The ICoN Integrated Communication and Navigation Protocol for Underwater Acoustic Networks

by

Rupesh R. Kanthan

Submitted to the Department of Electrical Engineering and Computer Science

in partial fulfillment of the requirements for the degrees of

Bachelor of Science in Electrical Engineering and Computer Science

and

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2005

Author . . . . . .

Department of Electrical Engineering and Computer Science

September 1, 2005

Certified by . . . . . .

Joel Schindall

Professor of Electrical Engineering and Computer Science

Thesis Supervisor

Accepted by . . . . . .

Arthur C. Smith

Chairman, Department Committee on Graduate Theses

The ICoN Integrated Communication and Navigation Protocol for Underwater Acoustic Networks

by

Rupesh R. Kanthan

Submitted to the Department of Electrical Engineering and Computer Science
on September 1, 2005, in partial fulfillment of the
requirements for the degrees of
Bachelor of Science in Electrical Engineering and Computer Science
and
Master of Engineering in Electrical Engineering and Computer Science

Abstract

The deployment of autonomous underwater devices has increased dramatically in the last several years, presenting a strong and growing need for a network protocol to mediate acoustic communications between devices. This network protocol must also provide an infrastructure for acoustic navigation, while ensuring that provisions for communication and navigation do not interfere with each other. To approach this difficult problem, we begin with a discussion of the limitations of traditional networking protocols when subjected to the complexities introduced by the underwater acoustic environment. We then present ICoN, a proposed network protocol, designed to integrate acoustic communication and navigation and optimized to operate in the low-bandwidth, high-loss underwater environment. A working description of ICoN and a discussion of its features are followed by analysis of the protocol through simulation, indicating its potential for improved performance over traditional networking protocols. The simulation results are reinforced through real-world experimental validation of ICoN, which, though limited, appears to confirm the effectiveness of the new protocol. We conclude with possible future extensions to ICoN, discussing various methods that might increase its potency in dealing with more demanding underwater acoustic applications.

Index Terms: acoustic communications, ad-hoc networks, baseline navigation, network protocol, underwater networks, wireless networks

Thesis Supervisor: Joel Schindall
Title: Professor of Electrical Engineering and Computer Science

Acknowledgments

Special thanks to Dr. Anthony A. Aponick for inspiring and setting in motion the collaboration between industry and academia that made this work possible, and guiding its progress as a tireless consultant for the two years this project underwent development.

Special thanks to Lee Freitag, who served as our direct analog at WHOI through the development of this project and the major facilitator of all real-world testing, as well as the architect of the modems that this project was instigated by.

Thanks to Dr. Thomas Swean, ONR, for enabling the funding for the final year of the project, and empowering the Woods Hole-MIT link that gave the project application and purpose.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The last decade has seen a dramatic increase in the requirement for underwater devices with the ability to exchange information. Historically, such communication links were limited to point-to-point transmission via acoustic modems, such as oil rigs communicating with a shore facility, or deep sea sensors passing messages to dedicated surface buoys for RF transmission to a base station. Today's applications are growing far more numerous, with the desire for small devices to perform tasks like oceanographic monitoring, ship hull inspection, deep-sea bottom mapping, and defense-related countermine warfare. These devices may be stationary, utilizing active or passive sensors and transmitting information via acoustic modems to a central location only when necessary. On the other end of the spectrum, devices may be mobile, actively collecting information at high speeds and emitting data acoustically at regular intervals.

With the property of motion, devices must be able to navigate in their environment, especially those designed to operate autonomously. Underwater, where GPS is unavailable, devices typically use active sonar navigation, emitting pings against transponders located at known coordinates in order to triangulate their own positions. The communication and navigation systems both share the acoustic channel, posing a large arbitration challenge to such devices. As both communication and navigation are conceptually independent and encompass their own set of requirements, a method of arbitration between the two independent systems must be created to allow them

to function dependently, a handicap not experienced by any network to date[10].

Furthermore, the underwater environment introduces unfavorable physical properties that affect network operation, most specifically high latency and low bandwidth. With acoustic transmissions that travel 200,000 times slower than typical wireless networks and baud rates as much as 1,250,000 times less than typical wired LANs, even the simplest of traditional networking techniques become difficult or impossible to accomplish. Network-layer protocol overhead must be kept to an extreme minimum, and simple retransmit on packet loss is inadvisable. Collisions must be infrequent, but rapidly detected in order for the network to quickly recover without wasting precious time. Navigation information must be provided to devices frequently enough to prevent crashes or loss, but without damaging communication signals in transit. These communication signals are nontrivial, often taking the form of mission-critical commands, or extended data packets representing imaging data, which may burden the acoustic channel for many seconds. The network protocol must have a mechanism for analyzing message types, so it may selectively discard packets that are non-critical, but must guarantee the receipt of messages that are essential to the mission.

The proposed integrated communication and navigation protocol, termed ICoN (Integrated Communication and Navigation), seeks to lessen or eliminate these challenges in the underwater environment for both stationary and mobile devices equipped with acoustic modems. Rather than requiring point-to-point or centralized polling systems, ICoN abstracts any arbitrary collection of nodes as a network and gives it a decentralized topology, based on fully asynchronous communication. With the introduction of generic nodes and links, and elimination of synchronous behavioral requirements, a resilient network layer can be constructed, incorporating a rudimentary adaptive behavior system which can also serve as a base for more intelligent ad-hoc structures in the future.

Chapter 2

Related History

The field of underwater acoustic communications is not new. Several communication topologies have been used, with their behavior dictated by their application. Before exploring the application-nonspecific protocol presented in this paper, it is useful to review existing architectures and constituent link types, discuss their strengths and weaknesses, and examine the useful technologies each employs that contributed to the features contained in the ICoN protocol.

2.1 Point-to-Point Topology

The first step towards the design of any network is the implementation of robust point-to-point links[1]. Early applications of underwater acoustic modems functioned in this strictly two-node, single-link manner. In these systems, nodes are each assigned a unique ID. To communicate, Node A simply sends an acoustic transmission into the water via its attached modem, with an embedded field containing the ID of its destination, Node B. In fixed-location applications, both nodes typically use directional phased arrays to maximize the energy amplitude in the direction of their known destinations. If the destination modem detects that the packet is addressed to it, it passes the message along to the connected node.

This process is implemented in a number of different ways, dependent on the listening method of the modem. Depending on the configuration, all modems might

pick up all messages they detect and pass them to their attached nodes, leaving it up to a higher layer to decide which messages to listen to. Other modems only pass messages with their own ID to the node. Modems optimized for power consumption may rely on an introductory transmission with embedded destination information to determine if the packet is for them, before turning on a more sensitive, power-intensive receiver.

In this paper we will assume that all modems are sensitive to all packets they detect, passing all of them to the protocol-implementation firmware. The protocol-implementation firmware decides which messages to pass to a node. For simplicity, we will refer to a node device (which may be a sensor unit, autonomous vehicle, or manned craft), the protocol firmware, and the attached modem collectively as a node.

Most existing bilateral point-to-point systems, have no safeguards to ensure that messages do not collide. As water is a linear medium, messages can pass through one another and continue towards their destination, retaining their original waveform. However, if portions of two messages, regardless of intended destination, reach the listening node at the same time, both messages interfere and are unintelligible to the listening node. A node also has an inability to hear incoming transmissions if it is currently in the process of broadcasting a message, similar in effect to half-duplex systems. Given the limited communication load required of simple two-node systems, and a tendency toward unidirectional information flow, collisions are generally not an issue. However, composing a multi-node network out of unregulated point-to-point links is unadvisable due to the rapidly-increasing probability of message collision.

2.2 Master-Slave Architecture

Topologies consisting of multiple nodes are frequently organized in a master-slave architecture. In this arrangement, one node is chosen to be the master, which typically doubles as the node that relays information to a base station via directed acoustic modems or above-surface RF transmitters. As with point-to-point architectures, nodes are given ID numbers, with the master node typically assigned an ID of 0.

Such an architecture can easily take advantage of a time division multiplexing (TDM) scheme where time is broken down into n slots, where n is the number of slave nodes in the system. However, instead of giving each node a long slot where it can either send its whole transmission or remain silent and waste network time, a more efficient master-slave based system uses round-robin polling, where the master sends a brief data request packet (DRQ) to the network, indexed sequentially across all n slave nodes. When a node receives a DRQ encoded with its own ID number, it now has authorization to use the shared acoustic channel to immediately transmit any desired data to the master node. When the master node is satisfied it has received all the expected data, either by internal means, or a timeout, it sends a DRQ to the next slave node in sequence.

No possibility of a collision exists in this architecture, as the master has complete control of when any node in the system sends a transmission, provided the master initiates this polling routine correctly. There are several drawbacks, however. As the number of nodes increases, but the chance of a typical node having data to send remains low, a large percentage of time is wasted polling nodes that have no data to report. The master node, if it is a remote buoy, consumes power continuously in an idle network. Most critically, the dependence on a master node creates a single point of failure that can render the whole network useless should the master malfunction or be cut off from the network.

In a stationary sensor net, this may simply demand a maintenance team to replace the buoy, which is likely an acceptable shutdown period for this application. However, this round-robin polling scheme is used in many mobile situations as well, where multiple autonomous underwater vehicles (AUVs) may be rapidly combing an area, conducting joint missions. Should the central master node be disrupted, even for a brief time, the dependent slave AUVs can fail their mission, which could mean consequences ranging from missing a target, to crashing into a ship's hull or seabed causing destruction of the AUV.

In addition to being a simple central-point-of-failure matter, forcing a central master control node greatly limits possible network topologies, and restricts mission

capabilities. A sensor net, if each node was given the power to decide when to transmit, could be scalable to very large n. Mobile nodes could function at very large n as well, but also be given more intelligent network-level behaviors, such as the ability to issue directions directly to other nodes on demand.

A master-centric topology also limits the physical extent of a network, forcing all nodes to lie within the master's acoustic transmit range. Several master nodes could be used to blanket a large region, but it is not clear how master nodes would arbitrate which master polled which slave nodes in the network, especially if the slave nodes were mobile, without the very complex algorithms that are common to devices such as cell phone towers. Systems that have solved this problem use tremendous amounts of overhead to arbitrate the handoff of slaves from one master to another, taking advantage of the high bandwidth available to them, a methodology not practical in the low bandwidth underwater environment.

In a mission where several unmanned underwater vehicles (UUVs) wished to co-operatively inspect the hull of a large tanker for possible leaks and commit repairs, a master node would need to follow the team around the ship ensuring no UUV fell into its acoustic shadow, also limiting the UUV team to all work together in only one area. If the UUVs could communicate freely with each other, multiple teams could work simultaneously on different parts of the ship, with a freedom to wander. The ability for each node in a network to initiate communications on its own accord is the most important fundamental in nearly every network we use today, making it very likely that it will be proven vital to the evolution of the underwater acoustic network.

2.3 Wired Networks

In the course of developing a protocol to handle the underwater dilemma, it is important to not only examine existing network topologies, but also the properties of the node-to-node links themselves. Though the underwater environment prompts a wireless network, important developments in wired networks led to many techniques used in wireless networks today.

The first evolution of wired multiple-node networks relevant to the underwater network issue is the token-ring network, which is a direct analogue to the round-robin master-slave polling architecture. This network has the same disadvantages of the master-slave system in that a token is passed around the network, used to choose the next node to transmit sequentially, and thus the network does not scale well for large n. In this configuration, there exists no central master node that handles arbitration, and thus no central point of failure. However, the token ring assumes a wired network where a token could never get lost by falling out of the wire, a situation that would be very common in the high-loss acoustic environment. Also, a token ring assumes a predictable topology where each node can hear the previous node in the sequence, something not guaranteed in an acoustic network where nodes may be mobile.

The solution to the sequential-waste problem is the Ethernet, where messages could be generated by any node at any time, and simply dumped into the ether, a wire that is shared and listened to by every node in the network. This is exactly how the water functions for an acoustic network. All nodes monitor the Ethernet, and when a packet comes by that has their destination ID, they read the packet from the Ethernet, otherwise the packet is ignored[15, 5, 16].

The arbitration involved in the multiple-source problem is handled by the Carrier Sense Multiple Access/Collision Detect (CSMA/CD) protocol. As messages travel across an Ethernet at the speed of electrons through copper, each other node on the Ethernet can tell almost instantly when the Ethernet is busy; this is the ability to Carrier Sense. With CS, the chance of a node transmitting while another node is also doing so is tremendously reduced.

If it so happens that two nodes decide to transmit at exactly the same time, each of the transmitting nodes will almost instantly hear each other on the Ethernet; this is the ability to Collision Detect. Upon detecting the other node's transmission through CD, each node stops transmission, waits a random interval, and tries to send the packet again. In a high-bandwidth Ethernet line, the collision avoidance feature of CS and the ability to recover through CD are enough to provide a fast, simple, yet resilient network.

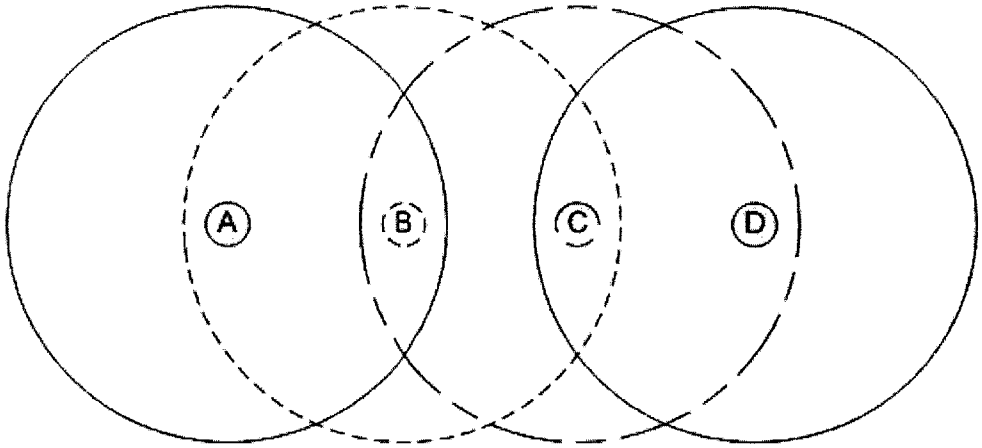

Figure 2-1: Illustration of "hidden terminal" problem. Nodes A and B can hear one another, as can B and C. However, A cannot hear C, as it is a terminal that is hidden from A. Thus, C may transmit, not knowing A is curently transmitting and cause a collison.

However, these simple tools assume a wired network where every node can hear every other, regardless of distance. As soon as we remove the range-independent Ethernet and replace it with a loss-prone, range-dependent medium, the ability to reliably CS or CD is lost.

2.4 Wireless Networks

The biggest challenge faced by the designers of wireless protocols like 802.11 was the CS problem, manifested in a scenario known as hidden terminal. Consider nodes A, B, and C in Figure 2-1. These nodes are arranged in a straight line, with A and B barely in range of each other, and B and C barely in range of each other. Thus, A and C are not in range of each other. In an Ethernet network, if A transmitted to B, C would hear A's transmission on the line. However, in wireless, where ranges are limited, when A transmits to B, C does not hear the transmission; it is the hidden terminal. Thus, C's CS ability is lost, and, seeing the channel is free, decides to send a transmission to B as well. Both the signals from A and C reach B at the same time, a collision results, and both messages are lost[4].

Worse still, neither A nor C realize their packets were disrupted by each other,

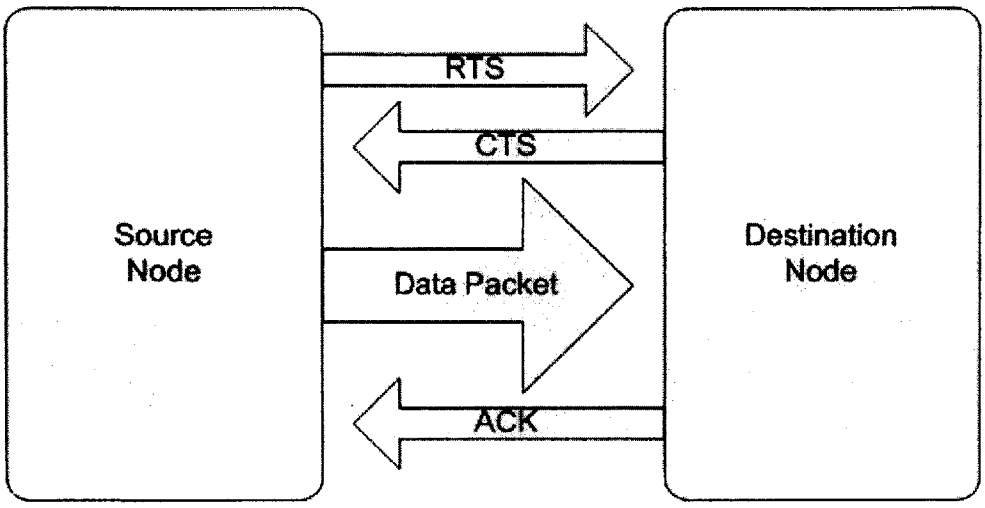

Figure 2-2: RTS-CTS-data-ACK handshaking sequence.

because the two nodes cannot hear each other. Thus, the ability to CD is also compromised. With both arbitration devices that are used in Ethernet disabled, wireless networks have to rely on an alternative system to carrier sense and collision detect virtually[3].

The solution is a message-based protocol that employs a sequential handshaking scheme between nodes that wish to transfer information. The four relevant packets are Request to Send (RTS), Clear to Send (CTS), Data (DAT), and Acknowledgement (ACK)[4].

To illustrate this handshaking scheme, consider nodes A through D in Figure 2-1. These nodes are arranged in a straight line with each sequential pair just within range of each other, such that concurrent transmissions from both neighbors of a node will interfere and be lost. Node B wishes to transmit a packet to Node C.

Node B begins by sending a brief RTS packet with C's address as its destination. As we are dealing with wireless, the packet is broadcast omnidirectionally from an antenna, and will be heard by both A and C (but not D, which is out of range). Upon hearing the RTS destined for C, Node A realizes Node B's intention to eventually transmit a data packet to Node C, and Node A silently considers the channel reserved by B for its use.

Node C receives the RTS packet, decodes that it is meant for it, and in response,

sends a CTS packet encoded with B as its destination. Node D and B (but not A) hear this packet. Node D responds in the same manner as A, silently considering the channel reserved by C for its use.

Node B, retrieving C's CTS, realizes the opening handshake of the protocol is complete, and B and C now have the channel reserved, which is known to all nodes within range of either B or C due to the RTS and CTS packets. B is free to transmit the intended data packet to C, and the DAT packet is sent, containing all the information B wishes to transmit to C in the first place. This packet can theoretically be as long as B wishes, because the channel is safely reserved for its use, but in reality, packets are typically limited to a timeout period to help recovery in the chance of failure, as will be detailed later. When the packet is complete, A and C sense that the transmission is no longer taking place, through either silence, or a final checksum visible in the packet. A knows the transmission is complete and B no longer has the channel reserved, but sets a short timer and waits for the final handshaking packet, ACK, to finish its work, for the benefit of node D.

C, on completion of reception of the DAT packet, generates and broadcasts the final of the four packets, the ACK packet, encoded for B's destination. This ACK is simply used by B to know that C received the whole DAT packet successfully, and it was not disrupted by environmental or other factors. B's job is done and the cycle is complete. D receives the ACK packet as well, which indicates to D that the cycle has completed, and C no longer has the channel reserved. Thus, the network is now silent, no node believes another node has the channel reserved, and any node can now send an RTS packet to reserve the channel for use to transmit to any other node.

Utilizing this message-based handshaking protocol, all but the most rare conflicts are handled in a well-arbitrated manner. To deal with lost packets, as the air is a lossy medium, each node sets timeout values at each phase of the handshaking cycle. If an expected message is not heard in an acceptable period of time, the message is presumed lost, and depending on which node detected a loss, a node might resend a message, assume the channel now free, or consider its cycle reset and try again at a later time. The combination of a set of timeout timers and the handshaking protocol

yield a virtual CS system, and an ability to recover from failure without CD[7].

2.5 Acoustic Properties of Water

An underwater acoustic network is closely related to in-air wireless networks. As with wireless, the abilities to CS and CD do not exist. Contemporary wireless protocols like 802.11 take many liberties in how aggressively they send packets, pumping packets rapidly and backing off when they detects high loss rates. They are able to use this scheme because when a packet is lost, the cost to retransmit it is very low; at bitrates as high as 11mbps a lost packet can be resent almost instantly. In addition, losses and hidden terminal situations can be detected in fractions of a second due to the extremely low latency afforded by transmitting EM signals at speeds close to that of light, over relatively short distances.

Limitations of sound propagation in water limit today's acoustic modems to speeds as low as 80 baud, making their data throughput over 1,250,000 times slower than Ethernet networks, and nearly 140,000 slower than wireless 802.11b. Underwater, the speed of sound is typically around 1,500 m/s, which is roughly 200,000 times slower than EM signals in wireless networks. With the diminished baud rates underwater and the tremendous increase in latency, lost packets are much more wasteful to the network, and recovery is not as easy as simply retransmitting packets until they get through.

These difficulties are representative of an ideal, pristine water environment. In reality, active bodies of water are far more hostile places for acoustic transmissions. Seabed surfaces are always irregular, building underwater sandbars and holes through natural wave activity. This multifaceted bottom acts as a reflector to acoustic signals like sets of dispersed mirrors, bouncing messages off different planes towards the destination node with different delays. This is a phenomenon known as multipath, and in addition to being a serious link-layer issue, multipath and other reflection-type anomalies can disrupt the careful timing required by many networks[18, 17, 11].

Wired and electromagnetic networks can occasionally suffer catastrophic effects

that disrupt their operation, such as shorting an Ethernet cable, or severe sunspot activity for in-air. But for underwater acoustic networks, a catastrophic effect is as common as a passing motorboat. Underwater networks must be prepared for frequent and total disruptions to all communications, due to other uncontrollable acoustic factors in the environment, and also exhibit the ability to rapidly recover and resume normal operation.

Perhaps even worse are the intrinsic salinity and temperature limitations of bodies of water. Gradual salinity gradients across a near-shore network can cause a variation in the speed of sound from one end of a network to the other, leading to unpredictable latencies and fouled timing calculations. The largest barrier to acoustic communications is the thermocline, a sharp boundary existing in all bodies of water between deep, cold water and warmer surface water. The thermocline is essentially impenetrable by acoustic communications, effectively cutting all nodes above the thermocline from those that venture just below it. Nodes that are disconnected from the network in such a manner must be prepared to weather the failure, but achieve a quick recovery as soon as contact is reestablished.

A protocol to handle communications underwater must be message-based to deal with the CS/CD issue in the same manner achieved with wireless, but many changes must be made to not only solve the problems of high latency and low bandwidth, but ideally function as a behavior system as well, with parameters that can be tailored to specific networks to optimize network goals.

The ICoN network protocol is designed for networks that are organized to solve a problem cooperatively. The protocol provides handles to allow network administrators, either human or automated, to optimize performance by tweaking goals like fairness, throughput, and fault-tolerance.

Chapter 3

The ICoN Protocol

The core principle of the ICoN protocol is fairness among all nodes participating in the network. Networks are assumed to be directed towards a cooperative task, though nodes are not likely homogenous. By this, we mean that each node has it's own communication and navigation requirements, which are all different among nodes of different configurations, mission types, and manufacturers. In this type of cooperative network, it is in every node's best interest to supply all other nodes with its required bandwidth and navigation pings, because the success of the cooperative mission likely depends on the success of every other node.

To avoid adding an excessive amount of overhead and thus reducing the already limited bandwidth available underwater, the philosophy of the ICoN protocol is intelligence over individual utilization. The prime value of the proposed protocol is not in pumping large amounts of data from one point to another, but in allowing a large number of varied nodes to accomplish their missions, accommodating their diverse requirements in a cooperative network infrastructure. Therefore, it is imperative that sending schemes be very conservative. The real value of the system comes from intelligent learning algorithms that each node can use to independently develop its perception of the network, rather than repetitive, aggressive probing.

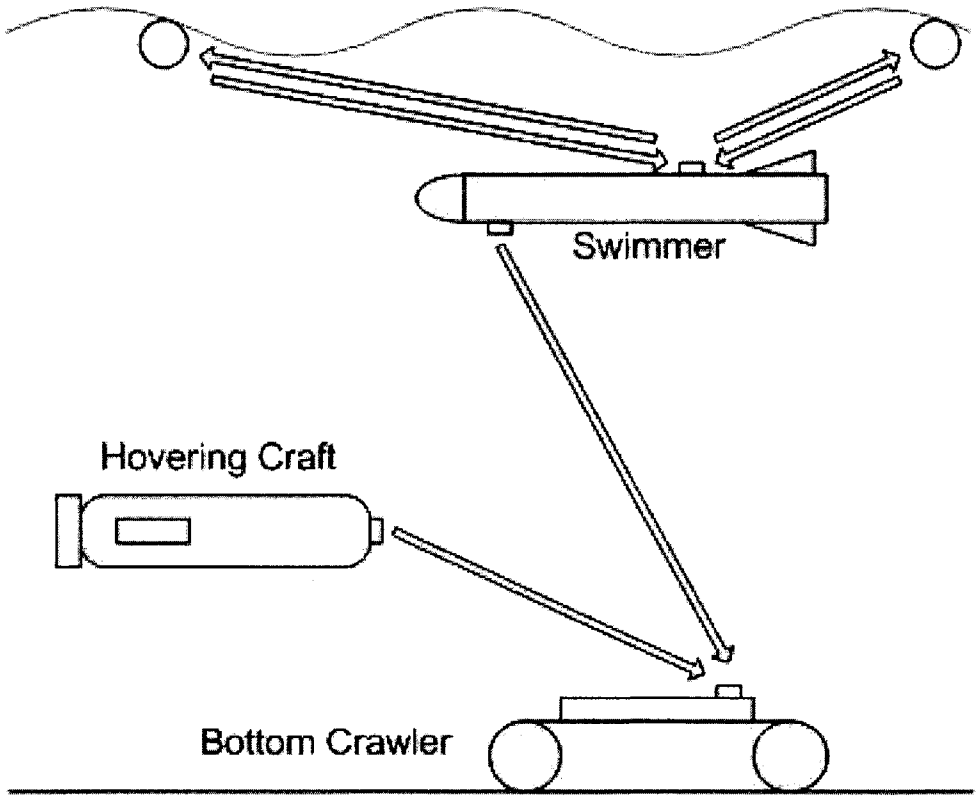

Figure 3-1: Conceptual situation involving three different classes of AUVs. Each AUV is equipped with acoustic transducers allowing it to communicate with the others. Additionally, all types of mobile AUVs use the same transducers to conduct periodic navigation pings to determine their locations from multiple navigation transponders.

3.1 High-Level Behavior System Concept

As an example, we will examine a hypothetical autonomous network attempting to examine a ship's hull underwater without the need to dry-dock, consisting of fast-moving survey AUVs, slower detailed-imaging AUVs, and repair AUVs. The survey AUVs, due to their speed, require navigation data every few seconds and thus need to emit an active sonar ping frequently, or they might crash into the ship's hull. They scan the ship's hull rapidly, detecting possible damage points as quickly as possible, and sending messages to the awaiting imaging AUVs to take a closer look at those suspect locations. Upon receiving an inspection request from a survey AUV, an imaging AUV spends time scanning the area more closely, and, depending on whether the anomaly is a clump of barnacles or a patch of rust, may dispatch a command to a repair AUV to come and effect the repairs. At the conclusion of the job, the repair AUV might send a confirmation to a shore base that a repair was executed at that certain location.

From this example, we quickly see three different levels of requirements in both communication and navigation from different classes of nodes. The survey AUV's navigation ping requests take a high priority in the network, to prevent the rapidly moving AUVs from crashing. The messages indicating possible damage points are less critical. Some may be false alarms, and in any event, the missed points can always be picked up again on later sweeps, when a different imaging AUV is nearby to investigate the location. However, once a damage point has been confirmed by an imaging AUV, it is very important that a repair command be dutifully serviced by a repair AUV, since these are not false alarms but definite positive matches. The concept of different message types and priorities are the central basis of the behavior system of this protocol.

Type	Expansion	Priority	Ack
CMD	Command	Highest	Yes
PNG	Navigation Ping	High	No
STA	Status Message	Low	Optional
DAT	Unspecified Data	Medium	Yes

Table 3.1: The four message types of ICoN and relevant properties of each.

3.2 Message Prioritization

There are a total of four user-visible message types that are introduced in this protocol. A Command (CMD) is a mission-critical directive message issued by a node or control station to another node to instruct it to carry out a task, change internal parameters, or change its otherwise preprogrammed operation. A Ping (PNG) is an instruction issued by a node to its own modem to emanate a navigation ping to retrieve current location data. A Status (STA) is a special data packet issued from one node to another node or control center to carry periodic vital signs, location data, or current sensor readings. A fourth generic Data (DAT) packet contains all other types of messages that do not fit into these three categories. Most typical missions consist of the previous three message types, unless the necessity to transmit large amounts of data is present.

In addition to the user-visible message types, there are three handshaking packet types that are utilized internally by the protocol. Those packets are the Request to Send (RTS), Clear to Send (CTS), and Acknowledgment (ACK) packets.

3.2.1 Selective Handshaking

There are two main effects of using four different message types in the protocol, the first involving handshaking, the second involving weighting. Each message, based on its type, employs a certain rule that governs its delivery.

Each of the four message types is handshaked differently. A CMD, which typically carries mission-critical commands to nodes to begin mandatory operations is given the highest priority. Because of this, the rule surrounding a CMD packet is that it

must be successfully acknowledged by its destination. If it is found that the packet is impossible to deliver, the source node must be explicitly notified that the packet has not reached its destination after a predetermined number of retransmit efforts.

To precede the actual CMD packet, an RTS and CTS are shared by the source node and destination node. Once the channel has successfully been reserved by this opening handshake, the CMD packet is transmitted. To ensure that the CMD has been successfully delivered, the destination node must reply with an ACK packet.

The STA packet is reserved to carry typical periodic status information returned by a node. For stationary nodes this may include salinity and temperature sensor readings, ambient seismic noise, gas concentration data, and other information encoded into a single string. For AUVs, the string may hold location and depth information, search pattern data, and other vital internals. UUVs may return mission progress data and diagnostic information. This message type is non-critical and also subject to staleness. The rules for a STA message state that the older a status message is, the more likely it is to be no longer relevant, and a newer status message is always preferred over an old one.

As such, a status message does not need to be explicitly acknowledged, but an ACK option enables the capability to help combat hidden-terminal scenarios in networks where the *netsize* is close to the maximum range of modems. A sending modem does not need to explicitly inform its attached node that a status message was not successfully transmitted.

The PNG packet is a special packet. It is not represented by a string, as the other three packet types are. A PNG is a very brief chirp emanated by a modem that is meant to be received by a device known as a transponder. A transponder is simply a repeater device that responds to a chirp with its own chirp. Each mobile node in the field is programmed with the known GPS locations of the field transponders prior to the start of a mission. By listening for the response chirps from each of these transponders and measuring time-of-flight intervals, each node can triangulate its own GPS-relative location based on the stored transponder GPS data.

A PNG is very short even compared to an RTS, represented by two momentary

clicks, leaving only the amount of time they take to propagate across the network as their relevant delay. As such, the RTS-CTS scheme is wasteful and unwise, and a PNG is transmitted without any handshaking scheme. Other nodes treat a received PNG as if it was an instantaneous RTS-CTS-data-ACK sequence, leaving the channel immediately free once more. A PNG, by nature, does not require an ACK, but only the response chirps from the transponders in the network.

By setting up different selective handshaking rules in this network protocol, significantly less time and bandwidth are wasted on less critical packets, while more effort is expended ensuring that the critical packets, the CMD packets, get through the network.

A DAT packet is given an intermediate priority between a CMD and a STA, because it is unknown how critical the unspecified data payload is to a mission. As a rule, DAT packets must be followed by an ACK, due to the likelihood that a DAT packet is a part of a sequential train of packets that represent a large file or constant data stream. Encoded in each RTS-CTS-data-ACK set is a sequence number. In this manner, each ACK can be correlated with its associated DAT packet. This aids both the source and destination nodes in determining which packets in the sequence were lost, and how to reorder the packets at the destination end, should they be reprioritized and resequenced in the network.

3.2.2 Packet Loss Scenarios and Network Recovery

There are a number of different scenarios that may take place depending on which messages, if any, are lost. The system depends on nodes to timeout, that is, reset themselves to a starting state if a handshaking error occurs in the network. Thus, if a node is expecting a certain handshaking packet to inform the node that the channel is now free and the packet does not arrive in a reasonable time, the node can resume normal operation instead of deadlocking.

If an RTS packet does not make it to its destination, the destination node cannot respond with a CTS packet. Most nodes in the network will not detect anything, and will pretend the channel is still clear, allowing them to send their own RTS.

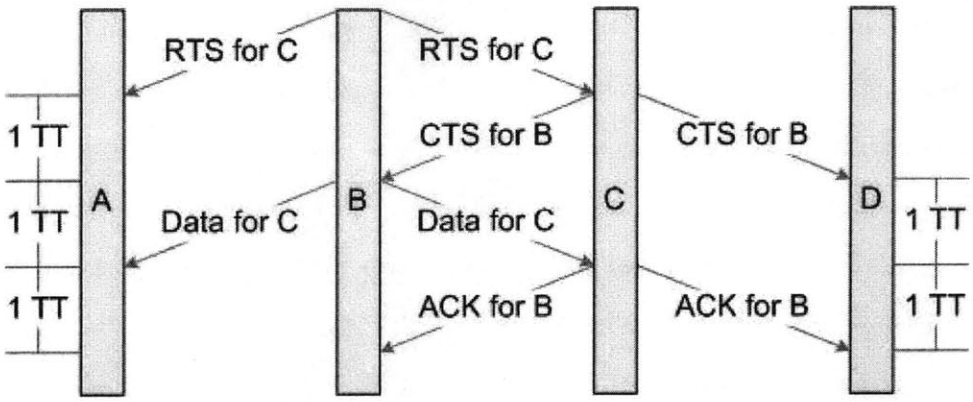

Figure 3-2: Timing diagram for timeout periods. One TT is given by the *traveltime* metric.

Those nodes that did hear the failed RTS, if they do not hear a data packet in a certain time, will experience a timeout, and consider the RTS as being a failed RTS. Then, they are free to transmit their own RTS. The source node, not hearing a CTS within the timeout period, will assume its RTS attempt failed and reset its internal timers, retrying the RTS at a later time. The recommended timeout metric is derived from the *traveltime* across a network, given by equation (3.1), where *netsize* is the maximum span of one hop in a network, and *spsound* is the speed of sound in water of the specific mission environment. This allows for the maximum time it could take for an expected message to arrive at a node from the maximum distance in a network, plus a little extra for anomalies resultant from the water. All actual timeout periods should be a multiple of *traveltime*.

$$traveltime = netsize/spsound * 1.2 \tag{3.1}$$

If the CTS packet is lost, the initial source node will assume its RTS attempt failed, reset its internal timers, and retry the RTS at a later time. The nodes that did not hear the RTS or CTS will proceed as normal, assuming the channel is free, transmitting their own RTS messages when they see fit. Nodes that heard either an RTS but not the CTS will expect a data packet after two *traveltimes*. If they do not see a data packet, they assume they simply missed it, and conservatively wait another *traveltime*, allowing the ACK to do its job in case the data packet was successful.

Nodes that heard only the CTS packet will wait for two *traveltime* periods, allowing time for the CTS to get back to the initial source node and the data packet to return to the initial destination node. If no ACK is heard from the initial destination node, it is assumed that the initial source did not hear the CTS, the following data packet was lost, or the ACK simply did not make it to our node in question. The channel is then free, regardless of what occurred.

In this situation, an optimization choice is available to the users of the network. One option is to be conservative, and now that the two timeout periods have passed, perhaps the handshaking failed and we can now send an RTS, or the data packet made it successfully and now that the channel is free, we can send an RTS. The other option is to assume that the handshaking was successful but the data packet may or may not have made it to its destination. Since the channel is already reserved, we can wait two extra *traveltime* periods to allow the two communicating nodes to attempt an immediate retransmit of the data. This intelligent retransmit scheme saves the overhead of extra RTS and CTS packets later, when the packet is scheduled to be sent again. In our implementation, we have chosen the conservative route for simplicity, leaving the option of intelligent retransmit for later developers.

A lost data packet will result in timeouts for all listening nodes, and will prevent the destination node from ever replying to the source node with an ACK. In this situation, depending on the intelligent retransmit option, the source node will either resend the data packet, or will also timeout and retry the data packet at a later time.

A lost ACK packet, depending on the intelligent retransmit option, will result in an immediate retransmit by the source node of the data packet, or will cause the source node to timeout and resend the data packet later. Other nodes in the network, based on whether they heard the data packet or happened to see the ACK packet, will either see the transaction as timed out, one *traveltime* period after seeing the data packet, or will see the ACK packet and determine the handshaking process completed gracefully.

As we can see, no matter where a packet may be lost in the whole handshaking routine, the network can always return to an idle state where everyone is given the

chance to send their own RTS; this shows a 100 percent ability for any node at any given time to recover from a packet-loss failure.

3.3 Weighted Probabilistic Transmit

The ICoN protocol was designed with a core requirement to use minimal amounts of overhead in order to conserve as much bandwidth as possible, since bandwidth is very expensive at the low bitrates that water allows. As a result, unlike most other ad-hoc protocols, ICoN uses a number of algorithms to maximize overall network performance cooperatively. These algorithms are entirely encapsulated within each node, without the need for inter-node coordination messages, to save as much bandwidth as possible for user-desired transmissions. The most important arbitration system used by ICoN is a system designed expressly for this protocol termed Weighted Probabilistic Transmit (WPT). The WPT system determines how likely a node is to transmit at any given time when the channel is free. By ensuring that nodes with more critical messages are more likely to transmit than nodes with less critical messages, the WPT system performs inter-node arbitration as well as maintains fairness across all nodes in a common network.

3.3.1 Queue Weighting

The basis for ICoN's WPT algorithm is queue weighting. Each node continuously maintains an internal priority queue of all messages the node wishes to output via its acoustic modem. When a node wishes to send a new message, the protocol adds this new message to its priority queue. As detailed in table 3.1, there are four distinct message types. Associated with each of these message types is a base *weight*, shown in table 3.2. The CMD has the highest weight, followed by PNG, DAT, and STA.

As messages are added to the node's priority queue, the weights of all messages in the queue are continuously summed to yield the queue's total weight, or *tweight*. In addition, the queue is periodically sorted to ensure the queue is constantly ordered with the message with highest weight first, decreasing as we proceed through the

Message Type	Sample *weight*
CMD	12
PNG	8
DAT	6
STA	5

Table 3.2: Possible weighting values for sample network. Modifications to these suggested weights dramatically change the behavior of the network.

messages stored in the queue.

3.3.2 Exponential Delay Curve

To ensure messages do not collide, the network as a whole must minimize the chances that two nodes will transmit messages at approximately the same time. If each node were to transmit its RTS messages as soon as it detected the channel was free, via timeout or a received ACK, there is a high likelihood that two RTS messages would collide with each other. To solve this issue, ICoN's arbitration mechanism is based on a variable length *delay* calculated individually by each node, which decides how long after the node detects the channel has gone silent does the node actually begin transmitting its RTS. By attempting to maximize the chance that any two nodes will have different *delay* values, the chance of RTS collision is minimized.

A node's *delay* parameter is derived from an exponential function known as the Exponential Delay Curve (EDC). The EDC exponential coefficient is derived from the specific *weights* given to each of the four message types, the size of the network, the baud rate, and the length of an RTS packet, and is thus, implementation-dependent. A sample EDC function is given in equation (3.2), where *rand* is a random value from -1 to 1, *rtslen* is the length of an RTS message in bits, and *bitrate* is the bitrate of a typical modem in the network.

$$delay = 10 * exp(-0.015 * tweight) + rand * (rtslen/bitrate + traveltime) \quad (3.2)$$

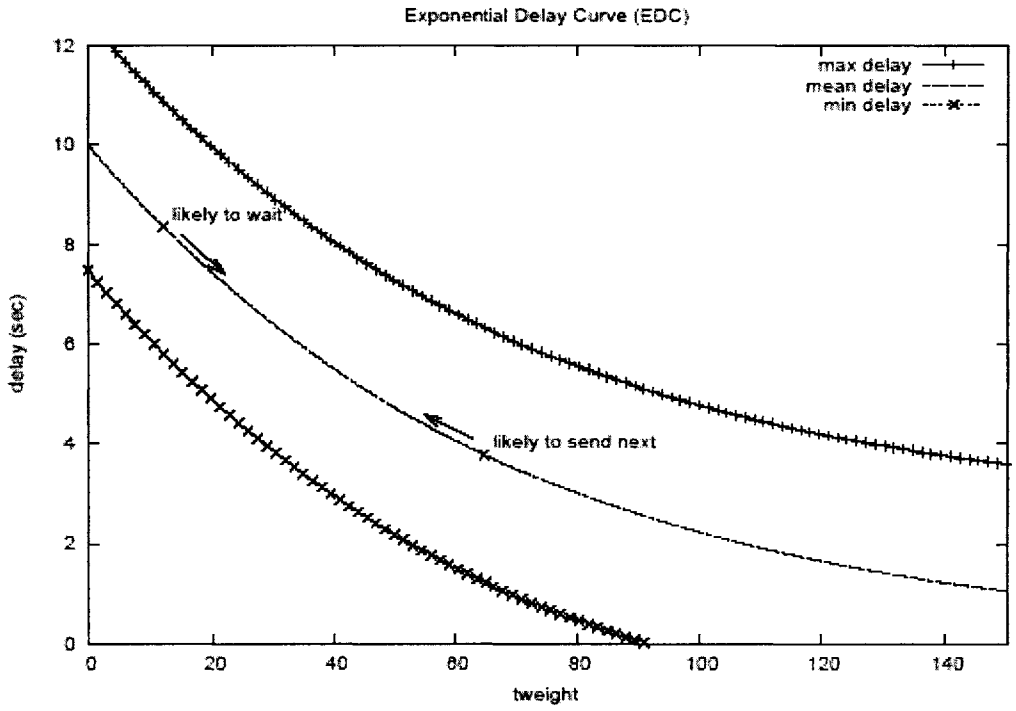

Figure 3-3: Exponential Delay Curve. Note the operating points of two nodes, and the direction in which they will most likely move along the curve in the near future.

The EDC sets upper and lower limits as well. It ensures that at the minimum possible total queue weight (one STA message pending), the *delay* is no greater than a maximum tolerable value, based on acceptable latency of the network. At the other end of the spectrum, if a node is heavily burdened with messages, the mean of *delay* should be no less than the amount of time it takes for a message to traverse the network plus the length of an RTS message at the operational bitrate.

The curve is tailored with the observation that typical acoustic networks operate at the less busy end of the graph with high probability, with the probability of existing at any given operating point decreasing as *tweight* increases. In normal operation, the majority of nodes in a network typically have very few messages queued for transmission. The exponential decrease of the EDC creates maximum separation between nodes of low *tweight*, where each node is probabilistically most likely to reside, while keeping within the aforementioned delay curve maximum and minimum parameters. As nodes queue more messages, their transmit *delay* value decreases. Thus, a node with a higher priority message type will typically transmit before a node with a low priority message type, and a node with more queued messages will likely transmit before a node with fewer queued messages.

To ensure proper operation in situations where all nodes have a typical transmission load, such as in a static sensor net, the curve is not a hard parameter, but a probabilistic distribution. Each time the desired *delay* is calculated, a randomization algorithm chooses a delay time in a window surrounding its *tweight*'s place on the curve. This window spans a region above and below the point on the curve by the length of an RTS divided by the bitrate, plus the specific network's maximum travel time. This ensures that there is enough variation each time that *delay* values are calculated, that two nodes with the same queue contents (and thus same operating point on the delay curve) will likely be able to choose delays far enough from each other that an RTS generated by one node can reach the other before it sends its own RTS.

The EDC faithfully maximizes the difference in delay before the RTS transmission for the region where nodes are probabilistically likely to lie densely, while still

maintaining a high degree of separation for the sparse distribution of nodes where they are less likely to operate. As we will see later, the EDC is not a fixed curve, and molds itself over the time of the network's operation to best fit emerging traffic pattern trends in real-time.

3.3.3 Convergence

A node's *tweight* is the independent parameter that determines where on the EDC a node is currently operating. A high *tweight* yields a low RTS-transmit delay, while a low *tweight* results in a longer delay. The reason for this architecture is based upon a cooperative network model. It is assumed that all nodes in a network are directed towards a common goal. In this manner, it is critical to afford nodes with more messages, and more critical messages, a higher probability of being the next node to transmit, versus nodes with fewer messages or less critical messages.

A node with a high *tweight* will wait a short *delay*, and then transmit a message, while the nodes with lower *tweight* stand by. Once the node with the highest *tweight* has successfully transmitted its message, it now has one fewer message in its queue, and thus, experiences a reduction in *tweight* equivalent to the weight of the transmitted message. Since each node works on a priority queue, this transmitted message is the queued message with the greatest weight for that node, making the decrease in *tweight* for that node likely to be significant.

As a result, the operating point on the EDC for that node shifts towards a lower *tweight*, and thus higher *delay*. The nodes that have been standing by, waiting for the node with greater *tweight* to transmit will either stay at their operating point on the EDC, or, if more messages have been queued while waiting, will shift on their own EDCs towards a higher *tweight*, and a lower *delay*. As messages are sent by the nodes with the lowest *delay*, and queued by the nodes with the highest *delay*, all nodes will tend to converge on a region on the EDC, as shown in Figure 3-3.

If this convergence happens successfully, it indicates that each node in the network is in equilibrium, with each of their transmission necessities adequately fulfilled by the network. This ability is adjustable by tailoring the weighting of each of the four

message types and how the queue manages the messages it contains, and is generally achievable in most network-traffic patterns.

3.4 Active Queue Management

The Active Queue Management (AQM) system is the second major innovation introduced in the ICoN protocol, and greatly affects the operation of the WPT algorithm. AQM provides a standardized way for a node to deal with its internal priority queue, and assures the queue conforms to a number of standards that WPT assumes for correct operation.

AQM enforces three major algorithms on the node's queue, in addition to maintaining invariants like *tweight*. These algorithms, like the rest of ICoN, are easily customizable for any desired network, with quickly modifiable basic parameters, and extended options to allow for optimizations in networks where traffic patterns are consistent or predictable.

3.4.1 Sorting

To maintain the priority queue structure of the queue, it must remain sorted with messages of the highest weight first, and lowest weight last. As such, a simple sorting algorithm is run periodically that keeps this invariant true. Being a queue structure, the first message is removed and given to the protocol for processing upon dequeue. New messages are added to the end of the queue upon enqueue. After each enqueue operation, the queue is sorted to install the newly added element in the proper location within the priority queue.

3.4.2 Excitation

To prevent message-starvation, and to combat staleness, an algorithm known as excitation is run periodically. As some message types, such as STA messages, are generally ranked with a lower weight than CMD messages, if a steady stream of CMD messages

is added to a queue, an STA may never reach the head of the priority queue within a node, and thus never be transmitted. In a similar fashion, as multiple CMD messages are added to the queue, depending on the implementation of the sorting algorithm, the most recent CMD messages might always be transmitted, neglecting older CMD messages that might be critical to the mission. In this scenario, CMD reordering also becomes an issue in missions where the sequence of messages is important.

In the network as a whole, a node with a low *tweight* as compared to the rest of the nodes in the network, will never get a chance to transmit, since its *tweight* will consistently be lower than the *tweight* of each other node. The lesser node's delay will always be higher than all other nodes, and its attempts to reserve the acoustic channel will always be beaten by the others. Likewise, any one node that can maintain a high *tweight* will have a monopoly on the acoustic channel, starving other nodes until they are backlogged enough to present a greater *tweight*.

The excitation algorithm alleviates most of these issues, promoting fairness across the network as a whole. Periodically, at an interval that can be configured for individual node types, the excitation algorithm examines the whole queue, and increases the weight of messages still waiting in the queue by a multiplier. In this manner, older messages will increase in weight, bringing forward in the queue and increasing their probability of being sent. A backlogged CMD message, for example, will therefore be moved toward the head of the priority queue, at the expense of more recent CMD messages. STA messages, if starved in the queue by CMD messages, will increase in weight and move forward in the priority queue, eventually jumping ahead of CMD messages in the queue. Nodes that have been starved for a short time will experience a cumulative increase in *tweight*, and be able to transmit, even in the presence of nodes with consistently high *tweight*, that would otherwise starve the more quiet nodes in the network[8].

The excitation algorithm allows customization of both the interval that determines how often it is run, and the multipliers that are individually set for each of the four message types. Modifying these four multipliers quickly changes the behavior of networks where enough traffic exists to allow queues to consistently contain pending

messages. However, care must be taken not to make the multiplier values too high or the excitation interval too short, such that all nodes in a network quickly accelerate along the WPT curve and end up operating in the high-congestion region, where all *delay* values are low and spaced close together, increasing the chance of collision. The parameters should be chosen such that nodes in normal operation should, with high probability, operate towards the low-congestion region of the WPT, allowing for maximum separation of delay values per node.

3.4.3 Filtering

The third major algorithm in ICoN's AQM is filtering. As a consequence of excitation, it is probable that a node that generates large numbers of status messages or rapidly occurring pings can obtain very large *tweight* values, though its messages are non-critical, and, due to the nature of STA and PNG messages, likely outdated. Thus, a major feature of AQM is the ability for a node to scan its queue, and ensure that at any given time, there is only one STA and one PNG message present. Since an STA string typically contains the most recent diagnostic or sensor data being reported by a node, it only makes sense to transmit the latest STA in the queue. In a similar manner, any PNG will return the current location information of a node, making multiple queued PNG messages redundant[6, 13, 9].

However, merely searching the queue for the most recently timestamped STA and PNG messages and eliminating the remaining STA and PNG messages is not quite correct. Continually replacing STA and PNG messages with newer ones knocks out messages affected by the excitation algorithm, inserting fresh, new messages with the standard low base weight. To correctly ensure that a STA or PNG message in a busy queue is able to properly increase in weight so it can be sent out, old STA and PNG messages are eliminated from the queue through filtering, and their weight transferred to the new, fresh STA and PNG messages, replacing their default low weight values. This effectively means each node keeps only one weight value each for STA and PNG messages as a whole, representing one STA and PNG message each.

In more application-specific situations, the filtering algorithm may be used to deal

with CMD and DAT packets as well, based on node purpose and design. The filtering algorithm can be modified by changing the rate at which filtering occurs, though since it is a maintenance algorithm rather than a state-changing algorithm, it should be run frequently, or be event-driven on every enqueuing of STA and PNG messages.

3.5 Learning and Adaptive Algorithms

The ICoN protocol contains a number of algorithms that learn based on network traffic patterns and adapt the previously described systems to optimize bandwidth utilization and throughput. These algorithms have the greatest effect in networks where traffic patterns are consistent and predictable. In networks where traffic patterns are constantly changing or erratic, the algorithms simply have no effect. One major algorithm handles scenarios where very little traffic exists in the network and the other functions when traffic is exceptionally high.

3.5.1 EWMA Curve Damping System

In networks where very little traffic exists, for example, in small sensor nets where each node is programmed to transmit a sensor reading once every minute in a round-robin fashion, a great deal of time is wasted using the EDC feature of this protocol. Given the low *tweight* of a node with only one STA message in its queue, each node will wait a long *delay* period before sending the message. This period is, as described earlier, dependent on the parameters chosen for the EDC system by the node designer, and can be made to be small. However, a designer may not be able to predict the right values for the EDC system in networks that are event driven, or incorporate nodes that may turn on and off at certain times. In these situations, where few nodes are transmitting at wide intervals, we would like the nodes to simply transmit their data as soon as possible, after a minimal random delay.

To accomplish this without specifically tailoring the EDC parameters to a tighter delay curve, the EWMA Curve Damping System (CDS) continually modifies the EDC function based on an estimated weighted moving average of network traffic frequency.

An internal parameter is maintained in each node that measures the last delay time recorded from the network; that is, the time between when the network was detected as going silent and the next detected transmission (which may be from the node itself). This value is averaged into an EWMA on each detected transmission cycle, yielding the average *delay* that the most busy nodes are running on in the recent past of the network. As with any EWMA, the EWMA weight multiplication parameter can be configured to make the CDS learn either faster or slower (deciding how recent the recent past is).

If the EWMA is high in relation to the node's operating point on the EDC, it indicates that the majority of nodes are operating at a consistently high *delay* value on the EDC. Since no nodes are operating at regions of shorter *delay* on the EDC, the majority of *delay* variation provided by the EDC is wasted, and a more shallow curve is called for. The CDS, based on the EWMA parameter it keeps and the current parameters of the EDC, determines a multiplication coefficient (in this case, smaller than 1) for the EDC system and applies it to the delay curve. By doing so, the EDC is made more shallow and less eccentric, reducing the unnecessarily high *delay* values applied by the unmodified EDC. As long as the randomization algorithm present in WPT is not modified, WPT will continue to provide enough variation around the EDC operating point (due to the rtslen/bitrate + travel time term) to minimize the probability of RTS collision. Following this model, we have eliminated most of the unnecessary latency created by the EDC, without diminishing our already established ability to avoid RTS collision.

The multiplication coefficient generated by the CDS can be either greater than or less than one, allowing the CDS to make the EDC either more or less eccentric, causing damping at low traffic patterns, and better delay separation in busy traffic patterns. Threshold levels are recommended in the latter case to guard against possibly extraordinarily long delay periods.

Due to the nature of EWMA, a network that has been operating in a stable traffic pattern for some time, and thus, operating with a CDS optimized EDC, will function sub-optimally the moment that the traffic pattern changes to a different pattern. If

the change is momentary, when the network returns to the old stable pattern, the EDC will remain at its optimal point. However, if the network changes to a new stable pattern, it will take several cycles for the EWMA to catch up, and allow the CDS to optimize the EDC to the new traffic pattern. Basing the CDS on an EWMA makes the network particularly resilient to temporary shifts, perhaps due to a new AUV wandering across an existing sensor net, but guarantees that the CDS will always adapt if a new stable pattern is desired.

3.5.2 Critical Zone Backoff Routine

The greatest difficulty facing the EDC system occurs when a large number of nodes build up a large number of messages to send, that is, have a high *tweight*, thus operating at a very low delay point on the EDC. This situation generally occurs when outgoing message volumes are high for most nodes in the network. However, this scenario poses a larger conceptual problem for the network in general, that is, the acoustic network only has a certain amount of bandwidth as a whole. If the sum of the desired outgoing traffic for all nodes in the network exceeds the network's total bandwidth, evidently the network will eventually fail.

Though this assumption must hold for the network over an extended length of time, it is conceivable that for a short period of time, more traffic than available bandwidth might be generated, and queues in a large number of nodes might back up, such as when a potential target wanders into a sensor net.

One option for handling such a problem is to give queues ample capacity to allow a large number of messages to back up in the queue, and adjust the EDC to be steeper to increase separation in *delay* values even at the low-delay region of the EDC. Then, we simply wait for each node's queue to slowly drain, once the object has passed.

However, where it is impossible to project the maximum quantity a queue should hold, or when the administrators of a network wish to retrieve only a sample of the excessive amount of data, a backoff routine has been devised to deal with these situations, similar in concept to congestion avoidance systems employed by high-end network routers today.

A node, by observing current traffic patterns, can easily get a feel for how busy other nodes are, by taking the same EWMA from the CDS, which quantifies the average delay time on the EDC where most nodes are operating, and comparing it to its own EDC. If the EWMA representing the observed *delay* values is very low, it indicates that at least one node is in a high-congestion region in the network. If the nodes own EDC is also operating at a region where the *delay* is comparably low, both nodes are at high congestion and are competing against each other for the channel. As this *delay* value gets smaller and smaller, we approach a region on the EDC known as the critical zone, where the difference in *delay* between nodes is small enough that RTS messages now have a high probability of collision.

By using this method, any node can detect whether it and at least one other node are both in that critical zone. To ensure that messages continue to get through, the nodes can increase the eccentricity of their EDC, as presented before, to increase the difference in delay, or either node can choose to activate the backoff routine. The critical zone detection algorithm periodically checks the delay EWMA, and compares it to its own calculated *delay* from the point on the EDC where it is operating due to its *tweight*. If its *tweight* is high, and both its own *delay* and the EWMA are very close together, the routine is activated.

The backoff routine scans the node's own queue, and begins eliminating unnecessary messages that are queued. As with all other algorithms, the manner of this paring can be customized. As a basic rule in the philosophy of ICoN, it is most important that any CMD and PNG messages be given the maximum chance to survive and be delivered. Thus, a simple solution is to wipe all STA and DAT messages from the queue. This can be done in multiple phases, depending on *tweight*, where first one type of message is wiped from the queue, and if necessary, the other is removed as well. Many implementations are possible, depending on the type and purpose of the node.

Depending on whether both nodes or one node trigger this algorithm, either both nodes will return to a considerate level of operation in the network and traffic will resume as normal, or one node will backoff and the other, suddenly seeing that other

nodes have eased their delay, will be free to transmit all its data at its existing delay derived from the EDC and its full *tweight*.

There are two ways to view this situation. One perception is that fairness has been violated. One or more nodes sacrificed their pending messages for the good of the network, and by doing so, a few nodes suddenly think the network is no longer very busy, and opportunistically take the liberty to send all their data. A second perception is best exemplified by a sensor network, where a potential target object triggers massive amounts of communication in the network, but it is only necessary for a small amount of that data to reach the network administration body. Many of the messages are redundant, indicating that a node has detected the same target that all other nodes also have. In this case, it is not only acceptable, but even preferable, for the majority of the nodes to drop their detection messages, promoting messages from only a sample of nodes to be successful. However, in networks where fairness is essential, the critical zone backoff routine should be disabled, and a better solution must be found to coordinate fairness between all nodes in the network.

3.6 Combined Protocol Analysis

The aggregate effects of different message types and prioritization, the WPT system, and the AQM system yield a stable network protocol that correctly deals with both communication and navigation messages in a best-effort probabilistic manner. Simple wireless-like handshaking handles the most basic transport-layer issues, while the philosophy of the WPT and AQM systems not only provide a method for minimizing RTS collision, but a whole behavior system for each node in a network.

The system works best when networks are cooperative, since nodes are designed to observe and anticipate the actions of nodes they can hear, each being courteous of each other as the network continues to develop. To allow optimizations in cooperative networks, many parameters can be easily adjusted in each algorithm, yielding infinitely customizable behavior patterns.

As a very basic methodology that comes from working in a low-bandwidth, high-

latency environment, the ICoN protocol uses neither probing nor detection messages to find the status of the network around it, nor nearest neighbor lists that are frequently maintained by ad-hoc networks by these means. Each node is self-contained in its traffic-detection abilities, utilizing only one-ended, passive techniques for network evaluation.

By integrating learning and adaptive algorithms, the core WPT system can be easily optimized for any network where traffic is relatively consistent, as most networks are. Furthermore, ICoN has the ability to adjust to any changes in general traffic patterns, without any need to reprogram or redeploy nodes. Small variations do not affect the operation of the protocol, due to long-history EWMA dependencies.

The ICoN protocol suffers limitations when congestion in the network is high and every node wishes to transmit a large amount of data. A backoff scheme has been introduced to combat this situation, however, there is no way to solve this problem while keeping traffic detection algorithms one-ended and fully passive. Work being done to increase bitrates for acoustic modems would tremendously reduce these limitations, in addition to providing other benefits, like greatly reduced probability of RTS collision. Shorter handshaking messages translate directly to tighter delay windows, making *delay* separation between nodes far more probabilistically likely, yielding much more robust operation of this protocol.

Chapter 4

Implementation

4.1 Historical Development

The history of the development of the ICoN protocol dates back to October 2003, shortly after AUV Fest 2003, held in August 2003 in Keyport, WA. Hosted by NUTEC (National UUV Test and Evaluation Center) at NUWC (Naval Undersea Warfare Center), and supported by ONR (Office of Naval Research) and NAVSEA (Naval Sea Systems Command), part of AUV Fest 2003 was to demonstrate the interoperability of AUVs of multiple types and manufacturers to accomplish a joint naval mission. The specific task was underwater mine detection and neutralization, funded as part of ONR's Very Shallow Water/Surf Zone Mine Counter-Measures Program. (VSW/SZ MCM).

The architecture used a central control node, employing round-robin polling. The individual AUVs communicated synchronously, using the polling scheme to arbitrate their communications to avoid collision. However, the AUVs used asynchronous navigation, broadcasting pings whenever they required a location fix. Watching navigation pings repeatedly collide with in-flight communications led to the realization that an arbitration scheme was necessary that incorporated both communication and navigation into a unified asynchronous protocol. In addition, the new protocol would have to support a broader concept of a network with an arbitrary number of nodes, instead of being a point-to-point system. Finally, the very low bandwidth available to

acoustic modems, combined with the high loss rates incurred whenever a disturbance occurred in the water, made it vital that overhead remain low, and recovery times be fast.

A collaboration between WHOI (Woods Hole Oceanographic Institution) and MIT was formed in December 2004, with the protocol issue delegated to MIT as part of the VSW/SZ MCM program, with WHOI focusing on underwater acoustic modem design. The partnership was orchestrated as a year-long investigation, with preliminary studies beginning in January 2004, and formal investigation beginning in June 2004.

The initial plan was to devise a protocol that would instantly overlay the existing transport layer, allowing implementation as a black-boxed translator layer between the AUV hardware and the acoustic modems. The first prototype of the protocol was dubbed the protocol implementation box, and was a single-board 586-based computer with serial RS-232 input and output. The PI box would simply be inserted in the serial link between any AUV and acoustic modem.

As the protocol developed, it became clear that this middle-of-the-road tactic was inefficient, and the protocol should take one of two forms. The first was to integrate the protocol into the AUV software, as an onboard API to interact with the modem. The second was to integrate it into the modem firmware, seamlessly joining the protocol and transport layers. As it was not feasible to modify the modem firmware, the first option was chosen, allowing the synergy of having easy handles into the protocol software to ease debugging, and provide greater operational visibility to the development environment.

In its current incarnation, the protocol software is a C++ based library, able to be placed in any AUV's code, though a full rewrite will eventually be required to clean up the heavy prototyping. The API offers function calls to send and receive all four types of messages over the network, clearly abstracting the barrier between application-level code and protocol-level arbitration, but also including cross-layer handles to monitor current protocol status and performance.

4.2 Protocol Implementation

The ICoN protocol is best implemented as integrated firmware onboard an acoustic modem to ensure that the network layer protocol cannot be decoupled from the transport layer mechanism. If this coupling is not enforced, any node can simply start transmitting data on the transport layer, disrupting all legitimate handshaking occurring on a network, leading to unfairness and starvation. As with the Internet, it is necessary to transmit all data utilizing some sort of protocol to ensure that the invariants that other nodes expect remain constant.

For first-run purposes, however, the protocol was implemented as a C++ library that runs on the node's onboard processor, rather than on the attached modem. To interact with the modem, the node was required to call protocol library functions. These functions allowed the capability to send each of the four message types, check to see if an incoming message has arrived, and to retrieve an arrived message.

As such, ICoN functioned as a clean API for the modem, in addition to being a network protocol, hiding all transport layer issues of the modem from the node. The abstraction barrier resulted in a situation similar to placing the protocol on the modem as firmware, because the details were hidden to the node of what was actually being transferred between the onboard library and the modem.

The protocol library commanded the modem to send messages via a serial link. The ICoN library handled all queuing, dequeuing, handshaking, and protocol algorithms. When it was time to physically send a message, a string was sent over the serial link to the modem and immediately dispatched to the network. Incoming network traffic was passed directly to the protocol library by the modem via the serial link, and was processed by the library.

The only limitation to this architecture was the capability of the node. Ideally, in this setup, a node would be designed with a multithreaded processor, with one thread explicitly assigned to a main function in the protocol implementation. This main function would continuously monitor the serial link, while running the protocol algorithms based on runtime timers, and calling library functions when events needed

to occur. On non-threading processors, the library is used in a polling configuration, where a catch-all function is called periodically by the single thread of the node's program. Via setting and checking delays and timers, the protocol is allowed to operate as desired, provided the protocol is polled frequently enough.

Since the serial port of a node is a physical hardware device and, therefore, hardware dependent, a virtual class needed to be created to abstract the send and receive functions of the port, specifically SendString(), IsString(), and GetString(). The manufacturer of the node is responsible for subclassing this virtual class, providing these three methods to the protocol library.

For simulation testing, a simulator was created in Java, and given a graphical user interface. Through a Java Native Interface layer, the Java simulator application is able to create as many instances of a node with attached protocol as desired, and allow those nodes to engage in artificial network traffic. The simulator mimics the water environment, functioning as an intermediary between all nodes. By faithfully reproducing propagation delays and collision effects, network traffic was effectively modeled for an acoustic environment for the initial troubleshooting of the ICoN protocol.

The final phase of evaluation involved a mobile testbench, equipped with three acoustic modems. The testbench used real acoustic modems in a true body of water, attempting to ascertain the effectiveness of this protocol in the everyday physical environment.

Chapter 5

Results and Evaluation

5.1 Testing and Evaluation Methodology

To rapidly collect a large amount of data regarding the performance of the ICoN protocol, a C++ based testbench was created in addition to the Java simulator, with a greater ability to vary individual component parameters. The C++ simulator chose random positions for each node created, taking into account pairwise travel times, as well as hidden terminal situations. Travel time jitter and random blackouts were also added to the simulation, as was random message loss. In addition, timing glitches were randomly inserted, to allow two pairs of nodes to effectively handshake across each other for dedicated access to the channel at the same time. This tested the ability of ICoN to recover from such situations that are bound to occur in the unpredictable underwater environment.

A four-node topology was organized, equipped with a centralized message-generation utility that randomly dispatched indexed messages to the four nodes, at various average rates. Messages in the system were tagged as the simulation progressed, indicating whether they successfully reached their destination node, or whether some phase of the handshaking cycle experienced a collision and was disrupted. The remainder of generated messages that went untagged were either dropped by the protocol due to staleness, or backed up in the nodes' queues due to inadequate sending bandwidth.

A second Control protocol was created to serve as a benchmark for the ICoN

system. The Control was equipped in the way that a decentralized acoustic network would be with today's existing technology, but without ICoN. Instead of using ICoN's four-stage handshaking, the Control set a random transmit timer for each message, and sent the desired message when the timer expired, with no overhead. Nodes hearing the beginning of a message held off their desired transmissions, until they determined that the current transmission ended.

Both ICoN and the benchmark protocol were subjected to varying average traffic rates dispatched to each node. Success and collision data was output to file for analysis. For greater detail on individual interactions, the Java simulator was consulted and frequently used to revise the operation of ICoN.

5.2 Model-Based Predictions

Initially, predictions based on simplified models of network traffic determined that at low sending rates, both the Control protocol and ICoN should perform similarly. Long delays between message generation would give each protocol plenty of time to send messages as they appeared, without messages accumulating in queues, though ICoN would add a significant latency to each message due to the EDC. This would make the Control protocol more optimal because messages are sent at once, eliminating the danger of messages going stale while waiting for the *delay* timer in ICoN's queue. At very low rates, the probability of collision is near zero, making the Control protocol safe.

As traffic rates increased for each node, there would be a period where the Control protocol would perform better than ICoN, when messages began building up in ICoN's queue, due to the imposed *delay* latency by the protocol. The Control protocol would not experience this queue buildup, and would pump messages out faster, resulting in a larger number of successfully delivered messages at the end of the trial. At these rates, probability of collision is likely low enough to only be a minimal factor.

As the delay between generated messages continued to diminish, traffic rates increased to the point where collisions started to affect the performance of the Control

protocol. From this point onwards, ICoN should perform significantly better in terms of collision avoidance than the Control protocol, due to its arbitration scheme, even though the *delay* inserted by ICoN will cause it to attempt to send far fewer messages than the Control during the fixed time interval.

5.3 Simulation Results and Evaluation

Extensive validation through simulation exposed some properties not identified though model-based prediction. As expected, at low sending rates, both the Control protocol and ICoN performed comparably, though the effect of collisions on the Control protocol was underestimated. This led to a modestly better performance (20 percent) of ICoN over the Control protocol at sparse sending rates, translating to roughly one message per node every 45 seconds, equivalently one message for the four node network every 11 seconds.

As sending rates increased, the number of messages delivered successfully to the destination node gently increased for both protocols, with ICoN steadily increasing its percentage lead over the Control protocol (peaking at 30 percent) for the number of successful messages per total generated messages.

The gentle increase for the Control protocol was resultant of two factors; the difficulty of sending packets due to an exponentially increasing chance of collision, and the sheer number of packets being generated with a dumb chance of success. To a point, the slim probability of each message getting through allows the number of successful messages transmitted using the Control protocol to increase as the number of messages generated increases exponentially, despite growing numbers of collisions, before congestion collapse occurs.

The gentle increase in ICoN initially follows the message generation curve as it begins to rise. As the protocol overhead introduced by the *delay* catches up with the frequency at which ICoN can transmit, the success curve diverges from the message generation curve. Since ICoN has reached a ceiling due to its overhead, one would expect it to level off; however, due to the ability to slide down the EDC as traffic

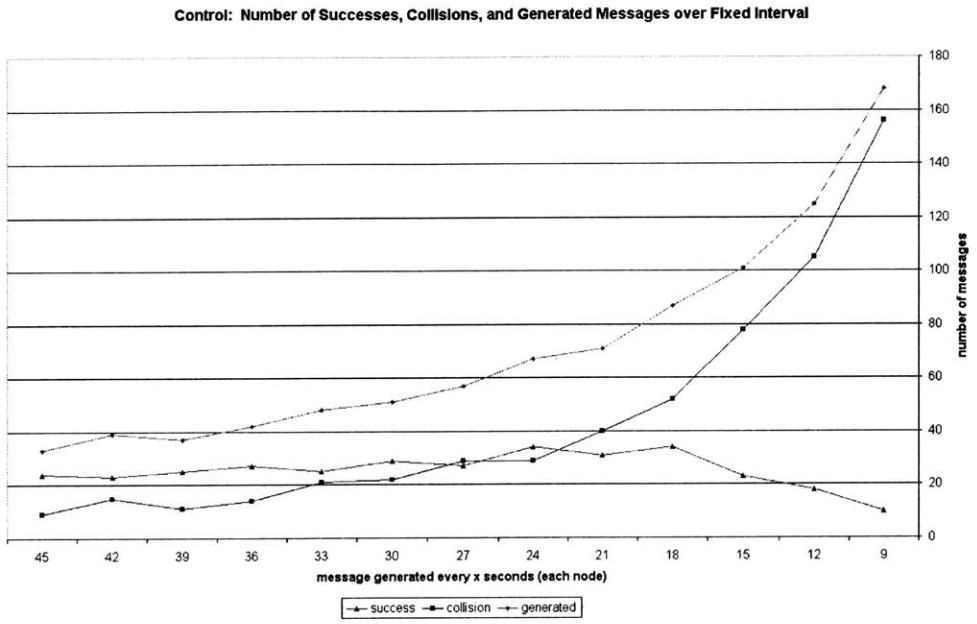

Figure 5-1: Successes and collisions in a fixed interval test using "Control" protocol. Note the congestion collapse as sending rates increase, and the reduction effect of collisions on successes.

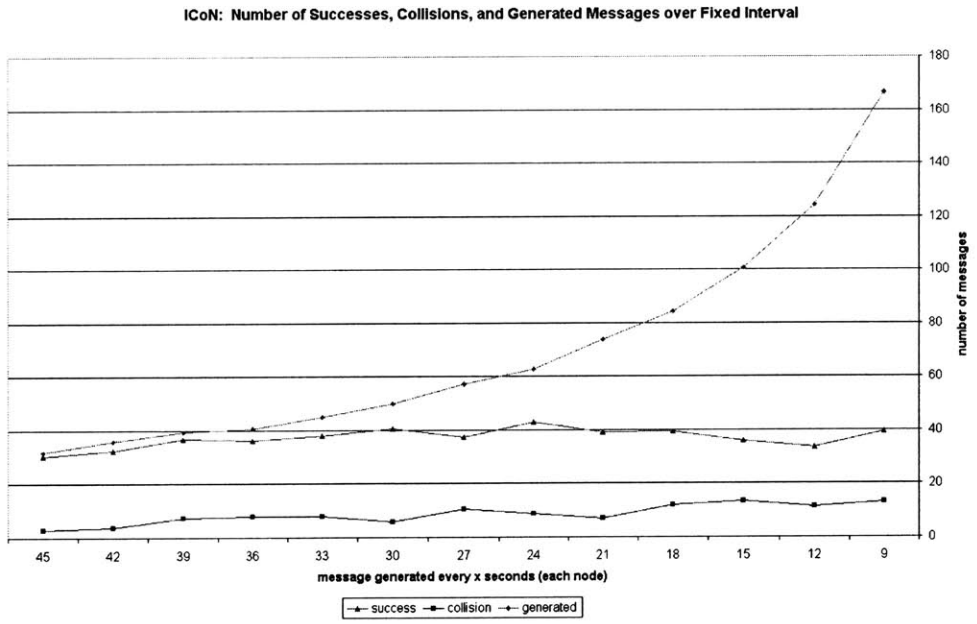

Figure 5-2: Successes and collisions in a fixed interval test using ICoN. Note the attempt of ICoN to follow the generation curve initially, until it levels off, without a congestion collapse. Collisions also do not detract from successes, due to retransmit.

increases, ICoN continues to steadily deliver increasing numbers of packets as more messages are generated, with modest but constant gains. The remaining generated messages that do not result in collisions or successful transmissions are simply backing up in each node's queue, destined to be sent when traffic slows down, or to be dropped by the node due to excessive traffic being detected.

All packets were considered high-priority in the simulation testing to observe the most mission-critical application of such a network, and the critical-point backoff routine was disabled, to see how poorly ICoN would fare in such a situation. As the sending rate reached the maximum imposed by the simulation and heavy queue congestion began burdening ICoN, this percentage lead declined, but maintained a reasonable percentage lead (15 percent) over the Control protocol even when all nodes had passed the critical point where ICoN was predicted to break down.

The Control protocol, as expected, flooded the channel with collisions as sending rates increased, though sheer volume of transmissions allowed it to successfully push a fair number of successful messages through. The rate of successes followed the aforementioned gentle increasing trend, before suffering a rapidly decreasing breakdown, as the number of collisions in the fixed interval of time increases exponentially, to nearly 90 percent of all messages generated.

In contrast, ICoN maintains a fairly steady number of successful transmissions, regardless of message generation rates. This is largely due to the ability of ICoN to avoid collisions using its four-way handshaking, and eliminate the hidden terminal problem. Ideally, ICoN should experience no collisions at all, but the simulator's random blackouts and timing glitches force handshaking messages to disappear and become reordered, and collisions to appear. ICoN manages to keep its response to this phenomenon steady, and more importantly, low (below 20 percent) for any transmission rate. Though at high transmission rates messages begin backing up in nodes' queues, a user can always be sure that a large percentage of messages that are actually transmitted are getting through, and little channel time is wasted by collisions.

The largest impact of ICoN is demonstrated by the profile of messages that are

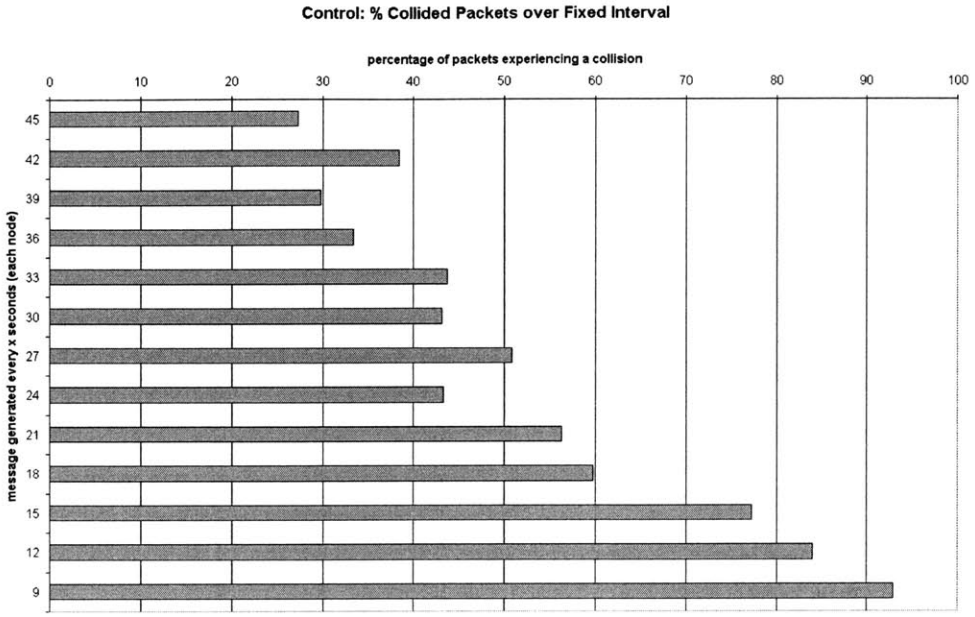

Figure 5-3: Percentage of transmissions resulting in a collision for "Control" protocol for various traffic rates.

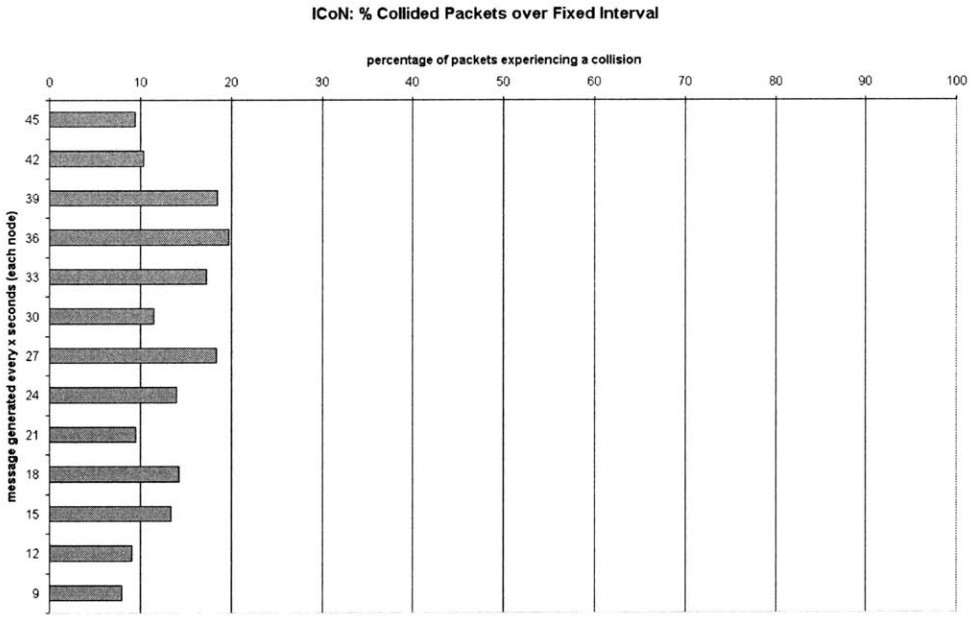

Figure 5-4: Percentage of transmissions resulting in a collision for ICoN for various traffic rates.

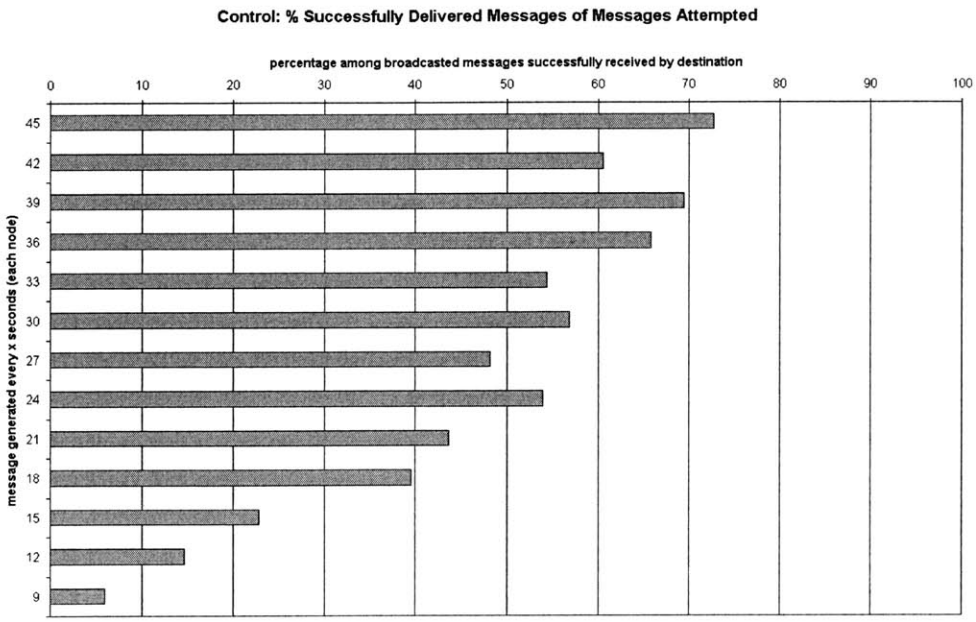

Figure 5-5: Percentage of attempted messages successfully delivered by "Control" protocol for various traffic rates.

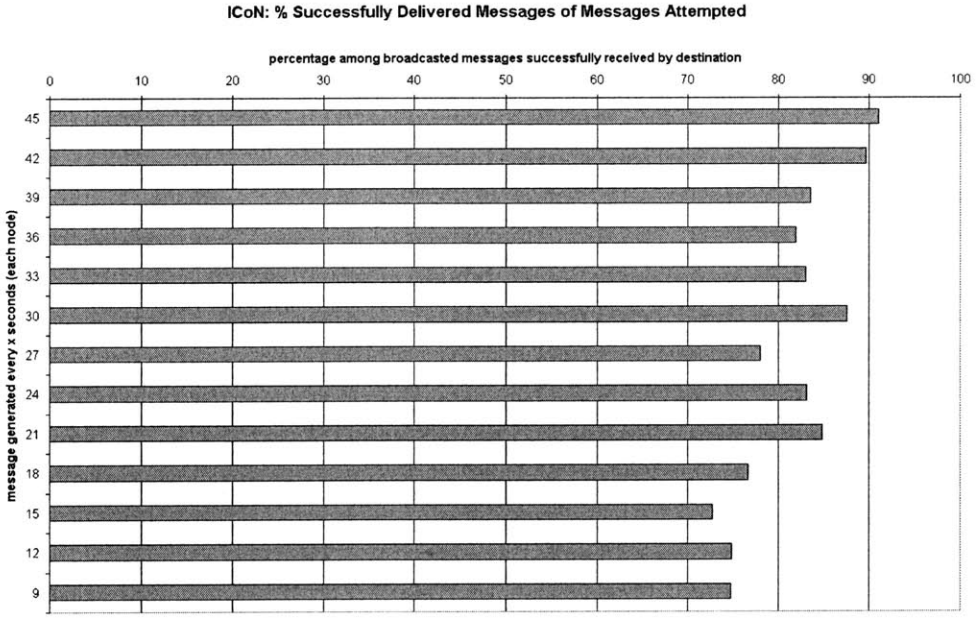

Figure 5-6: Percentage of attempted messages successfully delivered by ICoN for various traffic rates.

actually attempted to be sent across the network, in contrast to the messages that are simply generated and queued in a node. This percentage of attempted messages directly translates to the number of messages that an unacknowledged protocol will lose forever, and the amount of time in a network wasted by resending collided messages in an acknowledged network.

These test results indicate the ICoN protocol may lead to a roughly two-thirds increase in the percentage of successes per attempted transmissions, integrated across all sampled transmission rates. While this percentage deteriorates rapidly for the Control protocol, ICoN's percentage of success begins at 90 percent (15 percent higher than the Control) for very sparse traffic rates, and remains above 75 percent for all tested attempted transmission rates.

Though ICoN shows a significant improvement over the Control protocol for messages delivered successfully, there is a caveat in the statistics. The Control protocol is meant to replicate systems in use today. In this case, not only does the Control protocol not include the EDC, WPT, and other algorithms used in ICoN, but it does not employ retransmit either. Should the Control protocol be enabled with retransmit capability, it is likely that there would be improvements in its successful transmission statistics, but not in its ability to avoid collision. In addition, some of the retransmits might yield further collisions as well. The previously presented data should be viewed with this retransmit issue in consideration when choosing a protocol.

5.4 Real-World Validation

The ICoN system was validated in the field on 19 August 2005 off the coast of Falmouth, MA. The test setup consisted of three nodes, each equipped with a message generation module and the ICoN protocol, connected to an acoustic modem and an acoustic transducer. The acoustic hardware was provided by WHOI's Acoustic Communications Laboratory. The transducers were suspended 20 feet deep in the water column, and placed at various distances from each other at WHOI's facility.

Each of the three nodes was subjected to a test pattern that consisted of 7-

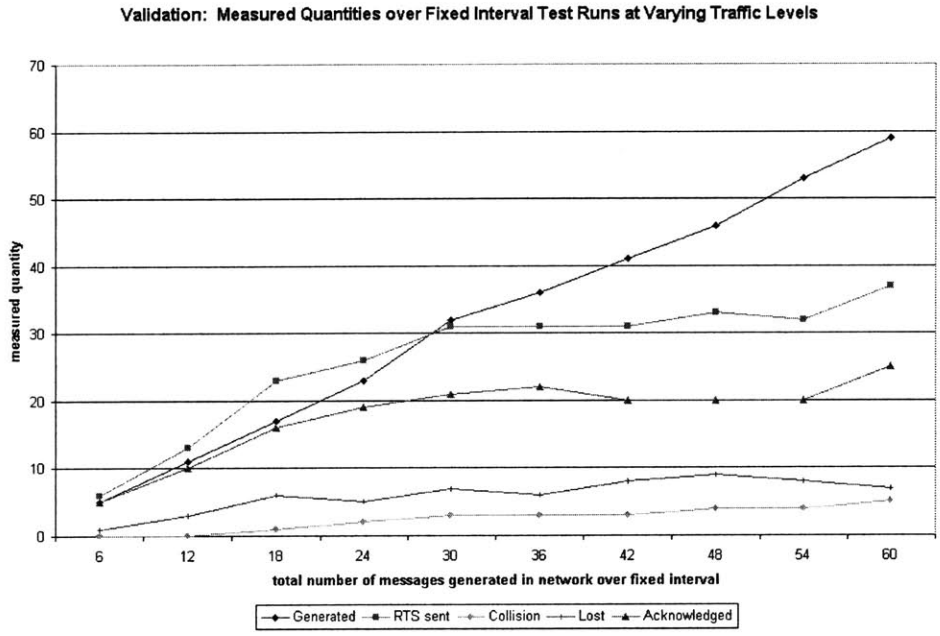

Figure 5-7: Line graph showing measured quantities of events over fixed interval tests. The number of generated messages is the independent variable, and increases linearly across various test runs. Note that the number of RTS messags sent is initially higher than the number of messages generated, since some generated messages require more than one RTS if the initial RTS collides with another RTS. Eventually, the number of generated messages surpasses the number of RTS messages sent, as the network approaches capacity. This is indicative of messages accumulating in nodes' queues. The number of collisions and packet losses remain low, with both gently increasing as traffic levels increase.

minute long tests, each subsequent test at greater traffic rates. Minimally, each node generated only two messages during the test interval, or roughly one every 200 seconds. At the highest traffic level, each node generated 20 messages during the test interval, or roughly one every 20 seconds. This maximum value was chosen to correspond with the maximum safe transmit rate of the acoustic modem's power system.

Each node generated its own log file of all messages it sent, received, and detected in passing. The log files were then parsed and analyzed, looking for RTS messages sent, CTS messages received, and ACK messages received.

Results were generally as expected. The number of acknowledged messages initially attempted to keep up with the number of messages generated, rounding off as traffic rates increased, eventually reaching a steady value. The quantity of RTS messages began higher than the number of messages generated, since each generated message might require more than one RTS if that RTS gets collided or lost. Eventually, the number of generated messages outpaces the number of RTS messages that the network has capacity for. Both the number of collisions and message losses remained low, with gradual increases detected in each.

As network traffic reached their highest levels, a slight change in network operation was observed. As the EDC began operating in higher-congestion regions, the delay applied to RTS messages decreased, allowing more to be sent more frequently. This led to an increase in acknowledged packets as well, though not a significant increase in collisions or losses. This phenomenon indicates that the EDC's parameters were not optimally tuned to the traffic patterns in the network, since the EDC should yield a continual increase in acknowledged packets though all traffic rates. However, though not optimally tuned, the EDC did have its intended effect in facilitating a gradual increase in successful transmissions as congestion increases at high congestion levels, instead of leading to a congestion collapse.

Percentage analysis of specific message characteristics yielded results of generally the same shape as simulation results. Success rates of generated messages begin high, rolling off as the rate of generation surpassed the capacity of the network. The effect

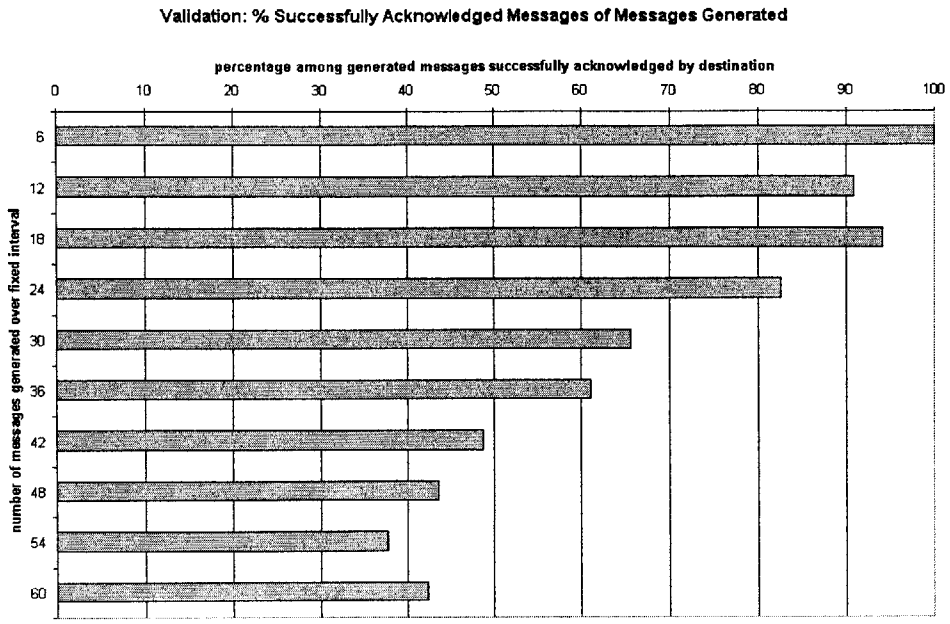

Figure 5-8: Percentage of generated messages successfully acknowledged in validation testing for various traffic rates. This value begins uniformly high, until message generation outpaces the capacity of the network, at which point there is a roll-off. ICoN's EDC system prevents the decrease from being sharp, and even helps the network catch up as traffic levels approach high quantities. These percentages can be improved by tuning the EDC's parameters.

of the EDC is seen at high traffic rates, as messages build up in each node's queue, pushing the node's operating point along the EDC, and forcing more messages to be sent.

Success rates of messages that eventually manifest themselves as an RTS remained high, as in simulation, though at slightly lower percentages. These percentages remained consistent across all traffic rates with a slight decline, and then an EDC-facilitated recovery. Optimally, EDC parameters should be tuned such that this percentage holds constant, instead of showing a decline and recovery.

As the most important effect of ICoN, through all traffic rates, collisions remained low. As traffic rates increase, the probability of RTS collisions increase as well. However, though traffic rates increased linearly through the tests, collision rates did not increase linearly, but logarithmically instead. This shows optimistic promise for nodes of higher transmit frequency or networks with greater numbers of nodes.

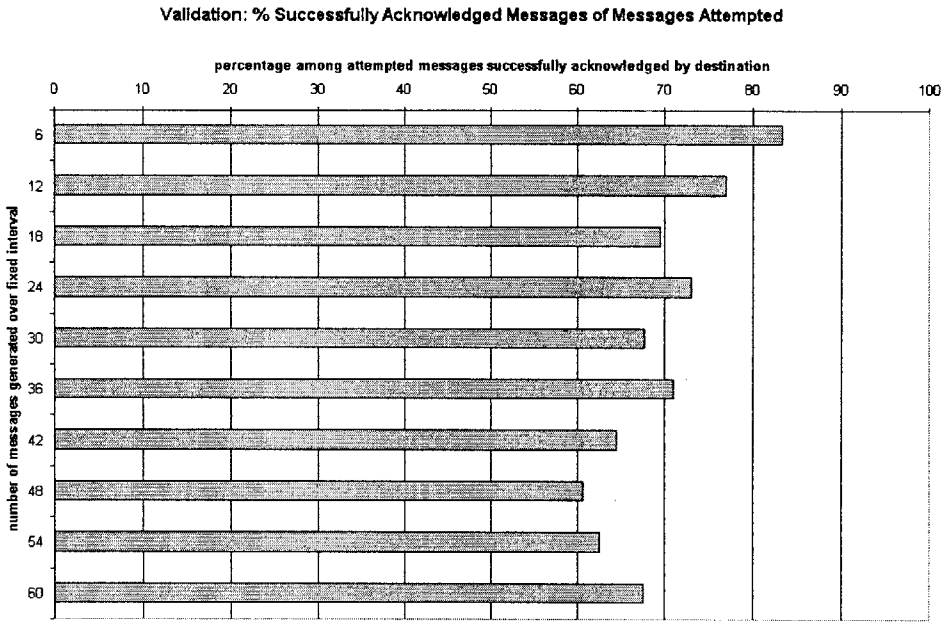

Figure 5-9: Percentage of attempted messages successfully acknowledged in validataion testing for various traffic rates. Note that this percentage starts and stays high, never sinking below 60 percent, and averaging just below 70 percent. The gradual decrease across the tests shows signs of recovery as traffic levels approach high quantities, due to the EDC. Once again, by tuning the EDC's parameters, this can likely be flattened out.

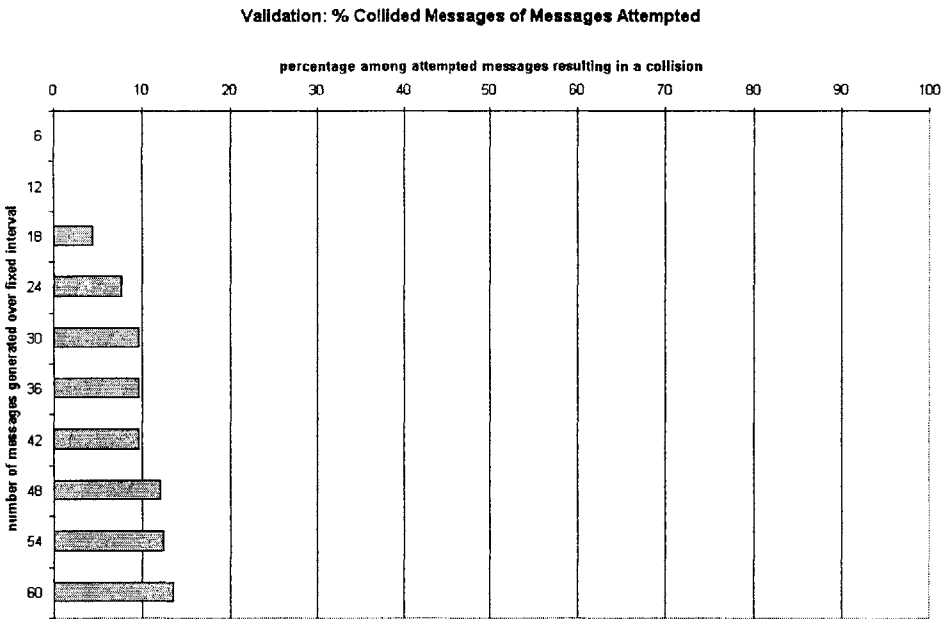

Figure 5-10: Percentage of collided messages of messages attempted in validation testing for various traffic rates. As expected, at low traffic rates, collisions are highly unlikely due to the sparseness of RTS messages. As the density of RTS messages increases, there is an increase in collisions, but that percentage increase appears logarithmic, instead of linear with respect to traffic rates. The percentage of time wasted due to collisions remains below 15 percent, averaging just below 10 percent.

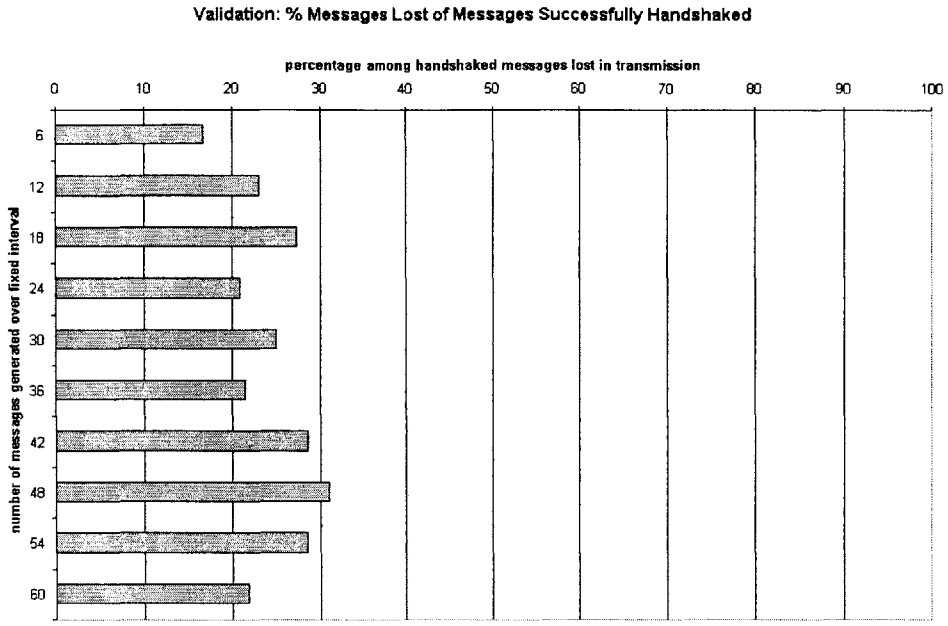

Figure 5-11: Percentage of lost messages of messages that were successfully handshaked in validation testing for various traffic rates. These are cases where the RTS and CTS messages have been successful, but the main data packet or acknowledgment gets lost due to acoustic anomalies. Regardless of traffic rate, this percentage of lost packets remains fairly constant with only slight variation. This is expected, as the physics of the link layer are independent of traffic rates. The measured percentage is generally confined between 20 and 30 percent, and averages around 25 percent.

Loss rates were defined as transactions that handshaked properly, but in which the message was never acknowledged. The interacting nodes had control of the channel, but due to environmental factors affecting acoustic transmission properties, the data or ack packet was not delivered. In these cases, the message was likely attempted again later, but a loss was recorded anyway for statistics purposes. Since environmental factors and link-layer effects are independent of protocol operation, node quantity, and traffic rates, this percentage remained relatively constant across all tests and traffic rates.

Through this validation using real off-the-shelf modems in a non-controlled body of water, ICoN has demonstrated its ability to function dutifully as a viable communication and navigation protocol for real-world applications. The resultant validation data, by being in line with simulated data, indicates that ICoN's systems and al-

gorithms do indeed function as they were designed, facilitating the operation of a decentralized asynchronous network.

Chapter 6

Future Expansion

The ICoN protocol is designed specifically to work well in today's underwater acoustic environment, and makes many sacrifices to keep protocol overhead as low as possible. The guiding philosophy is to ensure that all operation is single-ended, with all nodes passively monitoring the traffic in the acoustic channel, and making decisions based on what they detect.

However, if the bandwidth of underwater acoustic modems can be increased, the capability of the network rises, making significant enhancements possible. With increased bandwidth, overhead can increase, allowing nodes to probe for traffic, keep lists of neighbors, and take advantage of power-throttling technologies by using inter-node protocol packets that are more efficient at giving each node a correct picture of current network traffic as a whole.

The techniques presented in the following sections of this chapter may have great potential to increase the efficiency and effectiveness of underwater acoustic networks of the future, should the transport layer become faster and more reliable, an almost certain consequence of further research and development.

6.1 Explicit Awareness of Neighbors

Ad-hoc networks frequently optimize sending and retransmit schemes using continually updated nearest-neighbor lists. Each node periodically sends a probing message

to see which nodes around it are reachable and thus respond. By doing this on a regular basis, nodes always keep a table of all nodes that are their immediate neighbors.

In single-hop networks, by knowing which nodes are currently reachable from itself, nodes rarely waste time transmitting to a node that is out of range. Nodes can enqueue messages and hold them for nodes that are currently unreachable, and, once they determine that the destination node is now nearby, transmit the relevant data to that node.

Even more powerful is the ability to do routing and message redirection. Taking the probing message idea one step further, if a node embeds its own nearest neighbor list in the probe message, the immediate neighbors of that node know not only that the sending node is an immediate neighbor, but also what nodes are immediate neighbors of that node. By keeping a table of each immediate neighbors' neighbors, it is possible for every node to quickly derive a path to every other reachable node in the network, and, with a simple algorithm, determine the shortest such path. This makes it possible to relay messages from node to node, to reach destination nodes otherwise unreachable from the source node. Many other benefits are also afforded to a system that can keep track of neighbors and paths.

6.2 Cooperative Navigation

Currently, every node in a network that possesses the ability to navigate must emit active sonar pings off pre-located transponders in order to triangulate its location. As the number of nodes in a network increases, it is likely that soon, all available network bandwidth will be used solely to obtain navigation data from each node's active sonar pings. This is evidently an unacceptable situation, unless the only purpose of these nodes is to know where they are[14].

To function adequately on networks with large numbers of nodes, it is possible for nodes to navigate cooperatively off each other. Some nodes would need to remain stationary, with knowledge of their own location, analogous to the baseline navigation transponders of today. When faced with a navigation request by another node,

the stationary nodes would transmit a brief packet with their own location and a timestamp of exactly when the message was sent out.

By receiving three such messages, any node can use the location data and timestamp values to triangulate its own position. In this manner, each node needs to emit a ping barely strong enough to be reached by the closest three nodes, regardless of whether they are stationary or mobile nodes. From the data, the network continually passes navigation information through it, updating all parts of itself frequently enough to be correct with a reasonable level of probability[12].

Nodes can even be used as stationary beacons, broadcasting their location and a timestamp for everyone to hear. As long as any regular node hears three beacons, it can determine its own location. The only caveat to this system is that precise clock synchronization is needed network-wide. If this requirement is possible, and if nodes can reduce the volume of their pings to reach only the nearest nodes, it becomes possible for very large networks of mobile nodes to successfully navigate without monopolizing the channel.

6.3 Network Mapping

With the infrastructure provided by an ad-hoc cooperative navigation scheme, nodes could even construct internal maps of all nodes that are nearby in the network. With increased bandwidth, it becomes cheap for a node to encode its own position in every packet it sends. Combining this feature with reachable-neighbor lists, a node in the network could construct a spatial map of every node in the entire network.

Node designers could use this ability in many ways, depending on the special application of their nodes. A sensor net could be self-repairing, should any node be damaged. Fleets of AUVs could automatically coordinate with each other in mapping operations, forming their own adaptive spatial nets based on their own internal knowledge, without explicit commands from a command center. For power-conserving networks, the ability to form internal maps is vital for transmission power throttling.

6.4 Power Control, Ad-Hoc Subgrouping

As determined by previous ad-hoc network research, the lowest-power path between two nodes is not a direct path between them, but a sum of short possible hops between the two nodes. With internal maps, nodes can calculate power levels and determine the lowest-power paths to each possible destination node[2].

Perhaps even more powerful is the idea of ad-hoc subgrouping. With the ability for a node to throttle its power to be just strong enough to reach its desired destination, and an internal map of all nodes in the network, networks can form subgroups consisting of clusters of nodes that happen to be close to one another. Since communication messages are now quieter, each group can communicate independently, with multiple groups each communicating within themselves simultaneously. This greatly increases the effective bandwidth of such a network.

Furthermore, the idea of subgrouping implicitly creates a hierarchy in such networks. This leads to powerful network communication techniques, including multicast. The foundation for such self-organizing hierarchies is the ability to cooperatively navigate, form internal maps, and throttle a node's transmission power.

6.5 Burst-Transmit Throttling

Today's acoustic modems are already equipped with channel-diagnostic tools, which, upon reception of each transmission, quantify the integrity of the link with the source of the message. Keeping EWMAs of data quality factors, along with a nearest neighbor list can lead to power control optimizations. When the link quality with a neighbor node is poor, a node may choose to send packets sparsely, just often enough to probe the channel to check for changes in link quality. This saves a node from sending consistent high-intensity transmissions in order to overcome the poor link quality, conserving otherwise wasted power. When the link quality is good, a node can pump data at high bitrates, bursting data over the link at full speed. As the channel is likely cleaner, lower transmit power is needed, yielding the best efficiency per bit.

Chapter 7

Conclusion

Simulation testing comparing the ICoN protocol to the performance of decentralized networks using technology available today, and validation of its performance through real-world evaluation, demonstrates that ICoN achieves marked increase in network capability. Unlike other arbitration schemes, ICoN wastes no bandwidth on network-probing or inter-node status-sharing messages, and is fully single-ended, encapsulated in each node, using only passive means to monitor network traffic.

Despite the lack of status-sharing messages, ICoN is able to consolidate the most important information about local network traffic through indirect means. ICoN then plugs this sensed data into a number of algorithms that determine not only how a node decides to send messages, but also how it should manage its own internal message-handling resources.

Furthermore, ICoN serves as a rudimentary behavior system, setting message type priorities, and modifying the transmission profiles of nodes to better serve their specialized use. This behavior system and ICoN's algorithms are taken a step further, incorporating learning and adaptive algorithms to allow nodes to optimize themselves in real time within a changing network, and maintain stability in networks with a wildly changing environment.

ICoN is highly customizable, giving users access to a wide variety of parameters that, when changed, lead to very different behavior patterns, thus allowing the end user to manually optimize the performance of ICoN in networks where traffic patterns

are well known and predictable.

Limited ICoN testing in the underwater environment appears to validate ICoN's ability to perform as intended in the application for which it is designed. ICoN provides a clean interface to real-world hardware, showing itself integrable in today's modern systems.

Most importantly, the protocol is suitable for both stationary and mobile nodes, which is where ICoN gets its signature feature: the first network protocol that combines two independent systems, communication and navigation, in an asynchronous, decentralized arbitration scheme that allows both systems to use a shared communication medium cooperatively.

Appendix A

Figures

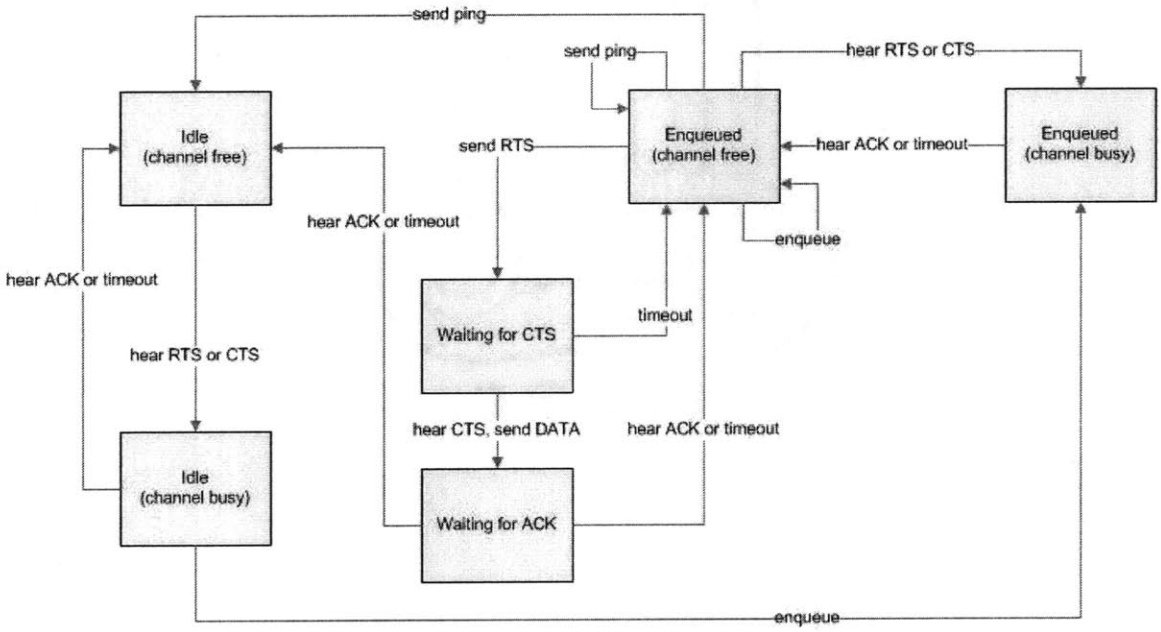

Figure A-1: System state diagram.

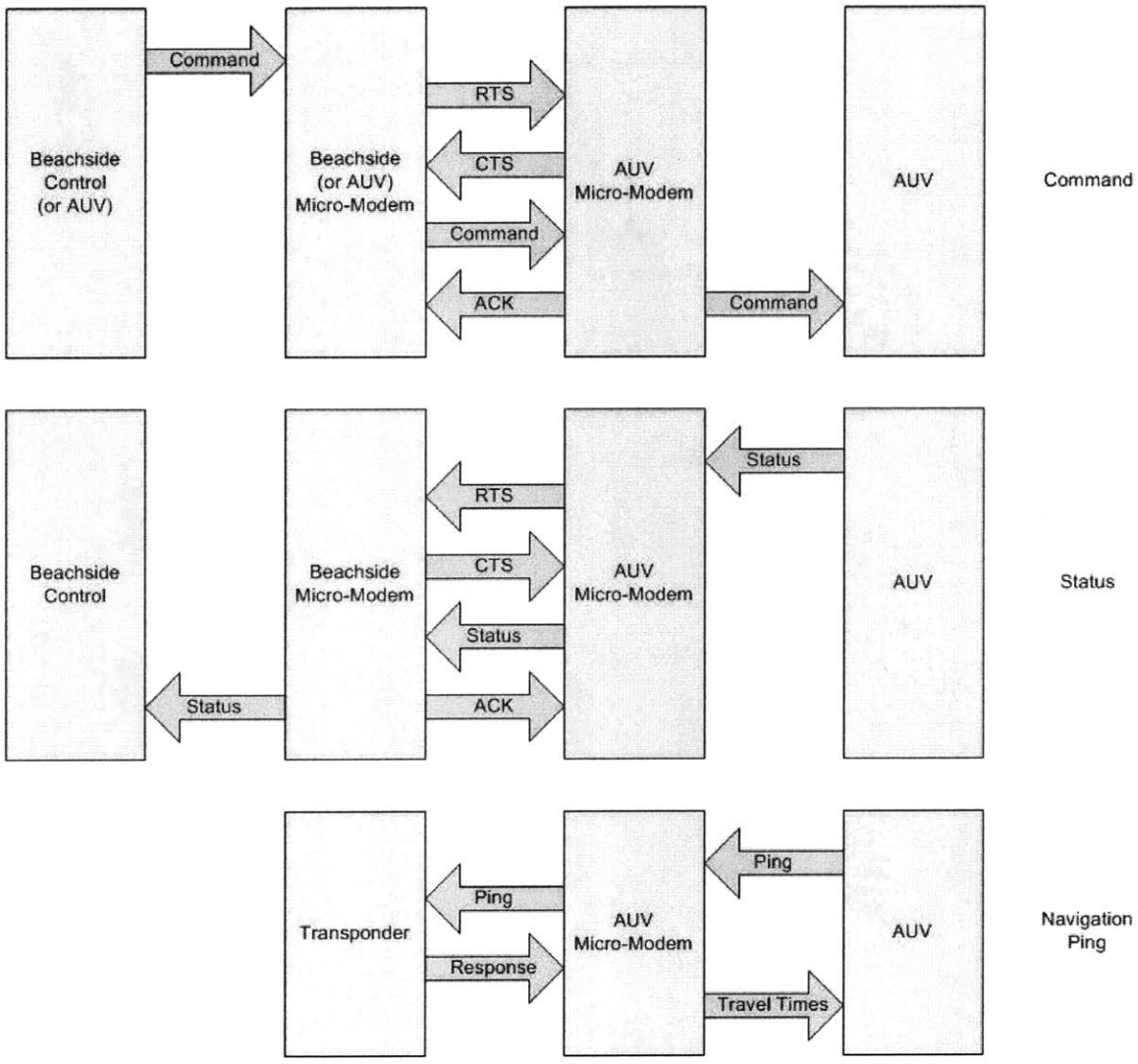

Figure A-2: Block diagram of interaction of protocol instances with each other and with their respective nodes, for different message types.

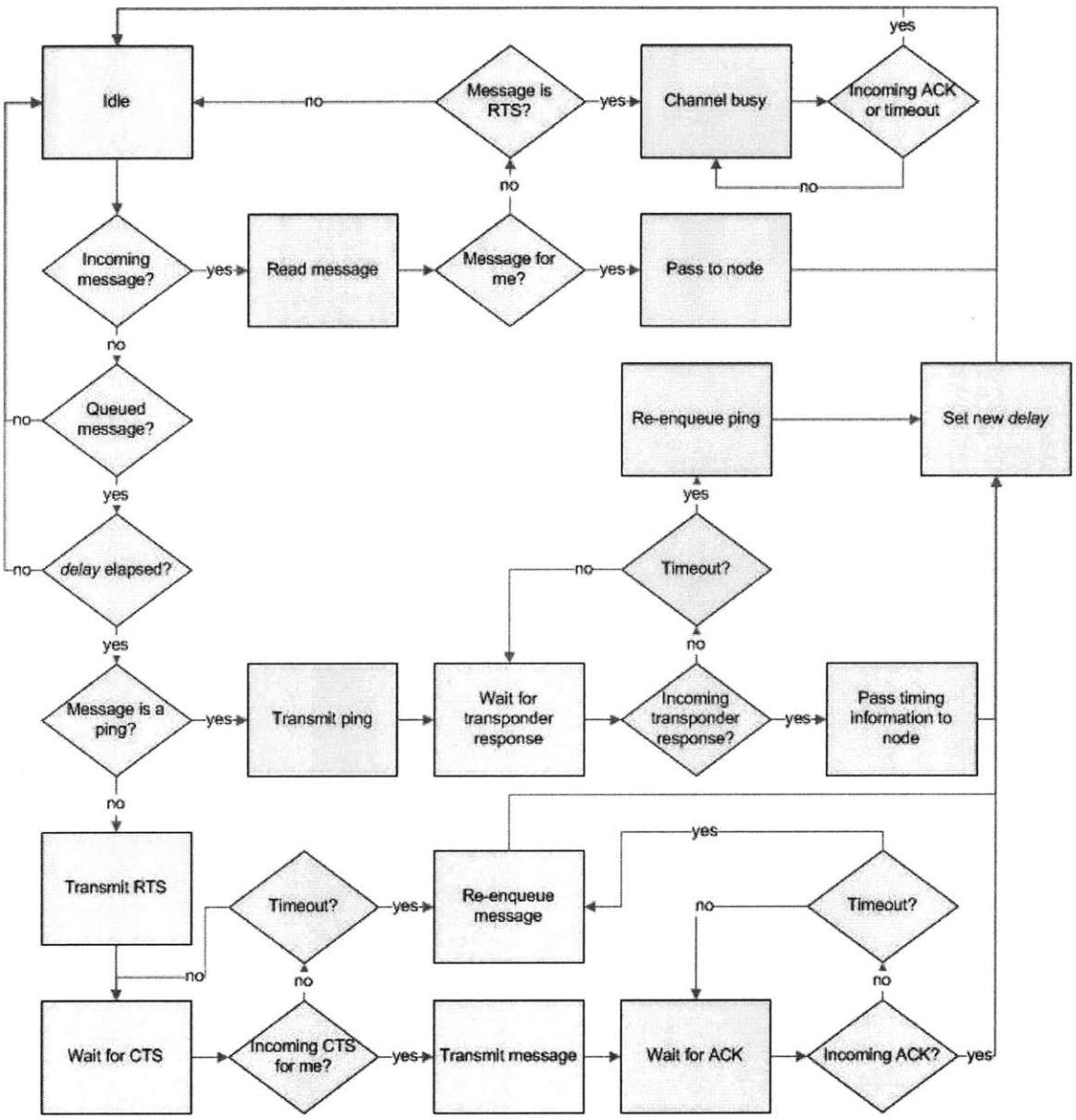

Figure A-3: Software flowchart. This presents a useful guide for implementation.

Appendix B

Protocol Code

The source code for the prototype ICoN protocol implementation can be found at "http://web.mit.edu/rkanthan/ICoN/".

Alternatively, a copy of the source code can be obtained by emailing the primary author at "rkanthan@alum.mit.edu".

Bibliography

[1] H. Balakrishnan. Single-link communication , 6.829 computer networks lecture notes. Technical report, Massachusetts Institute of Technology, 2002.

[2] H. Balakrishnan. Wide-area unicast routing, 6.829 computer networks lecture notes. Technical report, Massachusetts Institute of Technology, 2002.

[3] H. Balakrishnan, Venkat Padmanabhan, Srinivasan Seshan, and Randy H. Katz. A comparison of mechanisms for improving tcp performance over wireless links. *IEEE/ACM Transactions on Networking*, December 1997.

[4] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang. Macaw: A media access protocol for wireless lans. In *Proc. ACM SIGCOMM*, pages 212–215, London, U.K., September 1994.

[5] D. Clark. Design philosophy of the darpa internet protocols. In *Proc. ACM SIGCOMM*, pages 106–114, Stanford, CA, August 1988.

[6] D. Clark and W. Feng. Explicit allocation of best-effort packet delivery service. *IEEE/ACM Transactions on Networking*, 6(4):362–373, August 1998.

[7] D. Clark and D. Tennenhouse. Architectural consideration for a new generation of protocols. In *Proc. ACM SIGCOMM*, Philadelphia, PA, September 1990.

[8] A. Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair queueing algorithm. *Internetworking: Research and Experience*, 1(1):3–26, 1990.

[9] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, August 1993.

[10] L. Freitag, M. Johnson, M. Grund, S. Singh, and J. Preisig. Integrated acoustic communication and navigation for multiple uuvs. In *Proc. Oceans 2001*, pages 2065–2070, Honolulu, HI, 2001.

[11] L. Freitag, M. Stojanovic, S. Singh, and M. Johnson. Analysis of channel effects on direct-sequence and frequency-hopped spread-spectrum acoustic communication. *IEEE Journal of Oceanic Engineering*, 26(4):586–593, October 2001.

[12] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *Proc. ACM MOBICOM*, Boston, MA, August 2000.

[13] V. Jacobson and M. Karels. Congestion avoidance and control. In *Proc. ACM SIGCOMM*, Stanford, CA, August 1998.

[14] J. Li, C. Blake, D. De Couto, H. Lee, and R. Morris. Capacity of wireless ad hoc networks. In *Proc. ACM MOBICOM*, Rome, Italy, July 2001.

[15] Robert M. Metcalfe and David R. Boggs. Ethernet: Distributed packet switching for local computer networks. *Communications of the ACM*, 19(7):395–404, July 1976.

[16] V. Paxson. End-to-end routing behavior in the internet. *IEEE/ACM Transactions on Networking*, 5(5):601–615, October 1997.

[17] M. Stojanovic and L. Freitag. Hypothesis-feedback equalization for direct-sequence spread-spectrum underwater communications. In *Proc. Oceans 2000*, volume 1, pages 123–128, Providence, RI, 2000.

[18] M. Stojanovic and L. Freitag. Multiuser undersea acoustic communications in the presence of multipath propagation. In *Proc. Oceans 2001*, pages 2165–2169, Honolulu, HI, 2001.