

A Simulation Study of Connectivity Metrics for  
Wireless Ad Hoc Networks

by

Samuel Nelson

A Thesis

Presented to the Faculty  
of Bucknell University

In Partial Fulfillment of the Requirements for the  
Degree of Bachelor of Science with Honors in Computer Science

May 8, 2006

Approved By: \_\_\_\_\_

Luiz Felipe Perrone  
Thesis Advisor

\_\_\_\_\_  
Gary Haggard  
Chair, Department of Computer Science

*“The eyes of the Lord safeguard knowledge,  
but He defeats the projects of the faithless.”*

- Proverbs 22:12

# Table of Contents

1	Introduction.....	1
1.1	Concept Definitions .....	2
1.2	Wireless Ad Hoc Network Basics.....	5
1.3	Attacks and Connectivity Issues .....	6
1.4	Network-to-Metric Problem.....	8
2	Network Operation and Attack Details.....	12
2.1	Wireless Ad hoc Network Operation .....	13
2.2	Attack Details.....	15
3	Network-to-Graph Problem .....	20
3.1	Simulation and Connectivity Graphs .....	21
3.2	Redefining the network graph.....	22
3.3	Connectivity Graph Implementation in SWAN.....	25
4	Graph-to-Metric Problem.....	30
4.1	Connectivity Metrics.....	31
4.1.1	Weighted Metrics.....	33
4.1.1	Unweighted Metrics.....	35
4.2	Metric Implementation.....	36
4.3	Hypothesis Testing.....	39
4.3.1	Introduction to Hypothesis Testing.....	39
4.3.2	Hypothesis Testing Example .....	43

5	Experimental Results for the Graph-to-Metric Problem.....	45
5.1	Metric Scalability.....	47
5.2	Data Collection.....	49
5.3	Hypothesis Test Results on Grid Scenarios.....	51
5.4	Metric Comparisons.....	58
6	Conclusions and Future Work.....	67
7	Motivation.....	70
A	Metric Source Code Listings.....	74
A.1	Weighted Metrics.....	74
A.2	Unweighted Metrics.....	80
B	Standard Normal Cumulative Probability Table.....	84

# List of Tables

Table 3.1: Model Parameters .....	27
Table 3.2: PDR-edged graph.....	28
Table 4.1: PDR-graph input file.....	37
Table 4.2: Metric Values.....	38
Table 4.3: Example data sets .....	43
Table 4.4: Calculated Values .....	44
Table 5.1: Metric Listing .....	46
Table 5.2: Time / State Correspondence.....	50
Table 5.3: State / Mode Correspondence.....	52
Table 5.4: Summarized Data.....	58
Table 5.5: Overall Metric Rankings.....	59
Table 5.6: Jamming Metric Rankings.....	61
Table 5.7: Range Metric Rankings .....	61
Table 5.8: Scenario 1 Metric Rankings.....	63
Table 5.9: Scenario 2 Metric Rankings.....	64

# List of Figures

Figure 1: Network-to-metric problem decomposition .....	9
Figure 2.1: Jamming Attack.....	17
Figure 2.2: Range Attack .....	19
Figure 3: Grid Scenario.....	26
Figure 5.1: Scenario 1 .....	51
Figure 5.2: Scenario 2.....	51
Figure 5.3: Zoomed View of Green Nodes.....	52

# Abstract

Wireless ad hoc networks are a new technology, and the focus of much research. We have found, however, little literature investigating the quantification of network health, in a simulation environment. In this thesis, we propose a formulation and solution to the problem of summarizing the connectivity information of the network into a single real number, called a metric value. Using a divide-and-conquer based approach, we first summarize the useful connectivity information, for our purposes, into a weighted, undirected graph. The weight of an edge represents the ratio of discrete message units, called packets, successfully delivered over the link represented by that edge in a given time period. Using this graph as the foundation, the second step consists of applying a function, called a metric algorithm, to the graph to obtain a metric value. After cataloging a set of metric relevant algorithms, we use hypothesis testing to determine which metric algorithms best detect changes in network connectivity, due to malicious attacks on the network. Since all of our studies are done in a simulation environment, we have the ability to adjust the severity and time of an attack, allowing us to study how well each metric algorithm detects attacks under specific conditions. The results of this testing allow us to compare the metric algorithms and determine different sets of these algorithms best suited for detecting attacks within a specific range of simulation parameters.

# Chapter 1

## Introduction

Imagine the following scenario: A natural disaster, such as a flood or hurricane, devastates a major metropolitan area, leaving the city unreachable. Emergency response helicopters then fly over the area dropping tens of thousands of tiny battery-powered sensors, called smart dust [23]. They are about a cubic millimeter in size and have the ability to form a massive wireless ad hoc network; that is, automatically create and maintain connections to one another, forming a web of wireless communication. In other words, after deployed, the sensors immediately start detecting and establishing connections to other sensors nearby (say, within 200 meters). Furthermore, the emergency response team attempts to connect, using laptops, to some of the sensors that fell on the outskirts of the affected area. Each of the sensors then starts collecting data pertaining to its surroundings and, via the network the sensors created, relays that

information to the emergency response laptops. The response team now has real-time, constantly updated information about different densities, positions, velocities, and other data, of each section of the flood or hurricane.

When subsets of these nodes fail, get destroyed, or behave unexpectedly, the performance of the network is degraded. The question then is “how badly damaged is the network relative to its initial state”? It is immediately clear that many corollary questions follow. Is the state of the network now so bad that the data obtained is not trustworthy, accurate, or complete? How many more nodes should be deployed to achieve our goal of receiving an accurate and complete report on the area monitored? Before the underlying question about network damage is explored, it is important to define some terms and concepts that are used throughout the rest of this thesis.

## **1.1 Concept Definitions**

In the scenario presented above, the type of network formed is called a *wireless sensor network*. A wireless sensor network (WSN) is defined as “a network made of numerous small independent sensor nodes” which “are self-contained units consisting of a battery, radio, sensors, and a minimal amount of on-board computing power.” These nodes “self-organize their networks, rather than having a pre-programmed network topology.” [7]. A WSN is a type of *wireless ad hoc network*, which is a wireless network of computing nodes that is formed automatically without human intervention. Each node in the

network has the ability to discover its neighbors and to construct routes to reach other nodes in the collection. This is referred to as *self-configurability*. According to NIST, wireless ad hoc sensor networks must satisfy the following properties: “contain a large number of mostly stationary nodes”, not consume a lot of energy, self-organize the underlying network, “collaborate signal processing”, and have a “queuing ability” [19].

The summation of electromagnetic radiation at the location of a node is referred to as *background noise*. The *signal-to-noise* ratio (S/N) is defined as the “ratio between a signal (meaningful information) and the background noise” plus interference from other signals at the location [30]. When a message is sent to a destination, it is broken into discrete units, called *packets*. If the S/N ratio is too low, then the node cannot extract the meaningful data from the signals it is receiving and therefore fails to receive the packet. This is analogous to trying to hold a conversation when many other conversations are occurring close by. For instance, when person A wishes to speak to person B, the sound from person A will only be understood by person B if the background sound from the other conversations is low enough. The *packet-delivery-ratio* (PDR) for node  $n$  to  $m$  is defined as the number of packets node  $m$  receives from node  $n$  divided by the number of packets node  $n$  sends to node  $m$  [26].

All of our experiments regarding these wireless ad hoc networks are done via computer simulation. In order to simulate these networks, however, the network must be represented in the simulation as a model. An important mathematical construct used to

model a network for computer simulation is a *graph*. A graph is composed of a set of vertices ( $V$ ) and a set of edges ( $E$ ). A graph  $G$  is therefore represented as  $G = (V, E)$ . A *weight* is function with  $E$  as the domain, and the real numbers as the codomain.

Therefore, the weight of an edge is a real number which, in this thesis, represents the probability that a packet successfully transmits over that edge. The computational representation of a graph follows one of two forms: an *adjacency matrix*, which is a one-dimensional array containing, in each element, the weight of each edge, or an *adjacency list*, which is a two-dimensional array where each node stores a list of nodes that are connected to it, along with the weights of the connections. We say that a graph containing few edges, in comparison to the maximum number of edge it could have, is *sparse*, and a graph containing many edges is *dense*. We use graphs to represent wireless networks: a graph vertex corresponds to a network node and an edge to a network connection. A graph is said to be *connected* if there is a set of links one could follow to get from any node to any other node [29]. In this thesis we define *connectivity metric*, or simply *metric*, to be a numeric rating which attempts to describe the overall level of connectivity of a graph. This number attempts to quantify the level of connectivity of the network, or the *network health*. Therefore, the connectivity metric is a function with the graph  $G$  as the domain, and the real numbers as the codomain. We call an *isolated node* a node that has no links to other nodes; in other words, it is not connected to any other nodes. An *undirected graph* is a graph in which “relations between pairs of vertices are symmetric, so that each edge has no directional character” [15]. A *directed graph* is a graph in which each edge does have directional character. A *path* is a sequence of vertices where there

exists an edge between any two successive vertices in the sequence. A *cycle* is a path where each edge and vertex is distinct, except for the starting and ending vertex which are the same. Finally, a *tree* is a connected graph with no cycles [27].

## 1.2 Wireless Ad Hoc Network Basics

Wireless ad hoc networks are an emerging technology that potentially holds a very prominent place in technology history. The range of applications of these networks is astounding and includes military use, emergency response, surveillance, and scientific exploration of harmful environments (for instance, other planets or the deep sea), just to name a few.

There are many traits that a wireless ad hoc network must have. For one, it must have some set of rules, or *layers of protocols*, for communication without a wired medium. Next it must be self-configurable, meaning that each node discovers the existence of other nodes on its own. In other words, individual nodes become part of a collective, or set of nodes, that routes information to the appropriate destination. Each node acts as a router and guides packets along network paths to their destination. This introduces the issue of wireless routing protocols, which are explored in Chapter 2. Since data is transmitted wirelessly in an open, broadcast medium, these networks are inherently more vulnerable to security exploits and attacks by malicious nodes. Therefore the study of the

working properties of wireless ad hoc networks can indicate how they can be exploited by malicious parties.

### **1.3 Attacks and Connectivity Issues**

After understanding how wireless ad hoc networks operate, it is critical to understand what can go wrong. Since one of the primary goals of any network is to attain a high level of connectivity, we must explore how this connectivity drops. We define a *connectivity attack*, or simply an *attack*, on the network as some series of events that causes the network to decrease in its overall level of connectivity. This is a slightly inadequate definition for the following reason: there is no single, universally agreed upon metric to quantify the health of a wireless ad hoc network. So, how can we quantify the effect of an attack without some kind of measurement of how connectivity changed as consequence of the attack?

In simulation we know the exact moment an attack is occurring because we control the simulation of a malicious node. Simulation is a valuable tool because it allows us to have complete control over the physical factors (node placement, transmission times, environment), allows us to replicate and repeat experiments, and allows us to run experiments without physical equipment (with the exception of the computer). It is not hard to get someone to agree that an attack is occurring if, for instance, all sensors in some area suddenly cannot distinguish any transmissions from background noise (the

argument for this will be presented shortly). Even if our connectivity metric doesn't register this as an attack, it is common sense that an attack truly is occurring.

An analogy to everyday life makes this clearer. Most people would agree that human beings are born with the ability to determine good from evil. Although how we are endowed with this ability is currently under a great deal of disagreement, acts such as murder and theft are universally considered evil. Furthermore, human beings all have a direct guide to assist us in determining right from wrong, called our conscience. This leads us to believe that there truly exists an absolute good and an absolute evil, which were not created by human beings. Since this absolute good and evil never change by definition, governments since the beginning of human civilization have attempted to reward the overall goodness of a person by awards and recognition, and attempted to prevent them from achieving evil actions by setting up laws, rules, and prisons. Now, here is the analogy: we define evil as being a decrease in good, and we can say that evil is occurring in someone, even though we cannot completely measure what absolute goodness is. This is analogous to being able to say that we define an attack as being a decrease in connectivity, and we can say that an attack is occurring, even though we cannot completely measure what the overall level of connectivity is. We all simply have an idea of what connectivity is (like we all have an idea of what good is), and the goal of this thesis is to determine how to better measure it.

Now, one issue that may be raised is that an attack may actually increase or make no change in the overall connectivity of the network. This only happens, however, if the network is at a point such that adding more links decreases the connectivity, or overall health, of the network (since taking away those links increases the connectivity of the network). But, for all practical purposes, if the nodes are pumping out messages at the same rate and link capacities and/or number of links is suddenly increased, then it is clear that any reasonable network should not suffer from a connectivity loss. We would expect to see a leveling off of connectivity, but not any noticeable decrease. Therefore, for the remainder of this thesis, we assume that our simulated attacks should never increase our connectivity metrics (and if they do, then it is the sign of a bad metric).

## **1.4 Network-to-Metric Problem**

This thesis is primarily concerned with how well different connectivity metrics, or metrics that indicate how connected the network is, represent the overall connectivity of the network. The overall goal of this thesis is to describe a process by which connectivity metrics can be compared, mainly in terms of how the metrics react to attacks, as well as to perform the process on common metrics. The connectivity metric correspondent to a graph speaks for the health of the network. Much is known about connectivity metrics, and the computational algorithms used to estimate them [9]. For instance, work has recently been done allowing one to find the number of nodes needed, in a given scenario, to almost surely obtain a very connected graph [1]. In our research, however, we have

found sparse literature comparing graph theoretic metrics to each other, in relation to wireless ad hoc network simulators. It is beneficial to have comparisons of cost-benefit ratios for different metrics. Our contribution is to demonstrate the costs and benefits of applying different metrics to simulation experiments with networks that are under attack.

Since the connectivity metric is defined to have access *only* to the network graph in making its measurement, it is important that the graph itself contains all of the necessary information the metric needs from the network. We note here that a simulated network's adjacency matrix based solely on reception range only conveys potential links. When a message is transmitted between two nodes that are in reception range of one another, there is still the possibility that the message is not received by the intended receiving node. Two of the many factors involved with determining if a message is received are the level of interference at the receiver's location and if collisions with other messages occurred. We propose to look at end-to-end communication, in other words to only consider if the message has arrived or not, to build a graph that better reflects the effective network. In light of this, we have broken down the task of how a metric conveys the information in a network into two smaller problems – the *network-to-graph* problem, and the *graph-to-metric* problem. Below is the basic control path of the two sub problems (each represented by a  $\rightarrow$ ).

NETWORK  $\rightarrow$  GRAPH  $\rightarrow$  METRIC

**Figure 1: Network-to-metric problem decomposition**

The network-to-graph problem deals with how the graph should be structured and obtained from the simulator to contain all of the information that the connectivity metric needs. There are many factors that contribute to the probability that a message is successfully transmitted from one node to another. Some of these factors include the distance to the receiver and the level of interference at the receiver's location. Our solution to this problem considers only the *effective network*, which is concerned only with whether or not the messages were delivered correctly.

The graph-to-metric problem deals with how the connectivity metric extracts and summarizes the information contained in the graph. The overall metric should give a good summary of the overall connectivity of the network. In other words, now that we have a graph that properly represents the connectivity of the network, the problem is how to use this graph to produce a metric that accurately quantifies this connectivity. These two problems, along with our solutions to them, are explained further into this thesis.

At first glance, it seems that the solution to the graph-to-metric problem is the only problem that this thesis should deal with, since the primary purpose of this thesis is to compare and contrast connectivity metrics. It was critical, however, that we obtained a solution (or set of solutions) to the network-to-graph problem as well, as a solution to this problem provides a foundation for the graph-to-metric problem. Therefore, the time invested in this honors thesis was dedicated to solving and creating the frameworks need to implement the solutions to *both* of these problems. By taking into consideration the

network-to-graph problem, our comparisons for the connectivity metrics are much more accurate and fair. After a wide literary search, we are convinced that both our comparison of major graph-theoretic metrics in wireless ad-hoc networks simulation (the graph-to-metric problem), as well as the way we store the network connectivity information in our graphs to produce a more accurate depiction of the network (the network-to-graph problem) are novel ideas.

## **Chapter 2**

# **Network Operation and Attack Details**

In the previous chapter we have introduced the concept of a wireless ad hoc network, and described superficially how it operates. However, many critical details to the operation of these networks were glanced over and need further explanation. In this chapter, we explore the in-depth operation of wireless ad hoc networks, which is critical information for the understanding of this thesis. It is extremely difficult to recognize how these networks fail without grasping exactly how they operate. Later in this chapter we explore the innate workings and vulnerabilities of the network that allow attacks to occur, as well as the basic execution of these attacks.

## 2.1 Wireless Ad hoc Network Operation

To gain greater insight into how wireless ad hoc networks operate, we consider the three words (counting “ad hoc” as a single word) that compose it. The first word is “wireless”, meaning that every node communicates with the network wirelessly, via the electromagnetic spectrum. Federal Communications Commission (FCC) regulates this spectrum and assigns bands of frequencies to different applications/services. Some bands around 900 MHz, 2.4GHz, and 5 GHz are left unregulated. Wireless networking technologies work in these unregulated bands (currently, most work within the 2.4 GHz band), and therefore their signals compete with many other services, such as microwave ovens and cordless phones. The effects of this can range from minor performance degradation to complete network failure.

Skipping to the third word, we now explore the “network”. Wireless ad hoc networks follow the 802.11 standard for wireless local area networks defined by the Institute of Electrical and Electronics Engineers (IEEE). To attempt to minimize the effect of the unregulated band issue mentioned above, the standard defines a medium access control (MAC) layer that interfaces with three different physical (PHY) layers. The first layer is the *Frequency Hopping Spread Spectrum* (FHSS) in which the network may switch frequencies within the 2.4 GHz band in hopes of mitigating the effects of the interference. The other two are called the *Direct Frequency Spread Spectrum* (DFSS) and *InfraRed* [2]. The basic access mechanism for wireless local area networks is a protocol called

*Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA)*. Collision avoidance is critical for wireless networks because more than one node may access the medium for transmission at any given time, and hence the protocol states that nodes are not allowed to send messages until all their neighbors have stopped sending messages. Harmful collisions occur at the point of reception, and are due the wave fronts interfering with one another, making both messages either very difficult or impossible to read. Collisions are one of the reasons messages do not get delivered. The IEEE 802.11 standard defines a way of handling collisions, called the *Exponential Backoff Algorithm*. When a collision occurs, each node transmitting will backoff (not transmit) for a random period of time. It then tries to transmit again and, if another collision occurs, will backoff for another, greater, period of time. This process is continued for a set number of times. Furthermore, the backoff periods double with each collision, therefore increasing exponentially [2]. This algorithm lessens the degree of *contention*, or fight for the medium, and therefore decreases the number of collisions.

The second word in “wireless ad hoc network” is “ad hoc”, which is Latin for “to this”. When used as an adjective, it is defined as “improvised and often impromptu” [5]. Therefore, one would correctly assume that the network is “improvised”. In other words, there is no infrastructure, such as access points; all nodes in a wireless ad hoc network are used as routers for other nodes. This creates on-the-fly structure without any central controlling or governing point. Therefore, many routing protocols were developed to help aid the ability of these networks to work efficiently and improvise when need be. Five of

these network protocols were recently studied and compared in simulation and real life experiments [13]. The experiments indicate that the Ad hoc On-Demand Vector (AODV) protocol outperforms the other four. AODV works by creating links when need be, and by holding those links active to deal with link failure [13]. This type of protocol is considered a *reactive* routing protocol because it does not build routes before they are needed; instead it constructs them on demand. The other type of protocol is *proactive*, where routes are actively maintained. A newer, but popular, routing protocol is *Dynamic Source Routing (DSR)*, which is also reactive [20]. All experiments performed for this thesis use AODV.

## 2.2 Attack Details

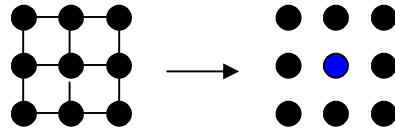
It can be argued that connectivity is one of the most important properties of a wireless ad hoc network [3] and that it may be used to quantify the “health” of the network. As mentioned above, what we have defined as *connectivity attack* on the network amounts to some sequence of events that causes the network to experience a decrease in its overall level of connectivity. Malicious connectivity attacks that make one ore more nodes in the network inaccessible are characterized as *malicious denial-of-service* attacks, or merely *denial-of-service* (DoS) attacks, as they attempt to prevent the node from access to the network. Carnegie Mellon’s Computer Emergency Response Team (CERT) defines a denial-of-service attack as an explicit attempt to prevent the client of a service from using the service [4]. It is possible that some inadvertent action denies the clients access to a

service, appearing like a malicious attack, but is rather the result of carelessness or natural effects. These attacks are referred to as *non-malicious denial-of-service* attacks.

In this thesis, we work exclusively with two different attack models, both of which are considered DoS attacks. These attacks are used to study how well the connectivity metrics collected represent the overall health of the network.

The first type of attack this thesis is concerned with is *jamming* attacks. These attacks occur when the frequency the network is transmitting on contains a high level of background noise, effectively decreasing the S/N ratio in the space surrounding the node. An attack may create this situation by broadcasting a strong signal with the hope of creating overwhelming interference on receivers, which perceives increased noise and is possibly not able to capture the actual signal. We define a *jamming node* to be a node that performs this kind of attack, whether it is part of the network or not. Hence, a jamming node attempts to create an environment in which the S/N ratio is so low that all nodes within a specific radius are not be able to communicate with any other nodes. These types of attacks can be either malicious or non-malicious, and can be effective in crippling network connectivity. It has been shown that an entire randomly distributed section or sections of a wireless ad hoc network with  $N$  nodes can be entirely disrupted by the well placement of  $k$  jamming nodes, where  $k$  is much less than  $N$  [31]. In other words, the operation of a network can be severely affected by a relatively small number of jamming

nodes. Figure 2.1 illustrates the potential links between nodes failing as the result of a single jamming node.



The middle node is replaced by a jamming node, rendering the network useless.

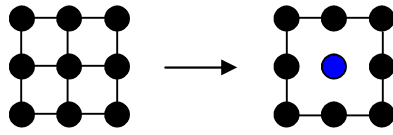
**Figure 2.1: Jamming Attack**

To make matters worse, these attacks are not necessarily difficult to implement. For instance, there are cordless phones that operate on the 2.4 GHz band, which is one band wireless ad hoc networks operate on. Consider the following hypothetical scenario: a major fire breaks out in a company's office building, where a wireless sensor network has previously been deployed. This network is used by the firemen to better understand the spread of the fire and to assess the damage done to different places of the building. Suddenly, a wireless phone using the 2.4 GHz band rings in the office, jamming an entire section of the building's nodes which effectively cuts off all communication to those nodes. This is an example of a non-malicious jamming attack, and shows how serious it can be.

Maliciously jamming attacks are, most likely, even worse, as they incorporate human intelligence. There are, however, difficulties that must be overcome by attackers. First, conserving battery life is important to the operation since the attacker wants to maximize

the effects of his disruption. If it is possible to use power sparingly in the attack and still cripple the network, then the length of the attacks may increase, since the attacker's battery life is depleted at a slower rate. Second, one of the physical layers defined by the IEEE 802.11 protocol uses frequency hopping (FHSS), meaning that the wireless network uses multiple frequency ranges (known as *channels*) to transmit data. Jamming nodes that wish to completely render the area useless would have to jam a large number of these channels, which is not likely to happen. It is easy to see that both malicious and non-malicious jamming attacks can influence the PDR of the nodes affected.

The second type of attack we consider is defined as a *range* attack and focuses on adjusting the ranges of different nodes to break and establish connections throughout network session. Different protocols specify different methods of connecting, disconnecting, and reconnecting to a network, but it is fairly universal that when these operations are performed, *control packets* (packets that carry no application data but instead are for establishing connections and routes) are transmitted. This causes the nodes that want to send application packet data to compete for the network's bandwidth, which in turn does not allow the network to communicate as quickly as if there were no control packets. Furthermore, the nodes that are not recipients of the control packets perceive them as interference, and therefore their S/N ratio decreases. Aside from the control packet issue, connectivity may be lost when a node's range decreases. Figure 2.2 illustrates what can happen to the links of a section of the network in the event of a range attack.



The middle node's range decreases,  
eliminating many links.

**Figure 2.2: Range Attack**

# Chapter 3

## Network-to-Graph Problem

An ideal solution to the network-to-metric problem would entail a direct translation of the connectivity information inherent in the wireless ad hoc network into a concise, accurate metric. Since we have found little literature on this topic, we have decided formulate our own approach to solve this problem in the context of simulation. The approach we follow to solve the network-to-metric problem applies the *divide and conquer* strategy to identify two sub-problems: the *network-to-graph* problem and the *graph-to-metric* problem, which we describe. The information first starts in topology of the network and then gets summarized into a weighted graph. Next the information from the weighted graph is used to compute the connectivity metric. The summarization process should be as accurate as possible; in other words, the essence of the connectivity information should

not be altered over the total transfer. This chapter is primarily concerned with the first part of the problem, known as the *network-to-graph* problem.

The network-to-graph problem consists of taking the network (here we are talking about all available data produced by simulation at any given time) and summarizing it into a *connectivity graph* that is capable of storing all the connectivity information the future metric needs. We have designed what we believe to be an accurate, manageable, and novel way to store this connectivity information in a graph data structure.

### **3.1 Simulation and Connectivity Graphs**

The Simulator for Wireless Ad Hoc Networks (SWAN) is the simulator we have been working with for the past two years. Its primary goal is to provide a reliable and scalable way of simulating wireless ad hoc networks.

SWAN, as well as other simulators, use an undirected, unweighted graph to represent the connectivity of the network [33][18]. Using only this type of graph to convey connectivity information is not an accurate method because the following questionable assumptions are implicit:

- 1) *Transmission and reception symmetry*. This brings to light the undirected property of the graph. The assumption that “if I can hear you, then you can hear me” is made.

2) *I can either hear you perfectly, or not at all.* This brings to light the unweighted property of the graph. The assumption that if there is a connection, then it is flawless (no lost packets) is made.

As concluded by Kotz et al, these assumptions are not appropriate to make and result in a connectivity graph that does not accurately convey the true connectivity status of the network [8].

## **3.2 Redefining the network graph**

As mentioned before, any adequate solution to the network-to-graph problem would have to address the issue of accurately and concisely summarizing the connectivity information from the network to the graph representation. In an attempt to address the assumptions made with previous connectivity graph constructions, we have decided to start from scratch and completely rethink how the graph should be constructed. The major flaws with the previous graph were the lack of direction and lack of weight; hence we have placed them at the highest priority in our new graph construction. Furthermore, understanding that the graph should reflect the dynamic nature of the simulated wireless ad hoc networks (especially mobile ones), we want our weights to represent the quantity of each link as seen in the simulator in a given time period. It is important to note that an undirected, unweighted graph may be exactly what one needs in some situations. For the purposes of quantifying network health, however, it is not as adequate as it should be.

With these stipulations in place, we take the original graph representation provided by SWAN and use it to construct another graph that better reflects the connectivity of the network as observed during a well-defined period of simulation time. In this construction, the weight on the edge (A, B) represents the *packet-delivery ratio* from node A to B in the given period of simulation time, with a penalty for a relatively large number of *control* or *broadcast* packets. Since the *packet-delivery ratio* could be different over the link from node A to B then over the link from node B to A, the graph is directed. Note that although not all broadcast packets are control packets, we assume for the purpose of this thesis that if a broadcast packet is sent, it is a control packet. Hence, the weight on edge A to B is defined by:

$$Weight\ A\ to\ B = \frac{UPR}{UPS + BPS} = \frac{Data\ received}{Data\ sent + Control\ sent} \quad (1)$$

In the formula above, UPR is the number of unicast packets received by B, UPS is the number of unicast packets sent by A, and BPS is the number of broadcast packets sent by A, all of which go over the link from A to B. Note that if the number of broadcast packets sent by A is much less than the number of unicast packets sent by A to B, the formula could be simplified to:

$$Weight\ A\ to\ B = \frac{UPR}{UPS} = Packet\ Delivery\ Ratio\ (PDR) \quad (2)$$

As defined in (1), the weight from A to B represents the packet delivery ratio from A to B, with a penalty for having too many broadcast packets. This allows the metric to better reflect the network health, by lowering the perceived delivery ratio in the case that there are many broadcasts.

One issue occurs when  $UPS + BPS = 0$ , meaning the denominator is zero and hence the weight is undefined. In this case, we define the weight from A to B as 0 (completely disconnected).

Another issue with using this new graph is backward compatibility with previous metrics. These metrics expect an undirected, unweighted graph. Therefore, we allow the graph to be altered in the following way, only if absolutely required by the metric:

- 1) If the metric requires an unweighted graph, then we define the weight to be 1 if  $weight > 0$  and 0 otherwise (in an unweighted graph, a 1 is a link and a 0 is no link).
- 2) If the metric requires an undirected graph, then we define the weight over link AB to be the average of the weight from A to B and from B to A.

After running many simulations with the new graph construction method, we are confident that this is an adequate solution to the network-to-graph problem. The next section illustrates visually how the new connectivity graph accurately reflects what we consider to be the network connectivity.

### **3.3 Connectivity Graph Implementation in SWAN**

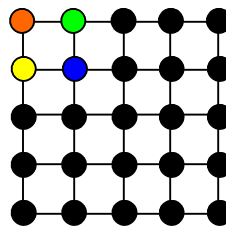
In order to implement the new connectivity graph construction in SWAN, we recorded all the unicast and broadcast messages sent, and all the unicast messages received. To do this in SWAN, we monitored the lowest possible level of the network protocol stack, while still being able to see the packets as a whole. Therefore, both the physical and data link layers were modified and closely monitored. Without going into detailed specifics, when a message travels down the protocol stack, the data link layer passes the message to the physical layer for processing and vice versa. It is immediately after these transmissions where the packets are recorded by incrementing either the “unicast packets received over this link” variable, the “unicast packets sent over this link” variable, or the “broadcast packets sent over this link” variable. After the timer has expired (specified by the user), the weight is calculated for each link using these variables, and written to a file specified by the user, along with the current time. The variables are then reset to 0 and the process is started over again.

One important consideration is how often to tally and reset the variables. In other words, what should be the length of the time interval used to recalculate the weights of the graph? If the length of the time interval is too short, then there might not be enough packets sent to obtain a fair reading of what the connectivity over that link should be. If the length of the time interval is too long, then the attacks are averaged in with the normal operation and the graph does not reflect the seriousness of the attack. Simulation

experiments have shown us that 25 second time intervals work well, for our attack types, relative to the length of the attack on-off cycle, and for the remainder of this thesis we assume that 25 second time intervals are being used.

To illustrate the advantages of this new graph construction, consider the following scenario.

- 1) A 9 node wireless ad hoc network is deployed in a 5x5 grid pattern where nodes are equally spaced and separated by 100 meters. Each node has a range of 130 meters.



Note: Red, Green, Yellow, Blue are Nodes 1,2,3,4 respectively

**Figure 3: Grid Scenario**

- 2) The single attack node is the one in blue. It changes its range from 130 down to 10 meters when the attack changes from off to on, and restores its range to 130 meters when the attack changes from on to off. The attack is off for 50 seconds and on for 50 seconds (of simulated time). The attack sequence starts at 100 seconds. All other nodes are normal.
- 3) We simulate 300 seconds of network operation.

- 4) Table 3.1 is a list of model parameters used in this simulation (this is reported for readers interested in repeating our experiments and is not explained in this thesis).

Propagation Model	2-ray
Carrier Frequency	2.4 GHz
System Loss	1
Antenna Height	1.5m
Temperature	290 K
Noise Factor	10 dB
Ambient Noise Factor	0 dB
Routing Protocol	AODV

**Table 3.1: Model Parameters**

The output of the simulation in regards to the connectivity graph is shown in table 3.2 (in adjacency matrix form). Only nodes 1, 2, 3, and 4 (Red, Green, Yellow, and Blue) are shown for space consideration, and all values are rounded to two decimal places. The columns run from nodes 1 to 4, left to right, and the rows run from nodes 1 to 4, top to bottom. The values are the weights over the links.

<b>Time = 0-25 seconds</b> 0.00 0.87 0.97 0.00 0.83 0.00 0.00 0.99 1.00 0.00 0.00 0.83 0.00 1.00 0.77 0.00	<b>Time = 25-50 seconds</b> 0.00 0.76 1.00 0.00 0.95 0.00 0.00 0.99 1.00 0.00 0.00 0.86 0.00 1.00 0.96 0.00	<b>Time = 50-75 seconds</b> 0.00 0.79 1.00 0.00 0.89 0.00 0.00 0.98 1.00 0.00 0.00 0.90 0.00 1.00 0.91 0.00
<b>Time = 75-100 seconds</b> 0.00 0.89 0.99 0.00 0.99 0.00 0.00 0.40 1.00 0.00 0.00 0.99 0.00 0.50 0.96 0.00	<b>Time = 100-125 seconds</b> 0.00 0.87 1.00 0.00 0.88 0.00 0.00 0.00 1.00 0.00 0.00 0.48 0.00 0.00 0.58 0.00	<b>Time = 125-150 seconds</b> 0.00 0.83 1.00 0.00 0.83 0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
<b>Time = 150-175 seconds</b> 0.00 0.84 0.86 0.00 0.94 0.00 0.00 1.00 1.00 0.00 0.00 0.92 0.00 1.00 0.83 0.00	<b>Time = 175-200 seconds</b> 0.00 0.92 1.00 0.00 0.87 0.00 0.00 1.00 1.00 0.00 0.00 0.89 0.00 1.00 0.86 0.00	<b>Time = 200-225 seconds</b> 0.00 0.90 1.00 0.00 0.91 0.00 0.00 0.43 0.99 0.00 0.00 0.28 0.00 1.00 0.31 0.00
<b>Time = 225-250 seconds</b> 0.00 0.92 1.00 0.00 0.95 0.00 0.00 0.00 0.96 0.00 0.00 0.00 0.00 0.00 0.00 0.00	<b>Time = 250-275 seconds</b> 0.00 0.96 1.00 0.00 0.96 0.00 0.00 0.87 1.00 0.00 0.00 0.96 0.00 1.00 0.92 0.00	<b>Time = 275-300 seconds</b> 0.00 0.87 1.00 0.00 0.93 0.00 0.00 0.98 1.00 0.00 0.00 0.96 0.00 0.98 0.92 0.00

**Table 3.2: PDR-edged graph**

During the first 100 seconds of the simulation, a few broadcast messages are exchanged in the network due to control packets sent by the Address Resolution Protocol (ARP). We consider a *transient period* in which simulation models are going through an initialization stage; data collected during this time does not reflect the typical operation of the network [22]. It is prudent to avoid collecting data until after the transient period has ended. The attack then occurs in the time interval from 100 to 150 seconds, and the adjacency matrix clearly shows a drop off for nodes near the attack node (node 4). The range is restored at 150 seconds, which is properly reflected in the adjacency matrix corresponding to 150-175 seconds. The attack occurs a second time during the time interval of 200 to 250

seconds, and again is properly reflected. Finally, a second recovery period starts at 250 seconds.

Many similar simulations have been run, all providing strong evidence that the connectivity is truly being captured in this new graph construction technique. Now that the foundation has been strongly set, we are finally in a position to start to compare connectivity metrics.

# Chapter 4

## Graph-to-Metric Problem

The graph-to-metric problem is the second and final step in the process of accurately and concisely conveying the connectivity information in a network. This step consists of using the connectivity information from a connectivity graph to determine a single real number, called a *metric*. The process maps a directed, weighted graph to single real number. One important aspect to note is that these metrics *cannot* be compared to each other without knowing beforehand when the attacks are occurring. Since we are performing simulations, we say an attack occurs when a range is decreased or jamming messages are flooding a part of the network.

A critical observation is that it is very hard to compare metrics directly. One major problem with comparing metrics directly is that the metrics may not all be in compatible

scales. Furthermore, there is not a commonly agreed upon metric that could be used as a basis for comparison. Therefore, we must analyze each one separately and independently. One possible way to do this is to compare them to our *expectations* of what the connectivity should be in a given scenario. In other words, since we have the power of simulation and therefore know the exact time and severity of each attack, we can compare the metric output to our expectations of what the connectivity should be. When this is complete, we then can indirectly compare the metrics to each other in terms of how well they met our expectations.

## 4.1 Connectivity Metrics

The heart of this thesis is to outline a *process* by which to compare connectivity metrics, and to give basic comparisons of some common metrics. The process that we believe best fits our needs is outlined in section 5.3. To the best of our knowledge, this process has never been used to compare connectivity metric for wireless ad hoc networks, and therefore is an original idea. Discarding this process for the moment, our starting point was to brainstorm and catalogue several common metrics of connectivity. First, let us outline common metrics that are used to describe connectivity, but would not be adequate for simulation studies of wireless ad hoc networks.

- *Vertex/edge connectivity* – The vertex connectivity of a graph is defined to be the minimum number of nodes, together with incident edges, that must be removed

from the graph before it becomes disconnected [27]. Edge connectivity is analogous.

- *Cost of minimum spanning tree* – A *spanning tree* is a tree that contains all vertices in the graph. A *minimum spanning tree* is a spanning tree where the sum of the weights is smaller than or equal to the sum of the weights of every possible spanning tree [27].
- *Graph diameter (longest shortest-path length)* – The graph diameter is the summation of the weights along the longest shortest-path belong to the graph [27].
- *Average shortest-path length* – The average shortest-path length is the average of all shortest-paths in a graph [27].

The primary reason that these metrics are not adequate for use in simulation studies of wireless ad hoc network scenario, especially in attack scenarios, is that the algorithms for computing these metrics all either require the graph to be connected, or else only apply to each connected component of the disconnected graph. In other words, every node must be able to reach every other node, whether directly or through a set of intermediate nodes. It is a perfectly feasible scenario for a node or set of nodes to become disconnected from the main network, and we would still like to provide a connectivity rating to a network of this kind. One example is during a jamming attack; all nodes in the jammed region most likely become isolated nodes. For this reason we must look elsewhere, for connectivity metrics that give meaningful results even when the graph becomes disconnected.

After a thorough literary search, we have discovered the following connectivity metrics that we believe would be useful for quantifying the network health. They are divided into two groups – metrics that are able to work on weighted graphs (we call these *weighted metrics*) and metrics that are not able to work on weighted graphs (we call these *unweighted metrics*). Since we concluded in the last section that the best way to convey connectivity information was by a weighted graph, we concentrate primarily on the weighted metrics as they have a greater potential to provide a better characterization of connectivity.

### 4.1.1 Weighted Metrics

The following metrics are capable of giving meaningful responses with weighted graphs, whether they are connected or disconnected:

- *Weighted density* – Weighted density is a measure of how much relative “linkage” the graph has. It is equal to the sum of the weights divided by the total possible sum of the weights (which is  $n * (n-1)$  for a directed graph, where  $n$  is the number of vertices) [25]. Let  $W_{ij}$  represent the weight of the link from node  $i$  to  $j$ . Then,

$$\text{Weighted Density} = \frac{\sum_{i,j \in V | i \neq j} W_{ij}}{n(n-1)}$$

- *Weighted clustering coefficient* – The weighted clustering coefficient of a node  $v$  (which we label  $C_w(v)$ ) is used in a weighted graph to measure the connectivity of

the nodes adjacent to  $v$ , also known as the *neighborhood* of  $v$  (which we label  $N(v)$ ). The number of neighbors of  $v$  is defined by  $K_v$ . We define it as follows:

$$C_w(v) = \sum_{i,j \in N(v), i \neq j} \frac{W_{ij}}{K_v(K_v - 1)}$$

This value gives an indication of how connected the nodes close to  $v$  are. The metric we report is the *average* of the weighted clustering coefficient over all of the nodes [14].

- *Weighted average 2-hop node density* – A node  $i$  is considered in the *2-hop neighborhood* of a link  $(a, b)$  (that is, the link connecting nodes  $a$  and  $b$ ) if the unweighted shortest paths (that is, 1 is weight  $> 0$ , 0 otherwise) from node  $i$  to node  $a$  and from node  $i$  to node  $b$  are both at most 2. The value obtained for each link is the number of nodes in the 2-hop neighborhood of that link. To transform this metric into a weighted metric, we multiply the value obtained by a link, by the weight of the link. The average over all of the links constitutes the metric [11].
- *Weighted average 2-hop link density* – A link  $(i, j)$  is considered in the *2-hop neighborhood* of a link  $(a, b)$  if either nodes  $i$  or  $j$  are in the 2-hop neighborhood of link  $(a, b)$ . The rest is analogous to the weighted average 2-hop node density metric [11].
- *Weighted Chemical Index* – The *connectivity index of an organic molecule with molecular graph  $G$*  is defined as follows:

$$R_\alpha = \sum_{(u,v) \in E} (d(u)d(v))^\alpha$$

where the summation goes over all possible pairs of adjacent nodes  $u$  and  $v$  in  $G$ . The function  $d(u)$  is the weighted degree (also known as *strength*, the sum of the outgoing weights) of node  $u$ . The *Randic connectivity index* is the connectivity index where  $\alpha = -0.5$ . This is also called the *branching index* [16]. We are using  $\alpha = 0.5$ , however, as this allows for the metric to increase as the connectivity increases. Although this metric is used in chemistry to characterize molecular connectivity, we believe that trying to apply metrics from the other sciences in the context of our problem can yield interesting, worthwhile results.

### 4.1.1 Unweighted Metrics

The following metrics are capable of giving meaningful responses only with unweighted graphs, but the graphs can be connected or disconnected:

- *Average number of neighbors* – This metric is simply the summation of the number of adjacent nodes each node has.
- *Clustering coefficient* – This metric is similar to its weighted counterpart found above. The clustering coefficient of a node  $v$  (which we label  $C(v)$ ) is used in a graph to measure the connectivity of the nodes adjacent to  $v$ . The number of neighbors of  $v$  is defined by  $K_v$ , and the number of edges between those neighbors is defined as  $E(v)$ . We define clustering coefficient as follows:

$$C(v) = \frac{E(v)}{K_v(K_v - 1)}$$

The metric we report is the *average* clustering coefficient over all of the nodes [14].

- *Connectivity index* - Let  $G' = (V, E')$  be the *transitive closure* of the graph  $G$  (if a node  $i$  is capable of reaching a node  $j$  in  $G$ , then in  $G'$  there is a direct link from  $i$  to  $j$ ). Then the *connectivity index* ( $CI$ ) is defined as:

$$CI = \frac{|E'|}{|V|^2}$$

Note that if the graph is connected, then the connective index is always 1.

Therefore this metric is only useful if the attack is powerful enough to disconnect the network [21].

## 4.2 Metric Implementation

After SWAN was programmed to produce a PDR-edged graph in accordance with our description, the next stage was to implement the catalogued connectivity metrics into a structured, modular program. This program, written in C++, is referred to as the *metric calculator program*. The program takes three parameters at runtime, the *input file name*, *output file name*, and *metric*. The input file (produced by SWAN in regards to the network-to-graph problem) contains snapshot of the PDR-edge graph in adjacency matrix form for each time interval (which is specified by the user of SWAN). The following is an input file (rounded to three decimal places), used to show the format.

100
0.000 0.875 0.000 0.893 0.000 0.000 0.000 0.000 0.000
0.989 0.000 1.000 0.000 0.979 0.000 0.000 0.000 0.000
0.000 0.816 0.000 0.000 0.000 0.968 0.000 0.000 0.000
0.935 0.000 0.000 0.000 0.947 0.000 0.982 0.000 0.000
0.000 0.968 0.000 0.999 0.000 0.997 0.000 0.997 0.000
0.000 0.000 0.999 0.000 0.999 0.000 0.000 0.000 0.976
0.000 0.000 0.000 0.996 0.000 0.000 0.000 0.982 0.000
0.000 0.000 0.000 0.000 0.908 0.000 0.992 0.000 0.963
0.000 0.000 0.000 0.000 0.000 1.000 0.000 0.992 0.000
200
0.000 0.925 0.000 0.934 0.000 0.000 0.000 0.000 0.000
0.958 0.000 0.999 0.000 0.968 0.000 0.000 0.000 0.000
0.000 0.841 0.000 0.000 0.000 0.974 0.000 0.000 0.000
0.916 0.000 0.000 0.000 0.948 0.000 0.974 0.000 0.000
0.000 0.990 0.000 1.000 0.000 0.981 0.000 0.999 0.000
0.000 0.000 0.993 0.000 0.997 0.000 0.000 0.000 0.998
0.000 0.000 0.000 0.995 0.000 0.000 0.000 0.994 0.000
0.000 0.000 0.000 0.000 0.952 0.000 1.000 0.000 0.991
0.000 0.000 0.000 0.000 0.000 0.996 0.000 0.986 0.000
300
0.000 0.891 0.000 0.957 0.000 0.000 0.000 0.000 0.000
0.984 0.000 0.998 0.000 0.946 0.000 0.000 0.000 0.000
0.000 0.879 0.000 0.000 0.000 0.921 0.000 0.000 0.000
0.991 0.000 0.000 0.000 0.881 0.000 0.957 0.000 0.000
0.000 0.999 0.000 0.974 0.000 0.982 0.000 0.994 0.000
0.000 0.000 0.997 0.000 0.970 0.000 0.000 0.000 0.994
0.000 0.000 0.000 0.935 0.000 0.000 0.000 0.999 0.000
0.000 0.000 0.000 0.000 0.969 0.000 0.951 0.000 0.977
0.000 0.000 0.000 0.000 0.000 0.992 0.000 0.964 0.000
...

**Table 4.1: PDR-graph input file**

This example shows the adjacency matrix for a graph of 9 nodes, where the time intervals are equal to 100 seconds. The values represent the weight of the edge between the two nodes (indicated by the row and column). For example, the first adjacency matrix is the

PDR-edge graph for the time period between 0 and 100 seconds. The second is the graph for the time period between 100 and 200 seconds, and so on.

The *metric* parameter should be a string containing one of the following values, each representing the metric associated with it:

<b>Value</b>	<b>Metric</b>
<i>wd</i>	Weighted Density
<i>wcc</i>	Weighted Clustering Coefficient
<i>w2n</i>	Weighted 2-hop Node Density
<i>w2l</i>	Weighted 2-hop Link Density
<i>wci</i>	Weighted Chemical Index
<i>ann</i>	Average Number of Neighbors
<i>cc</i>	Clustering Coefficient
<i>ci</i>	Connectivity Index

**Table 4.2: Metric Values**

The *output file name* is the name of the file that the program produces. It overwrites any file in the program's directory that already contains that name. The output file contains one row per time interval, with two values on each row. The first value is the time (as recorded from the input file). The second value is the metric, produced from the PDR-edge graph associated with that time interval. This format makes it easy to plot visually with either Microsoft Excel or GNUplot.

## 4.3 Hypothesis Testing

As previously mentioned, developing or applying a process by which to compare connectivity metrics is a central goal of this thesis. Being as such, it is of the utmost importance to have this comparison as mathematically sound as possible. Since, as previously noted, it is not beneficial to directly compare metrics, we have decided to give each metric a rating, which indicates *the strength of the statistical evidence against the possibility that the metric does not properly register an attack*. In other words, the value of the rating indicates whether or not there is enough statistical evidence to claim that the metric detected an attack. After the ratings are assigned, then the connectivity metrics are then indirectly compared based on their ratings. Before this rating method is described, the popular statistical concept of *hypothesis testing* must be introduced.

### 4.3.1 Introduction to Hypothesis Testing

Hypothesis testing is the statistical process of dividing possible conclusions of an experiment and then using probability theory to select one conclusion over the other [10]. The two competing conclusions are called the *null hypothesis*, denoted  $H_0$ , and the *alternate hypothesis*, denoted  $H_1$ . It may be easier to think about hypothesis testing as analogous to a court room trial. The defendant is either innocent (the null hypothesis) or guilty (the alternate hypothesis). Furthermore, the defendant is assumed innocent unless there is substantial evidence to prove him guilty. In the same fashion, the null hypothesis

is assumed to be true, unless there is substantial statistical evidence indicating that it is false.

The initial assumption of a trial is that the defendant is innocent. In hypothesis testing, the probability of obtaining a value that is as extreme or more extreme than the value that is “on trial”, *under the assumption that the null hypothesis is correct*, is called the *observed level of significance*, or *p-value* [10][32]. This is the rating that was discussed in the previous section. It is important to stress that the null hypothesis be assumed true for this process. The conclusion then is that if the p-value is very small, the null hypothesis is rejected in favor of the alternate hypothesis. On the other hand, if the p-value is not small enough, then there is not enough statistical evidence to reject the null hypothesis, and one must stick with the original assumption that the null hypothesis is true.

While the goal of a metric is to quantify network health, the primary goal of hypothesis testing in this thesis is to determine whether or not a metric successfully detects an attack, and to find the p-value associated with the test. Therefore, given a specific metric, a series of simulation runs were performed with the same parameters. Eventually, we collected a large number of independent random samples of the metric value when the attack is off, and a large number of independent random samples of the metric value when the attack is on. Since there are a large number of these samples, the *central limit theorem* states that the distribution of their average is approximately normal. For justification of this, the reader is referred to any elementary probability textbook such as

A First Course in Probability by Sheldon Ross [24]. The data collection specifics for obtaining these independent random samples are thoroughly outlined in Chapter 5.

A restatement of our goal in terms of hypothesis testing would be “is the average of the metric values during no attack equal to or less than the average of the metric values during an attack?” Note that we do not consider the case that the average of the metric values during an attack would be larger than the average of the metric values with no attack, because if this was the case then the metric would be extremely poor and we would still want to classify it as “did not detect attack”. Let  $\mu_1$  be the *true* average and  $\bar{x}_1$  be the *sample* average of the metric values with no attack. Similarly, let  $\mu_2$  be the *true* average and  $\bar{x}_2$  be the sample average of the metric values during an attack. Then,

$$H_0: \mu_1 - \mu_2 = 0 \text{ and } H_1: \mu_1 - \mu_2 > 0$$

The reader is now asked to take the following on faith. A justification can be found in most elementary statistics textbooks. The value

$$Z = \frac{\bar{x}_1 - \bar{x}_2 - (\mu_1 - \mu_2)}{\sqrt{\frac{\mathcal{G}_1^2}{n_1} + \frac{\mathcal{G}_2^2}{n_2}}}$$

where  $\mathcal{G}_1$  and  $\mathcal{G}_2$  are the true standard deviations of the average of the metric values with no attacks, and the and the average of the metric values during an attack, has a distribution that is approximately standard normal. Since  $n_1$  and  $n_2$  are sufficiently large, and we assume the null hypothesis is true ( $\mu_1 - \mu_2 = 0$ ), we conclude that

$$Z = \frac{\bar{x}_1 - \bar{x}_2 - 0}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}$$

where  $s_1$  and  $s_2$  are the sample standard deviations replacing  $\mathcal{G}_1$  and  $\mathcal{G}_2$ , is also standard normal [32]. The formula for a variance  $s^2$  is

$$s^2 = \frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n - 1}$$

where the  $x_i$ 's run through all members of the data set. Note that values can now be plugged into this formula, and a real number value is outputted, call it  $z$ . This is called the *test statistic*. From this point, the p-value is equal to the probability that a standard normal variable is greater than or equal to (more extreme or as extreme) the test statistic  $z$  [32]. This value is determined from the table in Appendix B.

The last question to answer is how low the p-value must be before we reject the null hypothesis. Like a court room, there is no exact answer to this question. The four most commonly used p-value cutoffs are 0.1, 0.05, 0.01, and 0.001. This means that if 90% of the time, 95% of the time, 99% of the time, and 99.9% of the time, respectively, the test statistic is as extreme or more extreme then a value picked at random from a standard normal distribution, the null hypothesis is rejected. For the purposes of this thesis, we use p-value cutoffs of 0.05 and 0.01, but report the p-value to the reader in all tests performed.

### 4.3.2 Hypothesis Testing Example

To illustrate more clearly the process of hypothesis testing in the context of our thesis, an example is presented. Assume data set #1 was taken when the attack was not occurring and data set #2 was taken during the attack. We perform a hypothesis test to determine if there is a significant difference between the averages of both data sets.

Data Set #1					
5.5	5.7	5.9	5.4	5.6	5.8
5.6	5.4	5.5	5.4	5.8	5.6
5.6	5.2	5.2	5.7	5.4	5.4
5.3	5.6	5.7	5.4	5.4	5.6
5.4	5.6	5.5	5.4	5.4	5.5

Data Set #2					
5.4	5.2	5.1	5.3	5.2	5.5
5.6	5.4	5.3	5.3	5.2	5.4
5.5	5.4	5.5	5.6	5.1	5.2
5.3	5.4	5.2	5.4	5.3	5.4
5.1	5.5	5.6	5.3	5.4	5.4

**Table 4.3: Example data sets**

The first step is to state the hypotheses. Using the same notation as above,

$$H_0: \mu_1 - \mu_2 = 0 \text{ and } H_1: \mu_1 - \mu_2 > 0$$

Furthermore, we calculate the following:

$\bar{x}_1 = 5.516667$	$\bar{x}_2 = 5.35$
$s_1 = 0.170361$	$s_2 = 0.145626$
$n_1 = 30$	$n_2 = 30$

**Table 4.4: Calculated Values**

Next, we compute the test statistic  $z$  to be

$$z = \frac{5.516667 - 5.35 - 0}{\sqrt{\frac{0.170361^2}{30} + \frac{0.145626^2}{30}}} = 4.07314$$

Finally, we compute the p-value to be the probability that a standard random variable is greater than or equal to  $z$ . After this is looked up in Appendix B, we obtain a p-value of approximately 0. This is overwhelming evidence that the null hypothesis is false, and we should accept the alternate hypothesis. Hence, if this were true metric data, this metric would be exceptionally good at detecting this attack given the specific network.

Now that we have layout out the basics for comparing metrics, and presented a strong solution to the network-to-graph problem, simulations and hypothesis testing is used to give us an equally strong solution, or set of solutions, to the graph-to-metric problem.

## Chapter 5

# Experimental Results for the Graph-to-Metric Problem

The main focus of this thesis is to provide a *foundation* and *method* for comparing connectivity metrics in terms of quantifying network health, and running these comparison tests against popular metrics. The foundation is provided in the PDR-edged graph, a strong solution to the network-to-graph problem, while the method is provided in statistical hypothesis testing, introduced in the previous chapter. Furthermore, a set of formal connectivity metrics were described and outlined. Now that the foundation, method, and set of metrics are all in order, simulations can be performed and analyzed, and the metrics can be ranked in accordance with our process.

The following metrics, all of which were described before, are used in the experiments:

<b>Metric</b>
Weighted Density
Weighted Clustering Coefficient
Weighted 2-hop Node Density
Weighted 2-hop Link Density
Weighted Chemical Index
Average Number of Neighbors
Clustering Coefficient
Connectivity Index

**Table 5.1: Metric Listing**

The goal of performing a hypothesis test on a metric is to statistically determine whether or not the metric can detect a difference from one state (with no attack or a relatively light attack), which we call the *base state*, to another state (with a relatively heavier attack), which we call the *attack state*. Note that it is possible for an attack to be occurring in the base state, but it would have to be a weaker attack than the one occurring in the attack state for the metric to detect the difference. The *scenario* simply refers to the physical layout of the nodes (in this case, location and transmission range). The *attack type* is the type of attack occurring, and is either *jamming* or *range*.

After the data is properly collected and the individual hypothesis tests are performed, the metrics are then ranked in accordance with how well they detected different types of attacks under different scenarios.

## 5.1 Metric Scalability

Before performing the hypothesis tests, it is useful to first look at metric comparison from another perspective – scalability. For this thesis, we only consider time scalability (as opposed to memory scalability), since that is most likely the determining factor in whether or not a simulator is efficient in using the metric. The concept of time scalability falls under the computer science study of *algorithm analysis*. The mathematical construct used to describe scalability is called *O-notation*. Formally, Sedgwick defines it as follows: *A function  $g(N)$  is said to be  $O(f(N))$  if there exist constants  $c_0$  and  $N_0$  such that  $g(N) < c_0 f(N)$  for all  $N > N_0$  [26].* Informally, if function  $g(n)$  does not grow faster than function  $f(n)$ , then we say  $g(n)$  is  $O(f(n))$ . This allows us to place upper bounds on how fast a function grows. Therefore, if we know a function  $f(n)$  is  $O(n)$ , then  $f(n)$  grows linearly with  $n$ , so as  $n$  increases linearly,  $f(n)$  also increases linearly. The lower we can make the upper bound on the metric's time to completion, the better the metric from the time perspective above. Note that the following calculations are based on the algorithms we have chosen to calculate the metrics with, and are not necessarily optimized for scalability. Furthermore, for many metrics the worst case scenario is far worse than the average case scenario, and therefore these comparisons should be more of guidelines than of rule.

- *Weighted Density* – The weighted density is calculated by summing all of the entries in the adjacency matrix. Since there are  $n^2$  entries where  $n$  is the number of

nodes, this metric is  $O(n^2)$ . The algorithm could be changed to allow each edge to be stored in an array, in which case the metric would be  $O(e)$  where  $e$  is the number of edges.

- *Weighted Clustering Coefficient* – The weighted clustering coefficient for a node must sum the weights of its neighbors, which in the worst case could be the entire graph. In this case, the metric for a single node would be  $O(n^2)$ , so the total metric would be  $O(n^3)$ . This upper bound is not tight, however, as if the network were not very dense it would scale far better.
- *Weighted Average 2-hop Node Density* – The algorithm chosen to calculate the metric for a link is to mark all nodes 2-hops away from one end of the link, mark all nodes 2-hops away from the other end of the link, and compute the intersection. For each link, it is possible, if the graph were very dense, that it would take  $O(n^2)$  time to compute the intersection. Since there are at most  $O(n^2)$  links, the metric would be  $O(n^4)$ . This is not a tight bound however, as the average case would scale much better.
- *Weighted Average 2-hop Link Density* – This metric is computed in a similar fashion to the weighted average 2-hop node density and therefore is  $O(n^4)$ .
- *Weighted Chemical Index* – For each link, this metric computes the strength of the vertices making up the link. Therefore, in a very dense graph, the metric is  $O(n^3)$ .
- *Average number of neighbors* – This is analogous to the weighted density algorithm, and hence is  $O(n^2)$ .

- *Clustering Coefficient* – This is analogous to the weighted clustering coefficient algorithm, and hence is  $O(n^3)$ .
- *Connectivity Index* – This algorithm uses a technique known as *weighted quick-union with path compression by halving*. Each link is  $O(n)$  at most, and hence the metric is  $O(n^3)$ .

As a very rough comparison, it seems that weighted density and average number of neighbors fare the best, followed by weighted clustering coefficient, weighted chemical index, clustering coefficient, and connectivity index. It is more important, however, to compare the metrics based on their actual hypothesis testing results.

## **5.2 Data Collection**

The criteria for hypothesis testing to be valid are met in the data collection phase. Most importantly, the sample must be large (greater than or equal to 30), randomly obtained, and independent. The sample must be large so, by the central limit theorem, the average of the samples is approximately normally distributed.

For each hypothesis test, 60 completely independent simulations of 1200 seconds were run (call these  $sim\_1, sim\_2, \dots, sim\_60$ ). For each simulation, the scenario, attack type, base state, and attack state were the same. Furthermore, all simulations performed

conform to the following structure:

<b>Time (seconds)</b>	<b>Network State</b>
0 – 400	Transient or Warm-up State
400 – 600	Attack State
600 – 800	Base State
800 – 1000	Attack State
1000 – 1200	Base State

**Table 5.2: Time / State Correspondence**

To obtain a large enough sample of the metric (the 25 second snapshot of the PDR-edged graph) in the attack state and a large enough sample of the metric in the base state (for the central limit theorem), while ensuring complete independence, we collected a total of 30 data points in the attack state and 30 data points in the base state, but only one data point per simulation run. Therefore, *sim\_1* to *sim\_30* each provided one data point from one of their two base states, and *sim\_31* to *sim\_60* each provided one data point from one of their two attack states. In other words, a time was chosen at random from 625, 650, 675, 700, 725, 750, 775, 800, 1025, 1050, 1075, 1100, 1125, 1150, 1175, and 1200 for each of *sim\_1* to *sim\_30* and a time was chosen at random from 400, 425, 450, 475, 500, 525, 550, 575, 600, 825, 850, 875, 900, 925, 950, 975, and 1000 for each of the *sim\_31* to *sim\_60* to provide a data point for the total sample. Therefore, we obtained an independent random sample of size 30 for the base state, and an independent random sample of size 30 for the attack state. At this point, hypothesis testing can proceed.

### 5.3 Hypothesis Test Results on Grid Scenarios

There are four previously defined variables that compose our simulation system – *scenario*, *attack type*, *base state*, and *attack state*.

- *Scenario* – There are two possible scenarios, both in a grid formation. In scenario 1, each node is connected to its adjacent neighbor, not including diagonals. In scenario 2, each node is connected to its adjacent neighbor, including diagonals. The following illustrates the location and links in scenarios 1 and 2 with no attacks. Furthermore, the nodes of interest later on are painted in green and labeled 1 to 9 below.

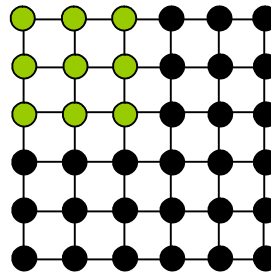


Figure 5.1: Scenario 1

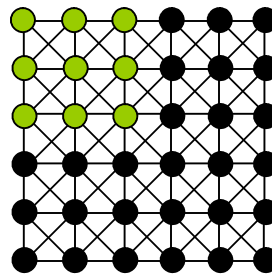
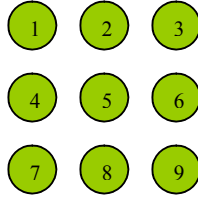


Figure 5.2: Scenario 2



**Figure 5.3: Zoomed View of Green Nodes**

Hence, scenario 1 is the less dense form of scenario 2. More specifically, each node is spaced 100 meters apart from its adjacent node ( $100\sqrt{2}$  for diagonals). The range for each node is 130 meters in scenario 1 and the range for each node is 150 meters in scenario 2).

- *Attack Type* – The attack type is either jamming or range. One or more nodes is selected (based on the base state and attack state) to be an attack node and cycles the attack on and off in accordance with the specified behavior. For a jamming attack, nodes that are selected for attack mode have a jamming range equal to the range of the scenario.
- *Base State / Attack State* – The base state and attack state are any of the following states with the criteria that the base state is a lesser degree than the attack state. All nodes are assumed to be in a non-attack mode unless explicitly mentioned.

State	Nodes in Attack Mode
Normal	None
Light	#1
Medium Light	#2
Medium	#1 and #2
Medium Heavy	#5
Heavy	#2 and #5
Very Heavy	#1 and #9
Very Very Heavy	#1, #5 and #9

**Table 5.3: State / Mode Correspondence**

These states and corresponding nodes numbers were chosen by hand and represent what we believe to be proper correspondents. There are many more possibilities for matches that can be explored in the future, but we believe this is a sound starting point.

A 4-tuple (*scenario, attack type, base state, attack state*) defines a complete simulation system for our purposes. Since there are a total of two scenarios, two attack types, and  $7+6+5+4+3+2+1 = 28$  (base state, attack state) pairs, there are a total of 112 possible complete simulation systems. To conduct a statistically solid hypothesis test for each one, it would require 60 simulations per system, which takes about 3 hours (running on a 2 GHz processor). Therefore it would require  $3 \text{ hours} * 168 = 14$  days straight to complete the experiments for all possible systems. For this reason, we have chosen four that combined a good mix of extreme with non-extreme scenarios. The four are, in (base state, attack state) format, (normal, light), (light, medium-light), (medium-heavy, heavy), and (very heavy, very very heavy). We therefore have 24 different systems, which required running 1440 simulations.

The tables below represent the data obtained from these simulations. The 4-tuple at the top identifies the system, and each metric has an associated z-value and p-value based on the z-value. The table then indicates whether the attack was detected at a cutoff level of 0.05 and a cutoff level of 0.01. The z-value is always rounded *down* to the hundreds position, in favor of the null hypothesis. Note that all attacks on scenario 1 omit weighted

and unweighted clustering coefficient since the possibility of the neighborhood of a node (where the neighborhood is greater than two) being connected is always zero.

### Jamming Attacks

System = (scenario 1, jamming, normal, light)

Metric	Calculated z-value	Looked up p-value	Detected attack at 0.05?	At 0.01?
ann	29.74	$\approx 0$	Yes	Yes
ci	5.25	$\approx 0$	Yes	Yes
w2l	2.41	0.008	Yes	Yes
w2n	0.80	0.2119	No	No
wd	19.38	$\approx 0$	Yes	Yes
wri	14.81	$\approx 0$	Yes	Yes

System = (scenario 1, jamming, light, medium-light)

Metric	Calculated z-value	Looked up p-value	Detected attack at 0.05?	At 0.01?
ann	29.12	$\approx 0$	Yes	Yes
ci	2.33	0.0099	Yes	Yes
w2l	6.50	$\approx 0$	Yes	Yes
w2n	6.25	$\approx 0$	Yes	Yes
wd	14.64	$\approx 0$	Yes	Yes
wri	10.83	$\approx 0$	Yes	Yes

System = (scenario 1, jamming, medium-heavy, heavy)

Metric	Calculated z-value	Looked up p-value	Detected attack at 0.05?	At 0.01?
ann	6.72	$\approx 0$	Yes	Yes
ci	2.68	0.0037	Yes	Yes
w2l	1.96	0.025	Yes	No
w2n	1.46	0.0721	No	No
wd	4.89	$\approx 0$	Yes	Yes
wri	3.21	0.0007	Yes	Yes

System = (scenario 1, jamming, very heavy, very very heavy)

Metric	Calculated z-value	Looked up p-value	Detected attack at 0.05?	At 0.01?
ann	10.20	$\approx 0$	Yes	Yes
ci	2.47	0.0068	Yes	Yes

w2l	2.59	0.0048	Yes	Yes
w2n	1.30	0.0963	No	No
wd	7.97	$\approx 0$	Yes	Yes
wri	7.52	$\approx 0$	Yes	Yes

System = (scenario 2, jamming, normal, light)

Metric	Calculated z-value	Looked up p-value	Detected attack at 0.05?	At 0.01?
ann	12.66	$\approx 0$	Yes	Yes
cc	1.36	0.0869	No	No
ci	24.94	$\approx 0$	Yes	Yes
w2l	7.18	$\approx 0$	Yes	Yes
w2n	6.44	$\approx 0$	Yes	Yes
wcc	1.87	0.0307	Yes	No
wd	16.45	$\approx 0$	Yes	Yes
wri	8.31	$\approx 0$	Yes	Yes

System = (scenario 2, jamming, light, medium light)

Metric	Calculated z-value	Looked up p-value	Detected attack at 0.05?	At 0.01?
ann	7.97	$\approx 0$	Yes	Yes
cc	2.42	0.0078	Yes	Yes
ci	15.66	$\approx 0$	Yes	Yes
w2l	4.56	$\approx 0$	Yes	Yes
w2n	6.83	$\approx 0$	Yes	Yes
wcc	2.11	0.0174	Yes	No
wd	12.73	$\approx 0$	Yes	Yes
wri	5.89	$\approx 0$	Yes	Yes

System = (scenario 2, jamming, medium-heavy, heavy)

Note: It is not possible for any metric to detect this attack. The reason for this is that node 5 jams the entire region of nodes 1-9, and therefore node 2 cannot do any more damage then node 5 is already doing. For this reason, the data has been omitted.

System = (scenario 2, jamming, very heavy, very very heavy)

Metric	Calculated z-value	Looked up p-value	Detected attack at 0.05?	At 0.01?
ann	5.03	$\approx 0$	Yes	Yes
cc	1.69	0.0455	Yes	No
ci	4.84	$\approx 0$	Yes	Yes
w2l	-0.85	0.8023	No	No

w2n	-0.07	0.5279	No	No
wcc	2.46	0.0069	Yes	Yes
wd	4.00	$\approx 0$	Yes	Yes
wri	1.79	0.0367	Yes	No

### Range Attacks

System = (scenario 1, range, normal, light)

Metric	Calculated z-value	Looked up p-value	Detected attack at 0.05?	At 0.01?
ann	12.75	$\approx 0$	Yes	Yes
ci	6.74	$\approx 0$	Yes	Yes
w2l	0.91	0.1841	No	No
w2n	1.54	0.0618	No	No
wd	15.78	$\approx 0$	Yes	Yes
wri	6.87	$\approx 0$	Yes	Yes

System = (scenario 1, range, light, medium-light)

Metric	Calculated z-value	Looked up p-value	Detected attack at 0.05?	At 0.01?
ann	4.66	$\approx 0$	Yes	Yes
ci	-0.50	0.6915	No	No
w2l	1.66	0.0485	Yes	No
w2n	3.57	$\approx 0$	Yes	Yes
wd	6.08	$\approx 0$	Yes	Yes
wri	2.20	0.0139	Yes	No

System = (scenario 1, range, medium-heavy, heavy)

Metric	Calculated z-value	Looked up p-value	Detected attack at 0.05?	At 0.01?
ann	7.88	$\approx 0$	Yes	Yes
ci	2.70	0.0035	Yes	Yes
w2l	0.02	0.492	No	No
w2n	0.86	0.1949	No	No
wd	8.47	$\approx 0$	Yes	Yes
wri	6.69	$\approx 0$	Yes	Yes

System = (scenario 1, range, very heavy, very very heavy)

Metric	Calculated z-value	Looked up p-value	Detected attack at 0.05?	At 0.01?
ann	8.48	$\approx 0$	Yes	Yes
ci	2.95	0.0016	Yes	Yes

w2l	3.54	$\approx 0$	Yes	Yes
w2n	2.97	0.0015	Yes	Yes
wd	9.19	$\approx 0$	Yes	Yes
wri	7.14	$\approx 0$	Yes	Yes

System = (scenario 2, range, normal, light)

Metric	Calculated z-value	Looked up p-value	Detected attack at 0.05?	At 0.01?
ann	0.04	0.4801	No	No
cc	-0.10	0.5398	No	No
ci	0	0.5	No	No
w2l	2.17	0.015	Yes	No
w2n	-1.30	0.9032	No	No
wcc	-0.71	0.758	No	No
wd	-0.01	0.504	No	No
wri	1.18	0.119	No	No

System = (scenario 2, range, light, medium-light)

Metric	Calculated z-value	Looked up p-value	Detected attack at 0.05?	At 0.01?
ann	3.42	0.0003	Yes	Yes
cc	0.20	0.4207	No	No
ci	0	0.5	No	No
w2l	-0.49	0.6879	No	No
w2n	0.22	0.4129	No	No
wcc	0.39	0.3483	No	No
wd	0.03	0.488	No	No
wri	-0.23	0.591	No	No

System = (scenario 2, range, medium-heavy, heavy)

Metric	Calculated z-value	Looked up p-value	Detected attack at 0.05?	At 0.01?
ann	2.18	0.0146	Yes	No
cc	4.14	$\approx 0$	Yes	Yes
ci	0	0.5	No	No
w2l	2.03	0.0212	Yes	No
w2n	2.97	0.0015	Yes	Yes
wcc	3.50	$\approx 0$	Yes	Yes
wd	2.44	0.0073	Yes	Yes
wri	0.73	0.2327	No	No

System = (scenario 2, range, very heavy, very very heavy)

Metric	Calculated z-value	Looked up p-value	Detected attack at 0.05?	At 0.01?
ann	3.96	$\approx 0$	Yes	Yes
cc	-1.15	0.8749	No	No
ci	0	0.5	No	No
w2l	1.34	0.0901	No	No
w2n	0.80	0.2119	No	No
wcc	-0.34	0.6331	No	No
wd	3.16	0.0008	Yes	Yes
wri	2.73	0.0032	Yes	Yes

**Table 5.4: Summarized Data**

Note: The reason *ci* is always 0 is because the graph never becomes disconnect.

## 5.4 Metric Comparisons

In analyzing this data, it is important to keep in mind that the data is meant to shed a light on the sensitivity of each metric given a certain type of simulation system. Therefore, the analysis should be done on different categories of systems.

The first category includes all of the experiments run. This “overall ranking” of the metrics is based on how many attacks each detected at the p-value cutoff level of 0.95 plus how many attacks each detected at the cutoff level of 0.99 (1 point per detected attack). As a side note, it is not fair to compare unweighted metrics to weighted metrics. This is because, even if an unweighted metric is ranked higher during these hypothesis tests, the weighted metric may still be preferred because it has the ability to convey important connectivity information not seen in these hypothesis tests. For instance, if a

simulation system (the 4-tuple defined previously) were plotted in Microsoft Excel, GNUplot, or any other plotting program, for one run with the  $y$ -axis as the metric and the  $x$ -axis as time, the weighted metric would indicate clear recovery periods, which would not be visible with an unweighted metric. This distinct advantage may be a desirable property for a scientist, even if the metric is not quite as good at detecting an attack. For this reason, weighted metrics are not compared against unweighted metrics. Below are the rankings of the metrics by overall points.

Overall Weighted Metric Ranking (total possible points = 30)

<b>Metric</b>	<b>Points</b>
Weighted Density	26
Weighted Chemical Index	22
Weighted 2-hop Link Density	16
Weighted 2-hop Node Density	12
Weighted Clustering Coefficient	6

Overall Unweighted Metric Ranking (total possible points = 30)

<b>Metric</b>	<b>Points</b>
Average Number of Neighbors	27
Connectivity Index	20
Clustering Coefficient	5

**Table 5.5: Overall Metric Rankings**

The top two metrics for the weighted metric category are the weighted density and the weighted chemical index. Therefore, if one were to run an attack or jamming simulation with an uncertain, random, or changing network structure and/or density, these preliminary results indicate that the weighted density and the weighted chemical index

would best measure network health for weighted graphs. If one were to use unweighted metrics then the average number of neighbors would be the best measure, followed by the connectivity index. It is very important to note that the connectivity index should not be used for a dense graph, since if the graph never becomes disconnected, the connectivity index does not indicate drops in connectivity.

It is also clear from the sample taken that both the weighted clustering coefficient and clustering coefficient perform poorly with dense graphs and do not perform at all with sparse graphs. These findings seem to indicate that clustering coefficient based metrics are overall poor choices for measuring network health.

The next category is attack-specific rankings. In other words, metrics are ranked in accordance with how well they performed within an attack framework. We first consider the jamming attacks:

Jamming Weighted Metric Ranking (total possible points = 14)

<b>Metric</b>	<b>Points</b>
Weighted Density	14
Weighted Chemical Index	13
Weighted 2-hop Link Density	11
Weighted 2-hop Node Density	6
Weighted Clustering Coefficient	4

Jamming Unweighted Metric Ranking (total possible points = 14)

<b>Metric</b>	<b>Points</b>
Average Number of Neighbors	14
Connectivity Index	14
Clustering Coefficient	3

**Table 5.6: Jamming Metric Rankings**

Compared to the overall ratings, the metrics are relatively clear-cut in terms of which properly detect the jamming attack. For the weighted metrics, the weighted density, weighted chemical index, and weighted 2-hop link density all performed well and would be recommended metrics to use. For the unweighted metrics, the average number of neighbors and connectivity index performed well.

We next consider the range attacks:

Range Weighted Metric Ranking (total possible points = 16)

<b>Metric</b>	<b>Points</b>
Weighted Density	12
Weighted Chemical Index	9
Weighted 2-hop Node Density	6
Weighted 2-hop Link Density	5
Weighted Clustering Coefficient	2

Range Unweighted Metric Ranking (total possible points = 16)

<b>Metric</b>	<b>Points</b>
Average Number of Neighbors	13
Connectivity Index	6
Clustering Coefficient	2

**Table 5.7: Range Metric Rankings**

Overall the metrics had a more difficult time detecting this type of attack. The reason for this is that in a jamming attack, the links surrounding the node performing the jamming attack are broken. In scenario 2 of the range attack, only the diagonal links are broken and therefore the graph remains connected. For this reason, the connectivity index did not perform well compared to its performance with the jamming attack.

The weighted metrics were more evenly dispersed, with weighted density and weighted chemical index performing the best. Again, the average number of neighbors is the top performer for the unweighted metrics.

The final category we consider is scenario-specific rankings. This makes clear how the metrics perform at different density levels (scenario 1 is less dense than scenario 2). For scenario 1, we obtain:

Scenario 1 Weighted Metric Ranking (total possible points = 16)

<b>Metric</b>	<b>Points</b>
Weighted Density	16
Weighted Chemical Index	15
Weighted 2-hop Link Density	10
Weighted 2-hop Node Density	6
Weighted Clustering Coefficient	0

Scenario 1 Unweighted Metric Ranking (total possible points = 16)

<b>Metric</b>	<b>Points</b>
Average Number of Neighbors	16
Connectivity Index	14
Clustering Coefficient	0

**Table 5.8: Scenario 1 Metric Rankings**

The ranking by scenarios greatly improves our understanding of the metric performance. The weighted density and weighted chemical index are again the strong leaders in the weighted metrics category, followed by the weighted 2-hop link density. For the unweighted metrics, the average number of neighbors and the connectivity index perform very well. The fact that scenario 1 is relatively sparse (or not dense), is the reason why the connectivity index performs so well. This indicates to scientists that connectivity index is a good metric if the attack is likely to disconnect the graph (i.e., the graph is sparse). The clustering coefficient metrics are 0 because the grid formation allows for no possible clusters. For scenario 2 we obtain:

Scenario 2 Weighted Metric Ranking (total possible points = 14)

<b>Metric</b>	<b>Points</b>
Weighted Density	10
Weighted Chemical Index	7
Weighted 2-hop Link Density	6
Weighted 2-hop Node Density	6
Weighted Clustering Coefficient	6

Scenario 2 Unweighted Metric Ranking (total possible points = 14)

<b>Metric</b>	<b>Points</b>
Average Number of Neighbors	11
Connectivity Index	6
Clustering Coefficient	5

**Table 5.9: Scenario 2 Metric Rankings**

Overall the metrics performance was not as good as with scenario 1, for the reason that the range attack in scenario 2 did not drop the connectivity as much as in the other systems. For the weighted metrics, the weighted density is the metric that detected the most attacks. For the unweighted metrics, the average number of neighbors performed the best.

The following summarizes the strengths and weaknesses of the metrics being analyzed.

- *Weighted Density* – The metric consistently performed best from the weighted metric category. Furthermore, it has the best scalability in relation to the other weighted metrics. For these reasons, this metric should be the top choice for scientists under all of the systems performed in this thesis.
- *Weighted Clustering Coefficient* – This metric consistently performed the worst in the weighted metrics category. One reason for this could be that the scenarios used in this thesis are highly non-clustered. Future work in this area should include a more in-depth study of how the weighted clustering coefficient metric ranks among the other metrics mentioned under a highly clustered scenario.

Looking purely at the results of the experiments run in this thesis, however, the weighted clustering coefficient metric is a poor choice to measure network health.

- *Weighted 2-hop Node Density* – Overall this metric performed at an average level in relation to the other metrics of the weighted metric category. The other metrics are, however, more scalable. Furthermore, the weighted 2-hop node density almost always performs worse than the similar weighted 2-hop link density, and therefore should not be a preferred metric.
- *Weighted 2-hop Link Density* – This metric performed better in scenario 1 than in scenario 2, whereas the weighted 2-hop node density performed at the same level. Since the scalability is not relatively good, this metric should not be favored to weighted density of weighted chemical index, but should be favored to weighted 2-hop node density.
- *Weighted Chemical Index* – This experimental metric used primarily in chemistry related issues performed at an exceptionally high level, second only to the weighted density metric. Furthermore, this metric performed well under all experiments run, and did not seem to struggle with any one in particular. The scalability is only slightly worse than weighted density. For these reasons, this metric is an excellent choice for an arbitrary scenario, but should not be favored to the weighted density metric.
- *Average Number of Neighbors* – This metric performed exceptionally high in the unweighted metric category. With a very desirable scalability, this metric is, without question, the best general metric in the unweighted metric category.

- *Clustering Coefficient* – This metric performed poorly in almost all systems, and therefore should not be a choice metric. The same comments as weighted clustering coefficient apply to this metric.
- *Connectivity Index* – This metric performed very well, but under a big restriction. If the network never becomes disconnected, then it is impossible for this metric to detect an attack. If, however, this scenario is unlikely, the connectivity index is a favorable metric in the unweighted metrics category, seconded only to the average number of neighbors metric.

# Chapter 6

## Conclusions and Future Work

The goal of providing a protocol for transferring connectivity information from the simulation model of the physical network into a concise, single real number is made easier by using a divide-and-conquer based approach. We have divided this problem into two equally important sub-problems: the network-to-graph problem and the graph-to-metric problem.

The network-to-graph problem is that of summarizing the connectivity information build via simulation into a mathematical graph, which can be both weighted and directed. We have proposed a solution to this problem by augmenting SWAN to produce PDR-edged graphs for every user-specified time interval. This novel solution allows important connectivity information, such as recovery time from an attack, to be clearly indicated

and places much emphasis on how well links truly behave. This method takes into account the packet-delivery ratio for every link in the given time interval, as well as the number of control packets sent over that link.

After providing the foundation of the PDR-edged graphs, we present a method that is used to compare and rank solutions (the metrics themselves) to the graph-to-metric problem. This method is hypothesis testing, and is a well developed field in statistics. After creating a catalogue of common metrics, including an experiment metric from another science, hypothesis testing was used to determine how well each individual metric performed as a solution to the graph-to-metric problem. The metrics were then indirectly compared using the results from the individual hypothesis tests, and conclusions were drawn as to which metrics performed the best.

This thesis was developed to address the lack of literature regarding the transporting of connectivity information from the physical network simulation model to connectivity metrics. We believe this thesis is a start, but there is much future work to be done in the area. First, due to the sheer number of simulations it takes to obtain good statistical data, a very limited number of possible scenarios were examined. For a more complete analysis, many more scenarios, especially those without grid formations, need to be incorporated. Second, we have examined only range and jamming attacks. The attack scenarios can be easily expanded to include many other kinds of attacks. One particular attack we are interested in is the reboot attack, where a node is periodically shutdown and

rebooted potentially causing an increase in control packets. Our PDR-edged graph solution to the network-to-graph problem provides an excellent foundation for this type of attack as penalties are introduced for a high control packet to non-control packet ratio. Third, a larger range of connectivity metrics should be tested, analyzed and compared to the basic set we have catalogued.

In conclusion, this thesis presents a starting point for understanding network health in a simulation environment. This is an important area of wireless networking, and we hope to spark a great deal of interest and future progress in quantifying network health in wireless ad hoc networks.

# Chapter 7

## Motivation

As a rising junior, having never done research before, I was unsure of what to expect when I approached Professor Perrone during the first day of my summer research program at Bucknell. The research sounded interesting and challenging – augmenting a state-of-the-art computer simulator to run attack scenarios on wireless ad hoc networks. After pressing through the learning curve of wireless network operation, discrete-event simulation, and the ever-present set of new jargon words, I became hooked. I was being paid to work on a project I actually enjoyed, which was a new experience for me. Furthermore, I was contributing to a project that would benefit many scientists and experts around the world, as well as our government. The benefits of the overall sensor network technology were staggering: military communication, scientific exploration, surveillance, etc. Just recently, in fact, I learned that professors of geophysics and

computer science from Harvard, University of North Carolina, and University of New Hampshire have collaborated to deploy two wireless sensor networks in active volcanoes located in Ecuador, in order to monitor volcanic eruptions [17]. I find this kind of research extremely interesting and very fun, which is my personal motivation for pursuing topics related to it.

After making a fairly good amount of progress over the summer, I decided that I would like to continue, if possible, exploring the theoretical side of these wireless ad hoc networks. Professor Perrone and I noticed that, while implementing the attack scenarios over the summer, the literature greatly lacked information regarding what connectivity metric is best used to record how effective the attacks are. In fact, there is even little literature quantifying the effects of attacks. This was a major problem since if the metric isn't good at recording what truly is happening in the graph, even the best attack implementations won't provide much insight into what's really happening. After thinking about the problem more, I decided that I would like to write an honors thesis on it.

One issue that a computer scientist or mathematician may struggle with is practicality. Wireless sensor networks, however, have a virtually unlimited amount of practicality in today's world. The sheer amount of potential they have is more than enough motivation for companies and governments to fund almost anything that is remotely connected to wireless sensor network research. Describing some practical scenarios puts into light how useful this technology is, which greatly adds to the overall motivation of studying these

networks and, particularly, the motivation for studying connectivity metrics of these metrics.

The first example scenario is the scenario laid out in the introduction of this honors thesis. It describes the emergency response use of wireless sensor networks, as the scenario creates an imaginary natural disaster or terrorist attack scene where the sensors are used to monitor the area for damage ratings. It is clear that in both a natural disaster and terrorist attack scenario, nodes and links are constantly being destroyed. Therefore, it is clearly useful for the emergency response teams to have information regarding the connectivity of the network as well as know in advance how the connectivity drops in the event of an attack on the network.

A second example scenario was mentioned above regarding the three research professors who deployed two wireless sensor networks to monitor active volcanoes. Since the area around and inside of an active volcano is obviously hazardous, one would expect nodes and links to be constantly destroyed. Therefore, the same knowledge of connectivity required in the first scenario would be extremely useful in this scenario as well.

For a third example scenario, imagine a massive wireless sensor network deployed on a battlefield and used to monitor the locations of enemy troops. As more and more ambushes from the enemy fail, the enemy eventually will realize that their positions are somehow being monitored and will attempt to jam the area surrounding them. Therefore,

the knowledge of how jamming effects connectivity would be very relevant and useful information for our military to have.

These plausible scenarios illustrate the practicality and application of wireless sensor networks, and demonstrate the need for the users of the sensor networks to have accurate information about the current connectivity of the network, as well as information allowing them to make predictions of how future attacks will affect their networks. The sum of all of these scenarios and illustrations provides a strong scientific and practical motivation for the research undergone in this honors thesis.

# Appendix A

## Metric Source Code Listings

The following source code listings are the C++ code used to implement the metrics discussed in this thesis. Helper methods are listed about the main methods. All methods belong to the Metric class.

### A.1 Weighted Metrics

#### Weighted Density

```
double Metric::WeightedDensity(Graph &inGraph) {  
    // Density is the total amount of "linkage" divided by the max  
    // possible "linkage"  
  
    double totalEdges = 0;
```

```

    for (int row = 0; row < inGraph.n; row++) {
        for (int col = 0; col < inGraph.n; col++) {
            totalEdges+= inGraph.adj_matrix[row][col];
        }
    }

    return (totalEdges / (double)(inGraph.n * (inGraph.n-1)));
}

```

#### Weighted Chemical Index

```

double Metric::Strength(Graph &inGraph, int a) {

    // Calculate the strength of node a
    // Strength = total weight exiting the node
    double strength = 0;
    for (int col = 0; col < inGraph.n; col++)
        strength += inGraph.adj_matrix[a][col];

    return strength;
}

double Metric::WeightedChemicalIndex(Graph &inGraph) {

    // Algorithm for weighted Chemical index:
    // 1) Get the first edge from the adj_matrix
    // 2) Obtain the strength (str) of the two vertices that make it up
    // 3) Calculate [str(u)*str(v)]^(.5) as the weight of the edge
    // 4) Repeat, keeping a summation of the weight of all edges

    // Declarations
    double str1, str2;
    double wEdge, wEdgeTotal = 0;

    // Loop through adj_matrix
    for (int row = 0; row < inGraph.n; row++) {
        for (int col = 0; col < inGraph.n; col++) {

            // Only process if an edge
            if (inGraph.adj_matrix[row][col] > 0) {

                // Obtain the strengths of each vertex
                str1 = Strength(inGraph, row);
                str2 = Strength(inGraph, col);

                // Apply the formula
                if (sqrt(str1 * str2) > 0) {
                    wEdge = sqrt(str1 * str2) * inGraph.adj_matrix[row][col];
                    wEdgeTotal += wEdge;
                }
            }
        }
    }
}

```

```

    }
}

// Return the metric
return wEdgeTotal;
}

```

#### Weighted 2-hop Node Density

```

bool Metric::isNodeInVector(vector <int> vec, int a) {

    //Is element a in vec?

    for (unsigned int i = 0; i < vec.size(); i++)
        if (vec[i] == a)
            return true;

    return false;
}

vector <int> Metric::NodesAdjacentTo(Graph &inGraph, int a) {

    // Returns the nodes adjacent to node a

    vector<int> NodesAdjTo;

    for (int col = 0; col < inGraph.n; col++) {
        // Be sure there is a bi-directional link
        if (inGraph.adj_matrix[a][col] > 0 && inGraph.adj_matrix[col][a]>0)
            NodesAdjTo.push_back(col);
    }

    return NodesAdjTo;
}

vector <int> Metric::TwoHopNeighborhood(Graph &inGraph, int a, int b) {

    // The easiest way to find all nodes that are at most 2-hops
    // away from link ab is to mark all nodes that are at most
    // 2-hops away from node a and to mark all nodes that are at
    // most 2-hops away from node b. The intersection of these two
    // sets are the nodes that are 2-hops away from both a and b.

    vector<int> nodes1HopFromA;
    vector<int> nodes1HopFromB;
    vector<int> nodes2HopFromA;
    vector<int> nodes2HopFromB;
    vector<int> intersection;

    // Store the nodes that are 1-hop away from A in nodes1HopFromA

```

```

nodes1HopFromA = NodesAdjacentTo(inGraph, a);
// Store the nodes that are 2-hops away from A in nodes2HopFromA
for (unsigned int i = 0; i < nodes1HopFromA.size(); i++) {
    vector<int> temp = NodesAdjacentTo(inGraph, nodes1HopFromA[i]);
    for(unsigned int j = 0; j < temp.size(); j++)
        nodes2HopFromA.push_back(temp[j]);
}
// Copy nodes1HopFromA into nodes2HopFromA
for (unsigned int j = 0; j < nodes1HopFromA.size(); j++)
    nodes2HopFromA.push_back(nodes1HopFromA[j]);

// Same for B
nodes1HopFromB = NodesAdjacentTo(inGraph, b);
for (unsigned int i = 0; i < nodes1HopFromB.size(); i++) {
    vector<int> temp = NodesAdjacentTo(inGraph, nodes1HopFromB[i]);
    for(unsigned int j = 0; j < temp.size(); j++)
        nodes2HopFromB.push_back(temp[j]);
}
for (unsigned int j = 0; j < nodes1HopFromB.size(); j++)
    nodes2HopFromB.push_back(nodes1HopFromB[j]);

// Copy nodes common to both into intersection (not
// duplicates though)
for (unsigned int i = 0; i < nodes2HopFromA.size(); i++) {
    int nodeInCheck = nodes2HopFromA[i];
    if (isNodeInVector(nodes2HopFromB, nodeInCheck) &&
        !(isNodeInVector(intersection, nodeInCheck)))
        intersection.push_back(nodeInCheck);
}

return intersection;
}

long int Metric::Weighted2HopNode(Graph &inGraph, int a, int b) {

    // Precondition: There is a non-zero weighted link between nodes
    // a and b

    vector <int> intersection = TwoHopNeighborhood(inGraph, a, b);

    // Now intersection contains all nodes (and no duplicates) that
    // are 2-hops away from node a and 2-hops away from node b.
    return intersection.size();
}

double Metric::Weighted2HopNodeDensity(Graph &inGraph) {

    // Calculate the average of the 2-hop node densities for each link

    double totalValue = 0;

```

```

long int totalNumOfLinks = 0;

for (int row = 0; row < inGraph.n; row++) {
    for (int col = 0; col < inGraph.n; col++) {
        // Everytime we encounter a non-zero link, we get the
        // metric value for that link and multiply it by the
        // weight of the link;
        if (inGraph.adj_matrix[row][col] > 0) {
            totalNumOfLinks++;
            totalValue += ( Weighted2HopNode(inGraph, row, col) *
                inGraph.adj_matrix[row][col] );
        }
    }
}

// Return the average
return (totalValue / totalNumOfLinks);
}

```

#### Weighted 2-hop Link Density

```

double Metric::Weighted2HopLink(Graph &inGraph, int a, int b) {

    // Precondition: There is a non-zero weighted link between nodes
    // a and b

    // Basically, we want to find all of the nodes in the 2-hop
    // neighborhood of link ab (just like the Weighted2HopNode), but
    // then we want to sum all of the links going out of those nodes

    vector <int> intersection = TwoHopNeighborhood(inGraph, a, b);

    double sum = 0;
    int currentNode = -1;
    for (unsigned int i = 0; i < intersection.size(); i++) {
        currentNode = intersection[i];
        for (int col = 0; col < inGraph.n; col++) {
            sum += (inGraph.adj_matrix[currentNode][col]);
        }
    }

    return sum;
}

double Metric::Weighted2HopLinkDensity(Graph &inGraph) {

    // Calculate the average of the 2-hop link densities for each link

    double totalValue = 0;
    long int totalNumOfLinks = 0;

```

```

for (int row = 0; row < inGraph.n; row++) {
    for (int col = 0; col < inGraph.n; col++) {
        // Everytime we encounter a non-zero link, we get the
        // metric value for that link.
        if (inGraph.adj_matrix[row][col] > 0) {
            totalNumOfLinks++;
            totalValue += ( Weighted2HopLink(inGraph, row, col) *
                           inGraph.adj_matrix[row][col] );
        }
    }
}

// Return the average
return (totalValue / totalNumOfLinks);
}

```

### Weighted Clustering Coefficient

```

double Metric::WeightedClusteringOfNode(Graph &inGraph, int a) {

    // Clustering for node a is the actual number of edges between the
    // neighbors of a (not including a itself), divided by the total
    // possible number of edges between the neighbors of a.

    // First, get the total number of neighbors
    long int totalPossibleEdges = 0;
    vector<int> neighbors;
    // Step 1: Create a vector holding the neighbors of a
    for (int col = 0; col < inGraph.n; col++) {
        if (inGraph.adj_matrix[a][col] > 0) {
            neighbors.push_back(col);
        }
    }

    // Step 2: Mathematically decide the max number of edges
    if (neighbors.size() == 0)
        return 0;
    if (neighbors.size() == 1)
        return 1;
    totalPossibleEdges = ( neighbors.size() * (neighbors.size() - 1) ) /
2;

    // Step 3: Determine the actual number of edges for the neighbors
    // of a
    int curRow, curCol;
    double actualEdges = 0;
    for (unsigned int rowPointer = 0; rowPointer < neighbors.size();
rowPointer++) {
        curRow = neighbors[rowPointer];
        for (unsigned int colPointer = 0; colPointer < neighbors.size();
colPointer++) {
            curCol = neighbors[colPointer];

```

```

        actualEdges += inGraph.adj_matrix[curRow][curCol];
    }
}
// Step 4: Calculate clustering coefficient of node a
double cluster = actualEdges / totalPossibleEdges;
return cluster;
}

double Metric::WeightedClusteringCoefficient(Graph &inGraph) {
    // Average clustering coefficients of all nodes
    double averageCluster = 0;

    for (int node = 0; node < inGraph.n; node++)
        averageCluster += WeightedClusteringOfNode(inGraph, node);

    averageCluster /= inGraph.n;

    return averageCluster;
}

```

## A.2 Unweighted Metrics

### Average Number of Neighbors

```

double Metric::ANON(Graph &inGraph) {
    // NOTE: This is for directed and undirected graphs

    long int totalNON = 0;

    for (int row = 0; row < inGraph.n; row++) {
        for (int col = 0; col < inGraph.n; col++) {
            if (inGraph.adj_matrix[row][col] > 0)
                totalNON+= 1;
        }
    }

    return (totalNON / (double)inGraph.n);
}

```

### Clustering Coefficient

```
double Metric::ClusteringOfNode(Graph &inGraph, int a) {

    // Clustering for node a is the actual number of
    // edges between the neighbors of a (not including
    // a itself), divided by the total possible
    // number of edges between the neighbors of a.

    // First, get the total number of neighbors
    long int totalPossibleEdges = 0;
    vector<int> neighbors;

    // Step 1: Create a vector holding the neighbors of a
    for (int col = 0; col < inGraph.n; col++) {
        if (inGraph.adj_matrix[a][col] > 0) {
            neighbors.push_back(col);
        }
    }

    // Step 2: Mathematically decide the max number of edges
    if (neighbors.size() == 0)
        return 0;
    if (neighbors.size() == 1)
        return 1;
    totalPossibleEdges = ( neighbors.size() * (neighbors.size() - 1) ) /
2;

    // Step 3: Determine the actual number of edges for
    // the neighbors of a
    int curRow, curCol;
    long int actualEdges = 0;
    for (unsigned int rowPointer = 0; rowPointer < neighbors.size();
rowPointer++) {
        curRow = neighbors[rowPointer];
        for (unsigned int colPointer = 0; colPointer < neighbors.size();
colPointer++) {
            curCol = neighbors[colPointer];
            if (inGraph.adj_matrix[curRow][curCol] > 0)
                actualEdges++;
        }
    }

    // Step 4: Calculate clustering coefficient of node a
    double cluster = ((double)actualEdges) / totalPossibleEdges;
    return cluster;
}

double Metric::ClusteringCoefficient(Graph &inGraph) {

    // Average clustering coefficients of all nodes
    double averageCluster = 0;
```

```

    for (int node = 0; node < inGraph.n; node++)
        averageCluster += ClusteringOfNode(inGraph, node);

    averageCluster /= inGraph.n;

    return averageCluster;
}

```

### Connectivity Index

```

double Metric::ConnectivityIndex(Graph &inGraph) {

    // Declarations
    int i, j;
    long int id[inGraph.n], size[inGraph.n];
    long int counter = 0;

    // Setup
    for (i = 0; i < inGraph.n; i++) {
        id[i] = i;
        size[i] = 1;
    }

    // Loop through the bottom left half of the adjacency matrix
    // looking for connected pairs
    for (int row = 1; row < inGraph.n; row++) {
        for (int col = 0; col < row; col++) {

            // See if it's a connected pair
            if (inGraph.adj_matrix[row][col] > 0 &&
                inGraph.adj_matrix[row][col] > 0) {

                // Main Algorithm

                for (i = row; i != id[i]; i = id[i])
                    id[i] = id[id[i]];
                for (j = col; j != id[j]; j = id[j])
                    id[j] = id[id[j]];

                if (i == j) continue;

                if (size[i] < size[j]) {
                    id[i] = j;
                    size[j] += size[i];
                } else {
                    id[j] = i;
                    size[i] += size[j];
                }
            }
            // Increment the counter, if the counter reaches
            // mNodeCount - 1, then the graph is connected
            counter++;
            if (counter == inGraph.n - 1)

```

```

        break;
    }
}

long int groups[inGraph.n];
long int totalNodes = (inGraph.n * (inGraph.n - 1)) / 2;
long int actualNodes = 0;

// Connected or not?
if (counter == inGraph.n - 1) {
    actualNodes = totalNodes;
} else {
    // Reset groups
    for (i = 0; i < inGraph.n; i++)
        groups[i] = 0;

    // This keeps a tally on how many nodes are in each
    // separate cluster
    for (i = 0; i < inGraph.n; i++) {
        groups[id[i]]++;
    }

    // Add up all combinations from each group
    for (i = 0; i < inGraph.n; i++) {
        actualNodes += (groups[i] * (groups[i] - 1)) / 2;
    }
}

return (actualNodes * 1.0 / (inGraph.n * inGraph.n));
}

```

# Appendix B

## Standard Normal Cumulative Probability Table

The following table was obtained from <http://www.math.sfu.ca/~cschwarz/Stat-301/Handouts/node122.html>.

z	Second digit of Z									
	0.00	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09
-3.5	0.0002	0.0002	0.0002	0.0002	0.0002	0.0002	0.0002	0.0002	0.0002	0.0002
-3.4	0.0003	0.0003	0.0003	0.0003	0.0003	0.0003	0.0003	0.0003	0.0003	0.0002
-3.3	0.0005	0.0005	0.0005	0.0004	0.0004	0.0004	0.0004	0.0004	0.0004	0.0003
-3.2	0.0007	0.0007	0.0006	0.0006	0.0006	0.0006	0.0006	0.0005	0.0005	0.0005
-3.1	0.0010	0.0009	0.0009	0.0009	0.0008	0.0008	0.0008	0.0008	0.0007	0.0007
-3.0	0.0013	0.0013	0.0013	0.0012	0.0012	0.0011	0.0011	0.0011	0.0010	0.0010
-2.9	0.0019	0.0018	0.0018	0.0017	0.0016	0.0016	0.0015	0.0015	0.0014	0.0014
-2.8	0.0026	0.0025	0.0024	0.0023	0.0023	0.0022	0.0021	0.0021	0.0020	0.0019
-2.7	0.0035	0.0034	0.0033	0.0032	0.0031	0.0030	0.0029	0.0028	0.0027	0.0026
-2.6	0.0047	0.0045	0.0044	0.0043	0.0041	0.0040	0.0039	0.0038	0.0037	0.0036
-2.5	0.0062	0.0060	0.0059	0.0057	0.0055	0.0054	0.0052	0.0051	0.0049	0.0048
-2.4	0.0082	0.0080	0.0078	0.0075	0.0073	0.0071	0.0069	0.0068	0.0066	0.0064
-2.3	0.0107	0.0104	0.0102	0.0099	0.0096	0.0094	0.0091	0.0089	0.0087	0.0084
-2.2	0.0139	0.0136	0.0132	0.0129	0.0125	0.0122	0.0119	0.0116	0.0113	0.0110
-2.1	0.0179	0.0174	0.0170	0.0166	0.0162	0.0158	0.0154	0.0150	0.0146	0.0143
-2.0	0.0228	0.0222	0.0217	0.0212	0.0207	0.0202	0.0197	0.0192	0.0188	0.0183

-1.9	0.0287	0.0281	0.0274	0.0268	0.0262	0.0256	0.0250	0.0244	0.0239	0.0233
-1.8	0.0359	0.0351	0.0344	0.0336	0.0329	0.0322	0.0314	0.0307	0.0301	0.0294
-1.7	0.0446	0.0436	0.0427	0.0418	0.0409	0.0401	0.0392	0.0384	0.0375	0.0367
-1.6	0.0548	0.0537	0.0526	0.0516	0.0505	0.0495	0.0485	0.0475	0.0465	0.0455
-1.5	0.0668	0.0655	0.0643	0.0630	0.0618	0.0606	0.0594	0.0582	0.0571	0.0559
-1.4	0.0808	0.0793	0.0778	0.0764	0.0749	0.0735	0.0721	0.0708	0.0694	0.0681
-1.3	0.0968	0.0951	0.0934	0.0918	0.0901	0.0885	0.0869	0.0853	0.0838	0.0823
-1.2	0.1151	0.1131	0.1112	0.1093	0.1075	0.1056	0.1038	0.1020	0.1003	0.0985
-1.1	0.1357	0.1335	0.1314	0.1292	0.1271	0.1251	0.1230	0.1210	0.1190	0.1170
-1.0	0.1587	0.1562	0.1539	0.1515	0.1492	0.1469	0.1446	0.1423	0.1401	0.1379
-0.9	0.1841	0.1814	0.1788	0.1762	0.1736	0.1711	0.1685	0.1660	0.1635	0.1611
-0.8	0.2119	0.2090	0.2061	0.2033	0.2005	0.1977	0.1949	0.1922	0.1894	0.1867
-0.7	0.2420	0.2389	0.2358	0.2327	0.2296	0.2266	0.2236	0.2206	0.2177	0.2148
-0.6	0.2743	0.2709	0.2676	0.2643	0.2611	0.2578	0.2546	0.2514	0.2483	0.2451
-0.5	0.3085	0.3050	0.3015	0.2981	0.2946	0.2912	0.2877	0.2843	0.2810	0.2776
-0.4	0.3446	0.3409	0.3372	0.3336	0.3300	0.3264	0.3228	0.3192	0.3156	0.3121
-0.3	0.3821	0.3783	0.3745	0.3707	0.3669	0.3632	0.3594	0.3557	0.3520	0.3483
-0.2	0.4207	0.4168	0.4129	0.4090	0.4052	0.4013	0.3974	0.3936	0.3897	0.3859
-0.1	0.4602	0.4562	0.4522	0.4483	0.4443	0.4404	0.4364	0.4325	0.4286	0.4247
-0.0	0.5000	0.4960	0.4920	0.4880	0.4840	0.4801	0.4761	0.4721	0.4681	0.4641

z	Second digit of z									
	0.00	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09
0.0	0.5000	0.5040	0.5080	0.5120	0.5160	0.5199	0.5239	0.5279	0.5319	0.5359
0.1	0.5398	0.5438	0.5478	0.5517	0.5557	0.5596	0.5636	0.5675	0.5714	0.5753
0.2	0.5793	0.5832	0.5871	0.5910	0.5948	0.5987	0.6026	0.6064	0.6103	0.6141
0.3	0.6179	0.6217	0.6255	0.6293	0.6331	0.6368	0.6406	0.6443	0.6480	0.6517
0.4	0.6554	0.6591	0.6628	0.6664	0.6700	0.6736	0.6772	0.6808	0.6844	0.6879
0.5	0.6915	0.6950	0.6985	0.7019	0.7054	0.7088	0.7123	0.7157	0.7190	0.7224
0.6	0.7257	0.7291	0.7324	0.7357	0.7389	0.7422	0.7454	0.7486	0.7517	0.7549
0.7	0.7580	0.7611	0.7642	0.7673	0.7704	0.7734	0.7764	0.7794	0.7823	0.7852
0.8	0.7881	0.7910	0.7939	0.7967	0.7995	0.8023	0.8051	0.8078	0.8106	0.8133
0.9	0.8159	0.8186	0.8212	0.8238	0.8264	0.8289	0.8315	0.8340	0.8365	0.8389
1.0	0.8413	0.8438	0.8461	0.8485	0.8508	0.8531	0.8554	0.8577	0.8599	0.8621
1.1	0.8643	0.8665	0.8686	0.8708	0.8729	0.8749	0.8770	0.8790	0.8810	0.8830
1.2	0.8849	0.8869	0.8888	0.8907	0.8925	0.8944	0.8962	0.8980	0.8997	0.9015
1.3	0.9032	0.9049	0.9066	0.9082	0.9099	0.9115	0.9131	0.9147	0.9162	0.9177
1.4	0.9192	0.9207	0.9222	0.9236	0.9251	0.9265	0.9279	0.9292	0.9306	0.9319
1.5	0.9332	0.9345	0.9357	0.9370	0.9382	0.9394	0.9406	0.9418	0.9429	0.9441
1.6	0.9452	0.9463	0.9474	0.9484	0.9495	0.9505	0.9515	0.9525	0.9535	0.9545
1.7	0.9554	0.9564	0.9573	0.9582	0.9591	0.9599	0.9608	0.9616	0.9625	0.9633
1.8	0.9641	0.9649	0.9656	0.9664	0.9671	0.9678	0.9686	0.9693	0.9699	0.9706
1.9	0.9713	0.9719	0.9726	0.9732	0.9738	0.9744	0.9750	0.9756	0.9761	0.9767
2.0	0.9772	0.9778	0.9783	0.9788	0.9793	0.9798	0.9803	0.9808	0.9812	0.9817
2.1	0.9821	0.9826	0.9830	0.9834	0.9838	0.9842	0.9846	0.9850	0.9854	0.9857
2.2	0.9861	0.9864	0.9868	0.9871	0.9875	0.9878	0.9881	0.9884	0.9887	0.9890
2.3	0.9893	0.9896	0.9898	0.9901	0.9904	0.9906	0.9909	0.9911	0.9913	0.9916
2.4	0.9918	0.9920	0.9922	0.9925	0.9927	0.9929	0.9931	0.9932	0.9934	0.9936
2.5	0.9938	0.9940	0.9941	0.9943	0.9945	0.9946	0.9948	0.9949	0.9951	0.9952
2.6	0.9953	0.9955	0.9956	0.9957	0.9959	0.9960	0.9961	0.9962	0.9963	0.9964
2.7	0.9965	0.9966	0.9967	0.9968	0.9969	0.9970	0.9971	0.9972	0.9973	0.9974
2.8	0.9974	0.9975	0.9976	0.9977	0.9977	0.9978	0.9979	0.9979	0.9980	0.9981
2.9	0.9981	0.9982	0.9982	0.9983	0.9984	0.9984	0.9985	0.9985	0.9986	0.9986
3.0	0.9987	0.9987	0.9987	0.9988	0.9988	0.9989	0.9989	0.9989	0.9990	0.9990
3.1	0.9990	0.9991	0.9991	0.9991	0.9992	0.9992	0.9992	0.9992	0.9993	0.9993
3.2	0.9993	0.9993	0.9994	0.9994	0.9994	0.9994	0.9994	0.9995	0.9995	0.9995
3.3	0.9995	0.9995	0.9995	0.9996	0.9996	0.9996	0.9996	0.9996	0.9996	0.9997
3.4	0.9997	0.9997	0.9997	0.9997	0.9997	0.9997	0.9997	0.9997	0.9997	0.9998
3.5	0.9998	0.9998	0.9998	0.9998	0.9998	0.9998	0.9998	0.9998	0.9998	0.9998

# Bibliography

- [1] Bettstetter, Christian. 2002. *On the Minimum Node Degree and Connectivity of a Wireless Multihop Network*. Lausanne, Switzerland ed. New York, NY, USA: ACM Press.
- [2] Brenner, Pablo. *A Technical Tutorial on the IEEE 802.11 Protocol*. Internet on-line. Available from <[http://www.sss-mag.com/pdf/802\\_11tut.pdf](http://www.sss-mag.com/pdf/802_11tut.pdf)>. (March 2006).
- [3] Calinescu, Gruia, and Wan, Peng-Jun. 2003. *Range Assignment for High Connectivity in Wireless Ad Hoc Networks*. ADHOC-NOW 2003:235-246.
- [4] CERT® Coordination Center. *CERT/CC Denial of Service*. June 4, 2001. Internet on-line. Available from <[http://www.cert.org/tech\\_tips/denial\\_of\\_service.html](http://www.cert.org/tech_tips/denial_of_service.html)>. (March 2006).
- [5] *Dictionary.com - Ad hoc*. (March 2006).
- [6] Gibbons, Alan. 1985. *Algorithmic Graph Theory*. Cambridge: Cambridge University Press.
- [7] *Google Defines*. Internet on-line. Available from <<http://www.google.com/search?hl=en&lr=&oi=defmore&defl=en&q=define:Wireless+sensor+network>>. (March 2006).
- [8] Kotz, David, Calvin Newport, and Chip Elliott. 2003. *The Mistaken Axioms of Wireless-Network Research*.
- [9] Krishna, Paul, RoyChoudhuri Romit, and Bandyopadhyay Somprakash. 2000. *Survivability Analysis of Ad Hoc Wireless Network Architecture*. In *Mobile and*

- [10] Larsen, Richard J., and Morris L. Marx. 2001. *An Introduction to Mathematical Statistics and its Applications*. Upper Saddle River, NJ: Prentice Hall.
- [11] Leguay, Jérémie, Timur Friedman, Vania Conan, and Serge Fdida. 2005. Connectivity Aware Routing in Ad-hoc Networks. In *10th IFIP International Conference on Personal Wireless Communications PWC'05*. Colmar, France.
- [12] Liu, J., L. F. Perrone, D. M. Nicol, M. Liljenstam, C. Elliott, and D. Pearson. 2001. Simulation Modeling of Large-Scale Ad Hoc Sensor Networks. In *Proceedings of the European Simulation Interoperability Workshop (Euro-SIW 2001)*.
- [13] Liu, J., Y. Yuan, D. M. Nicol, R. S. Gray, C. C. Newport, D. Kotz, and L. F. Perrone. 2004. Simulation Validation Using Direct Execution of Wireless Ad-hoc Routing Protocols. In *Proceedings of the 18th Workshop on Parallel and Distributed Simulation (PADS'04)*7-16.
- [14] Lopez-Fernandez, L., G. Robles, and J. M. Gonzalez-Barahona. 2004. *Applying Social Network Analysis to the Information in CVS Repositories*. In *International Workshop on Mining Software Repositories*. Edinburgh, UK.
- [15] Mathworld. *Undirected graph*. Internet on-line. Available from <http://mathworld.wolfram.com/UndirectedGraph.html>>. (March 2006).
- [16] Mei, Lu, Liu Huiqing, and Tian Feng. 2004. The Connectivity Index. *MATCH Communications in Mathematical and in Computer Chemistry* 51: 149-154.
- [17] *Monitoring Volcanic Eruptions with a Wireless Sensor Network*. August 2005Internet on-line. Available from <http://www.eecs.harvard.edu/~mdw/proj/volcano/>>. (March 2006).
- [18] *The Network Simulator*. Internet on-line. Available from <http://www.isi.edu/nsnam/ns/>>. (April 2006).
- [19] NIST. *Wireless Ad Hoc Networks: Smart Sensor Networks*. Internet on-line. Available from [http://w3.antd.nist.gov/wahn\\_ssn.shtml](http://w3.antd.nist.gov/wahn_ssn.shtml)>. (March 2006).
- [20] NIST. *DSR Readme File - NIST DSR Model*. December 2001 (April 2006).
- [21] Noubir, Guevara. *On Connectivity in Ad Hoc Networks Under Jamming Using Directional Antennas and Mobility*. Springer Berlin / Heidelberg.

- [22] Perrone, L. F., Yougu Yuan, and David M. Nicol. 2003. Modeling and Simulation Best Practices for Wireless Ad Hoc Networks. In *Proceedings of the 2003 Winter Simulation Conference*.
- [23] Pister, K. S. J., J. M. Kahn, and B. E. Boser. 1999. Smart dust: Wireless Networks of Millimeter-Scale Sensor Nodes. Highlight Article in 1999 Electronics Research Laboratory Research Summary.
- [24] Ross, Sheldon. 2002. *A First Course in Probability*. Upper Saddle River, NJ: Prentice Hall.
- [25] Scott, John. Chapter 4 - Density: Ego-Centric and Socio-Centric. In *Social Network Analysis: A Handbook*.
- [26] Sedgewick, Robert. 1998. *Algorithms in C++ / Parts 1-4*. Addison-Wesley.
- [27] Sedgewick, Robert. 2002. *Algorithms in C++ / Part 5 Graph Algorithms*. Addison-Wesley.
- [28] Tsirigos, A., Z. J. Haas. 2004. Analysis of Multipath Routing- PartI: The Effect on the Packet Delivery Ratio. *IEEE Transactions on Wireless Communications* 3: 138-146.
- [29] Tsukiyama, S., I. Shirakawa, and H. Ozaki. 1980. An Algorithm to Enumerate All Cutsets of a Graph in Linear Time per Cutset. *Journal of the ACM* 27, no. 4: 619-632.
- [30] *Wikipedia - Signal-to-Noise Ratio*. April 2006. Internet on-line. Available from <[http://en.wikipedia.org/wiki/Signal-to-noise\\_ratio](http://en.wikipedia.org/wiki/Signal-to-noise_ratio)>. (April 2006).
- [31] Wood, Anthony D., John A. Stankovic. 2002. Denial of Service in Sensor Networks. *Computer* 35: 54-62.
- [32] Vardeman, Stephen B., and J. M. Jobe. 2001. *Basic Engineering Data Collection and Analysis*. Duxbury Thomson Learning.
- [33] Zeng, X., R. Bagrodia, and M. Gerla. 1998. GloMoSim: A Library for Parallel Simulation of Large-Scale Wireless Networks. In *Proceedings of the 12<sup>th</sup> Workshop on Parallel and Distributed Simulation*.