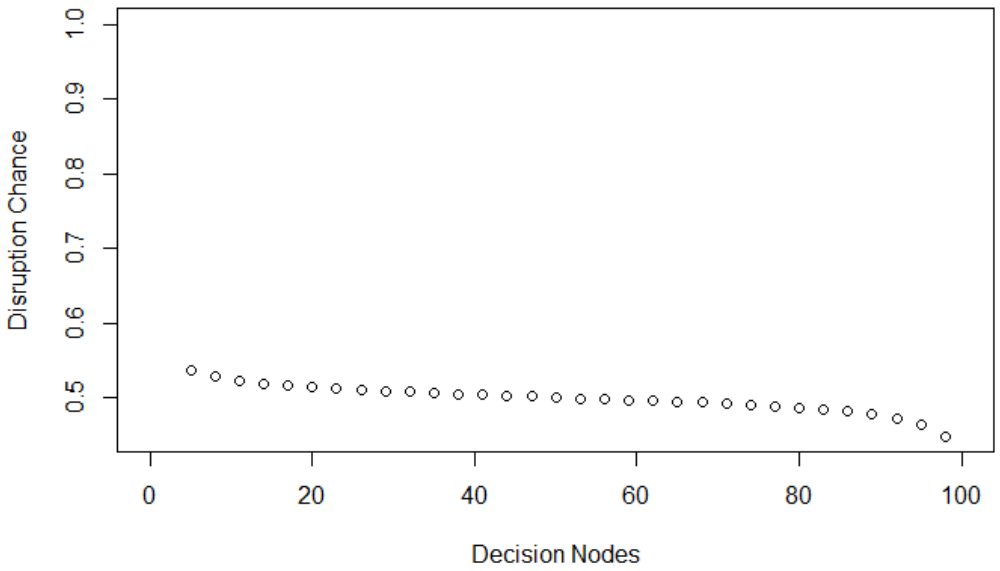

Figure 3.12 Comparison of Disruption Chance to Number of Decision Nodes of the Mod3=2 group using 33 Compromised Nodes out of 100 Total Nodes

The trichotomous nature of these node combinations makes a direct comparison impossible, as increasing both compromised and N_t exactly proportionately has varied results. If the value of N_c is increased from 33 to 66, then a value of 199 and not 200 should be used for N_t. This is due to each *k* value pertaining to 3 decision node values. In the case of 33 N_c, this is the *k* value for 98, 99, and 100 N_d with 100 falling in the Modulo3=1 group. When examining 66 N_c, the corresponding decision node values are 197, 198, and 199 with 199 falling in the Modulo3=1 group. So, a better comparison exists between 33 N_c out of 100 total, and 66 N_c out of 199 total. The results of 66 compromised and 199 N_t are shown in Figure 3.13.

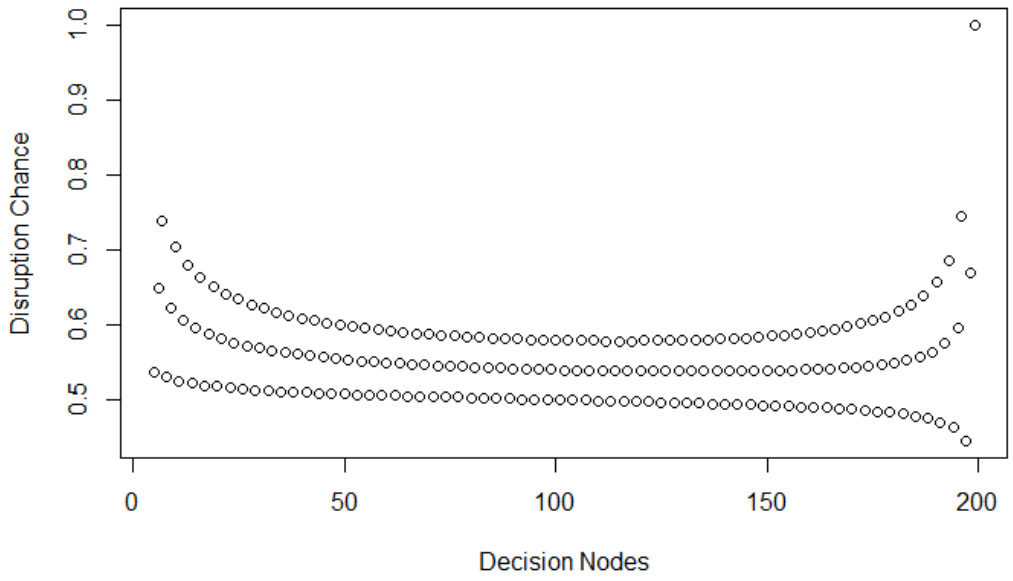

Figure 3.13 Comparison of Disruption Chance to Number of Decision Nodes using 66 N_c out of 199 Total Nodes

Figure 3.13 displays the same behavior as Figure 3.6 with a small discrepancy. The top two lines (Mod3=1 and Mod3=0) have a decreased average value and the bottom

line (Mod3=2) has an increased average value. Results were obtained from using higher numbers of compromised and N_t comparable to the previous sets and the resulting average values of each Mod group is tabulated in Table 3.1.

Table 3.1 Average Modulo Group Values of Proportionately Increasing Node Sets

Tot Nodes : Mod3=1					
Averages					
Compromised/Total Nodes	33/100	66/199	99/298	132/397	165/496
Mod3=0	0.574074	0.555091	0.5460413	0.540441	0.536527
Mod3=1	0.6513654	0.611781	0.5931466	0.58168	0.573691
Mod3=2	0.4983333	0.4991625	0.4994407	0.49958	0.499664

4. DISCUSSION

4.1. TRENDS IN EXPERIMENT 1 RESULTS

The first experiment was run to see how network performance of a predetermined system set up of N_d and N_t is affected by changes in the value of N_c. In the results, we observe a logistical curve. The line holds at zero until N_c increase past the *k* value of the N_d, when they become capable of disrupting a consensus. The compromise probability raises rapidly past this point.

We also observe in the results that increasing the number of N_d pushes the curve further along the x-axis so that a larger number of N_c is required to achieve the same disruption chance compared to situations with a lower value of N_d and equal N_t. Increasing the value of N_d to decrease disruption probability is nothing new, this is widely accepted to be an effective method of increasing security. But this is not the best method of optimization as increasing the value of N_d is the largest contributor to increased energy cost and latency times.

Increasing the value of N_d and the total number of nodes by proportionate amounts increases the value of N_c required to achieve similar levels of compromise probability. This means that, if run concurrently, a system could run more transactions simultaneously, the energy cost and latency times per transaction would not increase, and disruption probability would decrease.

4.2. TRENDS IN EXPERIMENT 2 RESULTS

Note the three groups formed in each graph in the second part of the simulations. Here, the disrupted nodes and N_t are held constant, and the disruption chance is calculated for each possible value of N_d. These three groups are characterized by their number of N_d modulo 3. The bottom line (best result) is where decision node modulo 3 is 2, the middle is 1, and the top (worst result) is 0. This is explained by the picking probability and the number of disrupted nodes required to disrupt a transaction.

Consider the disruption chance for a transaction when values of 5, 6, and 7 are used for N_d. The modulo 3 results for these numbers are 2,0, and 1 respectively, so from left to right they represent the bottom, middle and top lines, or best, moderate, and worst results. The reason that using 5 N_d results in the lowest disruption chance is that 2 disrupted nodes are required to disrupt this transaction, where 2 nodes are also able to disrupt a transaction using both 6 and 7 N_d. When the chosen number of N_d increases from 5, each additional decision node increases the chance of selecting a disrupted node, and therefore the chance of disrupting the transaction. This pattern is observable in each set of three numbers as the value of N_d increases.

4.3. PROPORTIONAL TRENDS

Most of the results show that the disruption chance converges to 0 at high values of N_d if the number of disrupted nodes is below one third the total number of nodes, and that it converges at 1 if the number of disrupted nodes is above one third. The only exception being if the number of disrupted nodes is exactly one third the number of N_t. This research shows the best number of nodes to use in a transaction with given total and

N_d and assumed disrupted nodes. Here, it is assumed that lower numbers of N_d are best for the system in terms of processing power, speed of transactions, and energy efficiency.

4.4. CASE EXAMPLE

Private companies who desire to create their own BFT Blockchain system can use the information presented in this study to design the physical architecture of their system in terms of how many total nodes to include and how many to include in each decision to maximize security while minimizing energy consumption and latency times. It is not meant to provide a final "best" answer to a system set up, instead, its primary purpose is to find a point that best balances the system organizer's priority metrics.

Some companies are looking to set up BFT Blockchain networks in their supply chains to record the handling of sensitive materials. In this case, the company is the system organizer and determines their acceptable ranges for number of total nodes and N_d and consult system technicians to determine approximate numbers of N_c possible in the system at a time. They could then use this model to obtain figures showing the compromise probability of their transactions with their preferred parameters. Minor alterations would then be made to the system setup to reach their optimal point.

This model is best tailored to Private Permissioned Blockchain networks where all nodes on the network are pre-approved and can be monitored by the system organizer. Certain companies, however, would prefer to share the system amongst partners where system nodes are spread across the various companies in the system. This system setup is known as a communal block chain, and could also be modeled similarly, but with a few adjustments.

Communal blockchains are the next logical step for this model, but more work could be done to alter it such that it becomes applicable to a much wider variety of system types to find their optimal setups as well. System developers are always looking for ways to improve their systems. This model provides another way for them to do just that.

5. CONCLUSIONS

BFT System developers who need to optimize their networks without delving into the inner workings of BFT algorithms should look to the methods shown in this research to optimize their nodal arrangements. The results of this study show that transaction disruption chance follows patterns that can be predicted statistically. By selecting a number of decision nodes where N_dMod3=2, they can optimize their system energy consumption while maintaining the lowest chance of transaction disruption.

While many believe that more decision nodes result in higher security, this research shows that this is not the case for variable node systems. The largest source of energy consumption in BFT networks comes from nodes running computations and returning decisions for consensus. Adding nodes to the system also increases latency times as there are more nodes requiring bandwidth and increasing the chance of messaging delays. Increasing the decision nodes not only increases the energy usage of the system and lengthens latency times, if it is increased to an amount not in the Mod3=2 group, it can actually decrease system security.

This research shows that administrators of variable node systems should consider altering the amount of decision nodes they use in consensus. In some cases, it is even preferable to counter-intuitively reduce decision nodes. If the decision node amount is not in the Mod3=2 group, the system could drop nodes to an amount in the group and increase security while simultaneously reducing energy costs and latency times. This holds great potential for variable node system optimization.

This method applies to Private Distributed BFT networks but, with some alteration, should be able to be applied to a wider variety of network types. Other models should be created to test its robustness and other factors such as differentiating between malicious and non-malicious faulty nodes.

APPENDIX

1. EXPERIMENT 1 CODE

```
library(tidyverse)

library(plyr);library(dplyr)

library(openxlsx)
library(rio)
library(combinat)

library(RcppAlgos)
library(writexl)

#Total Nodes
N = 100
#Decision Nodes
n =10
#Compromised Node information
a <- ceiling((n-1)/3)
b <- N-(n+1)



m = a:b
k = 0:(a-1)

failure <- c(list, length(m))

for(i in a:b){
  pmf = dhyper(k,i,N-i,n)
cbind(k,pmf)

failure[[i-(a-1)]] = 1- sum(pmf)
}

#png( paste(n," decision nodes out of ", N, " total nodes.png", sep="")
)

plot(m,unlist(failure),main = paste(n," decision nodes out of ", N, " t
otal nodes", sep=""), xlab = "Compromised Nodes", ylab = "Disruption Ch
ance")
```

2. EXPERIMENT 2 CODE

```
library(tidyverse)

library(openxlsx)
library(rio)

library(RcppAlgos)
library(writexl)
library(ggplot2)
library(colorspace)
library(ggpubr)

# N: Total Nodes
# n: Decision Nodes
# m: Total number of Compromised Nodes
# k: number of compromised nodes selected

#set info
#Total Nodes
N = 100
tempn=5 #starting point for iterations
start=5 #starting point not iterated
n=tempn:N
m = 33

#k = 0
iterations <- n
failureA <- c(list, length(iterations))
failureB <- c(list, length(iterations))
failureC <- c(list, length(iterations))

DecNodesA <- c(list, length(iterations))
DecNodesB <- c(list, length(iterations))
DecNodesC <- c(list, length(iterations))

A=1
B=1
C=1

failure <- c(double,length(n))


for(i in n){

k <-  (0:(ceiling((tempn-1)/3)-1))
```

```
  if((ceiling((tempn-1)/3)-1)>m){

     failure[[i-(start-1)]] <-0


  }else{

     failure[[i-(start-1)]] <- 1-sum(dhyper(k,m,N-m,i))

    if( i%%3 == 0){
    failureA[[A]] <-1-sum(dhyper(k,m,N-m,i))
    DecNodesA[[A]] <- i
      A=A+1

  }else if(i%%3 == 1){
    failureB[[B]] <-1-sum(dhyper(k,m,N-m,i))
     DecNodesB[[B]] <- i
      B=B+1
  }else if(i%%3 == 2){
    failureC[[C]] <-1-sum(dhyper(k,m,N-m,i))
     DecNodesC[[C]] <- i
      C=C+1
  }else{
    print("ERROR")
  }


  }

tempn=tempn+1
 k=NULL

}

plot(n,unlist(failure),main = paste(m," compromised nodes out of ", N,
" total nodes", sep=""), xlab = "Decision Nodes", ylab = "Disruption Ch
ance",xlim=c(0,N), ylim=c(0.4,0.85))

plot(DecNodesA, failureA, main= paste("Mod3=0: ",m,":Comp Nodes,", N, "
:Total Nodes", sep=""), xlab = "Decision Nodes", ylab = "Disruption Cha
nce",xlim=c(0,N), ylim=c(0.4,0.85))

plot(DecNodesB, failureB, main= paste("Mod3=1: ",m,":Comp Nodes,", N, "
:Total Nodes", sep=""), xlab = "Decision Nodes", ylab = "Disruption Cha
nce",xlim=c(0,N), ylim=c(0.4,0.85))

plot(DecNodesC, failureC, main= paste("Mod3=2: ",m,":Comp Nodes,", N, "
:Total Nodes", sep=""), xlab = "Decision Nodes", ylab = "Disruption Cha
nce",xlim=c(0,N), ylim=c(0.4,0.85))
```

BIBLIOGRAPHY

Castro, M., & Liskov, B. (2002). Practical Byzantine Fault Tolerance and Proactive Recovery. *ACM Transactions on Computer Systems*, *20*(4), 398–461. https://doi.org/10.1145/571637.571640

Concurrent Systems Research Group, University of Sydney, D.-C. (n.d.). *The Red Belly Blockchain Experiments*. *1*. https://redbelly.cdn.prismic.io/redbelly%2F7ad73d48-ea5e-49d2-8ec9-fca77c453ec1_redbellyblockchain-experiments.pdf

Da Silva, V. F., Coelho, M. N., Coelho, B. N., Coelho, V. N., & Coelho, I. M. (2019). A home ledger approach for IoT enabled devices. *Proceedings - Symposium on Computer Architecture and High Performance Computing*, *2019-Octob*, 227–233. https://doi.org/10.1109/SBAC-PAD.2019.00044

Franke, L., Schletz, M., & Salomo, S. (2020). Designing a blockchain model for the paris agreement's carbon market mechanism. *Sustainability (Switzerland)*, *12*(3), 1–20. https://doi.org/10.3390/su12031068

Ghosh, E., & Das, B. (2020). A Study on the Issue of Blockchain's Energy Consumption. *Advances in Intelligent Systems and Computing*, *1065*, 63–75. https://doi.org/10.1007/978-981-15-0361-0_5

Konnov, A., Makarov, A., Pozdnyakova, M., Safin, R., & Salagaev, A. (1999). Russia. *Laboratory for Computer Science, Massachusetts Institute OfTechnology, 545 Technology Square, Cambridge, MA 02139*, *February*, 359–368. https://doi.org/10.1007/978-0-387-95982-5_25

Lamport, L., Shostak, R., & Pease, M. (1982). The Byzantine Generals Problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, *4*(3), 382–401. https://doi.org/10.1145/357172.357176

Miller, A., Xia, Y., Croman, K., Shi, E., & Song, D. (2016). The Honey Badger of BFT protocols. *Proceedings of the ACM Conference on Computer and Communications Security*, *24-28-Octo*(Section 3), 31–42. https://doi.org/10.1145/2976749.2978399

Nair, R., Gupta, S., Soni, M., Kumar Shukla, P., & Dhiman, G. (2020). An approach to minimize the energy consumption during blockchain transaction. *Materials Today: Proceedings*, *xxxx*. https://doi.org/10.1016/j.matpr.2020.10.361

Sedlmeir, J., Buhl, H. U., Fridgen, G., & Keller, R. (2020). The Energy Consumption of Blockchain Technology: Beyond Myth. *Business and Information Systems Engineering*, *62*(6), 599–608. https://doi.org/10.1007/s12599-020-00656-x

Sternberg, H. S., Hofmann, E., & Roeck, D. (2020). The Struggle is Real: Insights from a Supply Chain Blockchain Case. *Journal of Business Logistics*, 1–17. https://doi.org/10.1111/jbl.12240

Team, R. C. (2021). *A language and environment for statistical computing.* Vienna, Austria: R Foundation for Statistical Computing.

Vizier, G., & Gramoli, V. (2020). ComChain: A blockchain with Byzantine fault-tolerant reconfiguration. *Concurrency Computation*, *32*(12), 1–26. https://doi.org/10.1002/cpe.5494

Wang, F., Ji, Y., Liu, M., Li, Y., Li, X., Zhang, X., & Shi, X. (2021). *An Optimization Strategy for PBFT Consensus Mechanism Based On Consortium Blockchain*. 71–76. https://doi.org/10.1145/3457337.3457843

Wang, Y. (2019). Designing a blockchain enabled supply chain. *IFAC-PapersOnLine*, *52*(13), 6–11. https://doi.org/10.1016/j.ifacol.2019.11.082

Zhao, W. (2009). Design and implementation of a Byzantine fault tolerance framework for Web services. *Journal of Systems and Software*, *82*(6), 1004–1015. ttps://doi.org/10.1016/j.jss.2008.12.037

VITA

Ian Robert Fulton received his Bachelor of Science in Mechanical Engineering in the Spring of 2019 from Missouri University of Science and Technology. He enrolled in the master's degree program in Engineering Management at Missouri University of Science and Technology in the Fall of 2019. He performed research for the Boeing company under Dr. Steven Corns beginning in the Spring of 2020. He received his Master of Science in Engineering Management and Certificate in Systems Engineering from Missouri University of Science and Technology in May 2022.