A MODULAR, VISUAL SIMULATOR OF UNDERWATER SENSOR NETWORKS

BY

THOMAS E. TAMAYO

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

IN

ELECTRICAL ENGINEERING

UNIVERSITY OF RHODE ISLAND

2011

MASTER OF SCIENCE THESIS

OF

THOMAS E. TAMAYO

APPROVED:

Thesis Committee:

Major Professor _____

_____

_____

_____

DEAN OF THE GRADUATE SCHOOL

UNIVERSITY OF RHODE ISLAND

2011

ABSTRACT

Wireless underwater sensor networks have many potential applications: disaster notification and recovery, waterway usage monitoring, and scientific applications. Truly underwater nodes have few communications options, of which acoustics is one option. Underwater acoustics communication networks have unique difficulties, notably that propagation speed is much lower than in radio sensor networks. The data throughput is also much lower with underwater acoustics at moderate distances (less than 1km). Examining underwater acoustic networks visually is useful to determine the cause of network failures. This thesis presents a network simulator geared to underwater acoustics, including an animated, visual component. A radio sensor network protocol from the literature is implemented and its failures in the underwater acoustic medium are examined.

ACKNOWLEDGMENTS

I would like to give great thanks to my professor, Dr. Peter Swaszek, for assisting me with this thesis. Special thanks goes to my wife, Christy, and my children: Julian and Eden (who was born during my graduate work) for putting up with my odd hours and alternating spurts of constant work and idleness.

Thanks also goes to my sponsors, The Naval Undersea Warfare Center division Newport (NUWC div NPT), The Center of Excellence in Undersea Technology (COEUT), and the University of Rhode Island Office of Special Programs.

Thanks, too, to my professors Dr. Harold Vincent III and Dr. Manbir Sodhi who have helped me in the COEUT Distributed Network Systems (DNS) program and with my thesis.

TABLE OF CONTENTS

Page

Page

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

1.1 Underwater Sensor Network Applications

Water makes up about 70% of the Earth's surface. Our waterways, littorals, and even mid-ocean sea lanes are used for many applications. These applications often go unmonitored even though their use continues to increase. A network of underwater sensors could provide numerous monitoring services for these applications. Examples range from disaster monitoring to usage monitoring and conditions monitoring. These services are expanded below:

- The 2010 BP Oil Spill [1, 2], named Deepwater Horizon, is one extreme example of the sorts of disasters that can occur as waterway use increases. While engineering and legislative actions are necessary for accident prevention, a large network of sensors could provide early detection of problems as well as the monitoring of disaster progress.

- Monitoring of harbor use has also been proposed [3]. This type of system would be able to identify ships entering a harbor and report back with alerts on unauthorized access. Similar applications outside of harbor entries would be useful, but would require more sensors and may not have nearby infrastructure.

- Science applications, monitoring the conditions of areas of interest, should also not be overlooked. The changing climate and pollution have profound effects on both the physical and biological aspects of ocean systems. Underwater sensors can measure these effects along the water column.

Each of these services involves one or more nodes, called "sources," gathering data and providing that data to one or more other, remote nodes, called "sinks."

Technically, a network of wired sensors makes sense for all static applications, but it is often not feasible as this requires a permanent structure. Without a permanent structure submersed nodes must rely on other means to communicate; when Radio Frequency (RF) is impossible, other possibilities must be explored.

1.1.1 Underwater Communications

Most RF signals attenuate very rapidly when exposed to water. Extremely low frequency RF, while not as easily attenuated, requires very large equipment. Light Amplification by Stimulated Emission of Radiation (LASER) has been used with very high throughput at very short distances (less than 50 m) and seems better suited for Autonomous Underwater Vehicle (AUV)-AUV or AUV-Sensor communications. Future advances may make longer distances feasible, but the issue of aiming the LASER and keeping it aimed would likely not be feasible for small, battery-operated devices.

Wired communications provide the most reliability and throughput, but they tend to be the most expensive and have the greatest logistic requirements in deployment and maintenance. They are simple to understand, working in the same way as any dry network. Wired communications are certainly the most correct solution in some cases. In the Harbor Shield example, for fixed sensors with surrounding shore infrastructure, both power and communications can be wired for a permanent solution. In many other applications a wired network is practically impossible.

Underwater Acoustic (UWA) communications are the most reasonable for mobile or temporary, underwater nodes with no available surrounding infrastructure, but its technical limitations must be accounted for. In understanding UWA, we will compare the parameters of UWA against the well-known, short-range RF medium.

1.1.2 Underwater Acoustics versus RF

It is useful to spend some time discussing the differences between Underwater Acoustic Sensors and RF Through-Air Sensors. Nearly all of the difficulties in migrating from the popular field of RF Sensor Networks to the less common Underwater Sensor Networks can be reduced to the differences in throughput and propagation speed.

In [4], the physical-layer difficulties of Multipath, where a signal is received multiple times in an overlapping formation, are discussed. Catapovic also goes into detail about transmission loss (TL), but in this case the most extreme losses are at multi-kilometer ranges. Kilfoyle and Baggeroer in [5] propose a rate-range envelope of 40 km-kbps where, at a 1 km range up to 40 kbps is possible. Actual, low loss rates are much lower.

While Kilfoyle and Baggeroer may be correct about this upper limit, reliable throughput rates are often much lower. The Teledyne Benthos commercial line of Subsea Acoustic Modems [6] are readily available and support data rates of between 140 and 15,360 bits per second (bps), which is dependent on distance and frequency. Two case studies are also cited by Teledyne Benthos at moderate ranges of 1-2 km with throughputs of 1200 bps. Informally, this commercial modem is reliable at data rates of 1200-2400 bps and ranges of 1 km. In other words, the modem reaches a third of the rate envelope at best and only 6% of the envelope when performing reliably. Note that "reliable," as discussed here, means an acceptable bit error rate (BER) resulting in no dropped packets.

Rates available on RF networks vary widely by frequency and modulation. Many popular sensor nodes employ ZigBee, which can operate at bandwidths of up to 250 kbps [7], which is about 100 times the data rate of reliable underwater communications. The ZigBee specification is covered by IEEE 802.15.4 which also

allows for data rates of 40 kbps and 20 kbps. On average, we will use a data rate of 100 kbps.

The limit on transmission speed affects an application's bandwidth. On the other hand, the propagation speed will affect the application's utility. The speed of sound in water varies slightly, but is approximately 1500 m/s. RF propagation is approximately the speed of light, 3×10^8m/s, or 2×10^5 times faster than underwater acoustics.

UWA's much slower propagation will result in more overlap of messages at the receiving node. It also limits the types of data that can be sent across an acoustic network; real-time applications, or those with frequent updates, will run into significant latency issues. Reliable protocols such as Transmission Control Protocol (TCP) / Internet Protocol (IP) are also more difficult with greater latency. Sending, acknowledgment of, and resending a message can take so long that the message's information may have expired by the time it arrives. This is greatly dependent on the application and is a major design consideration to be discussed later in this thesis.

In all mobile communications devices, energy management is a great concern. Underwater Acoustics is very expensive from an energy management point of view. Transmit power is about 25 times that of listening and receiving at up to 50 W transmitting, 0.08 W listening, and 3 W receiving [8]. Contrast this with ZigBee sensors which are on the order of 1mW to 10mW transmit power. This difference not only drives the engineering of a solution, but also the size and cost of the devices.

In this thesis we will create a network simulator that is able to visualize both data rate and propagation speed. At this time, energy management will not be included in this study and will be left for future improvements.

1.2 Sensor Network Simulation

This section discusses the Open Systems Interconnection (OSI) and TCP/IP model, which is an important abstraction used to discuss network communication; some of the simulators that already exist; and the Visual Studio environment the simulator was built in. Some high level object oriented design will also be discussed.

1.2.1 The OSI and TCP/IP Models

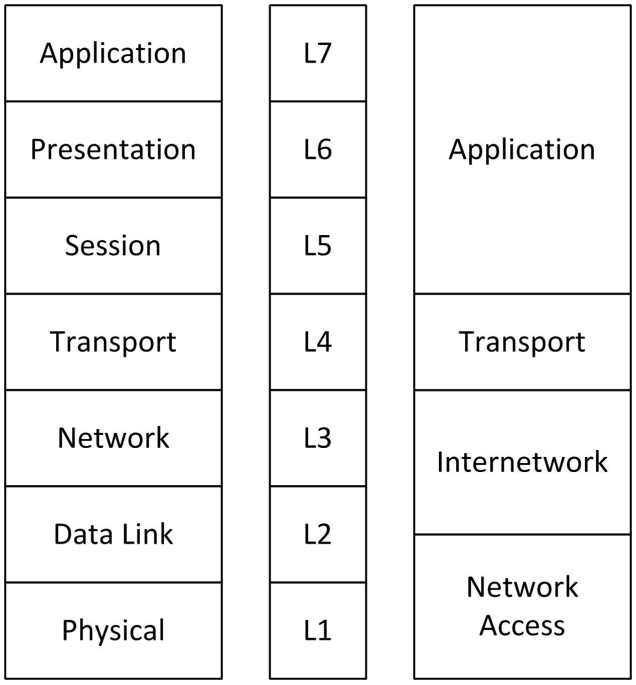

Figure 1. The OSI (left) and TCP/IP (right) Models

A network model describes a layered path for data communication. Each layer of the stack exists in at least two nodes. A given layer may generate data, which will communicate to the corresponding layer in another node. Such communication may only occur by traveling down the stack. For example, the Network Layer, Layer 3 (L3), often initiates communication to distribute routing information. This data gets encapsulated into the Data Link Layer, Layer 2 (L2), and is then converted into bits to transfer over the Physical Layer, Layer 1 (L1).

The OSI Model is deprecated, yet often referred to. Historically it was a protocol stack designed using open standards to compete with TCP/IP and other protocols. Little of this stack remains now (only the Network Routing protocol has been incorporated into the TCP/IP stack and is in use today). The model, however, is used to describe overall network concepts.

The TCP/IP Model is a modification of the OSI Model to fit better into how TCP/IP works. The Physical Layer is merged into the Data Link layer to form the Network Access layer. This is done primarily because network interface hardware tends to combine framing protocols with the physical Modulator-Demodulator (MODEM). The Internetwork layer takes over those parts of L2 that are not covered by hardware. The other change is that the unused Presentation and Session layers are removed to form the Application layer (the OSI stack had defined protocols covering these layers).

The OSI layers will be used in this thesis to discuss communication between nodes.

1.2.2 Sensor Network Simulators

Many network simulators exist, but none concentrate solely on the Underwater Acoustic Channel. There are a few that have modules that account for underwater acoustics. Notably, the ns2 simulator has been modified for underwater acoustics, as has the commercial simulator OPNET [9].

The ns2 simulator is a C++ and Objective TCL based simulator. It is open source and has extensive online documentation, but for protocol design and implementation it has a steep learning curve. Ns2 also has a visualizer called nam. Nam displays nodes and can show messages in transit, but this is a discrete animation better for showing message flow (path-based) rather than message interactions. This limitation makes sense for RF sensor networks because message in-air time

is negligible at the speed of light. The acoustic channel has been modeled with an add-on module to ns2 [10], but it has not made modifications to nam for visualization purposes.

OPNET provides an extensive pipeline between the radio transmitter to radio receiver, any stage of which is modifiable. Raysin, et al. in [11] modify the propagation delay, receiver power, background noise, interference noise, and Signal-to-Noise Ratio (SNR) modules in this pipeline in order to provide an underwater acoustic propagation model. This otherwise uses the OPNET visualization tools which show a message in transit, but do not animate packets in sim-time. OPNET combines a modified C++ language with graphical state machines for protocol implementation. OPNET's learning curve for protocol development is high, but somewhat less steep than with ns2. OPNET does provide a large suite of already implemented protocols and simulating them in a model network is relatively simple.

1.2.3 Visual Studio Environment

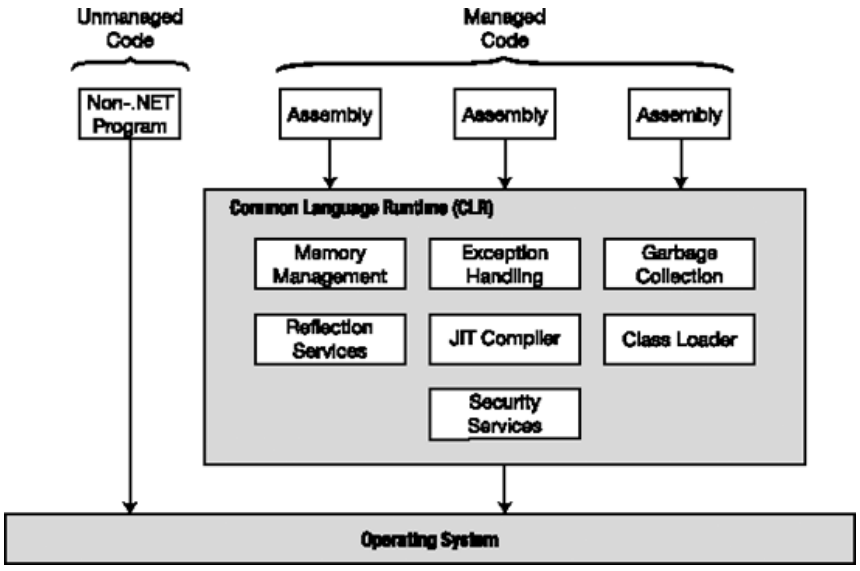

Figure 2. .NET Common Language Runtime [12]

The C♯ programming language is object oriented and easy to learn in comparison to ns2 and OPNET's design languages. C♯ employs the .NET Common Language Infrastructure (CLI) which contains the .NET Common Language Runtime (CLR). The .NET CLR manages a number of programming aspects that most other languages must include explicitly, such as memory management and garbage collection (freeing up memory), as shown in Figure 2. This means that, for the most part, the programmer does not need to be concerned with the background tasks .NET runs to support advanced programming techniques. One of the other benefits of this environment is that C♯, Visual Basic, and any other .NET enabled languages can be used to write third party libraries that anyone can use. Mersenne Twisters, a pseudo-random number generating algorithm, is one third party library used in the simulator described in this thesis. It is also possible to write wrapper classes so that many other programming languages can be used, resulting in a completely open architecture. The C♯ CLI also includes a Base Class Library (BCL) that contains thousands of managed classes: data structures like queues and stacks, mathematics, threads, graphics, and many others, which significantly reduces the burden of work on the programmer.

The Visual Studio Integrated Development Environment (IDE) allows graphical interfaces to be easily constructed. Windows Forms is used for typical user interaction and Windows GDI+ is used to paint in 2D to the screen, both elements of the BCL. The graphical environments are arranged on screen in the IDE and linked in the Forms event manager.

C♯ is a near-pure object oriented language. C♯ forces the use of interfaces and classes, unlike C++ which is more of a procedural language. One of the biggest benefits to object oriented programming is the use of well understood design patterns to perform common tasks. This makes for a quicker learning curve

compared with other languages. A well documented, object oriented simulator will provide an environment where the implementation of new, replacement modules is relatively easy.

1.2.4 Programming to Interfaces

Note: please refer to Appendix A for a description of the symbols used in diagrams throughout this document.

The simulator described in this thesis makes extensive use of interfaces. An interface can be referenced instead of a specific class, allowing that class to be replaced without refactoring all of the code. This allows the simulator to be modular.

One important use of interface programming is in the simple replacement of protocols. As long as a protocol is designed to the **INode** interface, the simulator will be able to deploy nodes using that protocol.

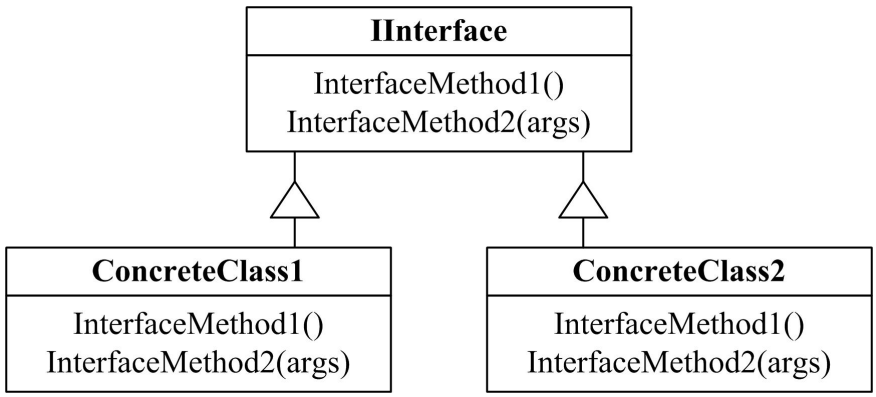

Figure 3. Two real classes that each implement the same interface.

Programming to an interface is very similar to its mechanical or electrical counterpart. A black box has a number of inputs and/or outputs that can be used by other objects. A programmatic interface can define a number of methods and/or variables. The interface description itself says nothing about the black box, these methods can be implemented in any possible way. A class that implements an

interface, however, is *required* to implement all the methods and variables described therein. This is a strict rule, and the class must adhere to the exact inputs and outputs of each method. The benefit comes when multiple classes implement the same interface. This is shown visually in Figure 3.

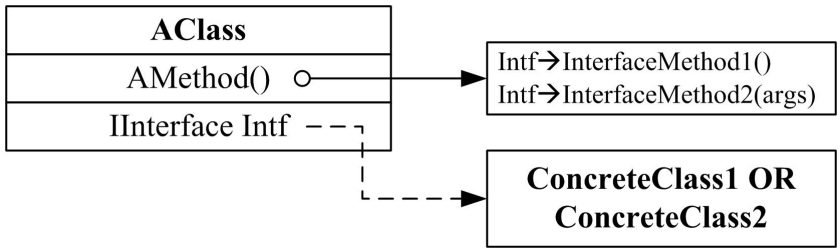

Figure 4. A class that references a real class through the use of a logical interface.

An object that uses this black-box need not know which class it is using. The utilizing object will have a reference to the interface, not the implementor, and can call the methods specified in the interface. This promotes modularity, as shown in Figure 4, as **ConcreteClass1** can be swapped for **ConcreteClass2** without changing the utilizing object.

1.2.5 Object Oriented Programming: Design Patterns Overview

Aside from basic modularity, Interfaces can be used to create Design Patterns. A Design Pattern is an object oriented algorithm for performing the same type of function without specifying the actual class that will be used. These prescribed cases appear frequently in different programming tasks.

One critical design pattern used in this thesis is the Abstract Factory pattern. This is defined in [13] as Figure 5. In essence, an object needs to create some number of other objects, but doesn't care what type of object they are. In the definition, the client may create either **Product1** or **Product2**; which depends solely on the Concrete Factory used. Referring back to interface programming, the client references the generic IAbstractFactory interface.

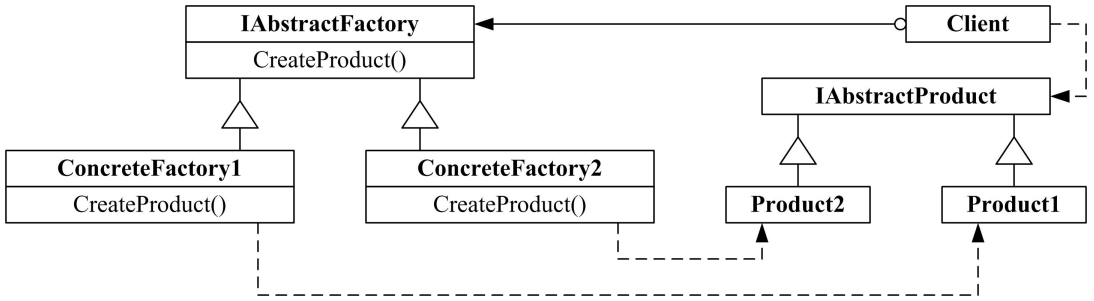

Figure 5. The Abstract Factory design pattern. Either Product1 or Product2 classes can be produced depending on the Concrete Factory used.

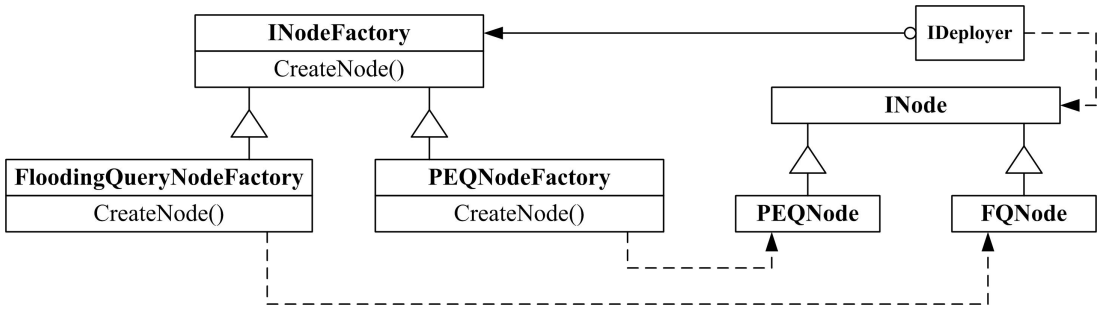

Figure 6. The Abstract Factory design pattern as used in the Modular Network Simulator to create nodes of a given protocol type.

Our important example is shown in the Figure 6 block diagram. In this case, we employ an **IDeployer** to create many Protocol Nodes. The **IDeployer** can be one of many deployment techniques: a random scatter, an Archemedian Spiral, a grid, or any number of user-generated classes. It would not be appropriate for a method of deploying nodes to be tied to a specific protocol; it's unlikely the two could have anything in common (aside from optimizing separation distances, and other deployment variables to a specific protocol). Thus, the IDeployer employs the INodeFactory to create many nodes. The specific INodeFactory that is assigned to the Deployer determines the type of node deployed. This can be determined at run-time through the user interface.

1.3 Problem Statement

The current state of underwater acoustic network simulation is lacking in a few elements. Foremost, no visualization tool exists that animates messages in-transit showing collisions and direction. Protocol implementation is difficult in existing simulators and while a panacea is not proposed, the result of this work will be object oriented and modular, allowing any of its parts to be replaced without modifications to any other part.

The simulator created for this thesis will be demonstrated with two protocols. The first protocol floods most of its network traffic and exists as a basic example of the simulator's functionality. The second is an implementation of a protocol from the literature. Most of the discussion and results will involve the second protocol.

Our first discussion is about the architecture behind the simulator and visualizer and about the protocols that will be presented. We will then move on to the results from the presented protocols. These results will be discussed in more depth and an overview of the visualizer will be presented. We will wrap up with a discussion on future work and possible improvements to the simulator.

List of References

[1] C. Cleveland, "Deepwater horizon oil spill," Encyclopedia of Earth, February 2011. [Online]. Available: http://www.eoearth.org/article/Deepwater_Horizon_oil_spill

[2] C. Robertson, "11 remain missing after oil rig explodes off louisiana; 17 are hurt," April 22, 2010 2010. [Online]. Available: http://www.nytimes.com/2010/04/22/us/22rig.html

[3] L. Faulkner, R. Granger, P. Hurst, W. Jankowski, D. Steinbrecher, and J. Tattersall, "Harbor Shield: A new technique for inspection of vessels below the waterline"," in *IEEE Conference on Technologies for Homeland Security (HST'09)*, 2009, pp. 221–226.

[4] J. A. Catipovic, "Performance limitations in underwater acoustic telemetry," *IEEE Journal of Oceanic Engineering*, vol. 15, no. 3, pp. 205–216, 1990.

[5] D. B. Kilfoyle and A. B. Baggeroer, "The state of the art in underwater acoustic telemetry," *IEEE Journal of Oceanic Engineering*, vol. 25, no. 1, pp. 4–27, 2000.

[6] "Telesonar underwater acoustic modems: Subsea wireless communication," 2007, teledyne Benthos. [Online]. Available: http://www.benthos.com/acoustic-telesonar-modems-undersea-sub-sea.asp

[7] L. Gang, B. Krishnamachari, and C. S. Raghavendra, "Performance evaluation of the IEEE 802.15.4 MAC for low-rate low-power wireless networks," in *IEEE International Conference on Performance, Computing, and Communications*, 2004, pp. 701–706.

[8] A. Harris III, M. Stojanovic, and M. Zorzi, "When underwater acoustic nodes should sleep with one eye open: idle-time power management in underwater sensor networks," in *Proceedings of the 1st ACM international workshop on Underwater networks (WUWNet'06)*, 2006, pp. 105–108.

[9] OPNET, "Application and network performance with OPNET." [Online]. Available: http://www.opnet.com/

[10] A. Harris III and M. Zorzi, "Modeling the underwater acoustic channel in ns2," in *Proceedings of the 2nd international conference on Performance evaluation methodologies and tools*, 2007, p. 18.

[11] K. Raysin, J. Rice, E. Dorman, and S. Matheny, "Telesonar network modeling and simulation," in *Proceedings of MTS/IEEE OCEANS '99*, vol. 2, 1999, pp. 747–752.

[12] D. Solis, *Illustrated C♯ 2008*. Apress, 2008.

[13] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, 1994.

CHAPTER 2

Methodology

This chapter outlines the architecture used to design a discrete, wireless network simulator with an animated visualizer, called the Modular Network Simulator (MNSim). It also describes the operation of two example protocols used to demonstrate and verify the simulator. Note that the simulator and visualizer are two distinct parts and will be discussed as such, but they are integrated into one computer program.

2.1 Simulation Setup

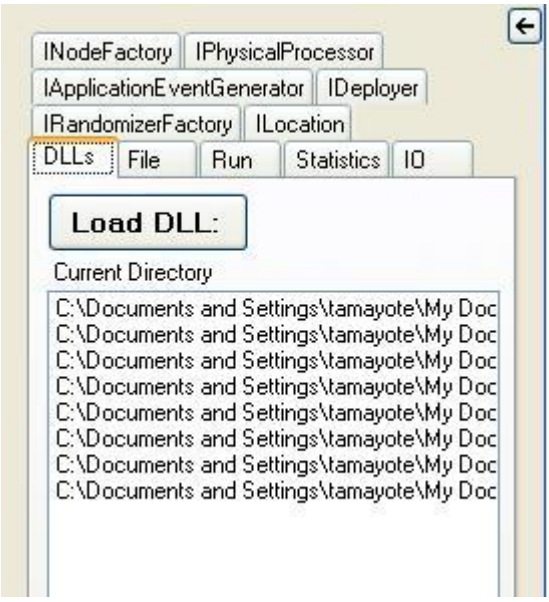

Figure 7. Simulator Set-up: Tabs

This simulator aims at a modular design. Some of these modules require user input, which can be customized based on which module implementation is loaded. When the visualizer is started, some core modules, presented as DLL files, are installed automatically. Additional modules can be loaded by the user

in the DLL tab. This and the other customizable parts are included in the tabs shown in Figure 7. The File tab is meant for saving and restoring settings, but is not yet implemented. The DLLs, Run, Statistics, and IO tabs are integral to the visualizer and are not a direct part of the simulator. While the visualizer aspect of the simulator is a core part of this thesis, it is not essential to the simulator itself and can be considered one of the replaceable modules.

The remaining tabs will be discussed individually.

2.1.1 ILocation Interface

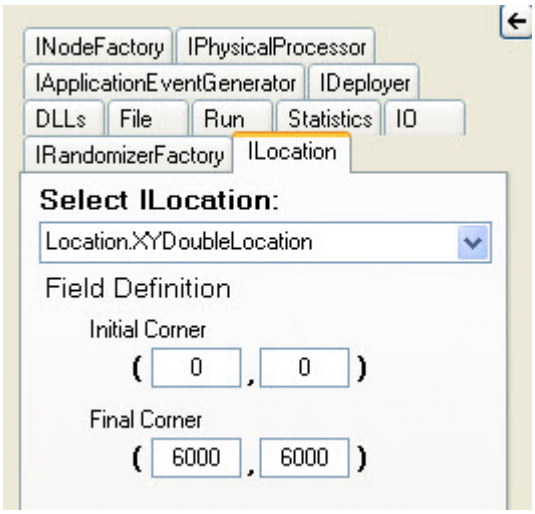

Figure 8. Simulator Set-up: Location Tab

The ILocation tab, shown in Figure 8, defines the coordinates type, which are currently limited to Cartesian coordinates without depth. This is also where the field is defined. The default field is a square area ranging from 0 to 6000 meters. The maximum size of the field is limited only by the double precision floating point number as described in IEEE 754-2008 (as implemented in C♯).

2.1.2 IRandomizerFactory Interface

The IRandomizerFactory tab is an Abstract Factory that creates random number objects. Currently, the only implemented randomizer is the Mersenne Twister,

using open sourcecode from CenterSpace Software, LLC [1]. A separate randomizer is provided for each node with a base randomizer used to set the seed of subsequent randomizers. This helps stabilize the replay of known seeds; tweaking protocol settings, for example, will not affect the location of the sink or sensor events.

2.1.3 INodeFactory Interface

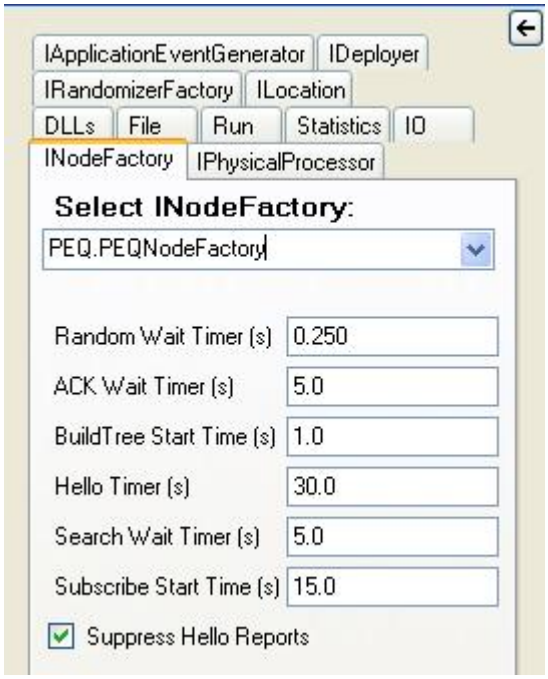

Figure 9. Simulator Set-up: Node Factory Tab

The INodeFactory is the abstract factory discussed in Section 1.2.5. The factory choice determines which protocol will be used during the simulation. The only options are PEQ and Flooding Query Node (FQN), both of which will be discussed later. Some variables can be set for PEQ, as shown in Figure 9, but none are available for FQN.

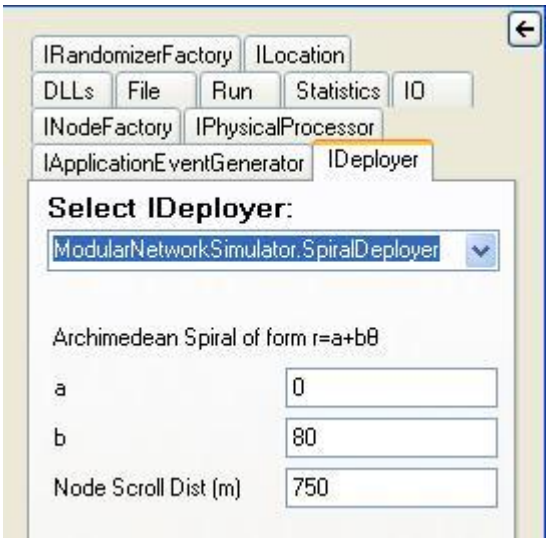

Figure 10. Simulator Set-up: Deployer Tab

2.1.4 IDeployer Interface

The IDeployer is only used during simulation set-up. As discussed in the introduction, the Deployer accesses a Node factory to create some user determined type of node. The user chosen IDeployer determines how nodes are placed in the simulation field.

Four options for deployment are available, and many other possibilities exist. Two options include deployment of nodes along a fixed grid, and random deployment (only the number of nodes and minimum separation are given). The third option, shown in Figure 10, shows an Archemedean Spiral, given by the equation (in polar coordinates):

$$r = b\theta \tag{1}$$

The desire was to have the nodes a set distance apart along the spiral, so knowing the scroll length is necessary. An approximation is derived from the separation, which is constant in Archimedes' Spiral. Let the separation s, times the number of turns n be an approximation for the radius. The area is approximated

by:

$$A = \pi s^2 n^2 \tag{2}$$

The area is also given by the scroll length l times the separation s. Setting the two equal,

$$ls = \pi s^2 n^2 \tag{3}$$

$$l = \pi s n^2 \tag{4}$$

From Equation 1, we can determine that b represents the thickness. At $\theta = 0$, $r = 0$ and at $\theta = 2\pi$, $r = 2\pi b$. We can also represent n as $\frac{\theta}{2\pi}$. Therefore,

$$l = \frac{b\theta^2}{2} \tag{5}$$

We will know l, so solving for θ,

$$\theta = \sqrt{\frac{2l}{b}} \tag{6}$$

Returning back to the configuration, the a value increases the radius at $\theta = 0$. While this deployment is the result of an approximation, it is enough for the purposes of the simulator.

It would also be advantageous to have a random distribution of the nodes along the spiral. It seems likely that buoys would be dropped with some random variation to b as well as a random distance between nodes along the spiral. This bears more investigation but is not within the scope of this thesis.

A fourth deployer, the PEQTestDeployer, was created to match the deployment used in [2] to better compare RF versus UWA. This is equivalent to the RandomDeployer except for five additional nodes situated equidistant along the left side of the field and one additional node set at the center of the right side of the field.

The right-hand node is set to be the sink node. The left hand nodes operate in conjunction with the PEQTestApplication to force events to occur at their

locations. These five nodes then become sources and will initiate five simultaneous events via the PEQTestApplication discussed in Section 2.1.5.

2.1.5 IApplicationEventGenerator Interface

The basic Application Event Generator creates an area where a supposed event occurs. Any node within the affected area receives an application layer message to be processed by the node. These events can be set to recur at specific intervals. Currently the event is static, but events that move over time are certainly possible.

The PEQTestApplication, mentioned in Section 2.1.4, is built off this basic Application Event Generator, creates five simultaneous events as discussed and can recur at specific intervals.

2.1.6 IPhysicalProcessor Interface

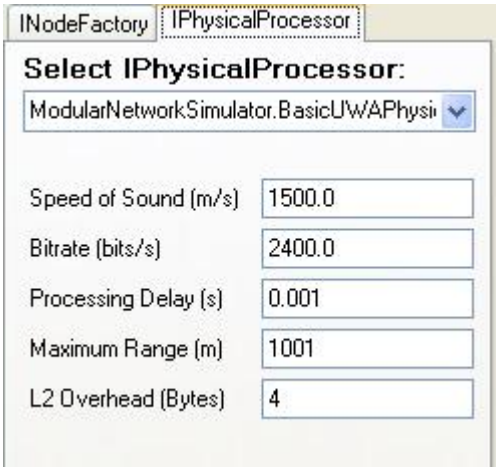

Figure 11. Simulator Set-up: Basic UWA Physical Processor Tab

The Physical Processor was designed as a Physical Layer model. Two such models were created for this simulator, the first being a very basic model of the underwater acoustic medium, whose characteristics are data rates on the order of thousands of bits per second (kbps) and latency on the order of seconds.

The configuration tab for this is shown in Figure 11. The limit to this system

is the maximum range, which is user defined.

A second Physical Processor was developed to include a simple statistical model of the underwater acoustic medium. This model includes Thorp's approximation for absorption, defined as:

$$A(f) = 0.11\frac{f^2}{1+f^2} + 44\frac{f^2}{4100+f^2} + 2.75 \times 10^{-4} f^2 + 0.003 \tag{7}$$

where f is given in kHz and $A(f)$ is the absorption in dB SPL per km. At low distances and low frequencies, this is minimal and transmission loss is dominated by spreading losses, the total given by:

$$TL = k \cdot 10 \log d + \frac{d}{1000\frac{\text{m}}{\text{km}}} A(f) \tag{8}$$

with d given in meters and where k is a spreading factor. [3]

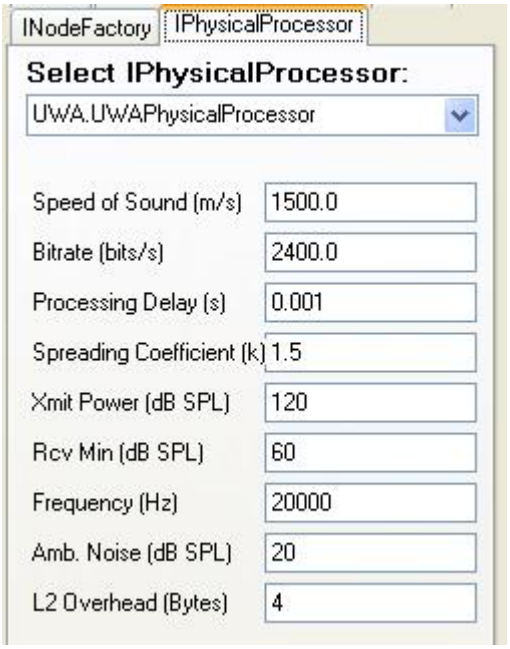

Figure 12. Simulator Set-up: UWA Physical Processor Tab

In Figure 12, the variables to set the above equations should be clear. Speed of sound and bitrate are required as in the Basic UWA Physical Processor. Also settable are the Spreading Coefficient, k, as described above, the transmit and

minimum receive power levels in dB SPL, frequency in Hz, and ambient noise in dB SPL. The sonar equation is used:

$$SL - TL \geq NL + DT \tag{9}$$

where SL is the source level, NL is the noise level, and DT is the detection threshold, [4] to determine reception power levels and interference levels.

The Physical Processor also shares some L2 responsibilities with the Node. Ultimately, the design of the physical layer models will lead to a future expansion of the simulator to include a separate L2 module responsible for the simulation of a modem. This may have the added benefit of allowing the simulator to be used for other physical media and L2 network types such as Frequency Division Multiple Access networks.

Without this additional L2 module, the UWA Physical Processor has problems mixing signal levels for each individual node. When receiving multiple acoustic signals, each additional signal can be seen as contributing to the noise. Thus, only the loudest signal is not masked. This loudest signal might or might not meet the detection threshold of the modem.

In the simplified model, all signal overlaps are considered collisions. This is a stricter analysis; any model that works in this type of analysis will also work in a simulation with greater fidelity. The rest of this thesis will focus on the simplified model. This thesis also distributes nodes at similar distances, so a true collision would usually occur when mixing signals.

Additional guard time is also not implemented, though multipath can increase the noise floor for some time after transmission is complete. Additional L2 overhead bytes are included and can account for this period of additional noise.

Future work and proposed architectures are discussed further in Chapter 6.

2.2 Simulator Operation

2.2.1 Simulator Network Model

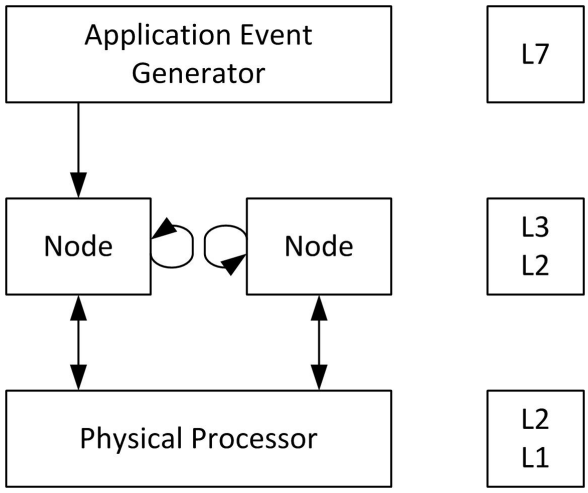

Figure 13. Modular Network Simulator Model

Figure 13 shows the high-level interactions between simulator components. It also shows many-to-one and one-to-many relationships between components. In most cases there is a single Application Event Generator and a single Physical Processor. As the Node's communication is under test, many are required.

The Node typically generates most traffic, at least until an Application Event occurs and is heard by the Node. The Node may also generate traffic to itself, either to be processed immediately or at some future time. To send messages to other Nodes, the Message Event is sent to the Physical Processor for processing.

Layer 2 activities are performed at both the Node and Physical Processor. For example, the Node is responsible for all collision detection. If the Node is busy (sending or receiving), and a new message begins to be received, a collision occurs. If the Node was sending, the message does still get sent as collisions in the Underwater Acoustic medium do not affect transmission.

2.3 Visualizer Architecture

The simulator visualization is distinct from the core simulation. The entire simulation is allowed to complete before visualization begins. There are three core parts that connect the simulator to the visualizer and either bypass the graphics or replay the simulation.

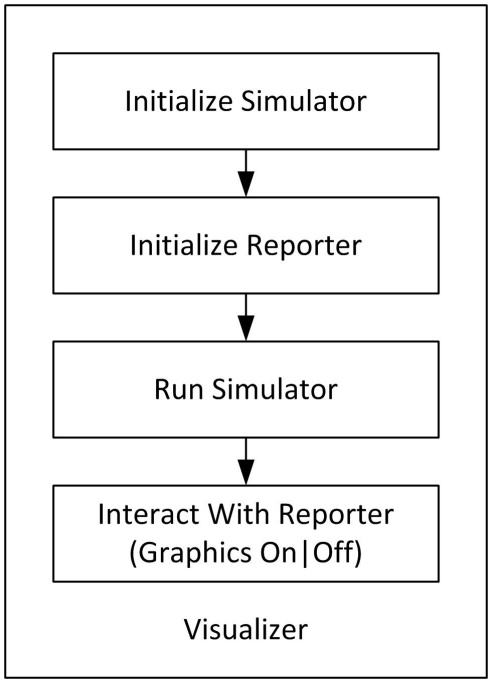

Figure 14. Visualizer High Level Flow

The following sections explain the flow shown in Figure 14.

2.3.1 Network Simulator

The Network Simulator's operation is discussed in depth in the previous sections. The Visualizer is responsible for configuring the network simulator's parts and stitching them together into a viable system.

2.3.2 Reporter

The Reporter provides the interface between the Simulator and Visualizer. The Reporter deals in IReport objects which fall into two classes: Informational

Reports and Graphical Reports. Informational Reports provide output in text or Comma Separated Value (CSV) formats. This could, for example, provide timestamps of each collision that occurred.

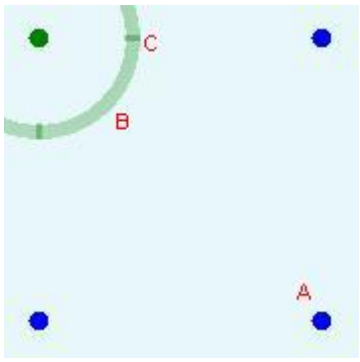

Figure 15. Examples of Circles (A), Annuli (B), and Lines (C)

Graphical Reports provide information back to the Visualizer about what type of graphics to display. These reports can be suppressed (for bulk simulation), but they are the primary means of using the Visualization capabilities of the overall simulator. These come in three major types: Line, Circle and Annulus. Typically, lines will be drawn to represent viable messages between two nodes (these messages may collide, but in a 2-node system, they would likely reach their destination). Circles are drawn for nodes and application events, and Annuli are drawn to indicate the entire wavefront of a message. In Figure 15, for example, the green node (circle) has sent a message whose wavefront is represented by the annulus (B). This will be received by all blue nodes except at (A), as represented by the darker lines (C). The node at (A) does not have closure (it is too far from the source).

The reporter has the ability to feed reports to the Visualizer in either the forward or reverse direction in time. It keeps a record of already viewed events and it has the ability to jump to any simulation time, which is controlled by the visualizer. There are two major types of graphics reports: static and instanta-

neous. The instantaneous graphics reports occur only in one particular discrete time frame. Static reports are cached and are redrawnin each frame. A node does not need to send a graphical report for each frame; it sends a single static graphical report and, should it die, a single "stop" action. The stop action has a pointer back to the initial "start" report, and the reporter is able to remove it from the cache when going in the reverse direction. Thus, simulated time travel is possible.

2.3.3 Visualizer

The Visualizer begins in the non-running state. The visualizer operates using a timer that expires every $1/F$ seconds where F is the number of frames per second (a setting in the main visualizer GUI). Each time the timer expires, it runs a method that draws an image to the visualization box in the GUI, then clears the buffered images. When in the non-running state there are no means to populate the buffered image, thus it maintains a cleared screen.

The user can select which modules to use and their configurations. Once the user presses the "Run Simulation" button or the Play button, Simulator set-up begins.

Simulator set-up involves instantiating the different modules and configuring them with the user supplied settings. There is a somewhat specific order required because some modules are used in the configuration of other modules For example, many modules make use of the IRandomizerFactory, which is in charge of handing out random variables, therefore the IRandomizerFactory must be instantiated first. Part of the configuration also involves running methods within those modules; for example, node deployment is run with the command "deployer.Deploy()."

The Visualizer waits while the simulator is running, and upon completion, starts the visualization phase of the simulator. This adds a method to the draw timer discussed above. This method keeps track of the sim-time and pulls all of

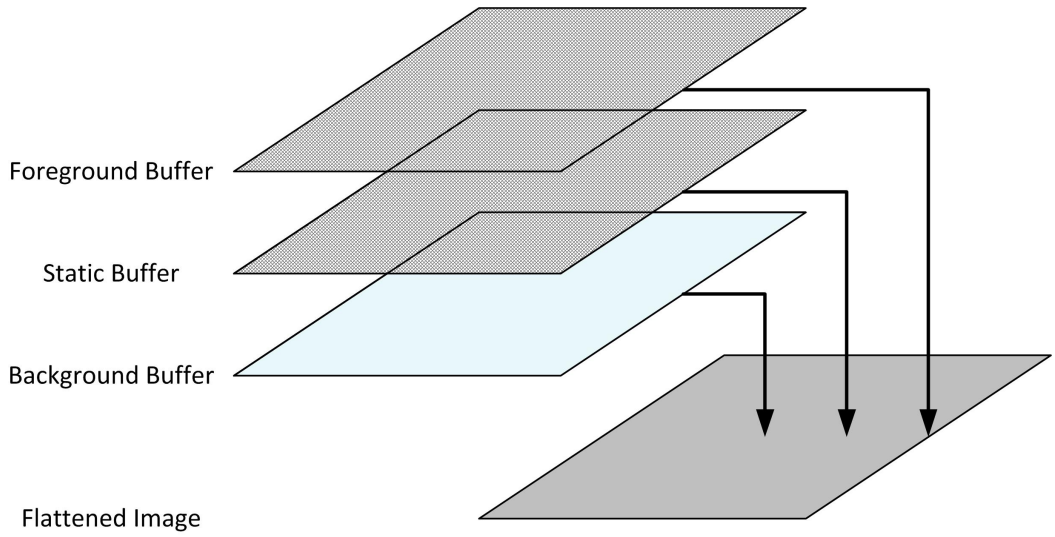

Figure 16. Transparent layers with opaque background are flattened for drawing

the graphical reports for that simulated time period from the reporter. These are drawn to the layered image buffers shown in Figure 16. The Foreground and Static buffers are transparent, the Background opaque. Any objects can be drawn to either the Foreground or Background, though most are drawn to the Foreground. The Static Buffer holds graphical reports that do not often change (such as nodes that have a static location). These reports are held in a special location and should an additional static report be added, one removed, or modified all of these reports will be redrawn to the static buffer. The static buffer only gets cleared under special circumstances such as stopping the visualizer or when it is redrawn.

What this results in is a composite image where Static objects (nodes) appear above Background objects (most often application event visuals) and below Foreground objects (the message visuals displayed in Figure 15).

The limitations of the visualizer are that, when a lot of Graphical Reports are present in any given timeslot (i.e. too many messages are active at any given time), the visualization slows down significantly. This can be adjusted by manually changing the frame rate (frames per second) before simulation. No ability to skip

frames or modify the frame rate during simulation exists. XNA, a Microsoft 3D gaming package, has the ability to drop frames to maintain graphics timing. It was initially explored, but was rejected due to the difficult learning curve of graphics optimization.

2.4 Flooding Query Protocol

The Flooding Query Protocol (FQP) was designed as a simple test of the simulator. It involves one sink node and one or more sources. Queries are flooded by the sink out to the network, which entails each node sending a received Query message as a broadcast unless the Query has already been heard. Data is flooded back unless a route is known. No action is taken for node failure and no acknowledgments are used, so the protocol is not viable for a real underwater acoustic network.

To detail this process:

1. The sink node initiates a Query message, indicating its desire to receive information.

2. The Query is propagated by a node the first time it hears the message from a given neighbor. Neighbor relationships are formed by Hello messages, sent every 30 seconds, but this is not a rigorous process and could be omitted. Currently, received Hello messages are dropped and are included to illustrate collisions.

3. The Query message received with the lowest hop count is used by the node to form a route back to the sink.

4. When an event occurs, nodes in "listening" range of the event send a Data message along their known path back to the sink. If no path is known, the message is flooded to neighbor nodes.

5. Each Data message continues throughout the network until it has reached the sink. The Data message can be split due to the forwarding process, so the sink will receive more than one of certain messages.

While this protocol is largely illustrative, it provides a simple and effective method of getting data back to the sink node.

2.5 Periodic, Event-Driven, and Query-Based Protocol [2]

A second protocol was chosen for its simple implementation and potential use in an UWA environment. Boukerche et al. in [2] present such an event based protocol.

Periodic, Event-Driven, and Query-Based (PEQ) protocol is periodic in name, but the actual workings of the protocol has no dependency on periodicity. It does, however, fit well into the event-driven, query-based problem space employed by FQP.

A PEQ network operates in three basic modes: tree building, sending data, and path repair. Some or all nodes may be employed during each of these modes. The first mode, tree building, has two parts: building a hop tree, and propagating Subscriptions. Both of these processes require the sink to flood a message to the entire network. A Build-Tree message carries information about the number of hops back to the sink. Upon completion of the Build-Tree, each node knows its own next-hop along the best path back to the sink, as well as the number of hops along that path. A Subscription message floods information regarding what sort of data the node is interested in. In the Subscription a coordinate attribute can also be set, potentially reducing the amount of flooding that needs to be done, but this is not outlined fully by Boukerche et al. and is not included in this simulation.

The sending data mode of PEQ occurs when a source node has information that meets the criteria of the query. When this occurs, the source finds the sink

node that is assigned to the matched Subscription to the best-path route to that sink. Data is sent to the next-hop node in this route. The receiving, next-hop node sends an Acknowledgement (ACK) message back to the previous-hop sender. Boukerche et al. also describe a reverse-path route that gets stored by each node along the data's path, allowing the sink to communicate back to the source; this functionality is not included in the MNSim implementation of PEQ.

If the sending node does not receive an ACK back from the next-hop node, the node starts the path repair process. This process is initiated when the sender broadcasts a Search message. All nodes in listening distance send a Response message containing their hop-distance to the sink. Boukerche et al. say that, "if any neighbor does not reply to the Search message, the sender node has to retransmit this message." This is not possible as multiple neighbors (existing in the neighbor table) may have failed. It may be that the neighbor table is run under some other process, but this is not described in the article. Therefore, the MNSim application of PEQ limits the Search message retries to 2. Once all Response messages have been received, or in the case of the failure described, the node chooses the best count to use as its next-hop route to the sink. The authors do not mention how hop counts are updated beyond the repaired path away from the sink. It is likely that more hops are required to get around the obstacle, but even the original paper has a diagram with unaltered hop counts far upstream. This will not be a problem given only one failure, but multiple failures will decrease the overall network stability.

Boukerche et al. do not give a complete specification for their protocol nor is any source code available. Therefore, some basic assumptions need to be made. In general, neighbor discovery occurs in a network protocol; Boukerche et al. assume this has already been completed. Our primary assumption is that Hello messages

are used for neighbor discovery in the MNSim implementation. Tables used in the protocol and message contents are mentioned by Boukerche et al., but no details are provided; the MNSim implementation provides many of these and other small implementation details.

PEQ will be discussed in more detail in the following chapters.

List of References

[1] C. S. LLC, "Free C♯ implementation of the Mersenne Twister algorithm," 2003. [Online]. Available: http://www.centerspace.net/downloads/free-stuff/

[2] A. Boukerche, R. W. N. Pazzi, and R. B. Araujo, "A fast and reliable protocol for wireless sensor networks in critical conditions monitoring applications," in *Proceedings of the 7th ACM international symposium on modeling, analysis and simulation of wireless and mobile systems (MSWiM'04)*, 2004, pp. 157–164.

[3] S. Joshy and A. Babu, "Capacity of underwater wireless communication channel with different acoustic propagation loss models," *International Journal of Computer Networks & Communications*, vol. 2, no. 5, pp. 192–204, 2010.

[4] L. Kinsler, A. Frey, A. Coppens, and J. Sanders, *Fundamentals of Acoustics*, 4th ed. John Wiley & Sons, Inc., 1999, pages 448-449.

CHAPTER 3

Results

3.1 Protocol Visualization

One major purpose of this thesis is to create a visual environment for the evaluation of communications in the UWA space. Unlike RF communications, the medium lends itself to visual observation. Typical results in numeric format will be less useful to our comparison of UWA and RF. Instead, a discussion of how this visualization will be used, as well as some of its various properties, will make up the bulk of the results for this thesis.

3.1.1 Visualization Use

To discuss how visualization can be used, the PEQ protocol will be stepped through from deployment of nodes to its failure point. Once the simulation has finished running, the field and the nodes it contains are displayed as shown in Figure 17. The sink node is green rather than blue and is marked by the arrow in the figure (marking arrows are included manually and will be noted in the discussion).

As discussed in Section 2.5, the sink begins by sending a Build Tree message. The Build Tree message propagates, flooded out from the sink node until all nodes that have heard a Build Tree send their own out as shown in Figure 18. In the figure, three Build Tree messages are shown as green annuli with darker green lines indicating the nodes that will receive the message. One node is actively receiving a Build Tree message, indicated at the node in yellow. No node will send more than one Build Tree message. The Build Tree is shown as a green annulus. (This color will overlap with Response messages, sent in response to the Search message).

Each node that has a route back to the sink shows its route with an arrow

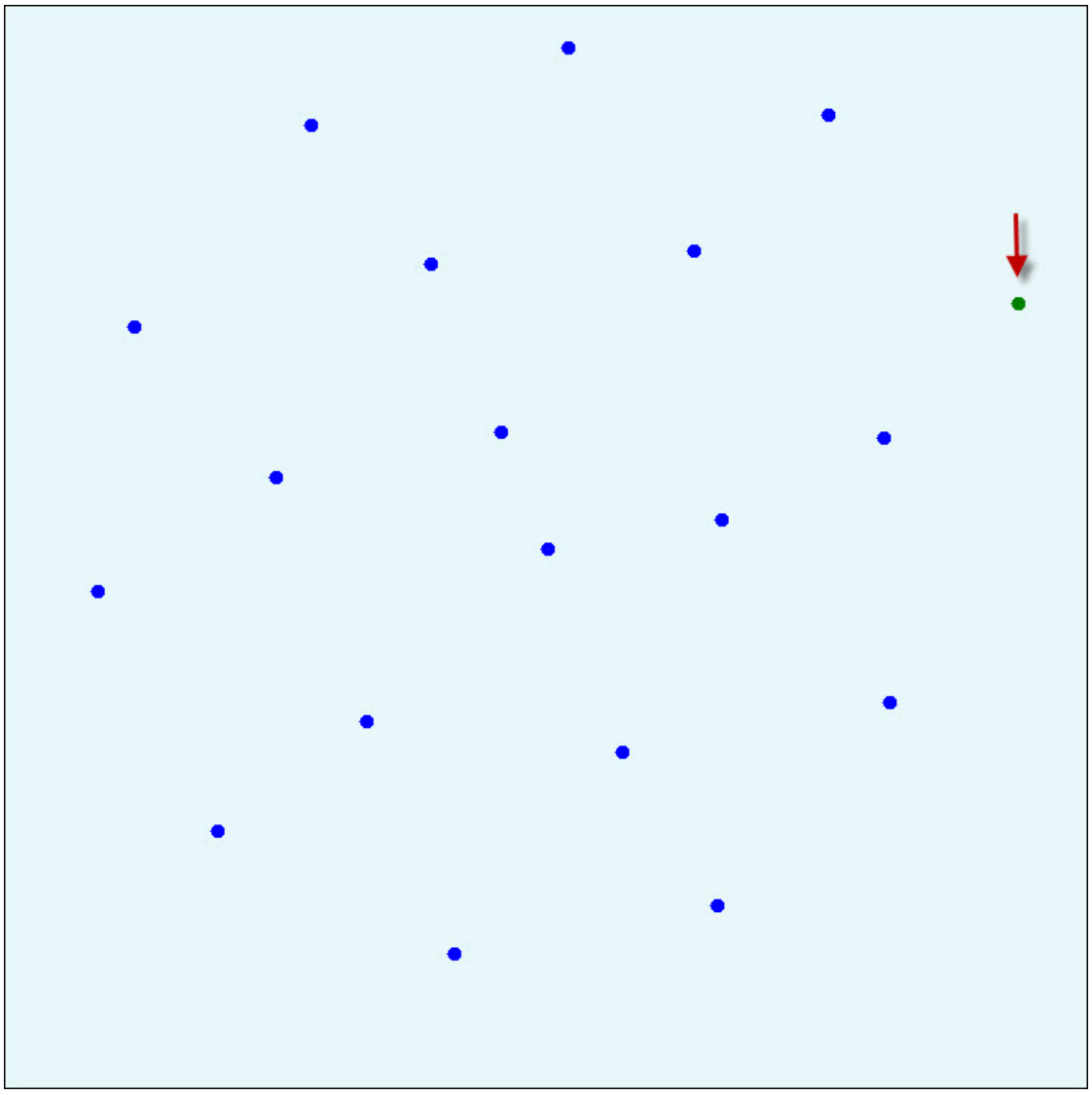

Figure 17. Nodes, sink (green) marked by arrow

pointing to the next-hop node. If the only available route is from a Build Tree message, the route arrow is shown in green. The purpose of the Build Tree is to provide a base route tree back to the sink. At this point, in Figure 19, the routing tree can be observed via the arrows pointing along the path back to the sink.

Shortly after the Build Tree message has been sent, the sink sends a Subscription message indicating its interest in a certain type of information (or from a certain location). In our case, we have only one type of interest and all interests

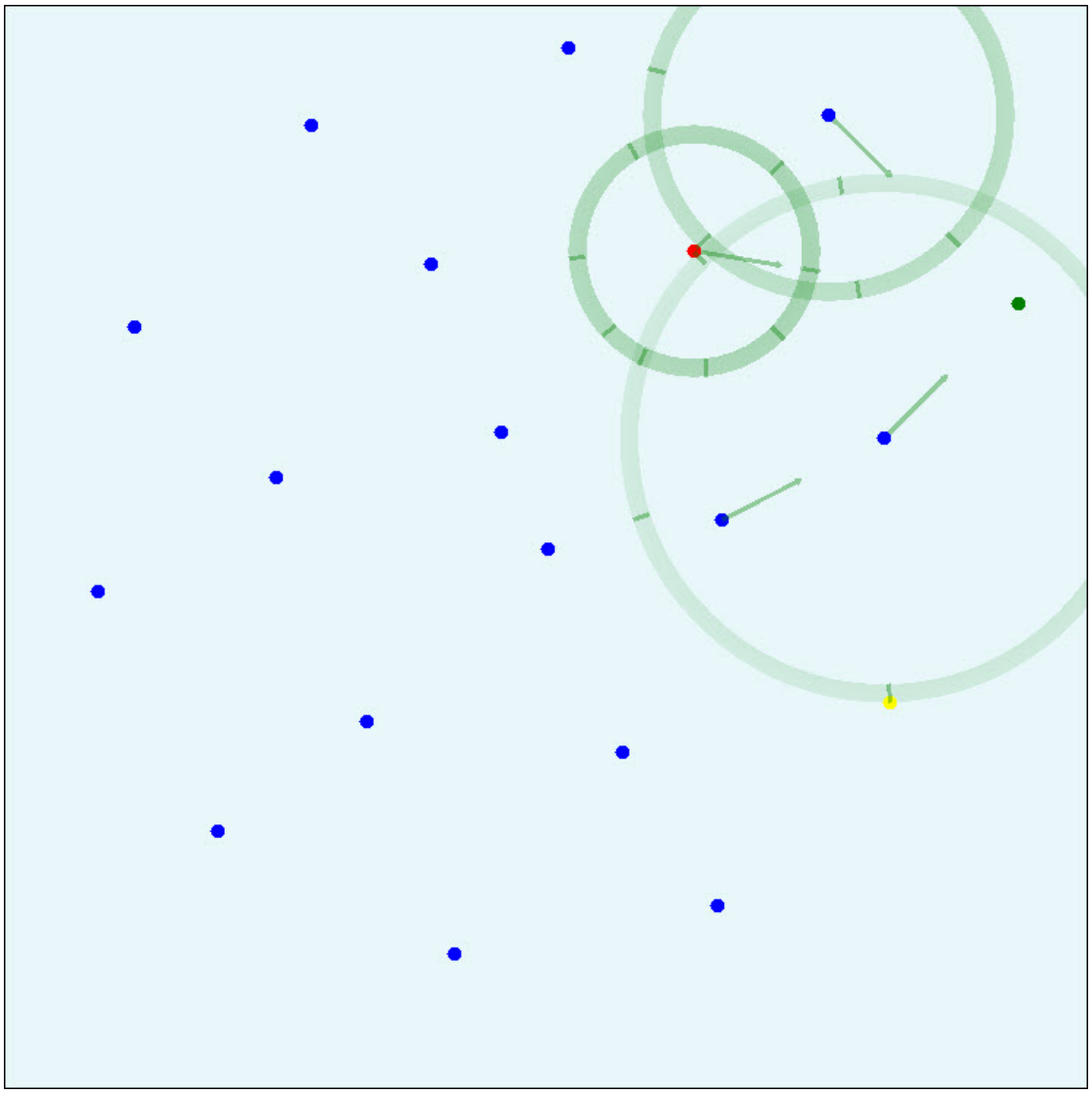

Figure 18. Build Tree message propagates

exist on a global scale. The Subscription message is shown as a red annulus, as shown in Figure 20.

As the Subscription propagates, it also carries the hop count back to the sink. This protocol gives routing precedence to the most recent message (given equal hop counts). To differentiate between the Build Tree and Subscription messages, the route arrow will change to red if learned from a Subscription. This is shown in Figure 21. This also means that the route can change, either due to a better

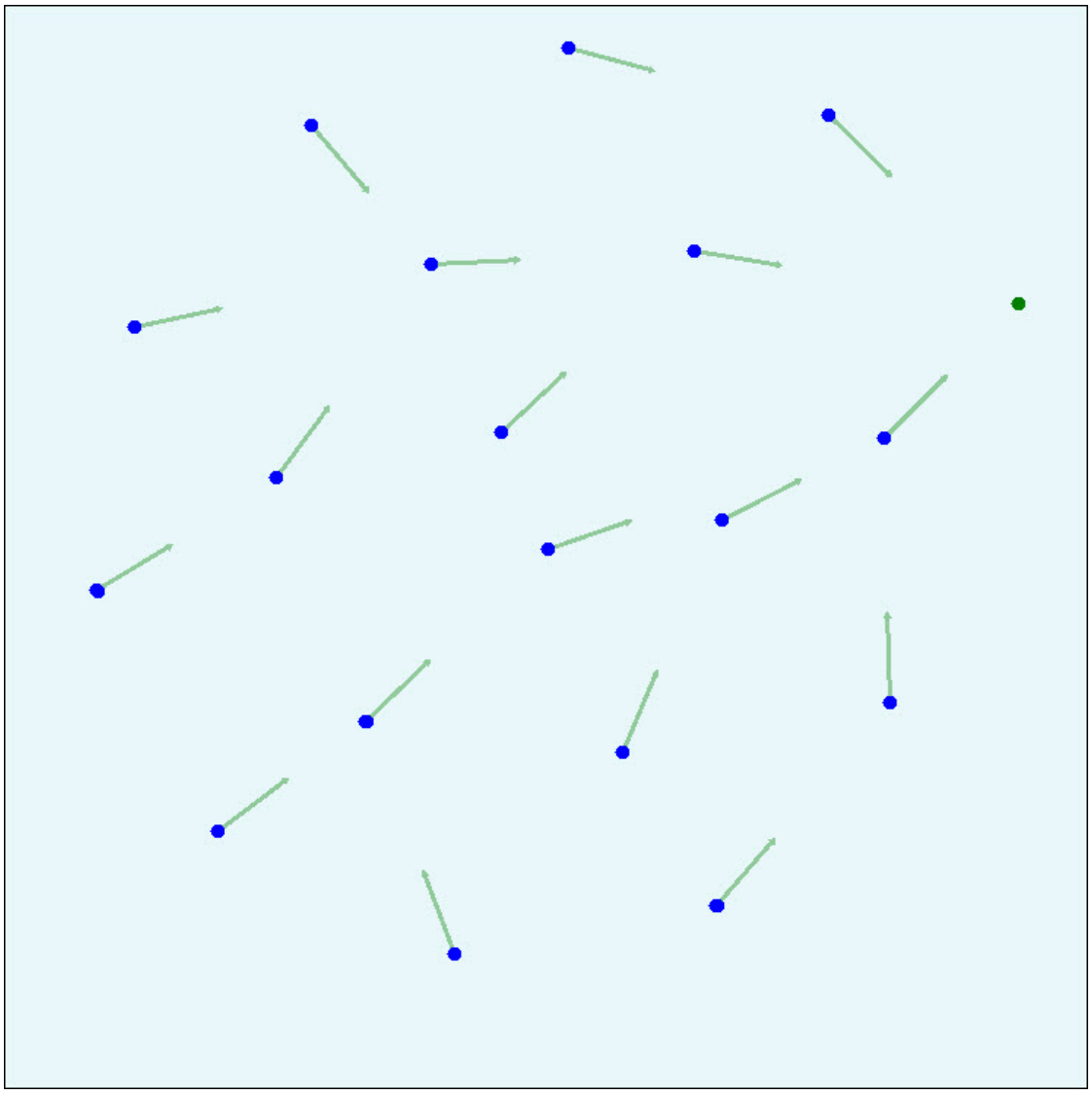

Figure 19. Build Tree complete with lowest-hop path back to sink

hop count or from the same hop count received from a different node. An example of this change happens between Figures 21 and 22 where the bottom-most node adjusts its route. In this case it moves to a better hop count because the Build Tree from the new next-hop neighbor was lost due to a collision (routing messages are not sent reliably).

In Figure 22, the Subscription has propagated completely and an event, displayed as the red circle in the background image, has occurred. This is a very large

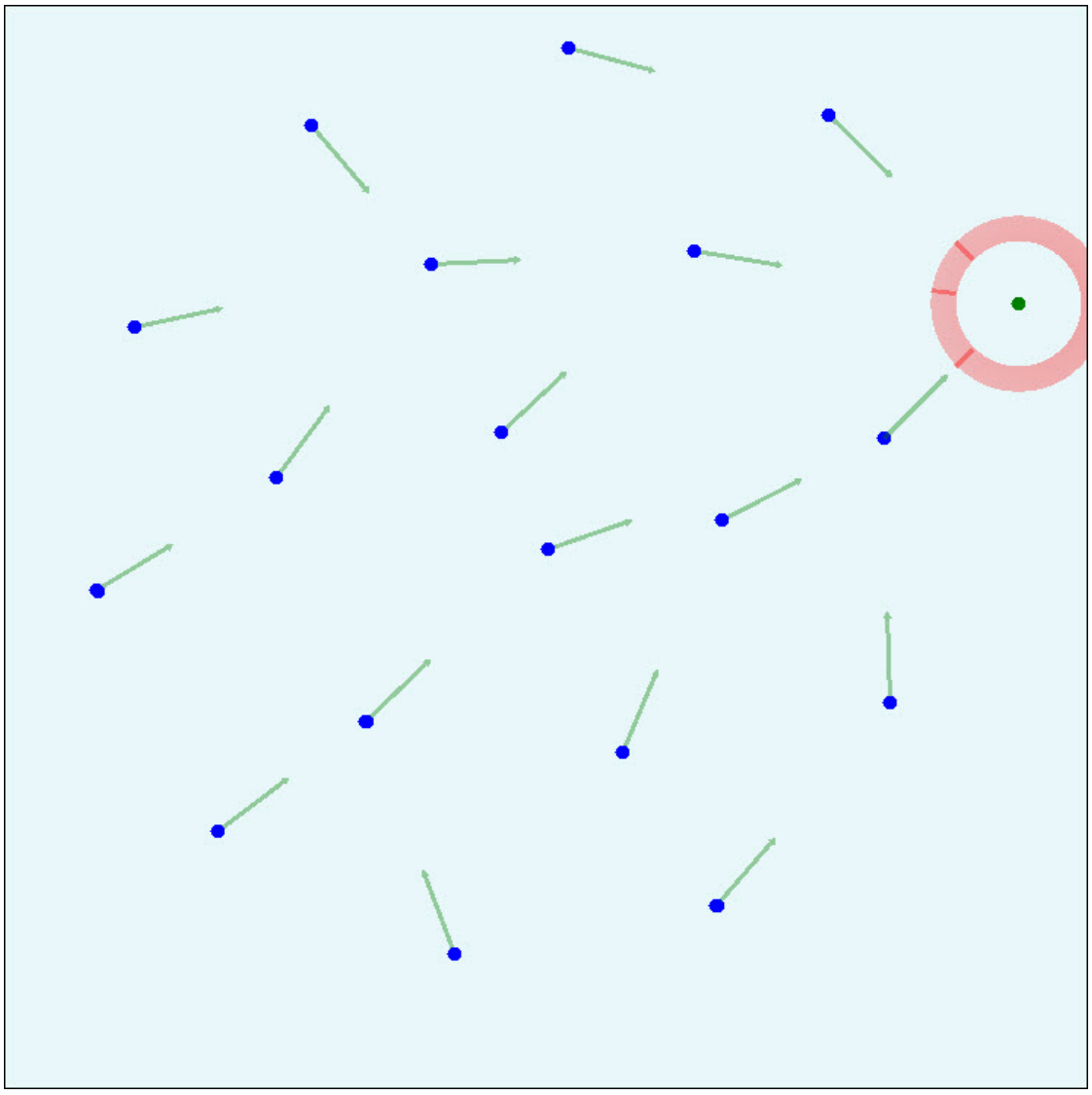

Figure 20. Sink sends a Subscription message

event for the number of nodes in the network, but it will serve to illustrate the failure of the protocol under these conditions. Four nodes have heard this event, and all have sent a Notification message directed toward the sink.

In Figure 23, an Acknowledgment message has collided with a data message at the node shown by the arrow. The node appears red to illustrate the collision, and the overlap between the gray annulus and red annulus is clear at that point. It is evident that far-off data streams can cause significant problems for data flows that

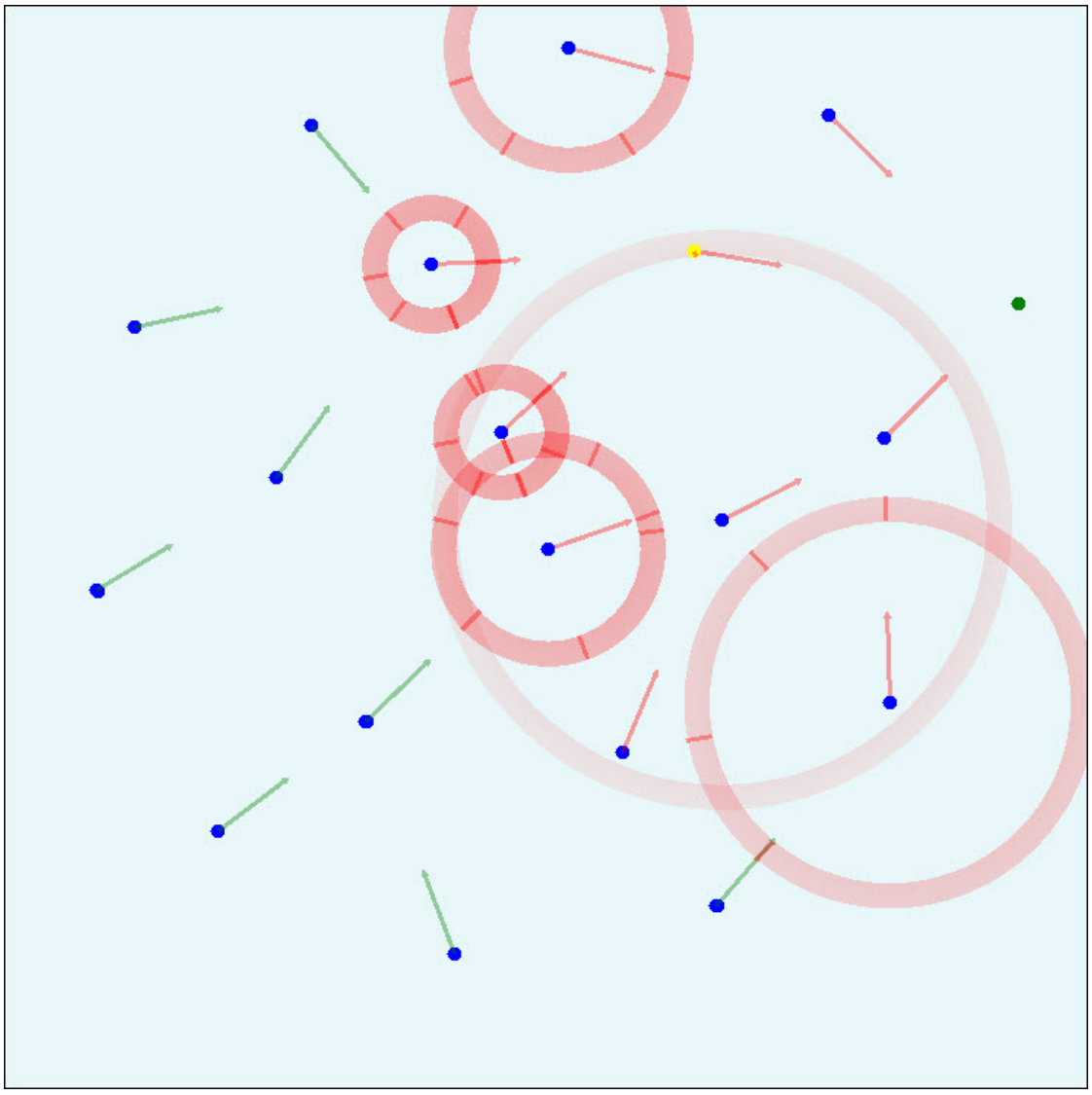

Figure 21. Subscription message propagates

follow completely separate paths. Collisions in UWA are a much more significant problem than in the RF space; this is quantified below.

First, ask the question: "At what range will no collision occur if two nodes transmit at the same time?" Assume that both messages are of the same size, B bytes.

The number of seconds required to propagate one bit, t_p, can be given by:

$$t_p = \frac{d_n}{S} \tag{10}$$

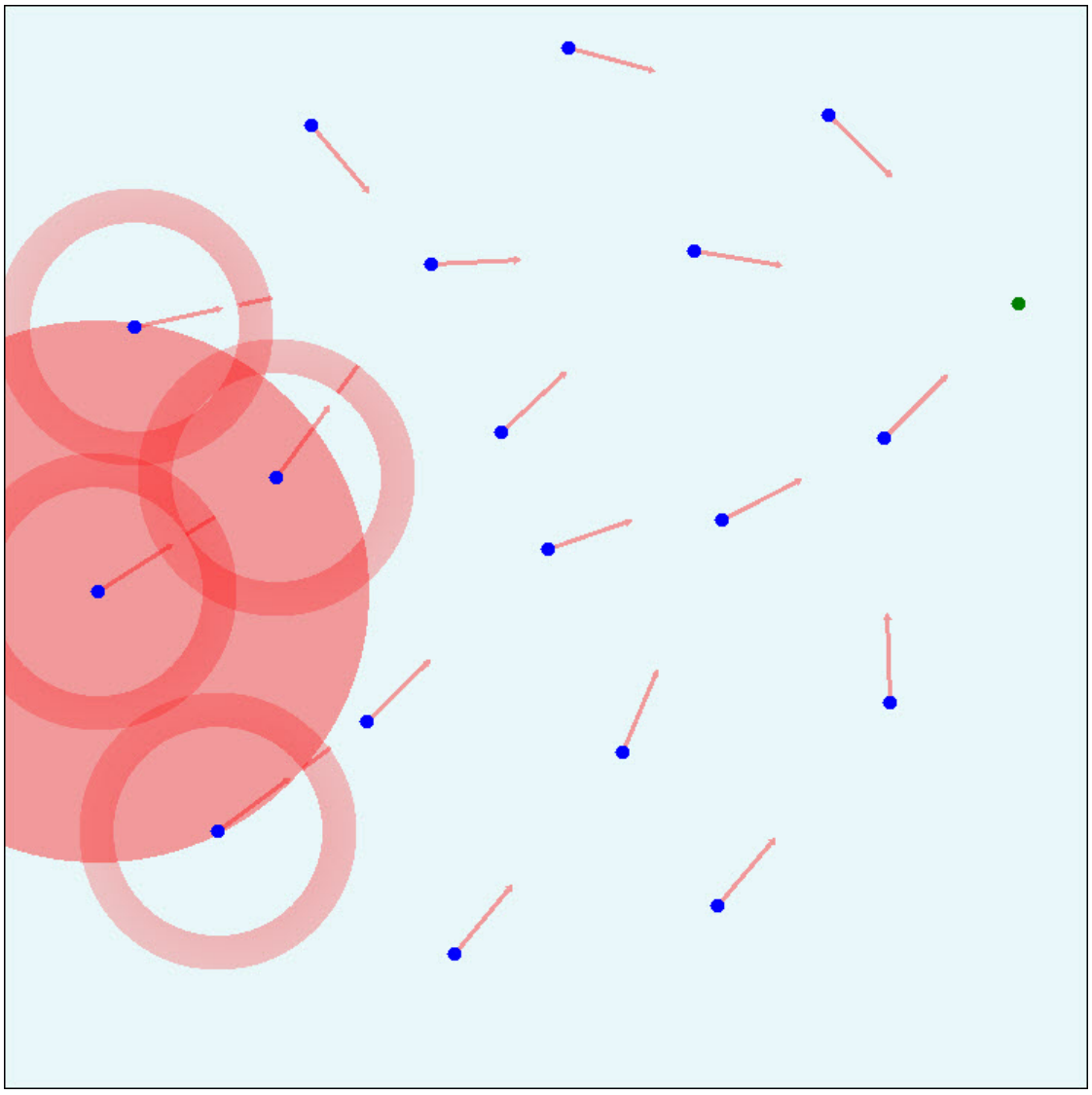

Figure 22. An event occurs, data is sent by listening nodes

where d_n is the distance between node n and a common destination node, and S is the propagation speed (speed of sound in UWA or the speed of light in RF). Also, the time it takes to transmit a message, t_t, which is the same as the time required to receive that message, is given by:

$$t_t = \frac{8B}{M} \tag{11}$$

where M is the transmission speed and B are the number of message bytes. We are assuming that $M_{\text{UWA}} = 2.4\text{kbps}$ and $M_{\text{RF}} = 100\text{kbps}$ as discussed in Section

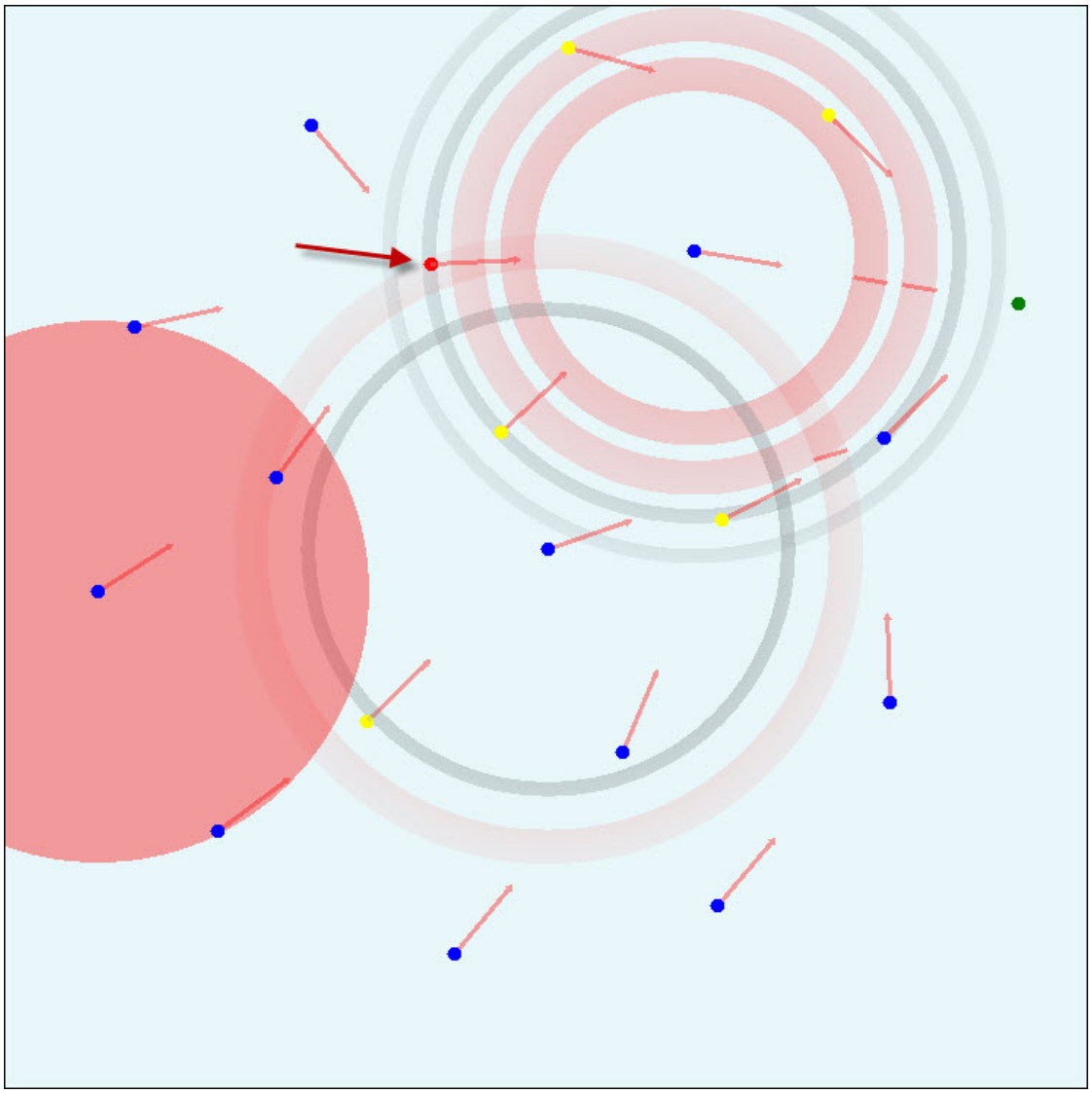

Figure 23. An ACK collides with data from another node

1.1.2. The factor of 8 converts bytes to bits.

We are then interested in the overlap where:

$$t_{1p} + t_t > t_{2p} > t_{1p} \tag{12}$$

Where t_{1p} is the propagation time for the first node and t_{2p} is the propagation time for the second node. Substituting back in:

$$\frac{d_1}{S} + \frac{8B}{M} > \frac{d_2}{S} > \frac{d_1}{S} \tag{13}$$

and subtracting t_{1p}:

$$\frac{8B}{M} > \frac{d_2 - d_1}{S} > 0 \tag{14}$$

which results in:

$$\frac{8B \cdot S}{M} > |d_2 - d_1| \tag{15}$$

This is most meaningful with real numbers and this is dependent on the size of the message. A very small message, i.e. 15 Bytes, will be used here to come up with the following values of Δd:

$$\Delta d_{\text{UWA}} > 75 \text{ m} \tag{16}$$

$$\Delta d_{\text{RF}} > 360 \text{ km} \tag{17}$$

In other words, two RF transmissions, sent at the same time, will always collide at the receiver so long as the distances to the receiver are in radio range. UWA transmissions, on the other hand, are much more spatially dependent. This means that in UWA, a 75m difference in distance corresponds to some certain time difference.

This begs the next question: "if the distances to the receiver are the same, what difference in transmit times avoids a collision?" We're now interested in the inequality:

$$t_t > |t_2 - t_1| \tag{18}$$

where t_1, t_2 are the initial transmit times for nodes 1 and 2 respectively. Substituting in Equation 11

$$\frac{8B}{M} > |t_2 - t_1| \tag{19}$$

Using the same real values gives:

$$\Delta t_{\text{UWA}} = 0.0500 \text{ s} \tag{20}$$

$$\Delta t_{\text{RF}} = 0.0012 \text{ s} \tag{21}$$

This means that, given the same message sizes (and number of messages transmitted), the UWA medium is almost 50 times more likely than in RF to have a collision. This is mitigated somewhat by modifying the frequency of messages transmitted.

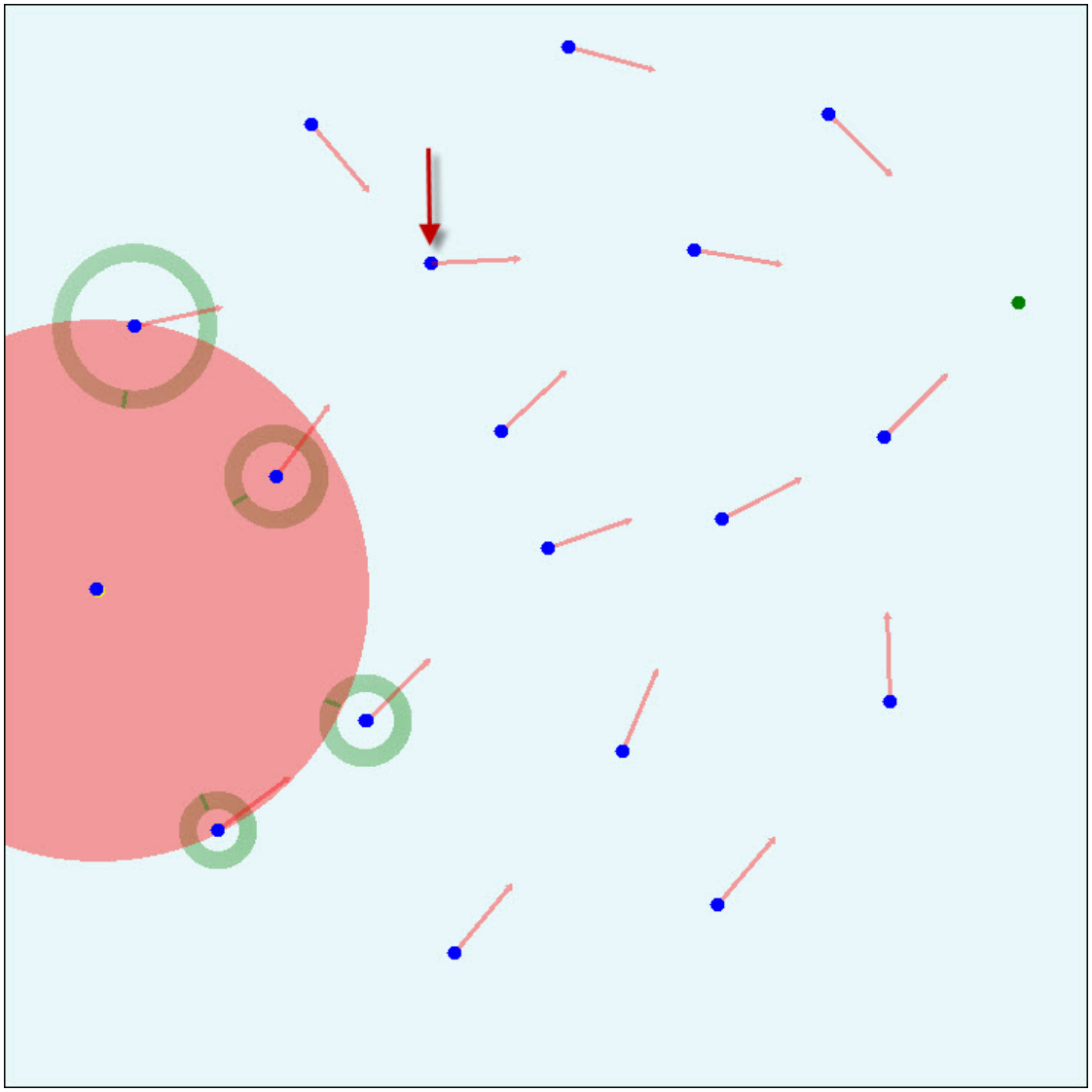

Figure 24. ACK time-out causes route to be lost (t^-)

Continuing on with the protocol flow, because of the collision the marked node will lose its route. No ACK was received for a data message sent so, at a certain time t, the node's route marked in Figure 24 will be lost.

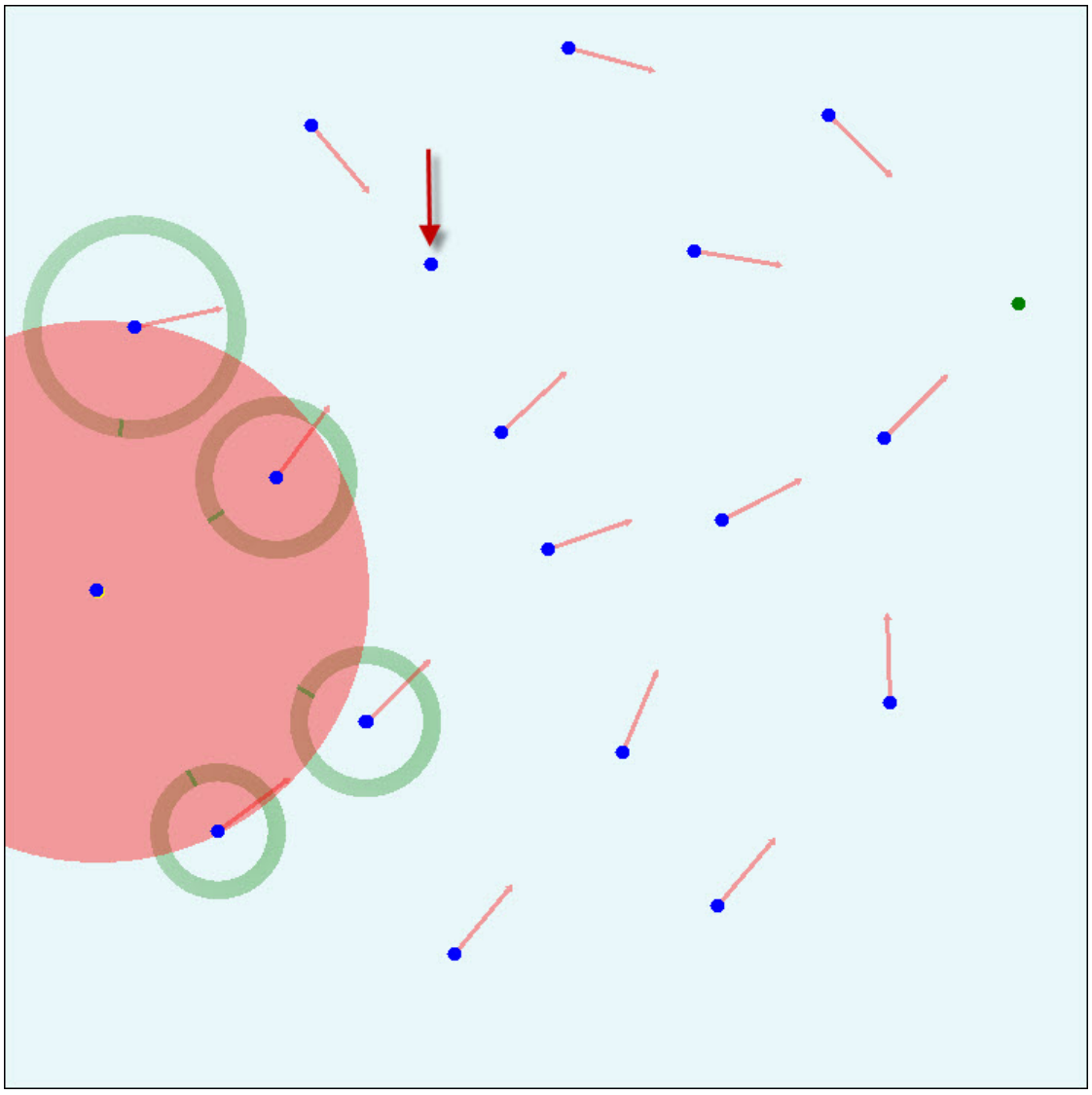

Figure 25. ACK time-out causes route to be lost (t^+)

Figure 25 shows the same node after it lost the route. Notice also that one other node in this small network has the same situation.

Once a node loses its route, it sends a Search message asking all its neighbors to provide their hop count back to the sink. Figure 26 shows this message, visualized as an orange annulus, being broadcast by the lost node.

Response messages are sent by neighboring nodes that heard the Search message, shown in Figure 27. The Response message is visualized as a green annulus.

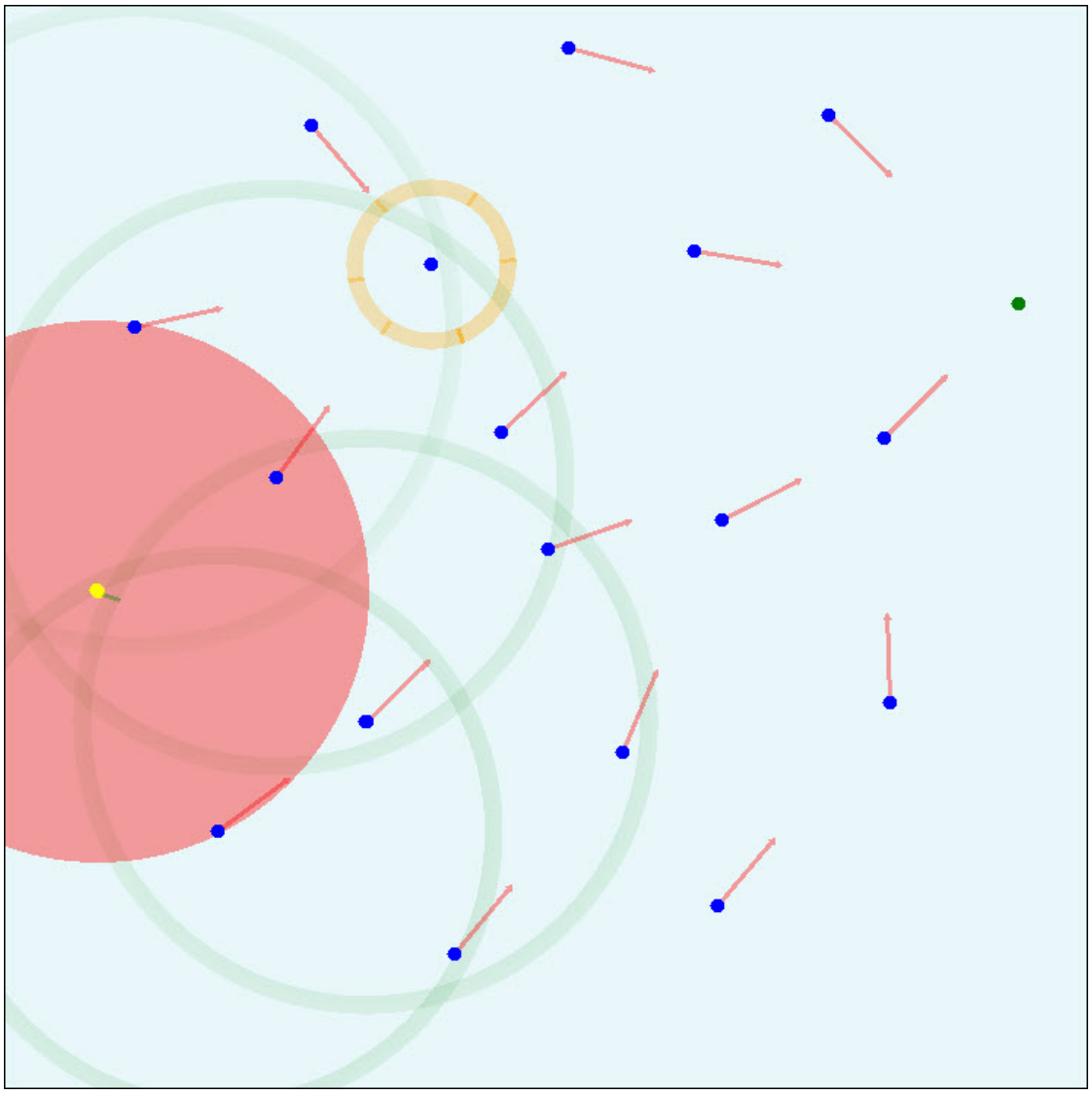

Figure 26. Lost node sends Search

There are two things to also note in this figure. Firstly, one other node has lost its route and is beginning the Search process. Secondly, the Search/Response produces a lot of traffic concentrated in one area over a certain amount of time. If the Searching node does not receive a response from all neighbors, it will make a second attempt before choosing its best remaining route. This chattiness, where multiple nodes in a close region are solicited into transmitting at nearly the same time, is immediately obvious when the visualization is animated and it can be

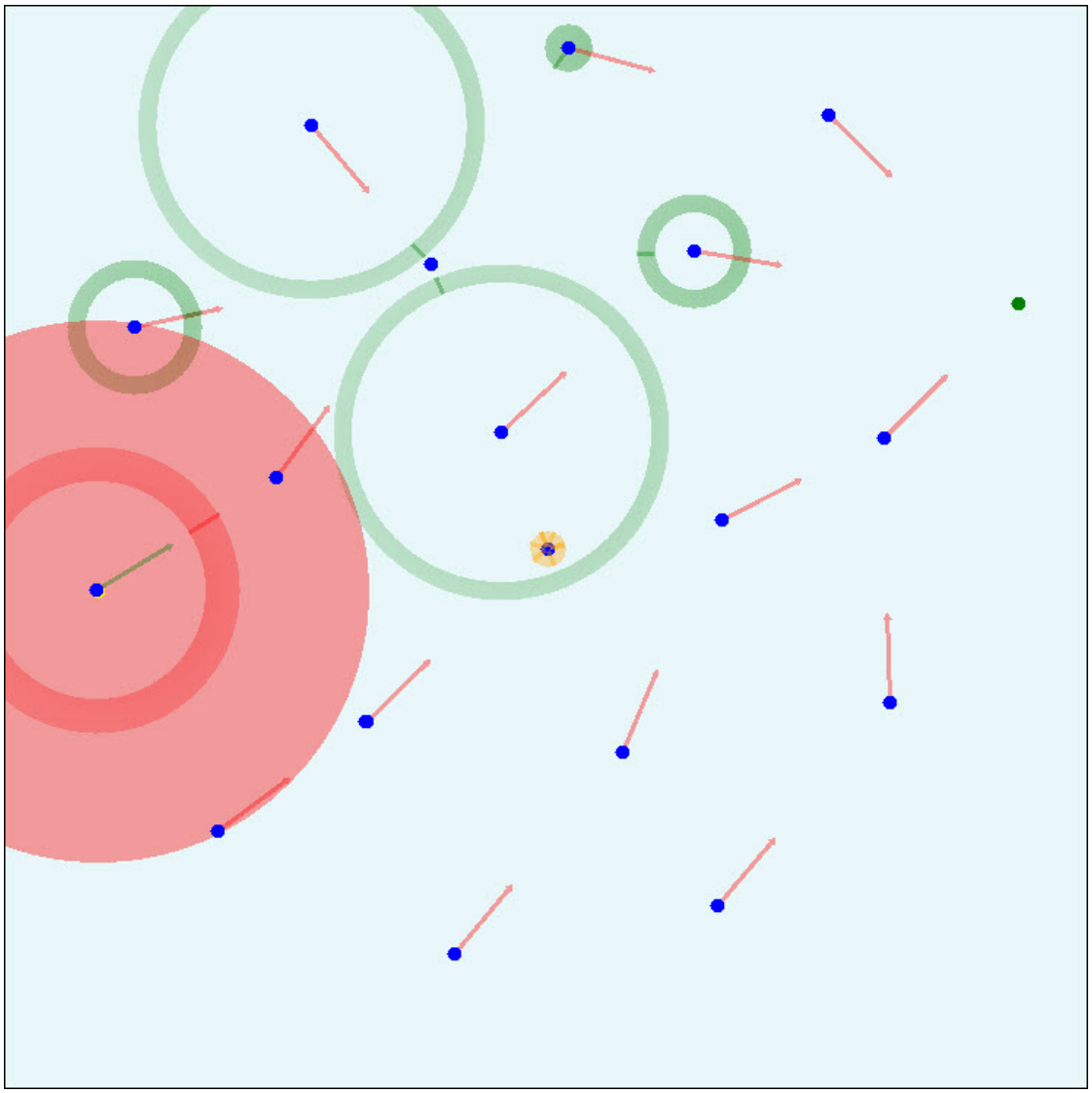

Figure 27. Neigboring nodes respond to Search

analyzed by stepping through the animation.

After two search messages, the lost node still hasn't received all the responses it needs, so it falls back onto the best route heard. The route is restored as shown in Figure 27. It's worth noting again that, the more traffic in a compressed time period, the more likely a collision becomes. In RF the collision rate will increase but still be quite rare. Boukerche et al. in [1] specify that search continues until all nodes have reported in. Logically this is impossible as dead nodes would

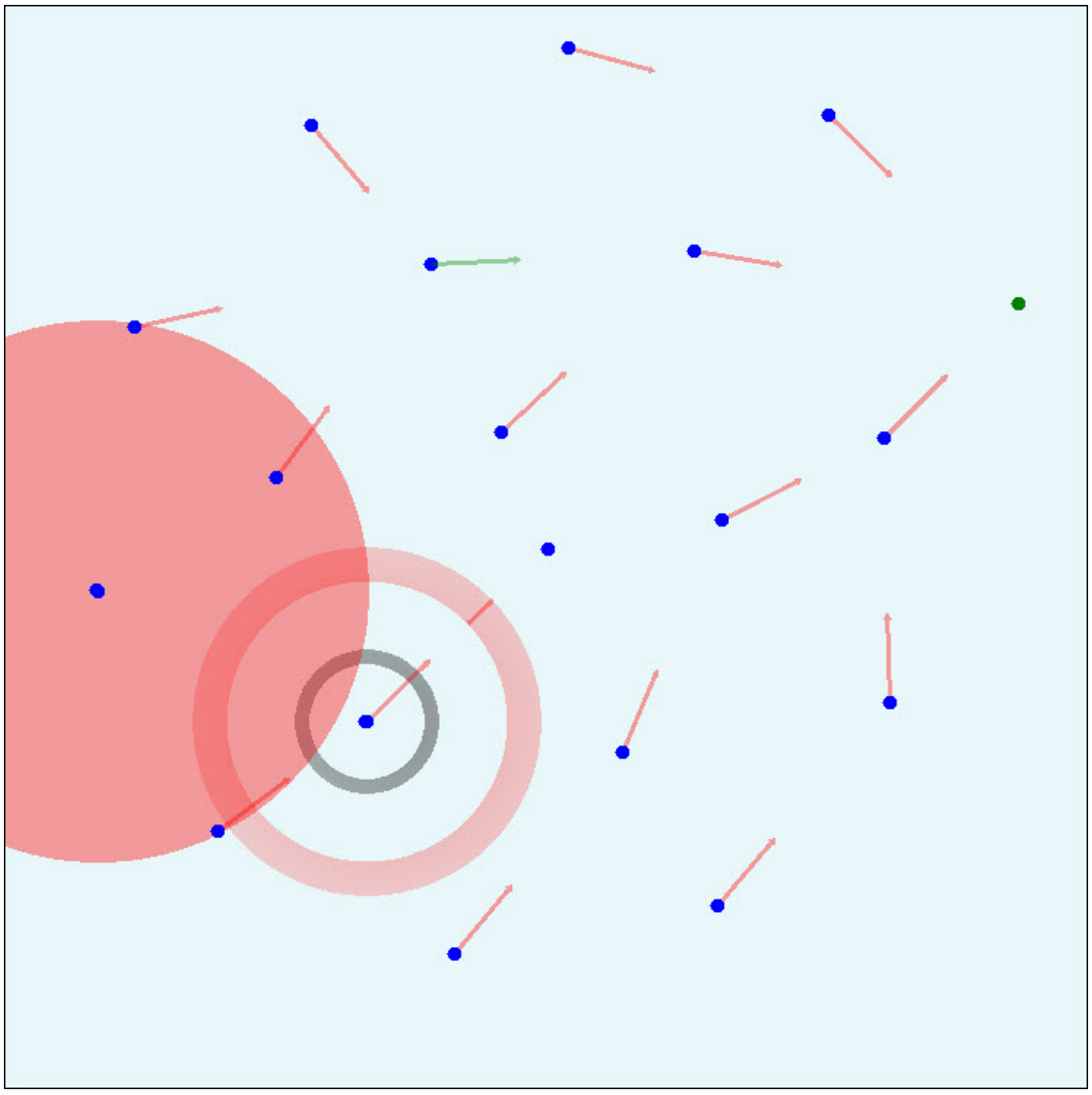

Figure 28. Eventually, the lost node recovers its route

never respond. Eventually, perhaps, the neighbor relationship to the dead node is removed through some other process, but this is not described by Boukerche et al. Repeating Search a second time in UWA results in a significant increase in source-to-sink delay for Data messages. Two Searches sent are enough to get a valid route, even if some Response messages are lost.

3.1.2 Visualization and Simulation Limits

The Visualizer and Simulator both have limits in various areas including frame rate or frame delay, number of nodes in visualization, and number of nodes in simulation (without visualization). These limits are clearly dependent on the computer used to run the simulation. In this case the hardware used was a Dell Precision M4400 laptop with an Intel Mobile Core 2 Duo P8600 (2.40 GHz) and 4 GB of DDR2-800 RAM.

Frame delay is the difference between the ideal real to sim-time ratio and the actual ratio. A user defined ratio of 5:1 is used in the examples in Figures 29 and 30. The first figure shows a visualization of 200 nodes with low chatter, few messages in the medium. This situation shows an actual time factor of 6.4 real seconds per simulation second. The second figure shows the same 200 nodes with somewhat more traffic, now at a ratio of 16.6. The number of nodes, route arrows, and message annuli all contribute to the worsening time ratio.

For the PEQ protocol, the following real to sim-time ratio ranges were observed in a 200 node simulation with a $(10,000\text{m})^2$ field:

PEQ Stage	Ratio Range
Build Tree	5.9 - 9.9
Subscribe	6.8 - 20.6
Notify	7.6 - 29.4
Search	31.4

Table 1. Sim-time to real-time ratio due to graphics complexity during the stages of the PEQ protocol.

Furthermore, during network failure due to many ongoing Search processes, the ratio reaches the range 56.4 - 132.0 and upward from there.

Memory use is affected by two primary causes: the number of nodes and the average number of neighbors per node. In a grid deployment, with nodes 1 km apart, the size of the field can be increased to increase the number of nodes, but

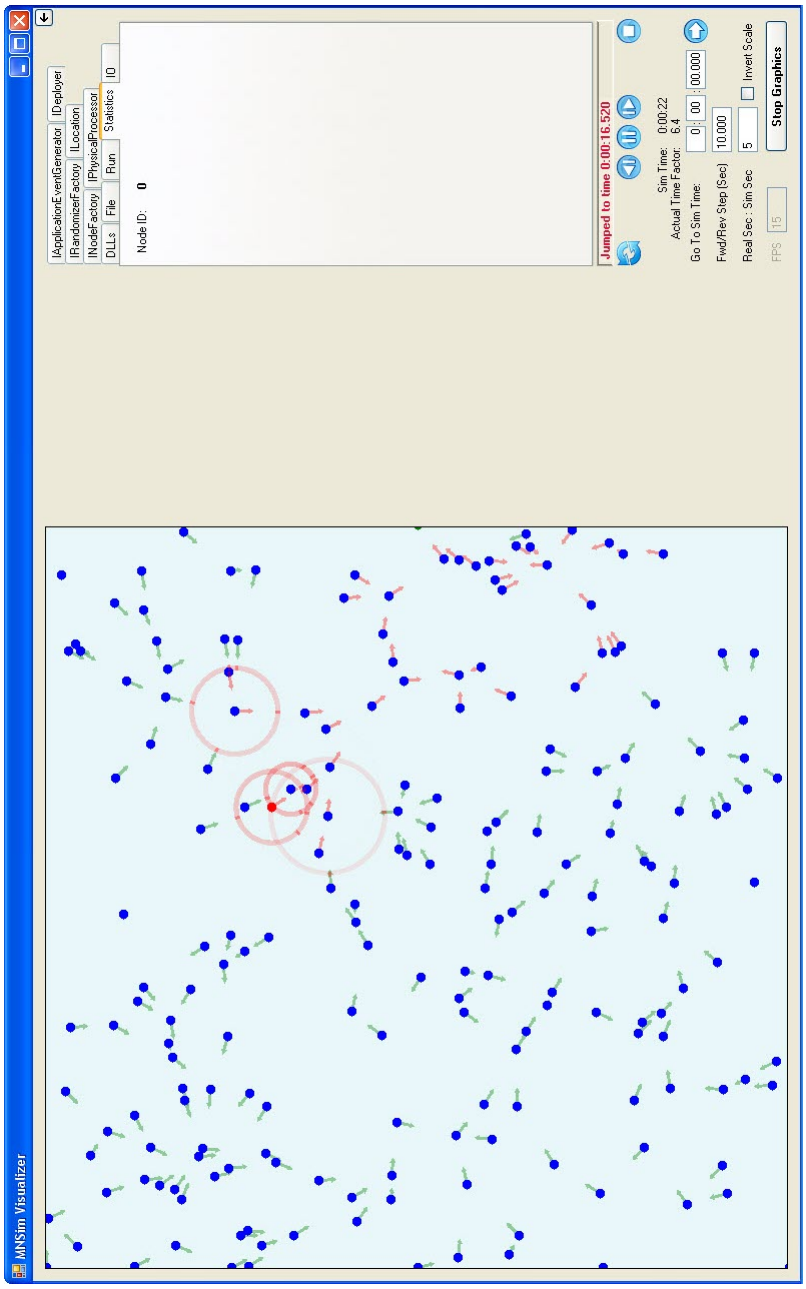

Figure 29. 200 Nodes with low chatter

not the number of neighbors per node. When this is done from $(4\text{km})^2$ to $(39\text{km})^2$, or from 25 to 1600 nodes, memory use goes from approximately 75 MB to 330 MB. The 1600 node field does not fit well into the visualizer window and is practically useless for visual simulation.

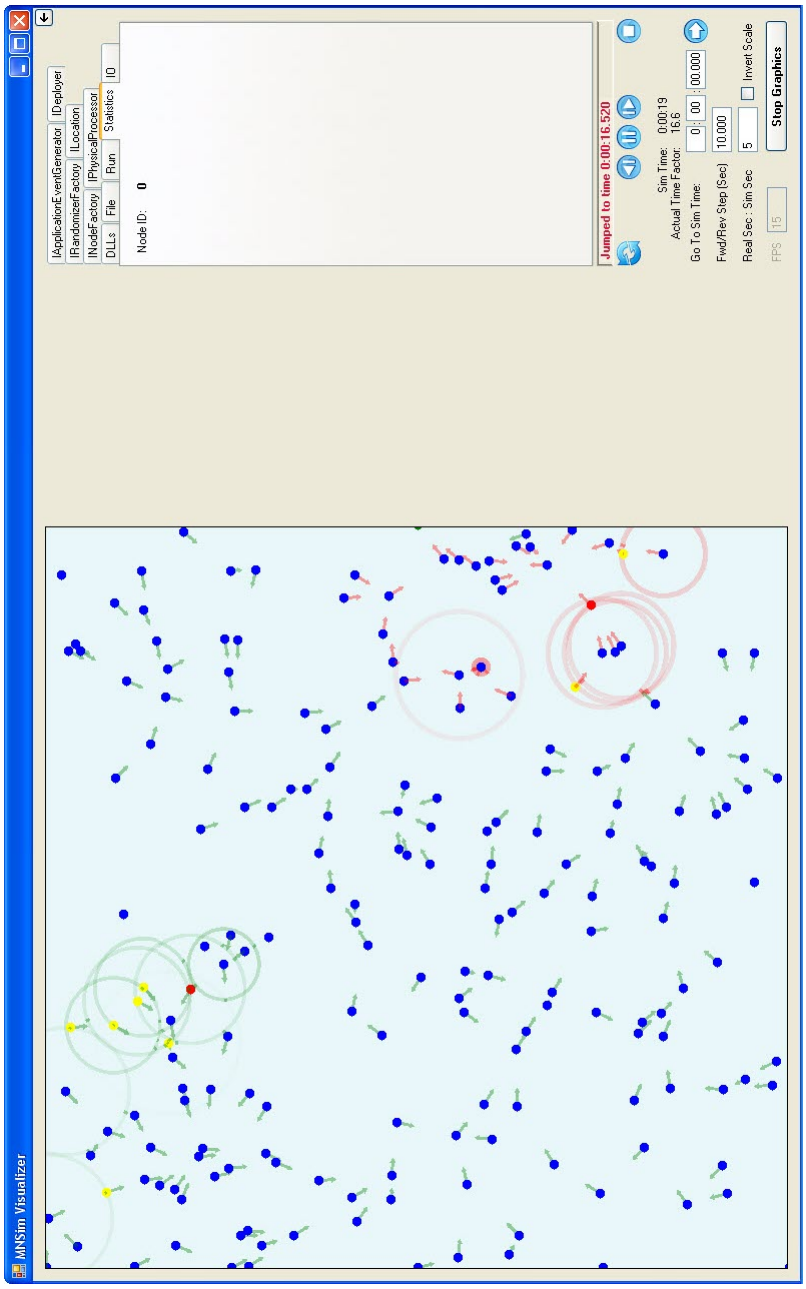

Figure 30. 200 Nodes with high chatter

Thus, the only memory limitation is on the number of nodes and average number of neighbors per node. Note that the number of neighbors per node is, on average, the number of nodes divided by the number of square km, since the node transmit distance is set to 1km. The limitation is graphed in Figure 31. Once 1500

nodes are reached on a $(10\text{km})^2$ field the memory used is approximately 1.44 GB; 2000 nodes resulted in an out of memory exception.

On the other hand, with the visualizer disabled and only the simulation and its textual output running, 3000 nodes in the same configuration peaks at 80 MB of memory used. The number of nodes and neighbors involved has more of an effect on the amount of time required to complete the simulation than on the memory used by the core simulation.

3.2 Protocol Simulation Statistics

3.2.1 Experiment Set-up

Boukerche et al. in [1] ran simulations, incrementing N by 100 nodes, from 100 through 500 randomly placed nodes. Five sources were placed at the opposite end of the field from one sink node. Their simulation parameters were given in the table below:

Parameter	Value
Simulation time (s)	500
Number of nodes	100 - 500
Source data rate (events/s)	10
Radio range (m)	20

Table 2. Original simulation parameters from [1] with non-applicable parameters removed.

Parameters outside the scope of this thesis, including energy dissipation and non-PEQ parameters, were removed from their table, above. The size of the field is missing, meaning that the average distance between nodes (for each N) is unknown. We can assume the field does not increase in size, as this would likely have been discussed in the article if true.

If we make the assumption that at the lowest N, $N = 100$, the average node separation is equal to the radio range, 20m, a square field would have sides of length l of 200m. In this experiment, our acoustic modem's range is set to 1000m.

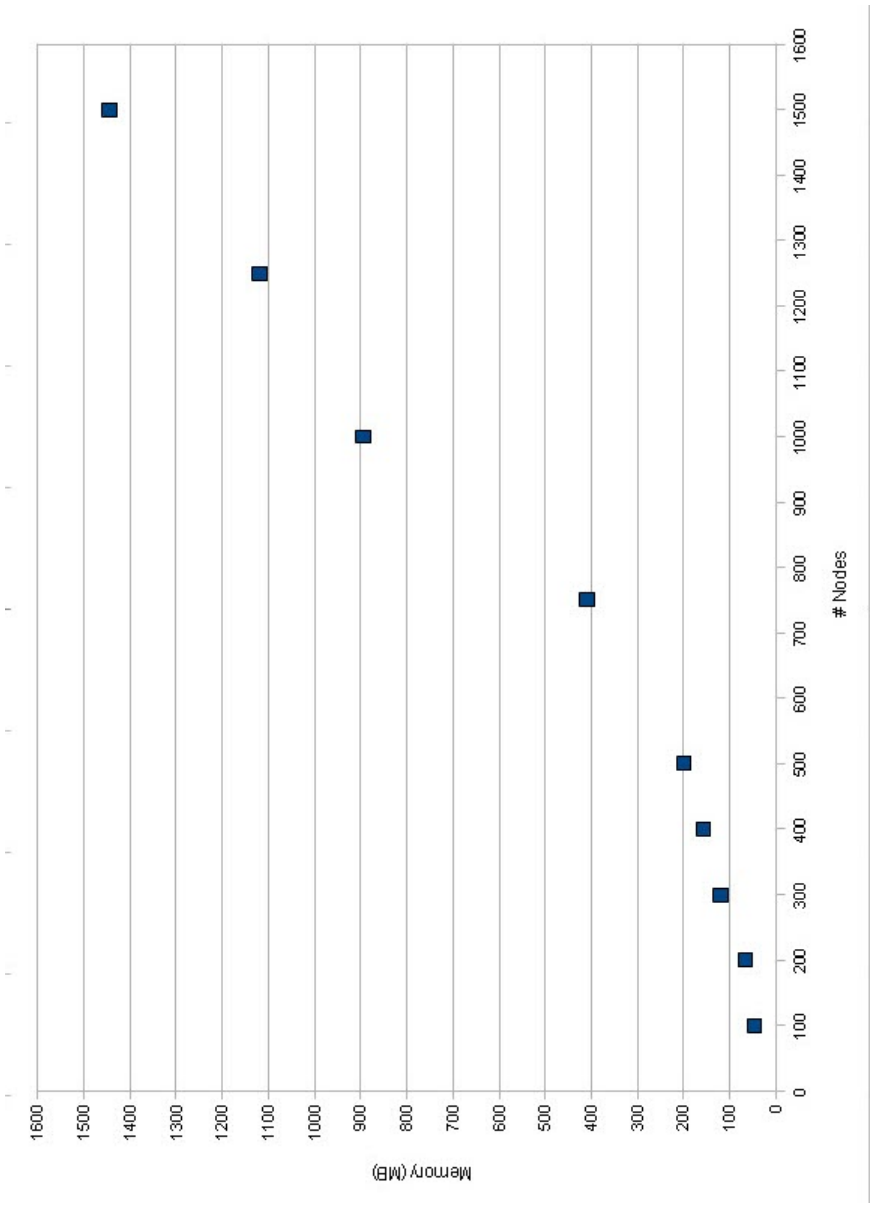

Figure 31. Number of Nodes vs. Memory Used, all other settings fixed

Thus, $l = 10km$. With this value, however, the likelihood of having complete closure between one or more sources and the sink is very low; once in 100 runs was observed. A field size of $(6 \text{ km})^2$ yields more reasonable results at $N = 100$, which yields an average node separation of 600m.

The source data rate is more interesting to the simulation. The value from Boukerche et al. is known, but as discussed in Section 3.1.1 these values do not

translate well into the UWA medium. By Equations 20 and 21 the rate should be:

$$R_{\text{UWA}} = \frac{0.1 \text{ s}}{\text{event}} \cdot \frac{0.0500 \text{ s}}{(\text{UWA})} \cdot \frac{(\text{RF})}{0.0012 \text{ s}} \cong 4.2 \frac{\text{s}}{\text{event}} \tag{22}$$

Unfortunately, this value doesn't tell the whole story. This relation works well for link-local messages that do not propagate and do not have a cause and effect relationship in the network. One example is a Hello message where, if an RF network were to use 30s Hello timers (no information on neighbor discovery is provided), the UWA network would 1250s timers.

In the PEQ protocol, a Notification message containing data is sent from the source node to the sink. If the Notification message passed only to its nearest neighbor, or even if the Notification messages passed from node to node in a purely unreliable way, 4.2s per message would not be a problem. Since the Notification message expects an ACK message in return, the process no longer takes a fixed amount of time. When a Notification is sent there are three outcomes: the Notification collided at the next-hop, the ACK collided at the current node, or the Notification and ACK are both received correctly. The first outcome results in the Search process starting at the current node. The second outcome results in the Notification continuing to propagate and the Search process starting at the current node. The final outcome results in only the Notification continuing to propagate.

In each of the three outcomes, different results occur which expend different amounts of network resources and take different amounts of time. The final outcome, where the Notification continues normally, is not a problem. The other two begin a process that can take approximately 10s. Meanwhile, another Notification will be coming 4.2s later. In other words, while a 4.2s event spacing is called for based on a standard collision rate ratio between UWA and RF, the clustering of network problems along the path may need a larger event spacing. In this thesis the 0.24 Hz event frequency (approximately 4.2s spacing) will still be used.

No visualization is used in this simulation and 1000 trials are run for each N. The following parameters, with every attempt to bring the original paper's into UWA, were used:

Parameter	Value
Number of nodes	100 - 500
Field Size	6000 m $\times$ 6000 m
Source data rate (events/s)	0.24
Number of events (each of 5 sources)	10
Hello timer	1250 s

Table 3. Simulation set-up for random node placement with 5 sources.

The simulation ends 60 seconds after the final event occurs. The five source nodes all receive the event at the same time.

Over the 5 simulation runs, $N = \{100,\ 200,\ 300,\ 400,\ 500\}$, the percentage of success, where the data reaches the sink, along with the average delay, and average number of Search processes started, are illustrated in the table below:

N	Success Rate	Avg Delay	Avg Searches	Avg Hops
100	14.5%	29.30	260	12.9
200	20.9%	34.55	261	12.0
300	17.4%	36.61	325	12.3
400	15.1%	37.45	344	12.6
500	13.8%	37.93	351	12.6

Table 4. Simulation runs with random node placement.

Due to the very low success rates, a second set of simulations was run with parameters similar to those above, but with a spiral deployment and only one source and one sink. Since the spiral deployment has fixed node distances, the field is increased to reach the correct number of nodes. The table for the spiral trials is below. Note that N does not exactly match the original trials because the spiral deployment exists in a square field and matching exact values of N is difficult.

Field Size	N	Success Rate	Avg Delay	Avg Searches	Avg Hops
$(7000\text{m})^2$	103	89.1%	12.53	24	10.4
$(9700\text{m})^2$	200	84.5%	19.65	60	15.9
$(12000\text{m})^2$	303	77.6%	29.97	144	22.6
$(14000\text{m})^2$	412	56.3%	41.32	189	24.0
$(16000\text{m})^2$	539	51.2%	46.15	242	27.1

Table 5. Simulation runs with Archimedean spiral node placement.

Graphing these values provides little insight that cannot be gained through a discussion of the failures and discrepancies of this protocol in comparison to its original report in [1]. This will be analyzed in Chapter 4.

List of References

[1] A. Boukerche, R. W. N. Pazzi, and R. B. Araujo, "A fast and reliable protocol for wireless sensor networks in critical conditions monitoring applications," in *Proceedings of the 7th ACM international symposium on modeling, analysis and simulation of wireless and mobile systems (MSWiM'04)*, 2004, pp. 157–164.

CHAPTER 4

Discussion

This chapter will focus on the analysis of the PEQ simulation data. The data we encountered will be discussed and will be enhanced with a visual analysis of what problems occur with the PEQ protocol in the UWA medium.

4.1 Discussion on Simulation Data as compared with RF

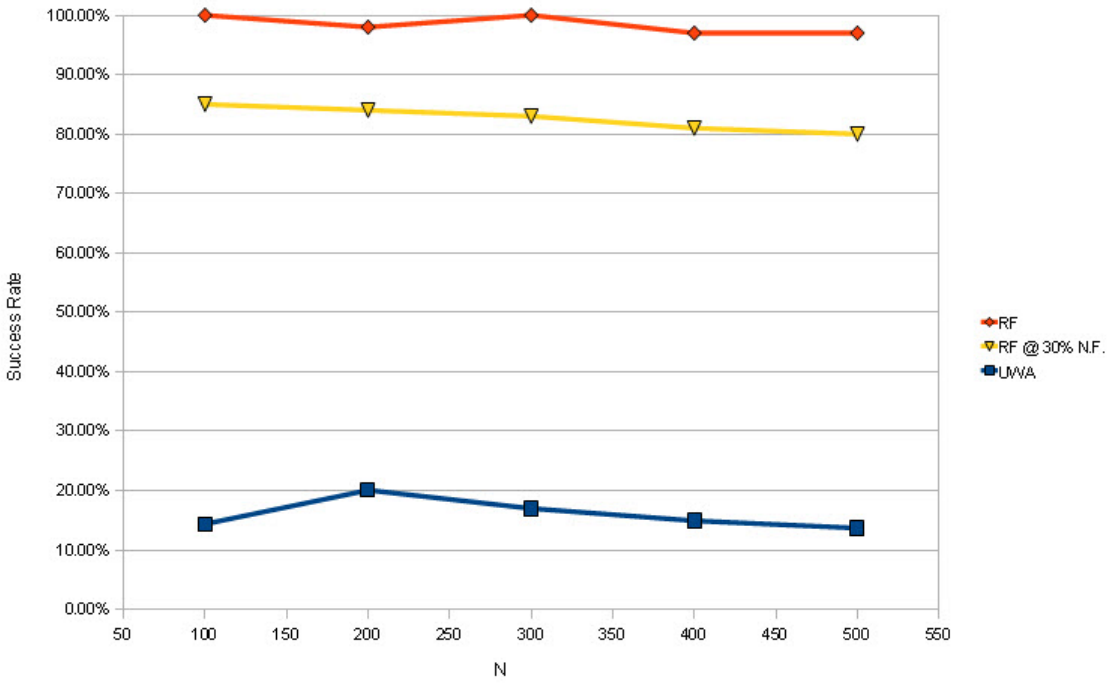

Figure 32. Comparing Success Rates of the PEQ Protocol in RF vs RF with 30% Node Failures (N.F.) vs UWA

The original, RF success rates are compared with the same rates of success in UWA in Figure 32. The original graphs for the RF case were presented in [1]. The field size(s) in the original paper are unknown, which represents a failure of this chart. It is impossible to replicate this experiment perfectly without knowing the node density. That being said, some of these results are easily explained. In

the following sections we will discuss these two outlying cases: firstly, the reduced success rate at $N = 100$; and secondly, the great disparity between the RF success rate and the UWA success rate. The Spiral deployment is discussed in the Success Rate Disparity section, Section 4.3.

4.2 $N = 100$ Random Nodes

The UWA success rate is lowest at $N = 100$ and highest at $N = 200$. Clearly, the node density at $N = 100$ is too low to have enough closure between the sources and sink node. Complete failures in the 100 node scenario are caused by a lack of a path back from all sources and doubling the node density solves this problem. From 200 through 500 node systems, UWA tracks the RF success rate fairly well, albeit at a large offset.

In the case of 100 random nodes, closure between source and sink is less common. Looking at the numbers, 6.8% of trials result in zero successes. This is a clear indicator of closure problems. Looking at the one of these cases, a randomizer seed of 1773791068 results in the network shown in Figure 33. In this case, the sink node has initiated the Build Tree process. A small network of nodes exists at the center of the outer Build Tree annuli. No annulus has a solid line to indicate a message will close with any other node (See Section 2.3.2). In all, or nearly all of the zero success trials, the sink node will be isolated from sources in some way. Usually this will be a break in the network near the sink as it is much less likely that all sources are isolated individually or that a break exists across the center field.

While it is possible for no data to pass from source to sink in the larger networks, it is increasingly uncommon and no specific cases were recorded.

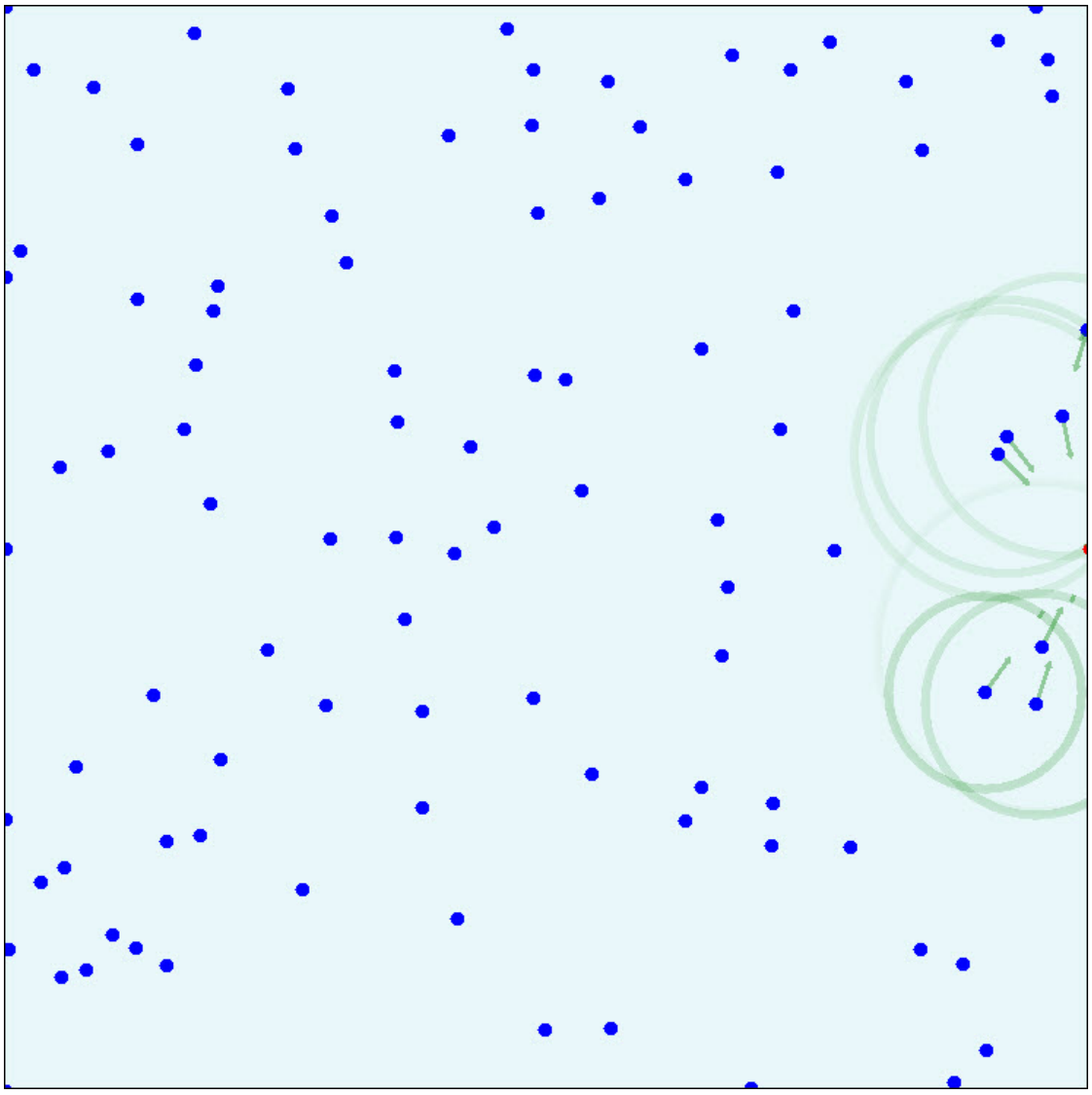

Figure 33. 100 Random Nodes, Run #10, Zero Data Successes

4.3 Successful Transmission Rate Disparity

Evident in Figure 32, there is an offset of approximately 65% for the RF case of 30% node failures and an offset of about 78% for the RF case of no node failures. This is a very large disparity between the UWA medium and the RF medium that has a number of possible causes. For the most part, PEQ's Search Process is at fault, but the reason for this problem is directly related to UWA's latency.

The Archimedean Spiral deployment has much more favorable results. The

node density in the spiral remains constant and this specific case has only one source and one sink. Fewer sources means that the spiral will have fewer collisions and a static node density means that the average number of neighbors per node does not change significantly. Only the path length increases. Note that the path length means more for the spiral deployment than for the random deployment because path length is not dependent on node density.

One source node is not a very realistic condition, but visually it's much easier to identify the problems with the protocol. It's also much easier to see what's going on with this type of geometric deployment because of the fairly even dispersement of nodes. Thus, our visual insight into the Search process will focus on the 100 node spiral deployment.

This discussion is broken down into the following three parts: firstly, the increased rate of Search in UWA vs. RF; secondly, the failure of the Search process in UWA; and finally, the increase of failures due to event frequency.

4.3.1 Increased Search Rate in UWA vs. RF

RF node failures are not well defined by Boukerche et al., but it is certain that node failures result in an increase in the number of Searches started. It is safe to assume that searches are extremely unlikely in the RF medium where node failures are at 0%. UWA tracks best with the 30% node failure RF scenario and, if the number of searches generated is an indicator of failure, the standard UWA network with 0% node failure likely has a significantly higher rate of search than any RF scenario.

Unfortunately, the rates of Search in RF networks are not provided in Boukerche et al., so we can only fall back on the fact that the collision rate in UWA is proven in Section 3.1.1 to be 50 times higher than that of RF. Higher collision rates will lead to a higher incidence of Search.

If the Search process worked correctly in the UWA medium, a greater incidence of Search couldn't be the only problem to account for the success disparity. Total network failures can only be explained by a failure of the Search process.

4.3.2 The Failure of the Search Process

The great offset between UWA and RF can be primarily explained by the failure of the Search process in the UWA environment. A Search message elicits a Response message. If not all responses are received, the Search process starts over. As written this process can continue indefinitely, though we have limited it to two Searches to prevent a totally chaotic, ineffective network. Since all neighbors send a response, the likelihood of additional collisions is extremely high, even when using random wait timers. This entire process takes a long time in UWA.

The failure of this process is difficult to determine using raw data, so the visual approach is preferred. This is illustrated using the Spiral deployment because the network failures are easier visualized with evenly distributed nodes and only a single source-sink pair. While an animated and interactive environment is most useful for this type of analysis, this thesis is written in a static medium so a number of key frames will instead be presented to illustrate how this failure occurs.

Because of the nature of this problem, and the fact that we're looking for a classic example rather than the first example of total network failure, Run #52 from the 100 Node spiral deployment has been chosen. This run used a random seed of 763679231. The event radius has been expanded from 1m to 150m so it would be shown visually, but it still covers only the single node; this change does not affect the simulation in any way.

In Figure 34, the node at **A** is the source node, characterized by the red circle representing the Event radius surrounding it. The green node at **B** is the sink node. The node at **Z** is about to receive a Notification message when a Hello

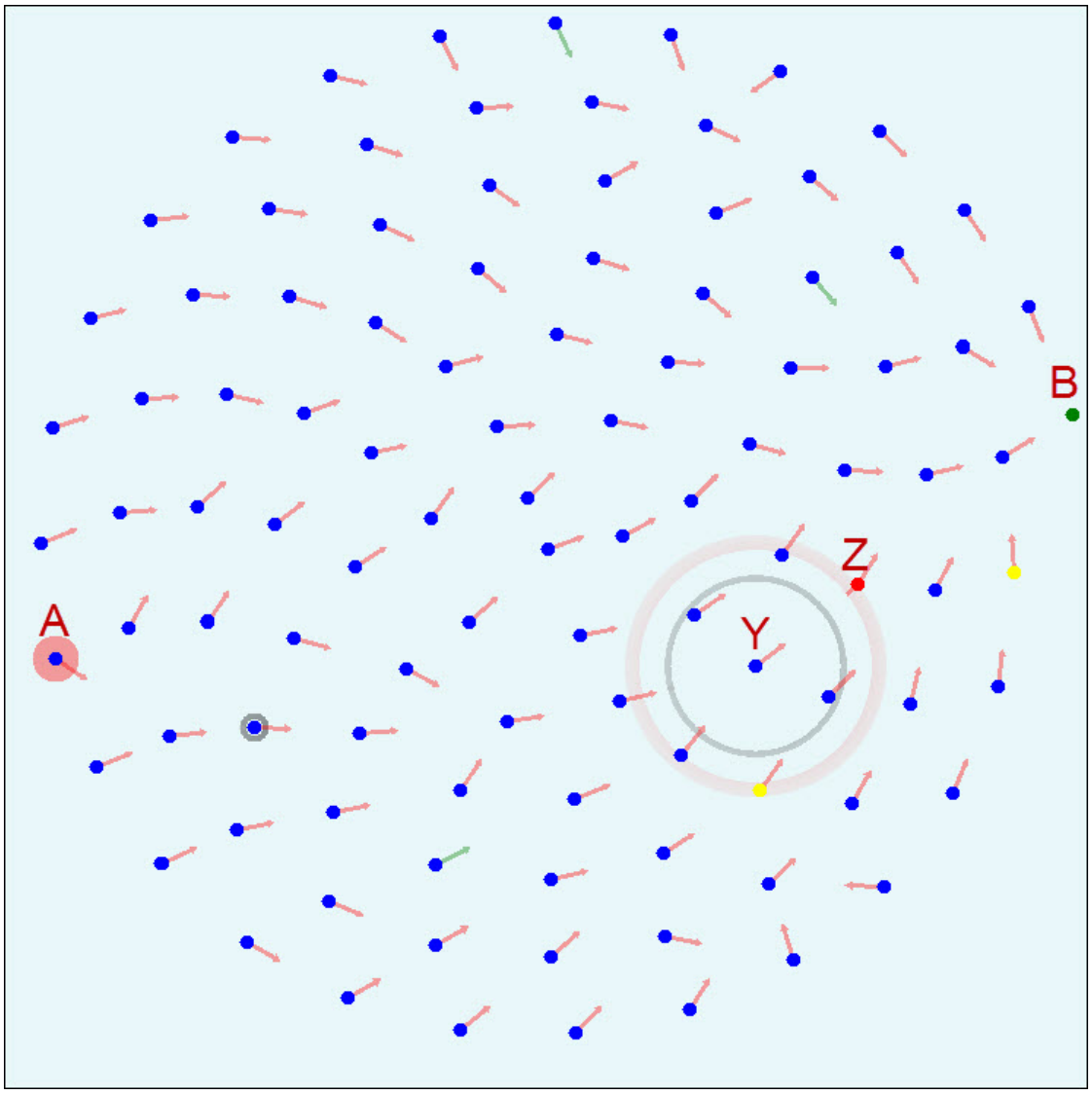

Figure 34. Node **Y** sends a Notification that collides at node **Z**.

message collided with it (note that Hello visuals have been suppressed for graphics efficiency). This collision occurs at approximately 1 minute 13.8 seconds. The most important fact of this collision is that the node at **Z** will not send an ACK message back. At approximately 1 minute 16.6 seconds, the sending node **Y's** ACK timer expires and its route to the sink is removed. (Note that all times given are from the start of simulation and given as approximate values to the nearest tenth of a second).

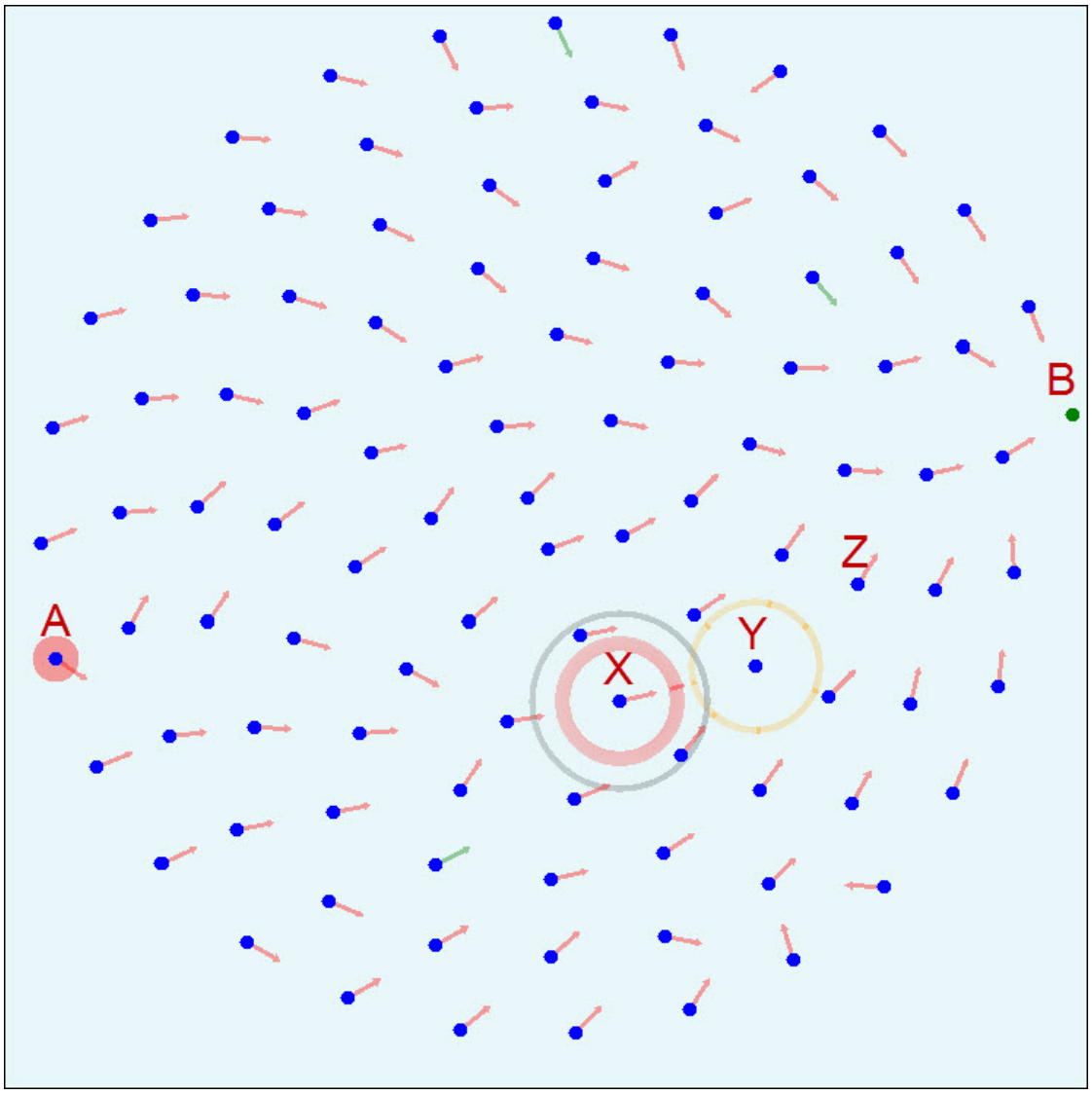

Figure 35. Node **Y** loses its route and begins Search. Meanwhile, node **X** sends a Notification to node **Y**.

At 1 minute 16.6 seconds, the node at **Y** in Figure 35 sends a Search message, the orange annulus centered at **Y**. Four of its seven neighbors' Response messages are received without collision, but the other three are lost due to collision.

The next data message is received by **Y** at 1 minute 17.3 seconds. It is not ACKed because no route exists. As specified in [1], "The neighbor node sends the ACK message right after it has forwarded the original packet." Although sending the ACK after the reception of any Notification message would help to solve

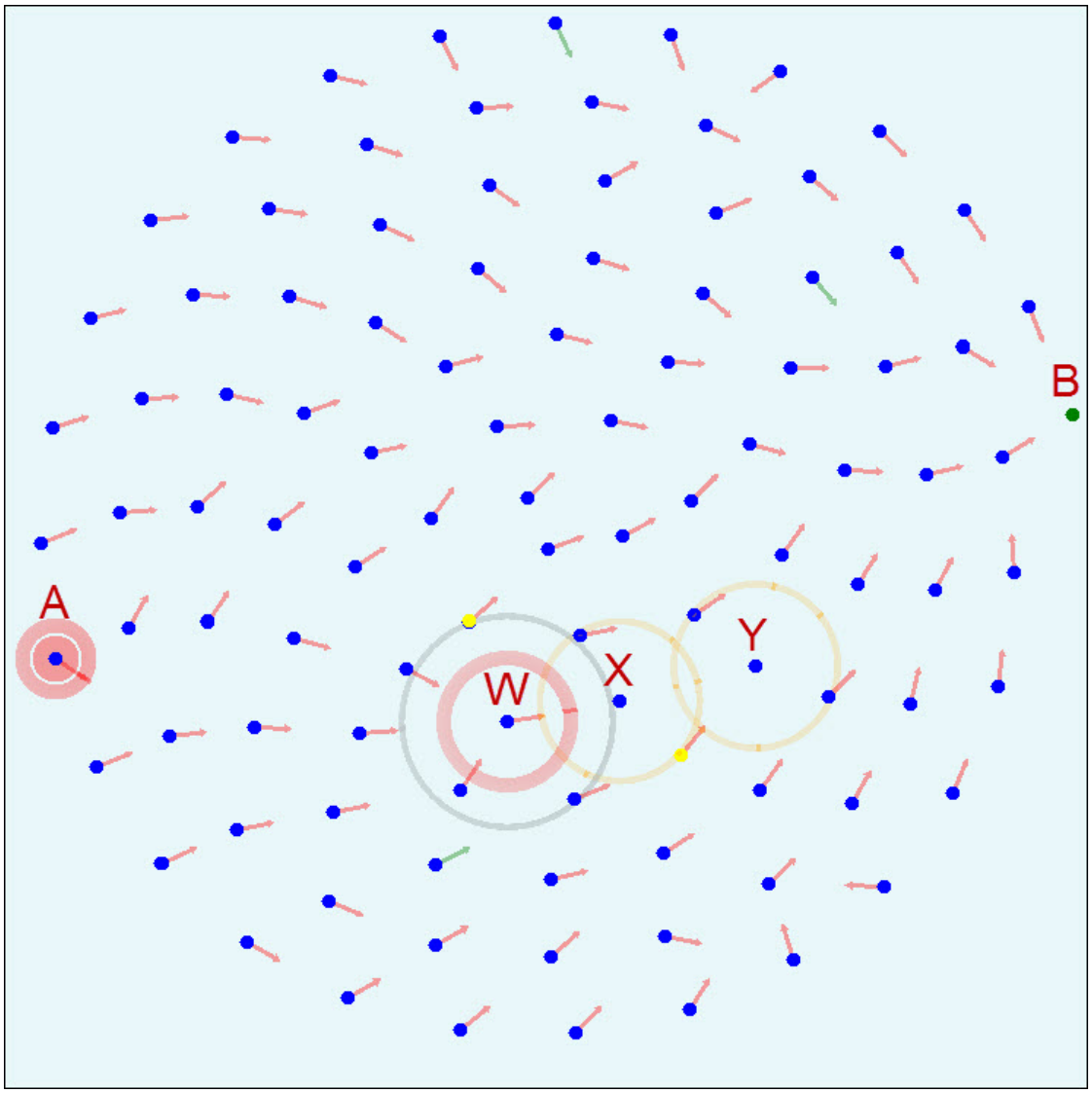

Figure 36. Node **Y** continues Search and node **X** enters Search. Node **W** sends a Notification to node **X**.

this problem, the increased collision rate during Search would likely disaffect the solution. The route failure results in the node at **X** in Figure 36 also entering Search at 1 minute 20.0 seconds. Meanwhile, at 1 minute 20.6 seconds, the Notification message sent by node **W** is received by **X** but collides with the second Search message sent by node **Y**.

This failure is repeated in Figure 37. At 1 minute 23.4 seconds node **W** loses its route and at 1 minute 23.6 seconds the Notification message from node

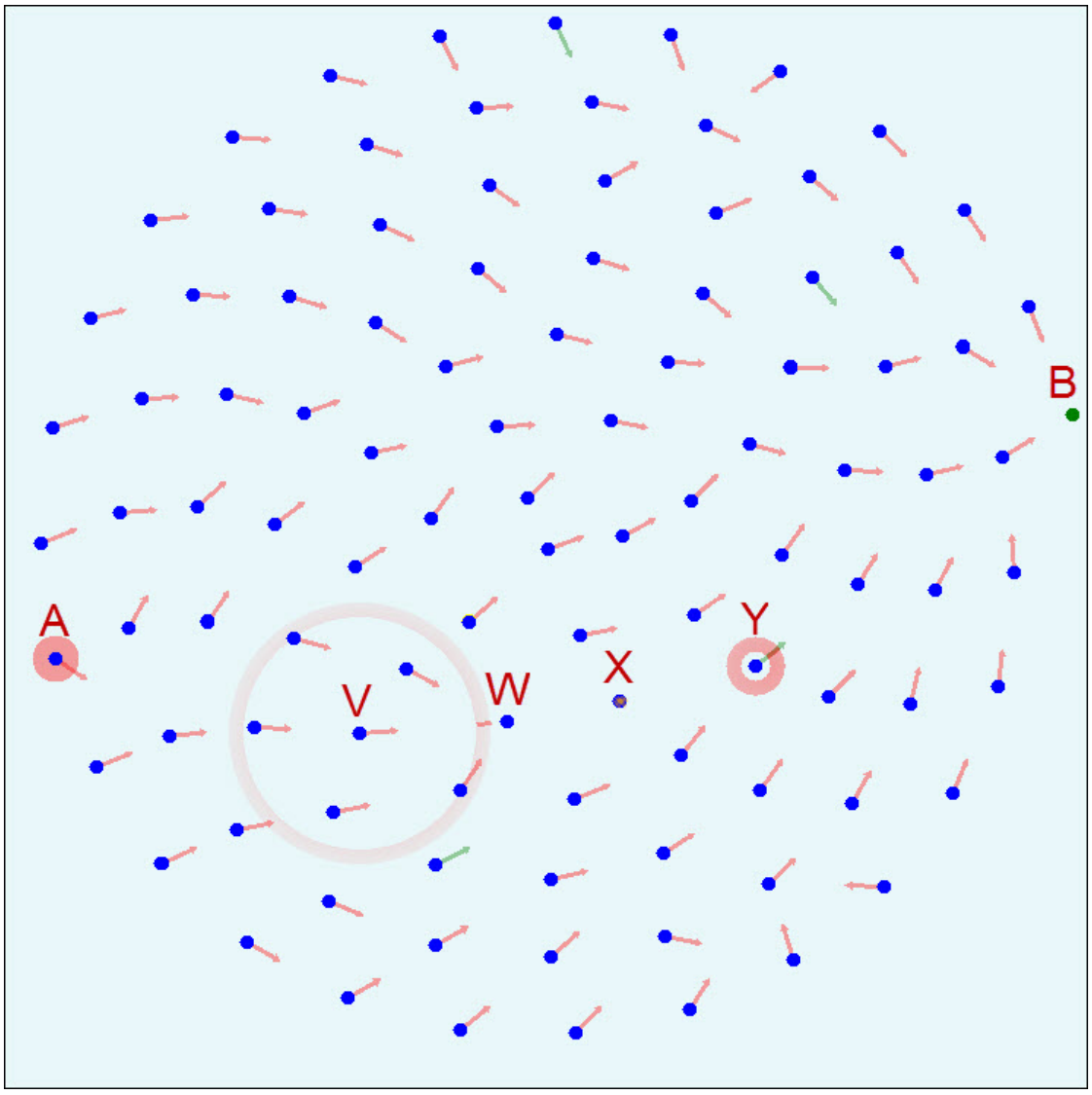

Figure 37. Node **Y** regains its route and continues sending the Notification message. Nodes **X** and **W** remain in Search and node **V** sends a Notification to node **W**.

V arrives. Eventually, the Search process completes at node **Y**, and the original message lost at 1 minute 13.8 seconds is resent at 1 minute 23.4 seconds. This does get the message to the sink, but with an additional delay of about 10 seconds. In just a few more steps the source node itself will fail and events will queue up at the source.

At 1 minute 33.3 seconds, node **T** in Figure 38 begins receiving a Notification

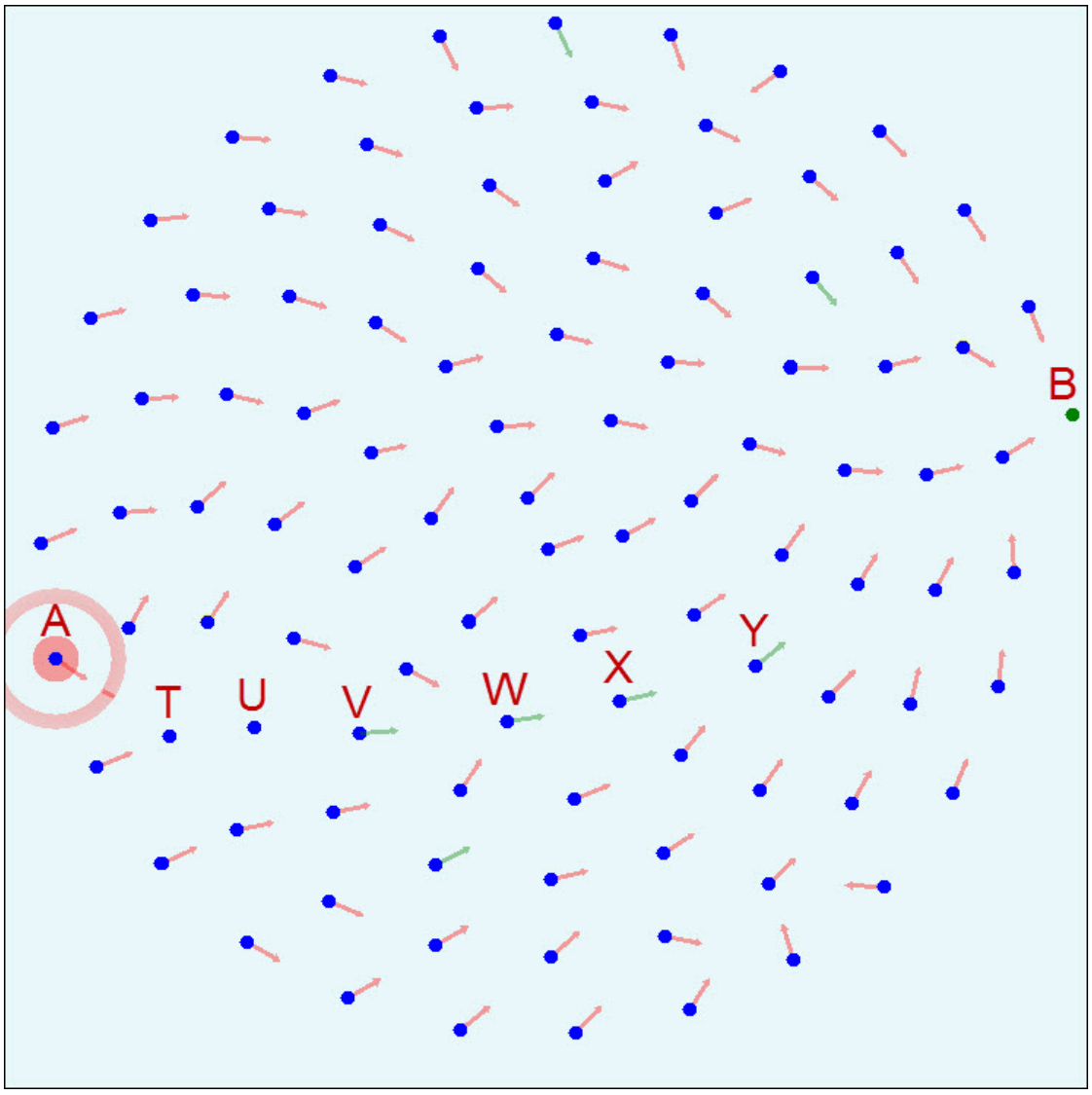

Figure 38. Eventually, the source at node **A** sends a Notification to Searching node **T**. Node **A** will enter Search.

message from the source, node **A**. Nodes **T** through **Y** have all undergone the Search process and nodes **V** through **Y** have recovered a route. At 1 minute 36.1 seconds, the source node loses its route and begins the Search process.

Until this point, the network has effectively delayed all events by three cycles, or

$$\text{delay} = 3 \text{ events} * \frac{1 \text{ s}}{0.24 \text{ events}} = 12.5 \text{ s} \tag{23}$$

This means that, once the source node enters Search, the lost Notification message is queued and two events are lost (an event is not sent as a Notification if no route exists).

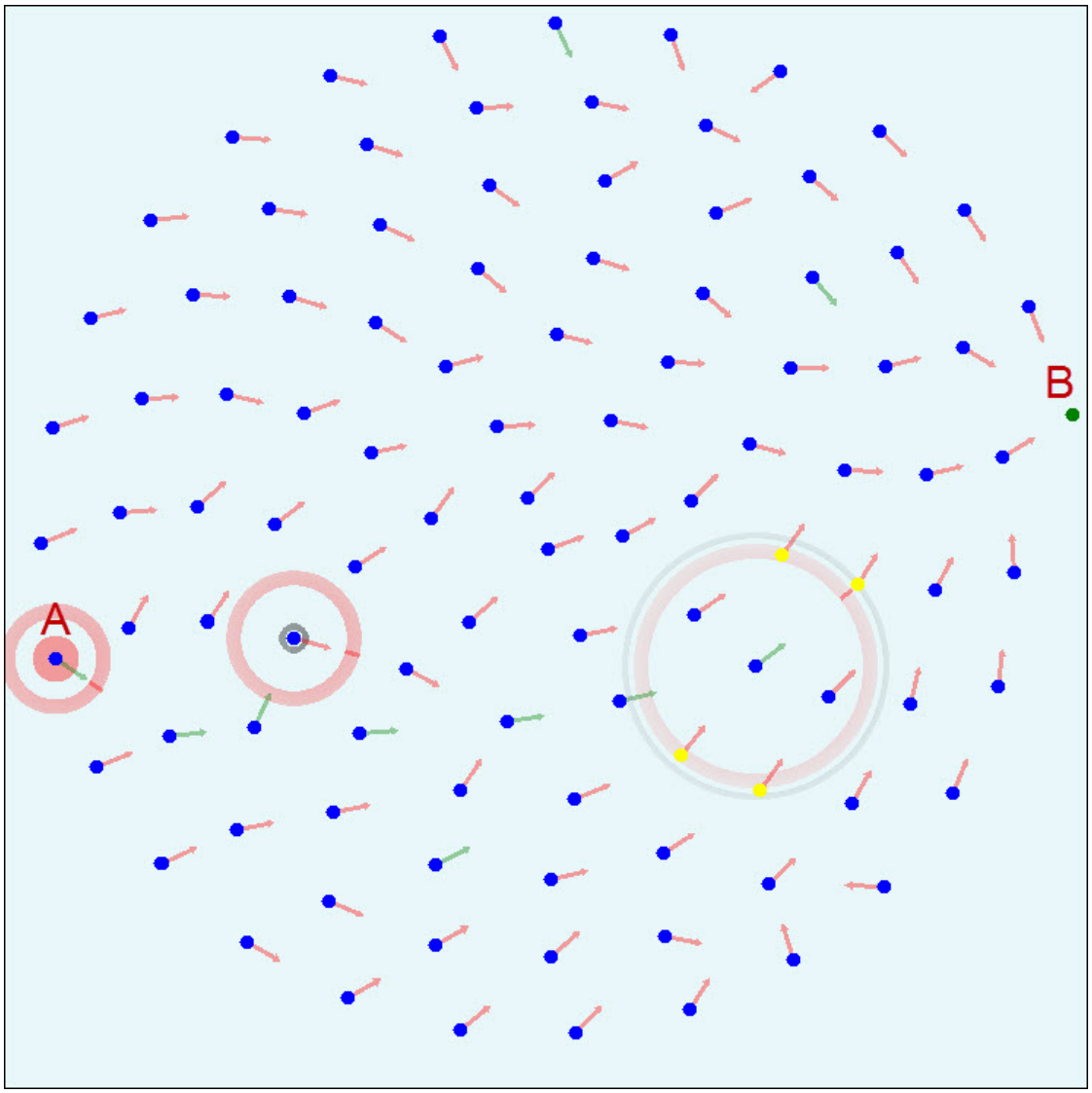

Figure 39. Node **A** completes Search and again begins sending Notification messages. Two application events were missed while node **A** was in Search.

In Figure 39, the network is restored. The overall problem is that node failures follow the data path in the reverse direction to the source node. Once the source node is affected, events are lost and the success rate drops. The example above is

a clean one where only two events did not make it to the sink node. Simulation runs with much higher rates of failure have a source node that is affected much more than in this example, often due to multiple node failures in the network.

Other problems resulting in major network failure are caused by a failure to recover from Search. It has been said that, due to the constraints of the UWA medium, the Search process has been limited to two attempts. If both attempts fail in finding any valid routes (due to collisions), the node will float indefinitely. This is an artificial failure we've imposed on the original protocol, but it is necessary to preserve basic network functionality.

4.3.3 Event Frequency

The calculated event frequency exacerbates the issue. Since the Search process takes a significant amount of time, it is highly likely that further events will dive directly into the area affected by Search. The more events that enter an affected area, the lower the likelihood that any such events will reach the sink. These messages are stored for transmission so, at the completion of the Search process, they will reach the sink, albeit with increased delay. When the Search process fails, which is the more likely case, the additional lost Notification messages will only increase the failure rate.

The event frequency is calculated in Section 3.2.1, which relates the propagation delay in UWA to that of RF. This conversion only relates the delay between a single pair of nodes. When a larger network exists, the transmissions of neighboring nodes may also affect such transmissions, complicating this rate conversion.

Event frequency is examined in [1] *(Figures 14 and 15)*, provided in Figure 40 in this thesis, indicating that at higher event rates delay and delivery ratios are reduced. It does not seem likely that doubling and tripling the event rate, as was done in the original article, will do more than erase the effectiveness of

this protocol in UWA entirely. Instead, one hypothesis may be that summing the frequency calculated by the path time differential between RF and UWA from Equation 22 and the estimated delay incurred by the Search process in Equation 23 will drastically increase the protocol effectiveness (and likewise reduce delay). The thought is that, given enough time, the network will tend to be restored.

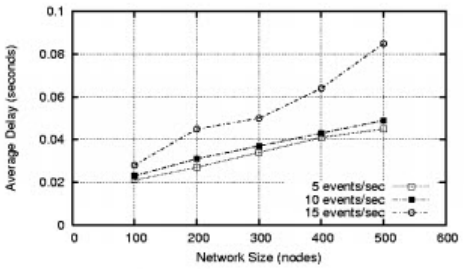

Figure 14. Average delay.

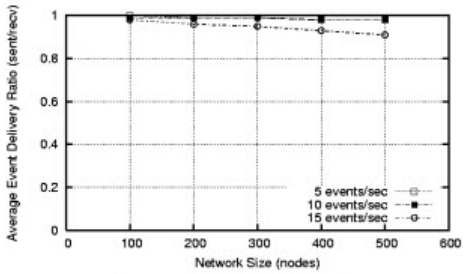

Figure 15. Average delivery ratio.

Figure 40. Original graphs from [1] comparing event frequencies.

An adjunct experiment was set up to test this hypothesis and prove that the event rate constitutes one of the major failures in bringing this protocol into UWA. In this experiment, we perform the exact experiments discussed in Table , changing only the event rate based on:

$$d_{\text{Search}} + R_{\text{UWA}} = 12.5 \text{ s} + 4.2 \text{ s} = 16.7 \text{ s} \tag{24}$$

Note that the above equation is rounded to the nearest tenth of a second while the experiment uses values in Hz of 0.06 Hz. The value from the equation above was also doubled, resulting in 0.03 Hz.

This experiment resulted in the graphs in Figures 41 and 42. The comparison of Delay between frequencies is actually opposite that of the original PEQ article, [1]. Notice that the delivery ratio shown by Boukerche et al. is consistently near 100% (and always greater than 90%). Lost data is not figured into delay values, since the delay is calculated by the sink node in the simulator. In our case it is likely that some of the Notification messages missed in the 0.24 Hz case are replaced by messages with greater delay in the 0.06 and 0.03 Hz cases.

Regardless of the delay comparison, the Average Delivery Ratio improves significantly as the time between events increases.

List of References

[1] A. Boukerche, R. W. N. Pazzi, and R. B. Araujo, "A fast and reliable protocol for wireless sensor networks in critical conditions monitoring applications," in *Proceedings of the 7th ACM international symposium on modeling, analysis and simulation of wireless and mobile systems (MSWiM'04)*, 2004, pp. 157–164.

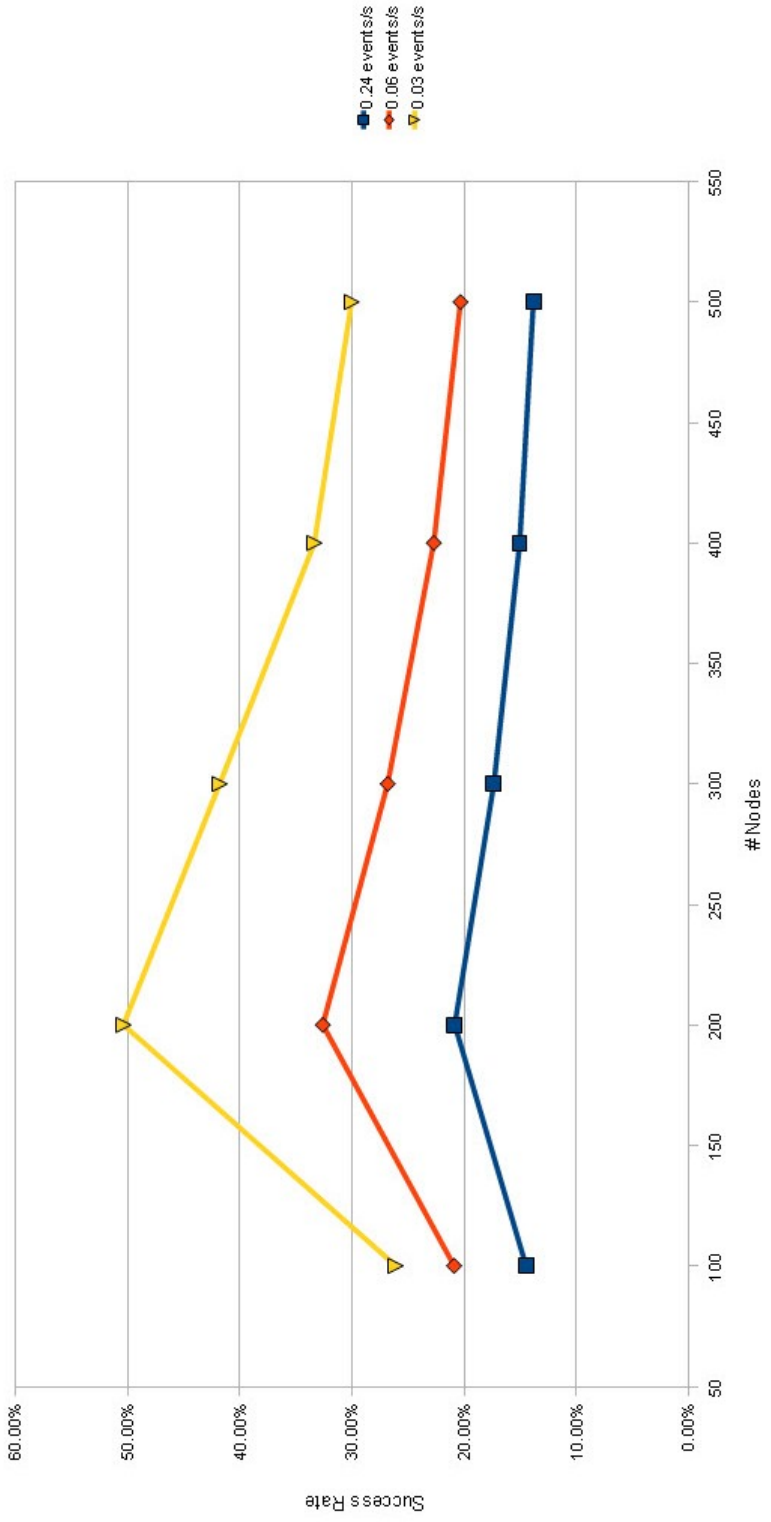

Figure 41. Success rate vs. network size for 0.24 Hz (4.2 s), 0.06 Hz (16.7 s), and 0.03 Hz (33.3 s) event rates.

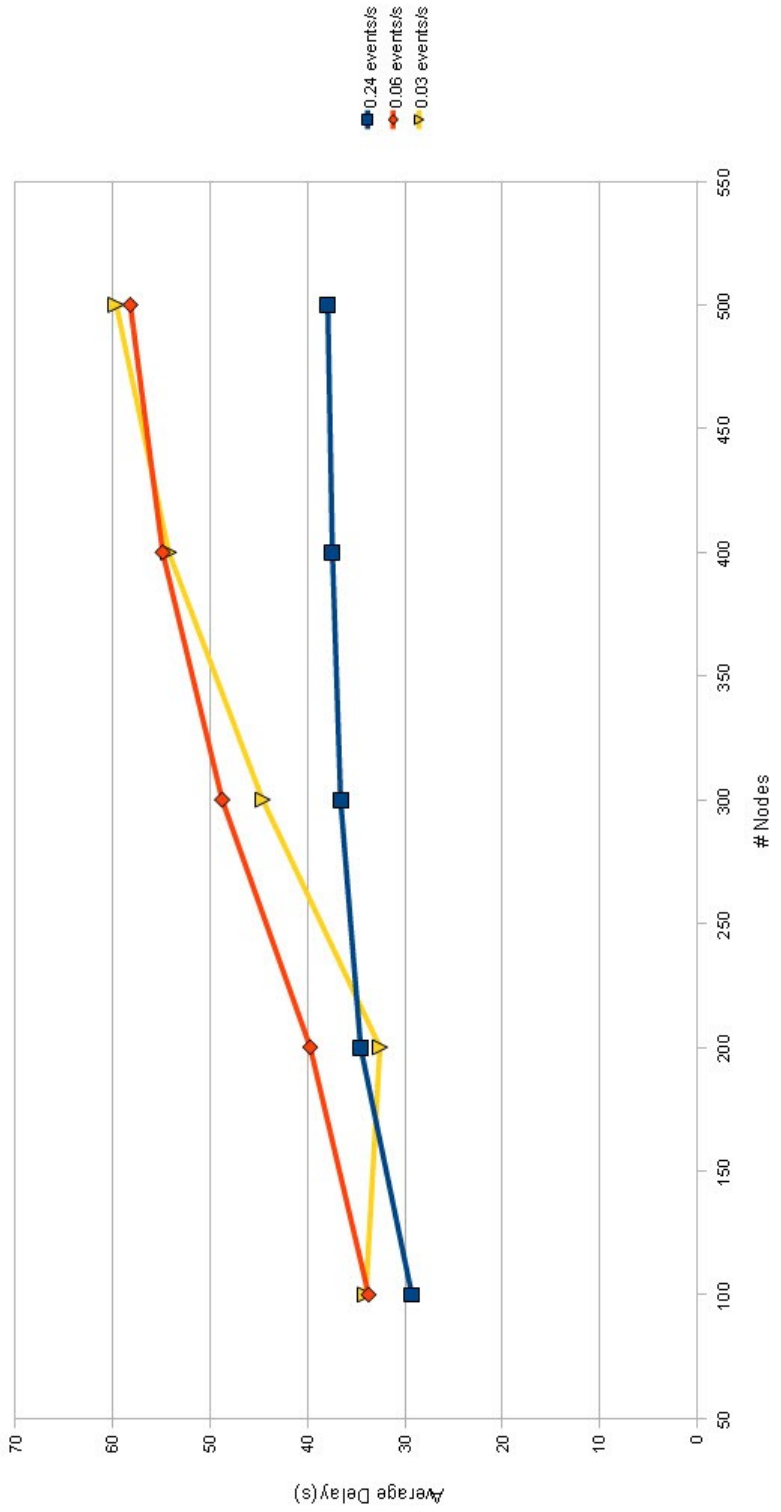

Figure 42. Average source to sink delay vs. network size

CHAPTER 5

Conclusions

This thesis set out to build a simulation engine and visualizer that enables the visual analysis of the highly latent underwater acoustic networks. RF networks are well known and RF protocols may be converted to the UWA medium. There are a number of differences between RF and UWA, but the most major of these are the propagation delay and throughput.

Throughput was not discussed in-depth in this thesis because it is an application layer problem and does not affect the network layer as significantly as delay. Propagation delay, as shown in this thesis, is a major, uncontrollable factor in determining the likelihood of collisions. All other things being equal, a UWA message is 50 times more likely than in RF networks to encounter a collision. Network algorithms optimized for RF networks may fail when subjected to a much greater collision rate, as seen with the PEQ Search algorithm defined in [1].

The much lower propagation speed in UWA allows a visual simulator to see the individual messages as they pass through the medium. While the same is true of small-scale RF sensor networks, the time a message takes to propagate between nodes is minuscule in comparison to the time between messages. This thesis proves that an animated, visual simulator of UWA networks is useful and more intuitive than simple data analysis of such networks.

The output of this thesis represents an example of what an animated, visual simulator can accomplish for UWA networks.

List of References

[1] A. Boukerche, R. W. N. Pazzi, and R. B. Araujo, "A fast and reliable protocol for wireless sensor networks in critical conditions monitoring applications," in

Proceedings of the 7th ACM international symposium on modeling, analysis and simulation of wireless and mobile systems (MSWiM'04), 2004, pp. 157–164.

CHAPTER 6

Recommendations

It is hoped that the release of this simulator will enable the visual study of UWA protocols. This is a living project, however, and future work can be split into two major areas: the continued study of UWA networks and the improvement of this simulator, used to study such networks. These two areas of improvement are discussed below.

6.1 The Study of UWA Network Protocols

In this thesis we examined one example of a sensor network protocol. The literature for wireless sensor networks has hundreds or thousands of protocol examples. Some of these have no place in UWA due to high bandwidth requirements; others could be of interest in UWA, but would need major algorithm changes due to the unique constraints the UWA medium imposes.

In [1], Akkaya and Younis categorize RF sensor network protocols into Data-centric, Hierarchical, and Location-based protocols. We've primarily focused on a Data-centric protocol, which means that a specific node ID is not queried, but instead specific types of data are requested from the network. One of these protocols, Directed Diffusion [2], is called a breakthrough by Akkaya and Younis, but would clearly have problems in UWA. It basically operates like PEQ in that an "interest" is sent out by the sink node and sources send data that matches the interest. The difference is that Directed Diffusion reinforces paths by generating statistics based on fixed intervals between data messages along multiple paths. It would be interesting to see this algorithm implemented in a UWA simulator, but the delay and increased collision rates will likely break the protocol.

Akkaya and Younis' Hierarchical protocols have node clusters where a lead

node responds back to a base station or higher level node network. This type of network has roots in early work in acoustic communications where all communications would be between one or more submersed clients and a base station. This can be seen in [3] and [4] where deep sensors communicate with a buoy or platform at the surface.

Location-based protocols involve geographic vector routing. This requires that the location of a node be known which, for underwater nodes, is a difficult problem on its own. If nodes are fixed, it is possible that the general location is known and such protocols can be used. Some vector protocols have already been proposed for UWA such as in [5]. These protocols have very limited usefulness if nodes are mobile.

Akkaya and Younis also describe Network flow and QoS-aware protocols. These protocols are completely aware of energy or network availability. As this isn't often the primary concern, many protocols in Akkaya and Younis' other categories also include these features.

None of these protocols are guaranteed to work in the UWA medium. The expectation is that few RF-based protocols would work without some degree of modification. Implementing these protocols and analyzing them in this simulator is the first step to understanding exactly what can be done with UWA communications.

The PEQ protocol requires work, especially the modification of the Search process. The first step in modifying this protocol for the UWA medium would be the entire removal of the Search process. Using the results of PEQ without Search as a baseline, the goal is to modify Search to improve upon that baseline. PEQ may not be the best protocol, but its simplicity helps it work in UWA to the degree it currently does.

6.2 The Improvement of the Modular Network Simulator

Just as this simulator is split into the core simulator and the visualizer, these are the two clear places where the Modular Network Simulator can be improved. The visualizer was initially written using a Microsoft 3D gaming package called XNA Game Studio. This imposes a certain frame rate and automatically drops frames when it can't keep up the speed. Unfortunately, the learning curve required of an optimized 3-D implementation led to the current implementation, which uses built-in drawing functions. The visualizer's major improvement is to optimize the graphics using either graphics system.

The core simulator had an initial design on paper before it was programmed. This initial design, while modular, did not take into account all desired modules and how they might interact with other modules. A combined, layered approach was used, as was discussed with Figure 13 in Section 2.2.1, reproduced in Figure 43.

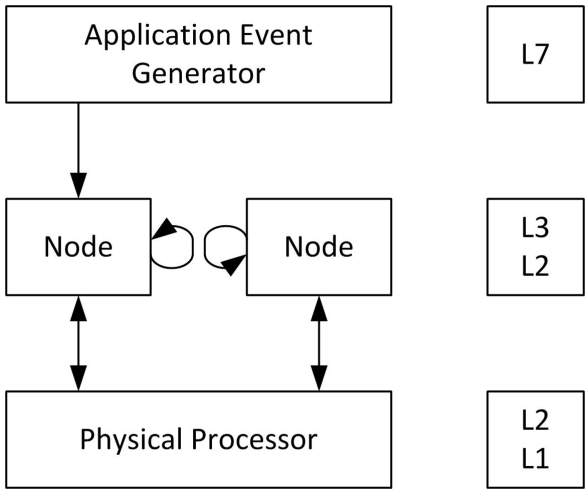

Figure 43. High Level Architecture used in the current Modular Network Simulator

The current core simulator does not have the flexibility to implement a Physical Processor that takes sending and receiving power levels into account without major modifications to the Node. This is a function that would nor-

mally be handled by the transducer and modem. Furthermore, without a distinct Layer 2, switching between a Time or Frequency Division Multiplex Access (TDMA/FDMA) and a Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) (as currently implemented in the simulator) link layer protocol would be impossible.

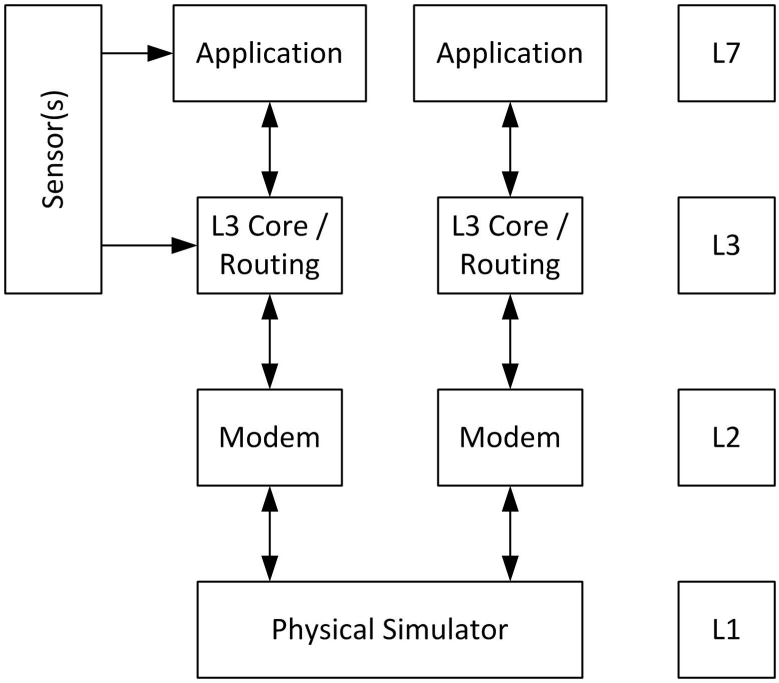

Figure 44. Proposed High Level Architecture for increased flexibility in the Modular Network Simulator

The proposed architecture would split the Data Link Layer functions currently shared by the Node and Physical Processor into a Modem layer. This way the Node would be represented in Figure 44 by L2, L3, and L7. The L3 Core / Routing would contain the bulk of the current Node. The benefit would be that L3 would only be responsible for sending a message to the modem. The modem would handle how that message reaches its destination. This provides for increased flexibility at the cost of increased simulator complexity.

Additionally, the Sensor(s) block in Figure 44 would be controlled by a Ap-

plication Event module, much as it currently does, but a new L7 Application layer would interpret the sensor data and initiate the event. This would make the event generation more flexible than it currently is. It would also allow for the simulation of the application layer in addition to the network and data link layers.

List of References

[1] K. Akkaya and M. Younis, "A survey on routing protocols for wireless sensor networks," *Ad Hoc Networks*, vol. 3, no. 3, pp. 325–349, 2005.

[2] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: A scalable and robust communication paradigm for sensor networks," *Proceedings of the ACM/IEEE 6th Annual International Conference on Mobile Computing and Networking (MobiCom'00)*, pp. 56–67, 2000.

[3] L. Freitag, M. Grund, S. Singh, J. A. P. J. Partan, P. A. K. P. Koski, and K. A. B. K. Ball, "The WHOI micro-modem: an acoustic communications and navigation system for multiple platforms," in *Proceedings of MTS/IEEE OCEANS '05*, vol. 2, 2005, pp. 1086–1092.

[4] J. Heidemann, W. Ye, J. Wills, A. Syed, and Y. Li, "Research challenges and applications for underwater sensor networking," in *Wireless Communications and Networking Conference (WCNC'06)*, 2006, pp. 228–235.

[5] P. Xie, J. H. Cui, and L. Lao, "VBF: Vector-based forwarding protocol for underwater sensor networks," in *Proceedings of IFIP Networking*, 2005, pp. 1216–1221.

APPENDIX

Symbols Used in This Document

TO APPENDIX

The following symbols will be used in succeeding sections of this document. Please refer back to this section as needed.

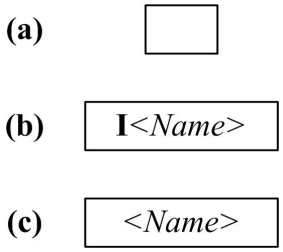

Figure A.1. Ways of defining Interfaces, Classes, and Objects.

In Figure A.1, a box such as (a) is a generic representation of an Interface, Class, or Object. An Interface will always begin with "I," as in (b), in this case named *<Name>*. The "I" will be followed by a capital letter, avoiding confusion with a class. A Class is simply named, as in (c): *<Name>*.

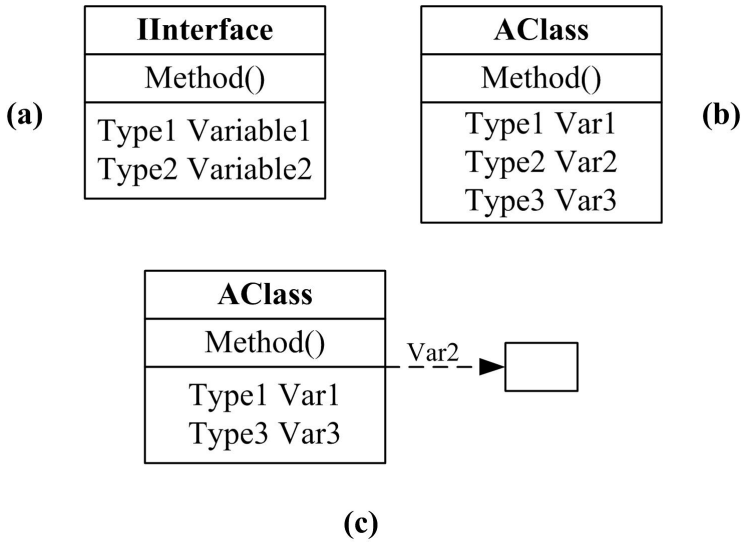

Figure A.2. Specification of Interfaces and Classes.

In Figure A.2, the specification of an Interface is given in (a) and a Class in (b) and (c). Zero or more methods are given in the middle section of each specification and Zero or more variables in the bottom section. If any section is empty it may be eliminated, but the name (IInterface or AClass in the figure) must be visible. If a variable is assigned to another object, as in (c), that variable can be omitted from inside the specification. Major groups of methods or variables may also be omitted from specific diagrams if they are off-topic (for example, methods relevant only to visualization when protocol specifics are being discussed).

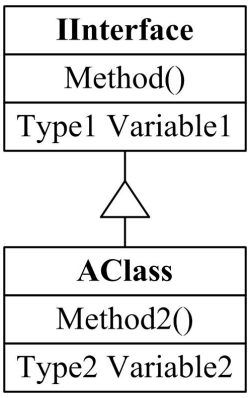

Figure A.3. A Class that implements an interface.

A Class that implements an Interface is specified with the symbol in Figure A.3. The Class need not specify the Methods or Variables that are specified in the Interface (the Class *must* contain these Methods and Variables), but may have its own Methods or Variables not required by the Interface.

A dashed line with an arrow indicates that an object is being created (usually by a specific method), assigned to, or accessed. In Figure A.4, (a) shows an object being created (by an unnamed method), while (b) shows its assignment to the named variables. What should be clear is that, if a variable is named, the figure will show an assignment relationship.

Figure A.5 shows the ways that external method calls can be described. A

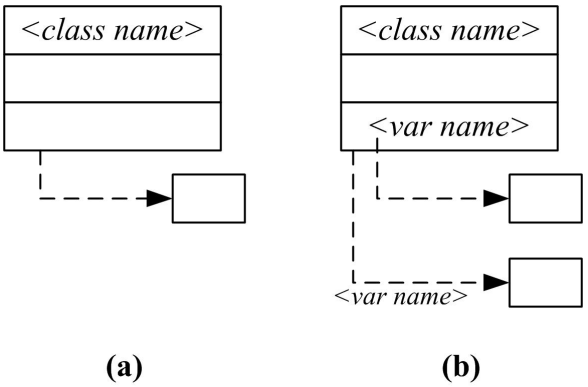

Figure A.4. Creating objects and assigning them to variables

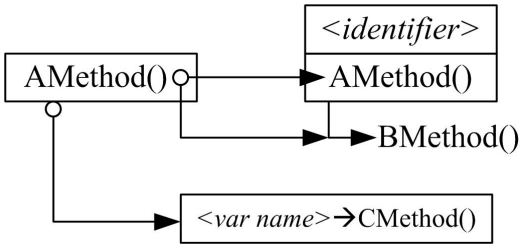

Figure A.5. Multiple ways of describing external method calls.

Class that has a reference to another Interface or Class can call the public methods specified by that reference. A small circle at the calling-side method with a solid line and arrow at the called-side method indicates that the calling method executes the called method.

Also note that "BMethod()" exists outside its specifying Interface or Class with a solid line and arrow. This is done to avoid diagrammatic confusion during an external method call.

BIBLIOGRAPHY

"Teledyne benthos undersea systems and equipment." [Online]. Available: http://www.benthos.com/

"Telesonar underwater acoustic modems: Subsea wireless communication," 2007, teledyne Benthos. [Online]. Available: http://www.benthos.com/acoustic-telesonar-modems-undersea-sub-sea.asp

Akkaya, K. and Younis, M., "A survey on routing protocols for wireless sensor networks," *Ad Hoc Networks*, vol. 3, no. 3, pp. 325–349, 2005.

Akyildiz, I. F., Pompili, D., and Melodia, T., "Challenges for efficient communication in underwater acoustic sensor networks," *ACM SIGBED Review*, vol. 1, no. 2, pp. 3–8, 2004.

Akyildiz, I. F., Pompili, D., and Melodia, T., "Underwater acoustic sensor networks: research challenges," *Ad Hoc Networks*, vol. 3, no. 3, pp. 257–279, 2005.

Akyildiz, I. F., Wang, X., and Wang, W., "Wireless mesh networks: a survey," *Computer Networks*, vol. 47, no. 4, pp. 445–487, 2005.

Al-Karaki, J. N. and Kamal, A. E., "Routing techniques in wireless sensor networks: a survey," *IEEE Wireless Communications*, vol. 11, no. 6, pp. 6–28, 2004.

Alcocer, A., Oliveira, P., and Pascoal, A., "Underwater acoustic positioning systems based on buoys with GPS," in *Proceedings of the Eighth European Conference on Underwater Acoustics*, 2006.

Boukerche, A., Pazzi, R. W. N., and Araujo, R. B., "A fast and reliable protocol for wireless sensor networks in critical conditions monitoring applications," in *Proceedings of the 7th ACM international symposium on modeling, analysis and simulation of wireless and mobile systems (MSWiM'04)*, 2004, pp. 157–164.

Braginsky, D. and Estrin, D., "Rumor routing algorthim for sensor networks," in *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, 2002, pp. 22–31.

Catipovic, J. A., "Performance limitations in underwater acoustic telemetry," *IEEE Journal of Oceanic Engineering*, vol. 15, no. 3, pp. 205–216, 1990.

Cleveland, C., "Deepwater horizon oil spill," Encyclopedia of Earth, February 2011. [Online]. Available: http://www.eoearth.org/article/Deepwater_Horizon_oil_spill

Cui, J. H., "The challenges of building scalable mobile underwater wireless sensor networks for aquatic applications," *IEEE Network*, vol. 20, no. 3, pp. 12–18, 2006.

Demirkol, I., Ersoy, C., and Alagz, F., "MAC protocols for wireless sensor networks: a survey," *IEEE Communications Magazine*, vol. 44, no. 4, pp. 115–121, 2006.

Faulkner, L., Granger, R., Hurst, P., Jankowski, W., Steinbrecher, D., and Tattersall, J., "Harbor Shield: A new technique for inspection of vessels below the waterline"," in *IEEE Conference on Technologies for Homeland Security (HST'09)*, 2009, pp. 221–226.

Foo, K. Y., Atkins, P. R., Collins, T., Morley, C., and Davies, J., "A routing and channel-access approach for an ad hoc underwater acoustic network," in *Proceedings of MTS/IEEE OCEANS '04*, vol. 2, 2004, pp. 789–795.

Freitag, L., Grund, M., Singh, S., Partan, J. A. P. J., Koski, P. A. K. P., and Ball, K. A. B. K., "The WHOI micro-modem: an acoustic communications and navigation system for multiple platforms," in *Proceedings of MTS/IEEE OCEANS '05*, vol. 2, 2005, pp. 1086–1092.

Freitag, L. E., Grund, M., Partan, J., Ball, K. A. B. K., Singh, S. A. S. S., and Koski, P. A. K. P., "Multi-band acoustic modem for the communications and navigation aid AUV," in *Proceedings of MTS/IEEE OCEANS '05*, vol. 2, 2005, pp. 1080–1085.

Gamma, E., Helm, R., Johnson, R., and Vlissides, J., *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, 1994.

Gang, L., Krishnamachari, B., and Raghavendra, C. S., "Performance evaluation of the IEEE 802.15.4 MAC for low-rate low-power wireless networks," in *IEEE International Conference on Performance, Computing, and Communications*, 2004, pp. 701–706.

Grimmett, D. J., "Message routing criteria for undersea acoustic communication networks," in *OCEANS 2007 - Europe*, 2007, pp. 1–6.

Harris III, A., Stojanovic, M., and Zorzi, M., "When underwater acoustic nodes should sleep with one eye open: idle-time power management in underwater sensor networks," in *Proceedings of the 1st ACM international workshop on Underwater networks (WUWNet'06)*, 2006, pp. 105–108.

Harris III, A. and Zorzi, M., "Modeling the underwater acoustic channel in ns2," in *Proceedings of the 2nd international conference on Performance evaluation methodologies and tools*, 2007, p. 18.

Heidemann, J., Ye, W., Wills, J., Syed, A., and Li, Y., "Research challenges and applications for underwater sensor networking," in *Wireless Communications and Networking Conference (WCNC'06)*, 2006, pp. 228–235.

Heidemann, J., Silva, F., and Estrin, D., "Matching data dissemination algorithms to application requirements," in *Proceedings of the 1st international conference on Embedded networked sensor systems*, 2003, pp. 218–229.

Heinzelman, W. R., Chandrakasan, A., Balakrishnan, H., and Mit, C., "Energy-efficient communication protocol for wireless microsensor networks," *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences*, vol. 2, p. 10, 2000.

Heinzelman, W. R., Kulik, J., and Balakrishnan, H., "Adaptive protocols for information dissemination in wireless sensor networks," *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking (MobiCom'99)*, pp. 174–185, 1999.

Intanagonwiwat, C., Govindan, R., and Estrin, D., "Directed diffusion: A scalable and robust communication paradigm for sensor networks," *Proceedings of the ACM/IEEE 6th Annual International Conference on Mobile Computing and Networking (MobiCom'00)*, pp. 56–67, 2000.

Intanagonwiwat, C., Govindan, R., Estrin, D., Heidemann, J., and Silva, F., "Directed diffusion for wireless sensor networking," *IEEE/ACM Transactions on Networking*, vol. 11, no. 1, pp. 2–16, 2003.

Jason, X., Rajesh, R. T., Anthony, M., and Mingyan, L., "AMRoute: ad hoc multicast routing protocol," *ACM Mobile Networks and Applications*, vol. 7, no. 6, pp. 429–439, 2002.

Joshy, S. and Babu, A., "Capacity of underwater wireless communication channel with different acoustic propagation loss models," *International Journal of Computer Networks & Communications*, vol. 2, no. 5, pp. 192–204, 2010.

Khemapech, I., Miller, A., and Duncan, I., "Simulating wireless sensor networks," *Technical Report, School of Computer Science, University of St. Andrews*, 2005.

Kilfoyle, D. B. and Baggeroer, A. B., "The state of the art in underwater acoustic telemetry," *IEEE Journal of Oceanic Engineering*, vol. 25, no. 1, pp. 4–27, 2000.

Kinsler, L., Frey, A., Coppens, A., and Sanders, J., *Fundamentals of Acoustics*, 4th ed. John Wiley & Sons, Inc., 1999, pages 448-449.

Krishnamachari, B., Estrin, D., and Wicker, S., "Modelling data-centric routing in wireless sensor networks," *USC Computer Engineering Technical Report*, vol. CENG 02, no. 14, p. 18, 2002.

Kussat, N. H., Chadwell, C. D., and Zimmerman, R., "Absolute positioning of an autonomous underwater vehicle using GPS and acoustic measurements," *IEEE Journal of Oceanic Engineering*, vol. 30, no. 1, pp. 153–164, 2005.

Lapierre, G., Beuzelin, N., Labat, J., Trubuil, J. A. T. J., Goalic, A. A. G. A., Saoudi, S. A. S. S., Ayela, G. A. A. G., Coince, P. A. C. P., and Coatelan, S. A. C. S., "1995-2005: ten years of active research on underwater acoustic communications in Brest," in *Oceans 2005 - Europe*, vol. 1, 2005, pp. 425–430.

Li-Ping, T., Wei-Kuan, S., Te-Chung, C., Sun, Y. S., and Meng Chang, C., "TCP throughput enhancement over wireless mesh networks," *IEEE Communications Magazine*, vol. 45, no. 11, pp. 64–70, 2007.

Library, O. A., "Acoustics toolbox," 2007. [Online]. Available: http://oalib.hlsresearch.com/

Lindsey, S. and Raghavendra, C. S., "PEGASIS: Power-efficient gathering in sensor information systems," *Proceedings IEEE Aerospace Conference*, vol. 3, 2002.

LLC, C. S., "Free C♯ implementation of the Mersenne Twister algorithm," 2003. [Online]. Available: http://www.centerspace.net/downloads/free-stuff/

Madden, S., Franklin, M. J., Hellerstein, J., and Hong, W., "TAG: a tiny aggregation service for ad-hoc sensor networks," *ACM SIGOPS Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI'02)*, vol. 36, 2002.

Manjeshwar, A. and Agrawal, D. P., "TEEN: a routing protocol for enhanced efficiency in wireless sensor networks," *Proceedings 15th International Parallel and Distributed Processing Symposium*, vol. 3, pp. 2009–2015, 2001.

Nemeroff, J., Garcia, L., Hampel, D., and DiPierro, S., "Application of sensor network communications," *IEEE Military Communications Conference (MILCOM'01)*, vol. 1, pp. 336–341, 2001.

OPNET, "Application and network performance with OPNET." [Online]. Available: http://www.opnet.com/

Ouimet, S. P., Hahn, M. J., and Rice, J., "Undersea communication network as a UUV navigation aid," in *Proceedings of MTS/IEEE OCEANS '05*, vol. 3, 2005, pp. 2485–2490.

Partan, J., Kurose, J., and Levine, B. N., "A survey of practical issues in underwater networks," *SIGMOBILE Mobile Computing and Communications Review*, vol. 11, no. 4, pp. 23–33, 2007.

Proakis, J. G., Sozer, E. M., Rice, J. A., and Stojanovic, M., "Shallow water acoustic networks," *IEEE Communications Magazine*, vol. 39, no. 11, pp. 114–119, 2001.

Qarabaqi, P. and Stojanovic, M., "Statistical modeling of a shallow water acoustic communication channel," in *Underwater Acoustic Measurements: Technologies & Results*, Nafplion, Greece, 2009.

Raysin, K., Rice, J., Dorman, E., and Matheny, S., "Telesonar network modeling and simulation," in *Proceedings of MTS/IEEE OCEANS '99*, vol. 2, 1999, pp. 747–752.

Robertson, C., "11 remain missing after oil rig explodes off louisiana; 17 are hurt," April 22, 2010 2010. [Online]. Available: http://www.nytimes.com/2010/04/22/us/22rig.html

Salva-Garau, F. and Stojanovic, M., "Multi-cluster protocol for ad hoc mobile underwater acoustic networks," in *Proceedings of MTS/IEEE OCEANS '03*, vol. 1, 2003, pp. 91–98.

Savvides, A., Park, S., and Srivastava, M. B., "On modeling networks of wireless microsensors," in *Proceedings of the 2001 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, 2001, pp. 318–319.

Sobeih, A., Hou, J. C., Lu-Chuan, K., Ning, L., Honghai, Z., Wei-Peng, C., Hung-Ying, T., and Hyuk, L., "J-Sim: a simulation and emulation environment for wireless sensor networks," *IEEE Wireless Communications*, vol. 13, no. 4, pp. 104–119, 2006.

Solis, D., *Illustrated C♯ 2008*. Apress, 2008.

Sozer, E. M., Stojanovic, M., and Proakis, J. G., "Underwater acoustic networks," *IEEE Journal of Oceanic Engineering*, vol. 25, no. 1, pp. 72–83, 2000.

Stojanovic, M., "Underwater acoustic communications," in *Electro/95 International. Professional Program Proceedings.*, 1995, pp. 435–440.

Stojanovic, M., "Recent advances in high-speed underwater acoustic communications," *IEEE Journal of Oceanic Engineering*, vol. 21, no. 2, pp. 125–136, 1996.

Stojanovic, M., Freitag, L., Leonard, J., and Newman, P. A. N. P., "A network protocol for multiple AUV localization," in *Proceedings of MTS/IEEE OCEANS '02*, vol. 1, 2002, pp. 604–611.

Sung, P., Andreas, S., and Mani, B. S., "SensorSim: a simulation framework for sensor networks," in *Proceedings of the 3rd ACM international workshop on modeling, analysis, and simulation of wireless and mobile systems*, 2000, pp. 104–111.

Sung, P., Andreas, S., and Mani, B. S., "Simulating networks of wireless sensors," in *Proceedings of the 33rd conference on Winter simulation (WSC'01)*, 2001, pp. 1330–1338.

Vasilescu, I., Kotay, K., Rus, D., Dunbabin, M., and Corke, P., "Data collection, storage, and retrieval with an underwater sensor network," *Proceedings of the 3rd international conference on Embedded networked sensor systems (SenSys'05)*, pp. 154–165, 2005.

Wang, C., Sohraby, K., Li, B., Daneshmand, M., and Hu, Y., "A survey of transport protocols for wireless sensor networks," *IEEE Network*, vol. 20, no. 3, pp. 34–40, 2006.

Xie, G., Gibson, J., and Diaz-Gonzalez, L., "Incorporating realistic acoustic propagation models in simulation of underwater acoustic networks: A statistical approach," in *Proceedings of MTS/IEEE OCEANS '06*, 2006, pp. 1–9.

Xie, G. G. and Gibson, J. H., "A network layer protocol for UANs to address propagation delay induced performance limitations," in *Proceedings of MTS/IEEE OCEANS '01*, vol. 4, 2001, pp. 2087–2094.

Xie, P., Cui, J. H., and Lao, L., "VBF: Vector-based forwarding protocol for underwater sensor networks," in *Proceedings of IFIP Networking*, 2005, pp. 1216–1221.

Yao, Y. and Gehrke, J., "The cougar approach to in-network query processing in sensor networks," *ACM SIGMOD Record*, vol. 31, no. 3, pp. 9–18, 2002.

Yu, J. Y. and Chong, P. H. J., "A survey of clustering schemes for mobile ad hoc networks," *IEEE Communications Surveys & Tutorials*, vol. 7, no. 1, pp. 32–48, 2005.

Zehavi, E. and Tiedemann, E., "The PCS CDMA system overview," in *Third Annual International Conference on Universal Personal Communications*, 1994, pp. 83–88.

Zhiyong, X., Rui, M., and Yiming, H., "HIERAS: a DHT based hierarchical P2P routing algorithm," in *Proceedings of International Conference on Parallel Processing (ICPP'03)*, 2003, pp. 187–194.