

Air Force Institute of Technology

AFIT Scholar

Theses and Dissertations

Student Graduate Works

3-16-2007

Adaptive Gravitational Gossip in Monitoring the Joint Battlespace Infosphere

Edmund Descartes Aban

Follow this and additional works at: <https://scholar.afit.edu/etd>



Part of the [Computer Sciences Commons](#)

Recommended Citation

Aban, Edmund Descartes, "Adaptive Gravitational Gossip in Monitoring the Joint Battlespace Infosphere" (2007). *Theses and Dissertations*. 3107.

<https://scholar.afit.edu/etd/3107>

This Thesis is brought to you for free and open access by the Student Graduate Works at AFIT Scholar. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of AFIT Scholar. For more information, please contact richard.mansfield@afit.edu.



**ADAPTIVE GRAVITATIONAL GOSSIP
IN MONITORING
THE JOINT BATTLESPACE INFOSPHERE**

THESIS

E. Descartes Aban, Captain, USAF

AFIT/GCS/ENG/07-01

**DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY**

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the U.S. Government.

AFIT/GCS/ENG/07-01

**ADAPTIVE GRAVITATIONAL GOSSIP
IN MONITORING
THE JOINT BATTLESPACE INFOSPHERE**

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science

E. Descartes Aban, BS

Captain, USAF

March 2007


APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

**ADAPTIVE GRAVITATIONAL GOSSIP
IN MONITORING
THE JOINT BATTLESPACE INFOSPHERE**

E. Descartes Aban, BSCPE

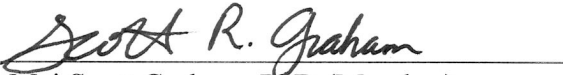
Captain, USAF

Approved:



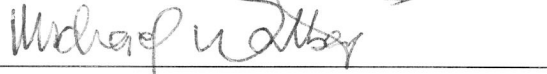
Dr. Ken Hopkinson (Chairman)

16 Mar 07
Date



Maj Scott Graham, PhD (Member)

16 Mar 07
Date



Dr. Michael Talbert (Member)

16 Mar 07
Date

Abstract

Future USAF operations will be heavily dependent on having the “right information” at the “right time”, and Joint Battlespace Infospheres (JBIs) are poised to fill that role. To do this, JBIs must be ubiquitous—always accessible, secure and responsive. Of all the literature written regarding JBIs, the most important problem to solve in order to make JBIs work in mobile scenarios are scalability, reliability and adaptability to changing battlefield conditions. This paper explores the use of SBCast, a novel adaptive probabilistic protocol, a delivery mechanism for JBI updates and as a possible solution towards guaranteeing these qualities. It documents tests of SBCast within a simulation environment configured with parameters based on actual military field operations. From these tests, the paper examines SBCast as an enhancer to JBI’s ability for overcoming transient network failures while managing different classes of subscribers by available bandwidth and priorities. By using the feedback from SBCast as a middleware layer controller, JBIs would be able to “dial up” traffic for parts of the network and “dial down” traffic in others based on dynamic changes in network congestion or traffic demands.

Acknowledgments

I would like to express my sincere appreciation to my faculty advisor, Ken Hopkinson, for his guidance and support throughout the course of this thesis effort. His insight and experience was certainly appreciated. I would like to thank my sponsor, Dr. Michael Talbert, from the Air Force Research lab, for both the support and latitude provided to me in this endeavor. My thanks also goes to Maj Graham and Lt Col Kurkowski, whose advice and guidance have pushed me onward through the course of producing this thesis. Without question, my peers and colleagues, those woeful souls who stayed late with me late into the cold, winter nights also deserve thanks for the support that they gave me. Last, but certainly not least, I would like to express my appreciation for my wife and my children, whose love, patience and support are without measure and without whose existence makes this endeavor meaningless.

E. Descartes Aban

Table of Contents

Abstract.....	v
Acknowledgments.....	vi
Table of Contents.....	vii
List of Figures.....	ix
List of Tables.....	xi
I. Introduction.....	1-1
1.1 Background.....	1-1
1.2 Definition of Terms.....	1-2
1.3 Problem Statement.....	1-6
1.4 Research Approach.....	1-6
1.5 Scope and Limitations.....	1-8
1.6 Summary.....	1-10
II. Literature Review.....	2-1
2.1 Chapter Overview.....	2-1
2.2 The Problem Domain: JBI.....	2-1
2.3 Distributed Systems Work.....	2-6
2.4 Adaptive Gravitational Gossip Background.....	2-9
2.5 Summary.....	2-12
III. Methodology.....	3-1
3.1 Chapter Overview.....	3-1
3.2 Problem Description.....	3-2
3.3 Scenario.....	3-2
3.4 The Novel Protocol: SBCast.....	3-9
3.5 Experiments.....	3-14
3.6 Objectives.....	3-17
3.7 Procedures.....	3-19
3.8 Summary.....	3-21
IV. Analysis And Results.....	4-1

4.1	Chapter Overview.....	4-1
4.2	Results of Simulation.....	4-1
	4.2.1 Observed-to-Target Graphs.....	4-1
	4.2.2 Standard Deviation Graphs.....	4-6
4.3	Investigative Questions Answered.....	4-13
	4.3.1 Can SBCCast be applied to an operational JBI scenario?.....	4-14
	4.3.2 How would SBCCast perform as a middleware controller in JBIs?.....	4-16
	4.3.3 Scalability.....	4-18
	4.3.4 Reliability.....	4-20
	4.3.5 Adaptability.....	4-23
4.4	Summary.....	4-24
V. Conclusions and Recommendations.....		5-1
5.1	Chapter Overview.....	5-1
5.2	Conclusion of Research.....	5-1
5.3	Recommendation for Action.....	5-3
5.4	Recommendations for Future Research.....	5-4
5.5	Summary.....	5-7
Appendix A: Development of Ruby Cast: The *Cast Script Generator.....		A-1
A.1	Appendix Overview.....	A-1
A.2	From Academic Exercise to Operational Environment.....	A-1
A.3	Software Development.....	A-4
A.4	Original *Cast Scenario Code.....	A-4
A.5	Ruby Cast Scenario Script Generator.....	A-9
A.6	Summary.....	A-21
Appendix B: Supplementary Diagrams.....		B-1
Bibliography.....		BIB-1
Vita.....		VITA-1

List of Figures

Figure 2.1: Organizational Diagram of JBI.....	2-2
Figure 2.2: Publish and Subscribe Software Architecture.....	2-4
Figure 3.1: Scenario Topology.....	3-3
Figure 3.2: A *Cast Group Level.....	3-3
Figure 3.3: Progression of Network Conditions.....	3-8
Figure 3.3a: No CBRs Active.....	3-8
Figure 3.3b: One CBR Starts.....	3-8
Figure 3.3c: More CBRs Start.....	3-8
Figure 3.3d: 75% of CBRs.....	3-8
Figure 3.3e: 90% of CBRs.....	3-8
Figure 3.3f: All CBRs Active.....	3-8
Figure 3.4: Original Simplified AGWCast <code>recv()</code>	3-10
Figure 3.5: <code>recv()</code> Modified for SBCast.....	3-11
Figure 3.8: Script-Generator Program.....	3-20
Figure 4.1: CBR Counts For All Experiments.....	4-2
Figure 4.2: Test 1 Results (All Links Saturated).....	4-3
Figure 4.3: Test 2 Results (Narrow Links Saturated).....	4-4
Figure 4.4: Test 3 Results (Wide Links Saturated).....	4-5
Figure 4.5: SBCast Standard Deviation Graph For All Links Saturated.....	4-7
Figure 4.6: AGWCast Standard Deviation Graph For All Links Saturated.....	4-7
Figure 4.7: GWCast Standard Deviation Graph For All Links Saturated.....	4-8

Figure 4.8: PBCast Standard Deviation Graph For All Links Saturated.....	4-8
Figure 4.9: SBCast Standard Deviation Graph For Narrow Links Saturated.....	4-9
Figure 4.10: AGWCast Standard Deviation Graph For Narrow Links Saturated.....	4-9
Figure 4.11: GWCast Standard Deviation Graph For Narrow Links Saturated.....	4-10
Figure 4.12: PBCast Standard Deviation Graph For Narrow Links Saturated.....	4-10
Figure 4.13: SBCast Standard Deviation Graph For Wide Links Saturated.....	4-11
Figure 4.14: GWCast Standard Deviation Graph For Wide Links Saturated.....	4-11
Figure 4.15: AGWCast Standard Deviation Graph For Wide Links Saturated.....	4-12
Figure 4.16: SBCast Standard Deviation Graph For Wide Links Saturated.....	4-12
Figure 4.17: An Example of Expected Behavior In Response To Traffic.....	4-17
Figure A.1: Typical *Cast Topology	A-2
Figure A.2: Original *Cast Scenario Procedure.....	A-5
Figure A.3: UML Diagram for C/C++ Script Generator.....	A-7
Figure A.4: Basic ns-2 Object Model.....	A-10
Figure A.5: Intermediate ns-2 Object Model.....	A-12
Figure A.6: Composite Topology	A-14
Figure A.7: Intermediate ns-2 Object Model With Composite Topology.....	A-15
Figure A.8: Interest Strategy Model.....	A-18
Figure A.9: PBCast Group.....	A-19
Figure A.10: Ruby Cast Complete Model.....	A-20
Figure B.1: Scenario Topology Expanded.....	B-1

List of Tables

Table 3.1: Groups.....	3-5
Table 3.2: Interest Levels.....	3-6
Table 3.3: Experiments.....	3-16
Table A.1: Typical Interest In Original *Cast Experiments.....	A-3

ADAPTIVE GRAVITATIONAL GOSSIP IN MONITORING THE JOINT BATTLESPACE INFOSPHERE

I. Introduction

1.1 Background

The idea behind the Joint Battlespace Infosphere is to tie all the disparate streams of data necessary for warfighters – to mean all personnel in a battle zone, from intelligence analysts, supply officers, communications units, infantry, tank crews, and pilots to general officers – to conduct operations, and process these streams into one useful information source specifically tailored to each particular warfighter’s mission. This is essentially a universal interface for all the information a warfighter needs, and thus it requires customizable and configurable standard interfaces as well as the ability to interoperate with both currently existing and future information systems. Directly from the executive summary of the JBI [25], the most complete definition of what a JBI is as follows:

The Joint Battlespace Infosphere (JBI) is a combat information management system that provides individual users with the specific information required for their functional responsibilities during crisis or conflict. The JBI integrates data from a wide variety of detail to users at all echelons...[25]

The JBI, being one of the cornerstones of future USAF operations (per the USAF Vision 2020 and Joint Vision 2020), is aimed at allowing our forces to dominate the Infosphere. The result will be Information Dominance (one of the USAF core competen-

cies), where forces will be able to act and react within a tighter OODA loop. In practical terms, this could mean:

- Tighter integration with sister services and Allied forces
- Correct intelligence with exact targets to strike
- Interrupting the Enemy's operations, or attacking before we are attacked
- Understanding and analyzing the Enemy

Operationally, the JBI is, or is comprised of, command and control (C2) systems.

Technically, it is the integration of all these C2 systems linked together through a platform of various protocols. This platform consists of fuselets to process and aggregate the different streams and middleware layers to translate, interface or monitor these systems,. These fuselets would all wrapped into a unified publish-and-subscribe (“pub-sub”) system, described later.

This paper proposes a possible protocol for inclusion in use for JBI as a controller middleware layer. This novel protocol, known as Slack Broadcast, or SBroadcast, is a probabilistic protocol based on gossip. The use of a gossip protocol, with its purported strengths in scalability, speed of updates and use for pub-sub systems, could provide JBIs the needed gains in the reliability, scalability and availability. The next section defines and explains gossip, and how it could extend JBIs.

1.2 Definition of Terms

This paper discusses the use of a customized, optimized gossip protocol. Because of the frequent use of gossip protocol terms, it is extremely necessary that they be prop-

erly defined before discussion. This section includes those definitions, and explains what these mean together. More importantly, this section provides a brief explanation as to what a gossip protocol means to JBIs.

A gossip protocol is a network-level protocol whose purpose is to propagate updates to communications nodes in a network by randomly selecting the next recipient of a message. This is where it gets its name; the underlying idea of gossip is that it imitates gossiping neighbors in a community. A great example would be the online community of customers for Apple Computers corporation line of digital music players, known as iPods. One member of this community might receive some especially attractive rumor, such as a report that states that Apple is coming out with a new portable music player that is smaller than a stick of gum. Meeting with his peers in chat rooms or in email, he would mention it, resulting in randomly updating his peers through casual conversation. Through these random updates, the news about the new music player travels fast throughout the group.

First developed by Al Demers and others in 1987 [7], the first basic gossip protocol, whose initial scheme consisted of using a simple random function (available in most implementations of C/C++), has diversified into a whole family of protocols. These protocols are all considered gossip because they vary only in the strategy they use to randomly select the next node. For example, a gossip protocol might be weighted, in that its random function selects the next recipient by some criteria associated with value. For example, a communications node might have a higher value than another adjacent node because it has higher bandwidth connections, and would most likely receive updates from

its neighbors. Another might use a different probability distribution function to randomly select a recipient. Instead of using the `rand()` function from the C libraries, the protocol would use some probability distribution function. Whatever the strategy involved, the common denominator amongst any gossip protocol is that it is *probabilistic*; that is, its behavior is modeled on a probability distribution function. From this quality, a gossip protocol is known as an *epidemic* protocol; the behavior of gossip is likened to a disease, where, as neighbors infected with the common cold might bring that cold to other neighbors, communication nodes would ‘infect’ each other with updates. Continuing this analogy, one characteristic that affects agents in a gossip network is its *infectivity*, the probability that it will send updates. Another characteristic is an agent’s *susceptibility*, the probability that it will accept an update. Like a contagious disease, updates spread in this way propagate rapidly [4].

This paper investigates an implementation of gossip, referred to as Slack Broadcast, or SBCast. SBCast was developed using the source code for gossip protocols based on Probability Broadcast (PBCast), which was an original implementation of a gossip protocol and was developed by Kenneth Birman of Cornell University [4]. PBCast has given birth to a number of variants, collectively referred to as the *Cast family. Each member of the *Cast family builds upon the previous one, and the variants differ by the addition of mechanisms to the probability calculation and the factors involved with that calculation. These factors include *gravitational weights*, a priority or rating given to agents in the gossip network according to observed infectivity or susceptibility. Being no different, SBCast adds the mechanism of target adjustment to the calculations accom-

plished by its ancestors, Gravitational Gossip (GWCast) [3] and Adaptive Gravitational Gossip (AGWCast) [13]. Some of these calculations are based on *interest* in, or priority of members to, a particular type of update, and the quality of bandwidth available based on success of past updates. These new methods of calculating protocols based on network conditions make the *Cast protocols *adaptive*, meaning that they are gossip protocols that are able to vary update rates based on perceived network conditions. SBCast and its relative *Cast protocols are discussed in-depth under “Research Approach” and expounded upon in Chapter II.

How would this benefit the uses of the JBI? Gossip protocols have been key components in bulletin board systems [3], and in Chapter II it will be shown that the publish-and-subscribe software architecture is a major design component for JBIs. The epidemic nature of a gossip protocol can be an asset in deployed ad hoc networks; because a gossip protocol is inherently decentralized, segmentation caused by the instability of links can be compensated. Therefore, utilizing SBCast will accomplish a number of benefits. First, it will enhance the JBI’s ability to overcome transient network failures. Second, it will allow different classes of subscribers to request different rates of traffic updates, according to their available bandwidth and interest, which could be very effective when dealing with traffic that is periodic in nature. Finally, the gossip feedback from the pub-sub system to get data can be used to infer current network conditions. By providing feedback to the JBI regarding current network conditions, it will be able to adapt its own behavior to better match the network’s capacity constraints. Additionally, because the JBI concept envisions use in various sizes of networks (small to large), scalability is a

must and many studies have shown that a gossip approach scales effectively and reliably [3].

1.3 Problem Statement

The core idea for this thesis is using SBCCast, a novel, adaptive gossip protocol to monitor publish/subscribe and quality of service to improve reliability and scalability of JBIs. This involves determining the applicability of SBCCast to an actual, military environment within which JBIs would be deployed. Therefore, the key investigative questions form the problem statement, are:

- Can SBCCast be applied to an operational JBI Scenario?
- How would SBCCast perform as a controller component in JBI middle-ware, specifically in regards to scalability, reliability and adaptability?

Therefore, the purpose of this thesis is to evaluate the use of SBCCast as a middle-ware layer to improve the reliability, scalability and adaptability of JBIs.

1.4 Research Approach

The work in this thesis simply follows the ongoing research started by Birman in his seminal work on Gossip in 1999 [4]. His work yielded one of the first implementations of a gossip protocol, PBCast. Since then, multiple researchers have continued with his work. From that base, Hopkinson and others went on to define GWCast, a *Cast variant that introduces the idea of “weights”; that is, weighted values that influence which of its neighbors a node in a PBCast network might update [3]. The next evolution of *Cast involved adjusting the quality at which updates were received-- raising or lowering the

infectivities in order to utilize bandwidth more efficiently [13]. This paper adds yet another iteration to the *Cast group of protocols-- the technique added by this research work involves interpreting the quality at which messages are received (and thus indirectly the bandwidth) and adjusting the gravitational weights for a node. From the perspective of evolution of protocols based off of PBCast, this paper investigates the *Cast family's applicability to monitoring JBIs.

From this perspective and the problem statement, the key investigative question is simply how well does SBCast perform in a JBI? If given a typical scenario that reflects battlefield conditions in some way, how can we expect the protocols to behave? This question is simply answered by creating some controlled environment and running all the *Cast protocols side-by-side and comparing the results. The next logical question is how the experiments can demonstrate how SBCast can manage the bandwidth by controlling or restricting updates? The results would be measurements of the quality of the updates. The purpose of gossip protocols are to propagate updates in publish-subscribe systems using methods with evenly spread overhead and probabilistically high reliability. The successive *Cast protocols enhance basic gossip by allowing users to maintain the same properties as in basic PBCast, but with the ability to lower the expected reliability in return for lower bandwidth utilization. If the new protocol can maintain the promised rate of reliability by commensurately lowering the bandwidth required then the protocol is successful.

As for creating this controlled scenario to run the protocols side-by-side, the only practical demonstration would be in a simulation test-bed. Currently, the *Cast protocols

source code has been developed using the VINT group's network simulator, ns-2, and exist as objects in the simulator. This ns-2 code could be used as a model to create (but not directly translate into) source for an actual SBCast implementation, but for the purpose of this research ns-2 code can be tweaked and modified to model SBCast's performance for use in the JBI. This approach is explained in greater detail in Chapter III.

1.5 Scope and Limitations

There are a number assumptions and limitations that constrain the scope of this paper. These constraints deal with time to conduct this research, and other constraints involve focusing the research to the important subject: the algorithms involved with making SBCast work in simulation. Without these constraints, what research that was accomplished would be unfocused.

One notable limitation is the matter of an actual wireless network. A wireless network's physical characteristics are not the subject of this paper; so considerations for the actual physical or data link layers in the OSI protocol stack, such as the use of IEEE 802.11b, g, n etc. or antenna type, signal strength or other characteristics are not factored in the simulation. Such considerations might add to ns-2 processing time, and would add extraneous factors to studying SBCast's behavior. To simplify matters, in the network simulation, the nodes are stationary and links are considered directional (e.g. similar to duplex links, the most basic link available in the ns-2 tool set). However, to simulate multi-directions, meshes are employed to link nearby nodes to each other.

Another is traffic, which is considered constant. General traffic and bandwidth usage are then abstracted and represented as constant bit streams. Likewise, so is the content of JBI traffic (what size of frames or packets are sent to the actual mission data encoded), which is considered as a uniform transmission.

The size of the scenario is restricted to a "local" network of units within close proximity, and so the physical size of the JBI network simulated covers only five to ten square miles of battlefield, with ten to fifty units participating as subscribers to the JBI services. Although the original documentation envisions JBIs to span AORs to deal with contingencies, the limits of ns-2 (see below) restrict scale. Additionally, for issues of scale, the idea of a JBI is that it can be composite [23], so a larger JBI can be a collection of smaller JBIs that are logically constructed/disassembled for a specific task. Therefore, the results for a larger topology can be inferred through experimental results of a smaller JBI network.

As mentioned, ns-2 has problems with scale. Besides being the original simulator environment for earlier implementations of *Cast protocols, it was chosen because of its modularity and "realism". However, due to that same realism, research experiments conducted using ns-2 tend to be within a range of twenty to forty nodes. According to the *Internet Draft Tools for P2P Network Simulation* [5]:

"Due to the realistic nature of the packet level NS2 simulator, scalability is a major issue. However, like most packet level simulators NS2 can run in parallel with a number of other machines. This can increase the maximum number of nodes for a given simulation but can become difficult to man-

age. Subsequently, NS2 is commonly used for simulating small networks and is generally unsuitable for modeling overlay networks. "

This characterizes the difficulty with using ns-2 to simulate larger networks with substantial amount of nodes. Additionally, there are some design choices (discussed in Chapter V) that limit the speed performance of the *Cast family in simulation, tradeoffs that yield a tighter, more comprehensive gossip model but result in longer simulation run-times if the network is too large.

1.6 Summary

This chapter covered the basis of this research-- the system or systems of JBIs, the concept of probabilistic protocols and specifically the *Cast family of probabilistic protocols, and the basis for this research. The successive chapters will follow the research outlined in this section. Chapter II expounds upon the background by outlining and summarizing the previous works in this field. Chapter III will outline the methodology of the experiments used to compare the performances of each of the *Cast family to each other and to SBCast. Chapter IV will provide in-depth analysis of the results of the experiments, and Chapter V will summarize the previous three chapters, draw conclusions from the experiments and post recommendations for future research.

II. Literature Review

2.1 Chapter Overview

This paper derives the use of Adaptive Gravitational Gossip to monitor publish/subscribe functions and quality of service Joint Battlespace Infosphere (JBI) systems to improve reliability and scalability of the overall collection of JBIs. In order to adequately address this problem, this chapter lays the foundation upon which to build. That foundation includes a description of how JBIs should work as well as distributed system ideas and concepts that play into JBIs, and the current research involving gossip protocols.

2.1 The Problem Domain: JBI

In solving any problem scientifically, the first step is to define the problem. The first literature sources reviewed in this section describe and define the concept of JBIs. These papers explain what JBIs are, their key objectives and describe challenges in establishing JBIs, and give an overview of current JBI research.

The first paper and foremost of research involving JBIs is by Marmelstein, and describes the overall JBI concept and the idea of Force Templates [19]. His paper identifies the overarching concept of JBI: to provide ubiquitous and uniform information dissemination between all US armed forces and Coalition allies, and to do so in a way that prioritizes the provided information by relevancy, mission and need-to-know [19]. The JBI's purpose is to address the barriers to information sharing among coalition military units. Historically, these barriers are "stove pipe" systems and classification issues. The key requirement is that information is broken up into discrete pieces which are easier to

control and disseminate to various entities (information customers, which could be anything from an infantry unit to a logistics office to a tactical fighter) according to various requirements, such as mission, geographic locale, “need-to-know”, classification level, weapon system, or priority. The chief means of accomplishing this parceling of information is the concept of “Force Templates”, the idea of specifying information customers or classes of customers via a “template” of properties (missions, classification level, nationality, priority, etc) and providing information according to these properties via a *publish and subscribe*, or *pub-sub*, system (defined below). The force template would provide the context and policies under which an entity would interact with the JBI. The overall motivation is to seamlessly integrate all coalition units into a unified Information Infrastructure to streamline command, control and intelligence operations. This is summarized as “with the right access, provide the right information to the right person in the right form at the right time.” Figure 2.1 diagrams the envisioned capabilities of JBI systems. This paper lays down the requirements for JBIs, but does not go into implementation details.

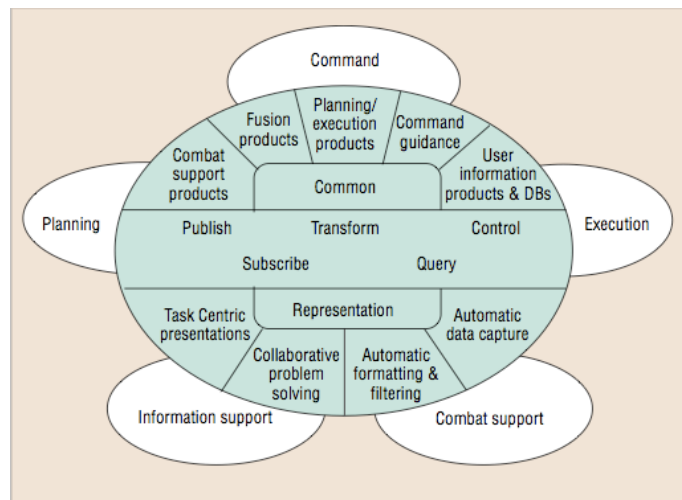


Figure 2.1: Organizational Diagram of JBI

Satterthwaite [23] builds on Marmelstein's work and extends the Force Template concept. In a Satterthwaite's work [23], the authors develop a specific force template to allow individual aircraft to become subscribers to a JBI. Called Guardian Agent, and part of the Insertion of Embedded Infosphere Support Technology (IEST) program, the system addresses the specific requirements to allow an aircraft to become an entity, such as how it links up and receives updates from a JBI via currently existing and future hardware. [23]. This paper provides a software model for system implementation and details interaction among physical parts of a network by which an entity using the Guardian Agent concept can interact with JBIs. The implementation makes use of Java Remote Method Invocation (RMI). A Guardian Agent system is an example of a client that interfaces with JBIs.

Focusing on the core JBI systems, Combs and Linderman [6] addresses the publish-and-subscribe capability of such systems. The key contribution of their work is the justification of the rationale for pub-sub systems with illustrative example code using the Java language and Jini technologies. But what exactly is a pub-sub system? Garlan and Shaw from Carnegie Mellon University describe this form of software architecture in a document. They provide a working definition in their paper describing traditional software architectures of computers and networks, many still in use today. Referring to pub-sub as a "blackboard", Garlan and Shaw's description keys in on the intent by Marmelstein. In their article, Garlan and Shaw describe the blackboard as a central repository of information, which could be composed of separate knowledge sources. Using the system, users could write to some common blackboard data structure, and then respond opportu-

nistically as changes are made to the blackboard [10]. An abstract diagram of this concept appears in Figure 2.2. In that figure, each of the users (ks1 through ks8) can either write

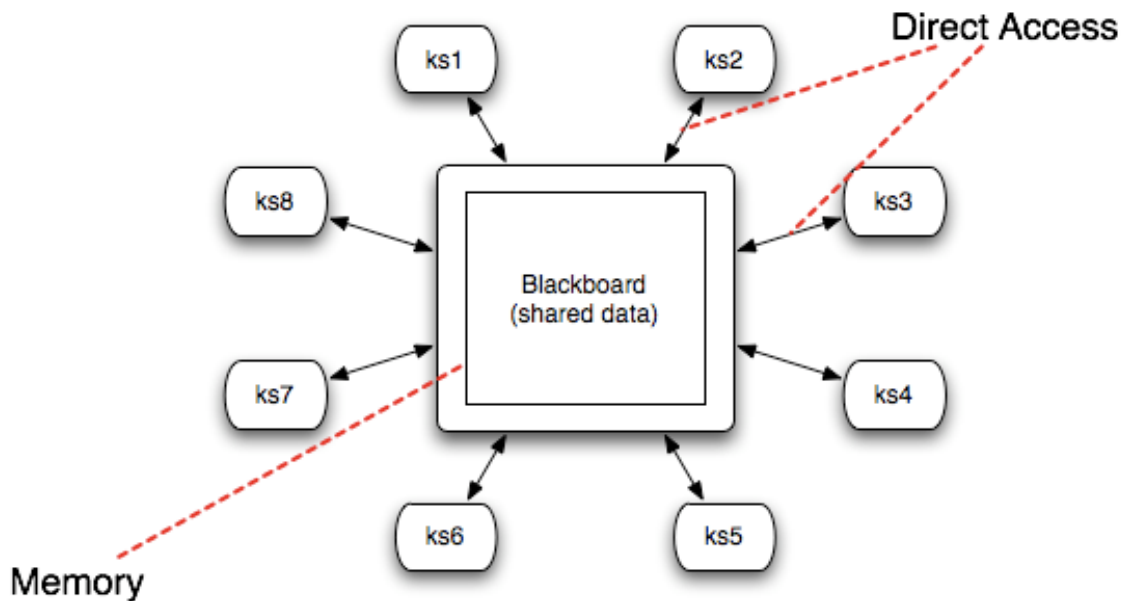


Figure 2.2: Publish-and-Subscribe Software Architecture

to or read from the shared data asynchronously. When the users read the shared data, they are *subscribers* and when viewing the shared memory receive updates. When the users write to the shared data, they are *publishers* of the data and are sending updates to other users. This extends upon Marmelstein's vision, which considers the collection of JBIs as a central repository that various, heterogeneous knowledge sources, such as fighter aircraft, tanks, ground units, supply depots, intelligence agents, satellites or any other military information customer can access and update information critical to their missions. In addition, this formulation as a blackboard makes the overall system more reliable as a distributed system because the central repository can still exist and be reachable, even if it is only intermittently available to customers suffering adverse network conditions.

The publish-and-subscribe architecture was chosen for JBI systems because it would decouple the various computer and communication systems that would make up, or interact with, JBIs; provide finer granularity and control of information released via JBIs; and address the challenges of unreliable or connectionless links of JBIs as distributed system. Ultimately, as a publish-and-subscribe system, entities would not need to be connected to a JBI or JBIs at all times; and information can be released in discrete pieces per the requirements laid out by Marmelstein [19]. Comb and Linderman's paper then provides an implementation of core publish-and-subscribe capabilities written in Java and utilizing Jini [6]. Jini is a service architecture for Java that provides network devices in distributed environments, with a network level plug and play capability. Unfortunately, at the time of publication, experiments using this code were not published. However, the value of the implementation is that it provides a schema for entities to interact with the JBI. The implementation also provides logic for the core middleware, independent of the quality of service or prioritization issues for the network upon which the entities and JBIs operate [6]. As such, Combs and Linderman illustrate one potential form of the protocols and data standards that will be employed for JBIs, and provides more detail into the design philosophy for JBI.

Loyall, Lawson and Duzan examined Quality of service (QoS) issues [18]. Their work outlines the specific quality of service challenges within the JBI and some context/example scenarios for these problems as well as proposed solutions. These QoS issues are: timeliness of the information, precision of the amount of data, the accuracy of the data, the importance between source and receiver, and trust (security, classification and

need-to-know). Loyall also discuss *where* the means of monitoring and controlling QoS can be placed in the JBI architecture. Taking these into account, the authors [18] state that QoS support in JBI needs sufficient control and management of:

- The sources of information,
- The infrastructure for transporting information,
- The users of the information,
- The competing demands of multiple information exchanges, and
- Dynamically changing environments, e.g., requirements, mission modes, participants, and resource availability and usage.

As this paper emphasizes QoS, with a special emphasis on the properties of priority and need-to-know, it lays down the particular challenges for this thesis project.

2.2 Distributed Systems Work

Part of determining solutions to the challenges described in the JBI problem domain is to survey the current offerings for distributed systems. In verifying whether or not a gossip-based solution will yield desirable results, it is necessary to discuss and examine comparable systems and how they might address the problem domain differently. Evaluations of gossip versus other approaches are needed to see if it introduces concepts or ideas that would be useful in optimizing our approach or avoiding some serious pitfalls. Some of these other resources also provide important background for inter-related ideas. Overall, perspectives from these other approaches should give an excellent vantage point

to observe, compare and evaluate SBCast's differences, strengths and weaknesses, and how to deal with them.

An excellent distributed system with which to compare requirements for JBIs is the Astrolabe system, developed by Van Renasse with Birman and Vogels at Cornell [26]. It is designed to provide information from a collection of diverse systems spread throughout a wide-area network. Astrolabe expresses some of the intent of the requirements for JBI. One use the authors envisioned the ability to query for a particular file available to the networked systems, regardless of the node that physically holds the file. Additional uses included the ability to configure systems to report the status of members (uptime, memory available, processors available, etc). Astrolabe is similar to the JBI architecture in that it utilizes SQL queries, which have support in many vendors' database client programs, to communicate using a reliable, publish-and-subscribe technique. However, unlike in the JBI, Astrolabe requires significant management overhead and it requires central management servers, similar to the root and zone servers used in Domain Name Service (DNS) in Astrolabe's most straightforward implementation [26]. Both of these conditions limit Astrolabe's usefulness in ad hoc, wireless environment.

Since monitoring network conditions is a critical part of this research paper, one paper by Ji and Elwalid discusses an interesting technique for sending updates [15]. However, what makes this technique, called 'network inference', valuable to monitoring QoS in JBI is that it uses measurements based on multicast messages to assess the state of the network. What this means is that network inference is able to pass updates and mes-

sages amongst its nodes without sending specific network status protocol packets. As a result, the noisiness and chatter of Gossip-based protocols is significantly reduced[15].

Another good example of a distributed system geared towards monitoring and controlling a network is the system described by Gjermundrod et al [11]. It employs a status dissemination model useful for modeling the spread of updates; it is therefore not using a strict and noisy packet stream, but instead utilizes the semantics of status data to send and receive updates. What this means is that each subscriber in a status dissemination model receives an update and derives various QoS measurements from attributes of that update, such as when it was received, who it was sent by and so forth [11].

For direct comparisons with Gossip and another distributed system, Rodriguez et al provide a comparison between traditional gossip against a custom protocol called Lola in a wide area network[22]. The most important lesson from this paper is the weakness of using Gossip or any other probabilistic protocol in a WAN because gossip tends to have high latency in these environments. Rodriguez then designs and describes Lola, whose key characteristic is that it builds upon gossip by enacting a weighted update scheme that prioritizes which nodes will receive updates based on speed/bandwidth available. Although the author emphasizes a distinction from gossip, this is inherently identical to the concept of a weighted gossip protocol such as GWCast by Hopkinson et al [3].

A project that is analogous to JBI efforts is the Network-friendly Epidemic Multicast (NEEM) project, also by Rodriguez [21]. NEEM focuses on preserving bandwidth and keeping costs down on transmissions. These two qualities are what military communicators are seeking to enable them to execute network operations in chaotic, ad hoc

wireless environments. The paper examines different criteria for buffer management, and concludes that a semantic approach, which provides updates and messages within the protocol itself, such as placing update information in headers, instead of placing them in a message payload, works best. This reasoning is similar to Ji and Elwalid's work [15], where both Rodriguez and Ji conclude that the wrapping, unwrapping and decoding of messages in TCP and UDP are a cause of unreliability. Although there are many similarities among these papers and focus on JBI, the problem domain differs; Ji and Elwalid's research is geared towards making online multiplayer games more efficient. The intent and overall analogy is useful because the requirements for the game are similar to the requirements for JBIs.

2.3 Adaptive Gravitational Gossip Background

Finally, after building upon the problem domain explanation and background review of similar distributed systems, the next task is to fully define what we mean by *adaptive gravitational gossip* (extending from the "adaptive gossip" definition from Chapter I), and to examine its background and ongoing research in its area. Studying these resources, we can note what has been explored before, and extend upon it; or expand the field towards areas that were not observed. However, knowledge of other experiments involving gossip within the perspective of the problem domain will provide valuable insight and guidance towards formulating research and working towards a solution. More importantly, this section discusses the previous experiments and characteristics of the earlier *Cast protocols, PBCast, GWCast and AGWCast.

As mentioned in the Definitions section in Chapter I, one of the first papers to conceptualize gossip protocols is by Kenneth Birman [4], and it is the source of PBCast, the original epidemic protocol that SBCast is derived from. Birman conceived PBCast to be an asynchronous system that is both scalable and reliable. It addressed the traditional distributed system issues of atomicity, stability and scalability; Birman ran numerous tests to demonstrate this.

Extending on Birman's work on Bimodal Multicast, Hopkinson and Jenkins derived the concept of "gravitational gossip" in a successive paper [3]. Gravitational gossip, or GWCast, was a protocol where the probability of updates going to a node was based on a given weight, and this weight was derived from various factors, such as node processing ability, congestion or, in the context of this paper, available bandwidth. This paper provides some preliminary tests simulated in a lab environment, but does not specifically address a problem domain [3].

Hopkinson continued work on GWCast by extending it and adapting it to a specific problem domain. He extended the idea of gravitational gossip by making it *adaptive*; that is, the weights attributed to nodes were dynamically adjusted depending on current network conditions. The problem domain studied the power grid domain examined by Gjermundrod [11]. The result was the next version of *Cast, AGWCast, which differs from GWCast with its use of adaptive weights. Hopkinson's paper provided further technical and implementation details for building an adaptive gravitational gossip system, the same details that will be used for the systems and experiments described in this thesis. This thesis then, extends upon Hopkinson's work by applying adaptive gravitational gos-

sip concept, with enhancements produced by target quality adjustment in SBCast, to the JBI problem domain.

Simultaneous research also conducted on gossip protocols includes an “adaptive gossip” protocol by Rodriguez et al [22]. Rodriguez compiles a compendium of existing gossip implementations, and derives a technique that is a gossip protocol that adapts based on the resources available and the network congestion. This technique differs from Hopkinson’s [13] because it is not weighted, which means that it utilizes the age of a message to determine where to send updates.

Another related work that chooses a problem domain similar to the JBI environment is by Qi Zhang and Agarwal, and that work applies gossip to ad hoc networks [27]. In that paper, the researches compare multiple approaches to the probability a node utilizing Gossip would update: they send updates based on a counter, according to network distance, based on location or by neighborhood knowledge. After investigating all of these variants, the researchers introduce a new hybrid approach that combines a fixed value with a counter as the basis for the gossip probability weights. The results demonstrate that the hybrid approach is the most scalable and reliable, and also provides insight into some of the drawbacks with probabilistic epidemic approaches.

Addressing a different problem, Jelasy et al. consider a gossip application for peer sampling services [23]. They examine the scalability of gossip by measuring its performance in small diameter networks to large clusters of nodes. The authors analyze the performance of gossip in complex networks with statistical physics techniques. Their paper thus provides a useful basic analysis of gossip-based networks[23].

2.4 Summary

This chapter has summarized important work related to gossip protocols and JBIs. The research described in this thesis draws on the previous works described in this chapter, introduces new concepts, and combines and formulates the sum of the previous concepts described here into a proposed schema for accomplishing quality of service and prioritization for JBIs using adaptive gravitational gossip.

III. Methodology

3.1 Chapter Overview

This section covers the methodology used to create and explore the capabilities of SBCast, an adaptive gravitational gossip technique based on AGWCast. The general problem is evaluating SBCast in a scenario that models JBI in an actual, operational environment. The research approach is to implement SBCast in simulation and run experiments on this simulation. The simulation is comprised of a topology representing a possible battlefield network of units as JBI subscribers that are all running SBCast as the means for receiving JBI resource updates, and various conditions used to test SBCast's reactions. The experiments vary by changing the conditions in the simulation. This is to meet the objectives of the experiment, which is to determine if the SBCast protocol can maintain the quality of updates despite increased or reduced network conditions. If the quality of the most important updates can be maintained, perhaps at the cost of reducing the bandwidth available to less important information streams, then this will allow users to make best use of the total bandwidth available according to their own priorities. For perspective, the experiments will run trials involving SBCast and will also compare against the other members of the *Cast family (GWCast, AGWCast and traditional PBCast). This chapter discusses the formulation of the problem, describes the scenario used, defines SBCast and its implementation, discusses about the experiments and procedures used to execute the experiment, and outlines the development of the code to make it all happen.

3.2 Problem Description

The formulation of the specific problem is this: given a local, battlefield arrangement of assets (troops, vehicles, etc) how can we maintain the level of information updates despite periodic bursts in traffic of competing background traffic. Towards this end, we consider forty-one nodes, which can represent vehicle units or troop formations. These nodes are connected via directional or directionless wireless links, and these connections are not only shared by the members of this gossip network but also by nodes that are not running *Cast agents but who contribute to the overall traffic. The scenario can be customized to create a specific instance of the problem.

3.3 Scenario

The scenario consists of three components. Its purpose is to simulate a possible formation of air assets in order to accomplish an air superiority mission. The three components flesh out these roles. The first part, the topology, represents the units themselves, and are nodes in a graph that represent an air unit (for example, A-10 ground support aircraft or F-22 attack aircraft) running a *Cast agent. The second part consists of the links, or edges, in the graph. The nodes and the links form the full graph, and represent the logical relationship among the nodes. All the nodes receive updates in this arrangement, and these links could represent directional wireless links, such as a direct radio transmissions or satellite feeds. The third part of the scenario is the set of data streams that are updated via the *Cast system. These application streams represent the JBI in the scenario, and all nodes subscribe or send for these streams. The last part of the scenario is the set of

conditions, which represents other existing network traffic besides the JBI application streams. Implementation details for the scenario appear in Appendix A.

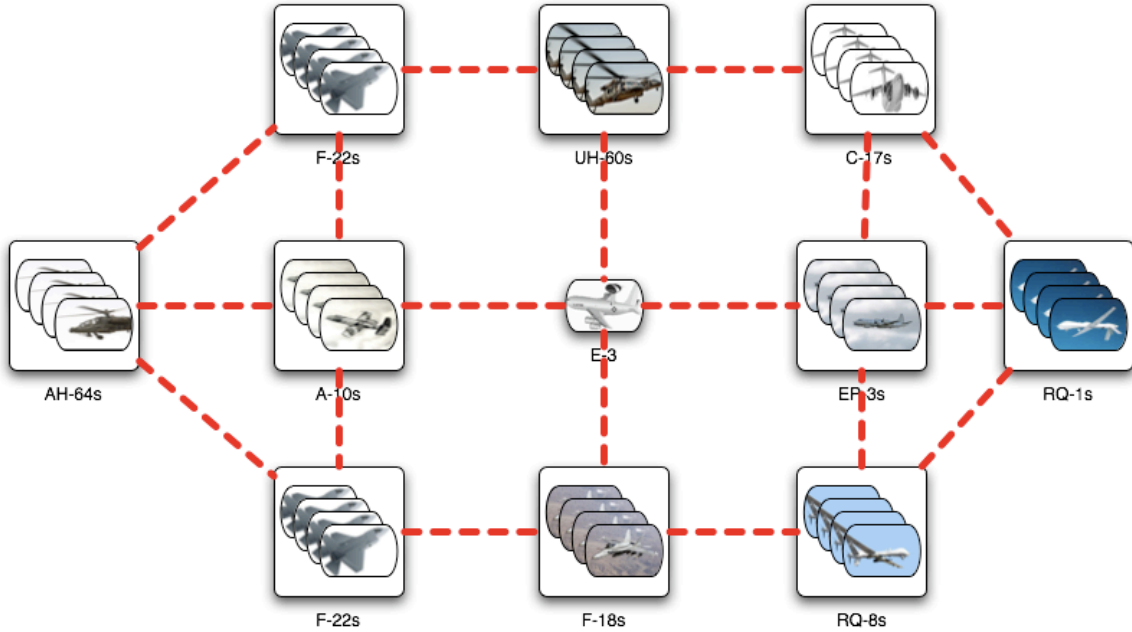


Figure 3.1: Scenario Topology

The topology appears in Figure 3.1. Figure 3.1 is a simplified view of the diagram, with clusters of four represented by the bigger icons. These clusters are referred to as *group levels* and represent like units meshed together, as seen in Figure 3.2. These group levels are considered to be in close proximity with each other and share the same

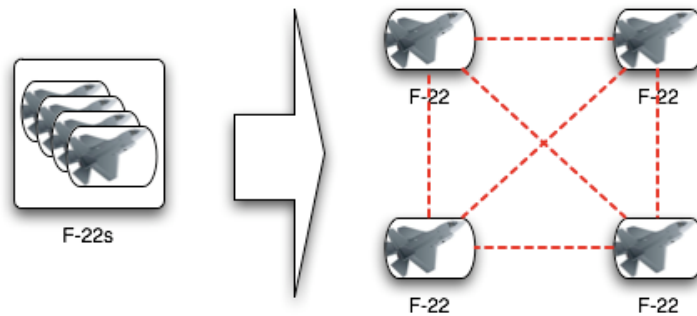


Figure 3.2: A *Cast Group Level

interest levels in JBI application *groups* (see below). There are forty standard client nodes and one sender node in the topology. The sender node pushes out *Cast traffic, and all the nodes *Cast agents receive it, and probabilistically forward the updates onward. In ns-2, all the nodes are node objects, and each have *Cast agents, software that represents the JBI application streams sending out the *Cast traffic, or receiving and forwarding the *Cast traffic along the edges.

Each edge is a logical representation of some sort of wireless link, whether it is a radio link, satellite link or a combination of both. In the figure, thinner edges represent lower bandwidth links, and thicker edges represent higher bandwidth links. It is important to note that edges in the figure represent which nodes are “closer” to each other and thus the direction in which updates arrive. This forms a logical relationship where the units communicate/relate to each other via a gravitational gossip protocol. Additionally, to better portray a wireless network, the subgroups within the gossip groups are totally connected meshes. Overall, the topology is strung together as a combination star-mesh network. Finer grained detail of the topology can be found in the expanded diagram in the appendix (Figure B.1).

As mentioned earlier, each node subscribes to one of four different pub-sub applications, or *groups* of the JBI. Groups could also be considered the collection of all the nodes that are *subscribed* to the same pub-sub application. Subscription means that the node can receive and read incoming updates, and publishing means being able to write updates. These updates represent the information published in a JBI; for example, an update might be an amendment to the Air Tasking Order (ATO), or the status of forces in a

given area. On the other hand, updates could represent, as they would under the stated requirements of the JBI, an update of an already existing data feed, such as a satellite imagery stream or an intelligence database query result. Groups push out information in periodic bursts, which are scheduled and predicted. Periodic bursts originate from *senders*. In this topology, the only sender is the AWACS (E-3) in the center of the topology. A collection of the data streams, which model possible group applications for use in JBIs, appears in Table 3.1.







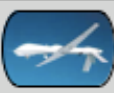

Table 3.1: Groups

Group	Name	Description
ATO1	Air Tasking Order Stream #1	Zone assignment, targeting and mission data; important to fighter and bomber aircraft.
ATO2	Air Tasking Order Stream #2	Additional information not included in the first Air Tasking Order stream
CAS	Combat Air Support Data	Specific target and tasking information in support of ground forces; important to ground support aircraft, combat rescue and troop transportation aircraft.
INT	Intelligence Imagery Feed	Provides updates to a satellite and reconnaissance imagery data repository; important to reconnaissance aircraft and unmanned aerial vehicles.

Interest is a characteristic that describes the priority that nodes are subscribed to with respect to a given data stream. Another important term is *group level*. Whereas a group refers to all subscribers to a given information stream, the group level, or subgroup, refers to a subset of the overall group with its own level of interest in the informa-

tion stream. For example, a group might have two subgroups. The first subgroup might be interested in receiving roughly half of the overall updates while the second subgroup might only wish to receive a quarter of the information updates. Interest appears in Table 3.2. Each column represents a particular unit/agent, and each row denotes that interest to the data stream as a percentage. The greater the percentage, the most likely a node will

Table 3.2: Interest Levels

Unit/ Agent								
	F-22	A-10	F-18	AH-64	UH-60	E-3	RQ-1	RQ-8
ATO1	80	70	70	65	65	75	60	60
ATO2	80	70	70	65	65	75	60	60
CAS	50	80	65	85	90	35	35	55
INT	50	45	65	66	66	80	90	85

receive and pass on a message from that particular stream. Also referred to as *target qualities*, for GWCast and AGWCast, the target qualities are constant. For SBCCast, the technique is to lower or raise the target qualities by the status of the network, so these fluctuate as interest in each specific data stream waxes and wanes with changes made by the protocol. In other words, interest is dialed up and down depending on the heuristic used by SBCCast.

The last part to be modeled is the simulation of traffic other than the JBI application streams. Referred to as the *network traffic conditions*, these represent other data transmissions that occur in the battlefield. Sampling real world traffic would be too com-

plex and varied, and would result in longer simulation run times or inconsistent results. The controlled environment requires only that there are high or low levels of traffic. Therefore, the way these conditions are handled in the ns-2 simulation is as a set of constant bit rate (CBR) traffic generators at each node for each link. These CBRs can be turned off and on to simulate transmissions occurring at each node, and turning on more CBRs can simulate more traffic.

To create rolling waves of traffic, the CBRs are turned on, one-at-a-time, at periodic intervals. For example, at 500 ms one CBR is turned on. At 1000 ms, the next is turned on, and so fourth. This would occur until all the required CBRs for the network traffic condition are on and transmitting data. Once all the CBRs are active on all links, the scenario will continue for a set time. After that period is over, the traffic will recede, and this is done by simply turning off each CBR, one-by-one, until all the links are free does this.

Figure 3.3 illustrates the steps of this procedure. It shows successive views from the ns-2 network animator (nam) user interface. nam is a tool that is used to visualize ns-2 simulation events. Here, it is used to illustrate, on a simple example mesh network, the gradual activation of the CBRs. Figure 3.3a shows the ns-2 graph where no CBRs are active; Figure 3.3f, on the opposite end, shows the graph when all CBRs are active. One CBR is turned on a link towards the start of the simulation, as shown in Figure 3.3b, and successively CBRs are each link are turned on, gradually building the number of CBRs active (Figures 3.3c, d and e) until finally all links are active in Figure 3.3f. For recessing the traffic, the scenario needs to only script events to occur in the reverse order, from

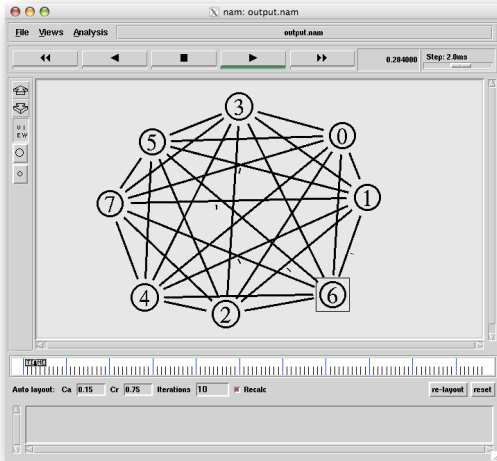


Figure 3.3a: No CBRs Active

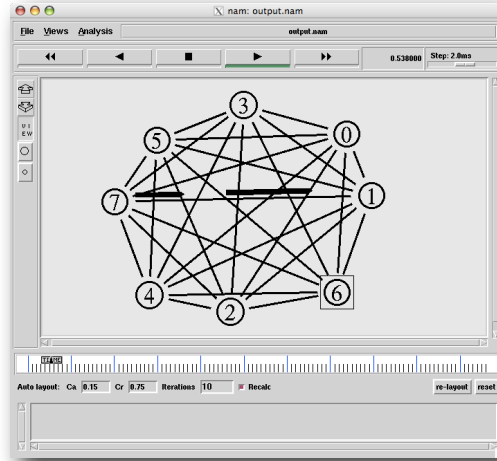


Figure 3.3b: One CBR Starts

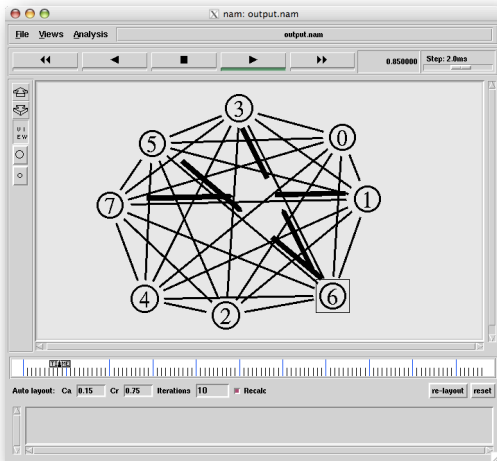


Figure 3.3c: More CBRs Start

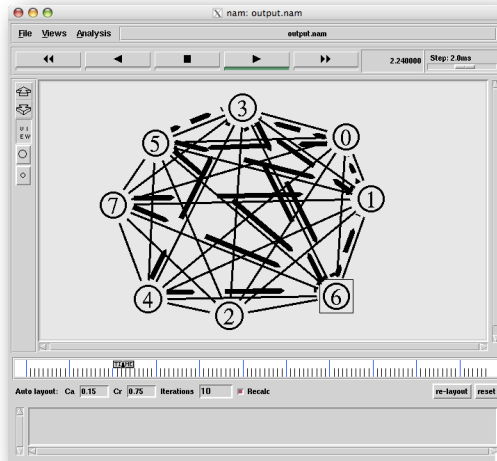


Figure 3.3d: 75% of CBRs

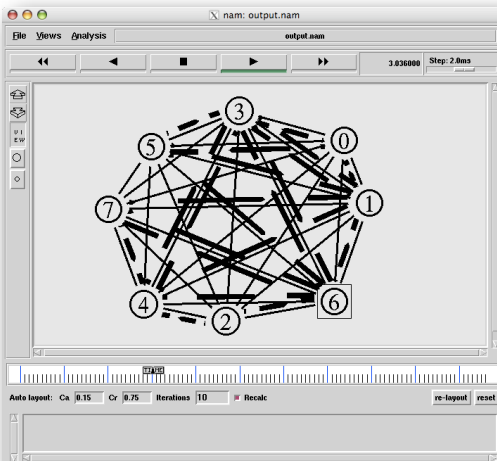


Figure 3.3e: 90% of CBRs

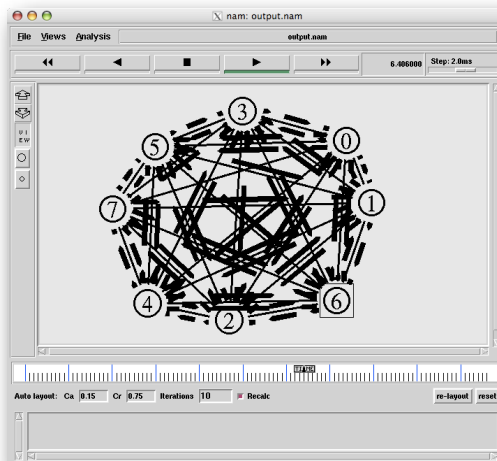


Figure 3.3f: All CBRs Active

Figure 3.3: Progression of Network Conditions

Figure 3.3f back to 3.3a. This is the basis for the network conditions in the experiment.

3.4 The Novel Protocol: SB^{*}Cast

The novel protocol and subject of this paper is called SB^{*}Cast. It is a variant of the ^{*}Cast protocols, and evolves directly from AGW^{*}Cast. SB^{*}Cast is a descendant of PBC^{*}Cast, and thus retains qualities of PBC^{*}Cast, GW^{*}Cast and AGW^{*}Cast. As a result, SB^{*}Cast is similar to its ^{*}Cast family members; it is probabilistic per PBC^{*}Cast, it prioritizes updates via

```
1  AGW*Cast::recv(packet)
2  round // current round we are in
3  weights
4  targets
5  if packet is of type... // other cases
6  end-if
7  if packet is of type timer
8    update round
9    send weight updates
10 end-if
12 if packet is of type weight update
13   evaluate weight updates
14   evaluate targets
14   recalculate weights,targets
15 end-if
16 End
```

Figure 3.4. Original Simplified AGW^{*}Cast `recv()`

gravitational weights in the same exact technique of GW^{*}Cast, and also incorporates the adaptive gravitational algorithm of AGW^{*}Cast.

The difference that sets SB^{*}Cast apart is the “slack” routine it adds to the ^{*}Cast protocols current behavior. The idea behind SB^{*}Cast’s slack routine is to lower the target qualities for the nodes so that they can send updates more easily. SB^{*}Cast stands for “Slack Broadcast”, and its name derives from this technique of lowering the targets (giving slack) to enable the protocol to “catch up” by backing off on the intended probability

```

1  SBCast::recv(packet)
2  round // current round we are in
3  weights
4  targets
5  if packet is of type... // other cases
6  end-if
7  if packet is of type timer
8    update round
9    send weight updates
10 end-if
12 if packet is of type weight update
13   evaluate weight updates
14   evaluate targets
15   TargetStrategy::adjust_targets
16   recalculate weights(weights,targets)
17 end-if
18 End

```

Figure 3.5: `recv()` Modified For SBCast

of sending updates. Once the updates have caught up, through slowed rate of updates or increased bandwidth, the target qualities are restored at a rate commensurate of the restored bandwidth. This can be also described as “backing off.” SBCast agents at different nodes sense network conditions in their part of the network, and by coordinating with their peers within the same group level, “back off” of their original intended target qualities and accept sending messages at lesser rates in order to allow bandwidth usage to clear up. However, the intent is not to simply turn off outgoing traffic. This defeats the purpose of making the application services reliable. Instead, the objective is to maintain an expectation of updates, albeit at some reduced rate. This expectation is expressed as an *observed-to-target ratio*, which is a ratio of the observed infectivity to the target quality. By maintaining this ratio, SBCast will be able to maintain an expectation of receiving updates proportionate to the original priorities required for a given subscriber.

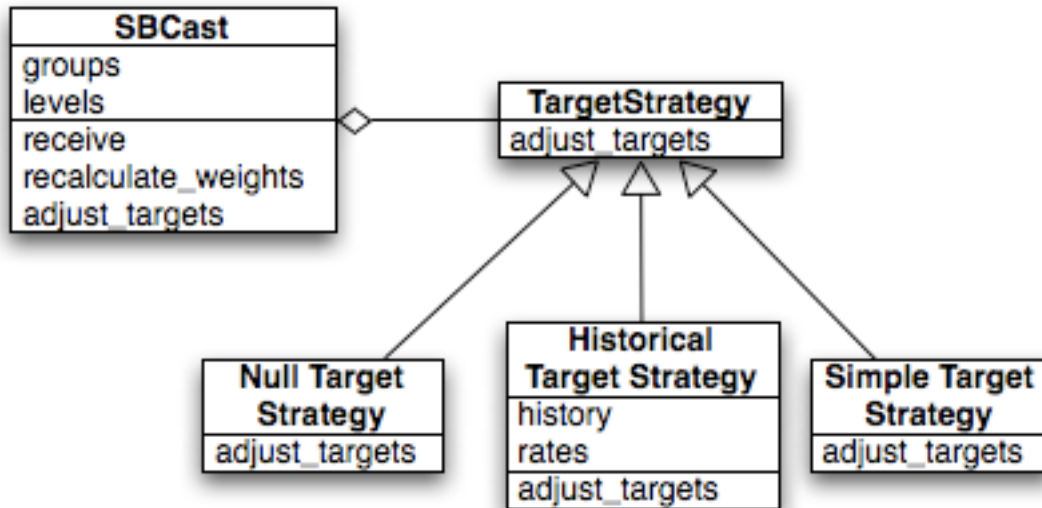


Figure 3.6: UML For SBCast Target Strategy

Implementation-wise, adding SBCast’s capability to the *Cast source involved adding a call to a subroutine to the algorithm that handles the receipt of packets. Figure 3.4 and 3.5 illustrate this convergence.

Basically, the first block of pseudo-code (Figure 3.4) shows AGWCast’s general algorithm: the `recv()` routine. The `recv()` is a common method call for protocols implemented for ns-2, and it is the method called when an agent in an ns-2 simulation must deal with an incoming packet. The second block (Figure 3.5) shows the changes (in bold) that were made to that original algorithm to create the `recv()` for SBCast. The change is an addition of a call to a subroutine, but with a subtle twist. That twist is that this new subroutine for SBCast is implemented as a software engineering *design pattern* known as a *strategy pattern*. A design pattern is an often-repeated pattern of code that has been used repeatedly and successfully in numerous software programs due to its ability to increase flexibility, performance or reuse [2]. A strategy pattern is a software engineering

design pattern that encapsulates a commonly used subroutine into a separate object, or *strategy*, so that either the strategy can be used again and separately later, or that a different strategy can be interchanged in the original call from the original routine to the sub-

```
1 HistoricalTargetStrategy::adjust_targets(curr_infectivity,
    curr_susceptibility,curr_target)
2   history := new array
3   rates := new array
4   adjustments := new array
5   min_slots := 10
6   threshold := 0.005
7   best_target
8   history.add curr_infectivity
9   rates.add curr_susceptibility
10  if (history.size,rates.size or adjustments.size < min_slots)
11    adjustments.add curr_target
12    return curr_target
13  else
14    if (almost all history less than curr_target)
15      new_target := history.max
16    else if (more than half rates < rates.average)
17      if (rates going down)
18        new_target := curr_target+(best_target-curr_target)/2
19      else if (best_target<history.max or rates.avg<threshold)
20        new_target := best_target
21      else
22        new_target := curr_target + (best_target -
          curr_target)/min_slots
23      end-if
24    else
25      new_target := curr_target
26    end-if
25    adjustments.add new_target
26    return new_target
27  end-if
28 end-if
28 End
```

Figure 3.7: SBCast Historical Target Strategy

routine [12]. In this case, SBCast's capability is encapsulated in the call to ad-

`just_targets` and a strategy object. The new form for the `*Cast` code appears below, in a simplified UML format in Figure 3.6.

The net result is that any target adjustment can be defined and substituted into `SBCast`, altering `SBCast`'s behavior. For example, in Figure 3.6 there are three strategy subclasses defined: the Null Target Strategy, Simple Target Strategy, and Historical Target Strategy. Null Target has a method `adjust_targets` that does nothing; Simple Target Strategy has an `adjust_targets` that adjusts target qualities based on the last round of data observed. These two strategies were developed prior to Historical Target Strategy, the actual strategy used for these experiments and explained in the next paragraph. Because all three classes are subclasses of the type `TargetStrategy`, by Liskov's principle, which states that any subclass of a class with identical interfaces for identical methods, can substitute for that superclass [17] any one of these target adjustment strategies can be used by `SBCast`. The strategy patterns opens up numerous avenues for future research, of which some are explored in Chapter V. This architectural change is what makes `SBCast` different from its relatives: it adds an additional step of adjusting targets between reading network conditions and recalculating gossip weights, and encapsulates this behavior in a strategy.

Although different target adjustment strategies could be employed, this experiment used what is termed a *historical* approach. The algorithm is illustrated in pseudocode in Figure 3.7.

The premise of the historical approach is to observe a number of rounds (a "history") and make a decision to adjust target qualities based on those observations, re-

corded as “slots” in that history. The observations involve looking for rising or falling trends in observed infectivities and susceptibilities. From these observations, the algorithm adjusts the target quality in one of three different ways:

- Adjust down to the best observed infectivity
- Adjust up to the original target quality divided by the number of slots
- Adjust up to half of the original target quality

The algorithm was designed to back off aggressively but recover gradually. This technique should allow an immediate reaction to significant losses of bandwidth and reduce churning up the network once bandwidth is restored. This way, sudden losses are addressed while newly recovered bandwidth is utilized gradually and not hoarded by a single group level. The result of employing this algorithm should result in a gradual recovery of the observed-to-target ratio. The results of these experiments will verify if this is the case.

3.5 Experiments

Tests consist of ten runs of different network traffic conditions and cases based on the above scenario. Due to ns-2’s long run-times using the scenario outlined above, each test will be run for 400 seconds of simulated time. Each test is repeated ten times. Each test run is executed over a scenario with one of the specified network conditions and one of the listed cases. There are three different types of network conditions and four different cases, as explained below.

The three conditions differ in the set of CBRs that are turned on, as described in the scenario section. To create a situation where network bandwidth will be increasingly utilized at a constant rate, the CBRs are turned on, one at a time in that set, until all the CBRs are on and bandwidth is saturated. Then, to decrease the traffic uniformly, the CBRs are periodically turned off one-by-one. Each traffic condition is specified as follows:

- *Wide bandwidth links saturated.* The links with the higher amounts of bandwidth, which also form the backbone of the topology, become saturated-- that is, waves of interference (or other traffic) are flowing across the topology, limiting the bandwidth available between group levels.

- *Narrow bandwidth links saturated.* Similarly, the links with lower bandwidth suffer from these same waves of background traffic. These links are the edges within a group level. This traffic condition limits nodes connectivity to each other within their

Table 3.3 Experiments

	Wide Bandwidth Links Saturated	Narrow Bandwidth Links Saturated	All Links Saturated
PBCast	Test 1A	Test 2A	Test 3A
GWCast	Test 1B	Test 2B	Test 3B
AGWCast	Test 1C	Test 2C	Test 3C
SBCast	Test 1D	Test 2C	Test 3D

own group levels.

- *All links saturated*. Combining aspects of the first two conditions, this condition basically sends waves of interference throughout the entire network. All CBRs are on in this condition, and this should prove to be the greatest challenge to the *Cast protocols.

These conditions are then used to vary the cases, whose controlling variable is the type of the gossip protocol used. These cases are:

- PBCast - The original implementation of Gossip will be used. This will serve as a baseline for the performance of the other *Cast protocols and validate the experiments run on these protocols in previous papers. In this case, the agents have equal probability of updating any of the other agents. For implementation purposes, this means that all interest levels are set at 98%.

- GWCast - The nodes all run GWCast agents, which differ from PBCast only by setting interest levels according to Table 3.2.

- AGWCast - The nodes not only use GWCast, but also use adaptive behavior as well. The adaptive behavior involves sensing the current network conditions and adjusting the infectivity, also known as the probability that the node will receive an update.

- SBCast - This is the new protocol created as part of this thesis. Besides using weights defined in Table 3.2, and the adaptive behavior of AGWCast, this protocol behaves as outlined in the previous section: it senses current network conditions and adjusts the target qualities accordingly.

The combinations of conditions and cases comprise the experiments, as outlined in Table 3.3.

These experiments were run ten times with different random number seeds each time. The random number seeds are used to seed the random number generators from the standard C/C++ libraries used by the PBCast code. In addition, to simulate aircraft loitering in an area, all nodes will move in simple orbits but not leave a pre-defined area; in the ns-2 simulation, they will be configured as normal, non-mobile nodes.

3.6 Objectives

The objectives of these experiments are to compare the performance of the algorithms of the four gossip schemes, which include PBCast, GWCast, AGWCast and SBCast, under varying levels of waves of interference. The first series of tests (Test 1s) demonstrate the performance of traditional gossip (PBCast), the second and third tests (Tests 2 and 3) test gravitational and adaptive gravitational (respectively), and the last set tests adaptive gravitational gossip (SBCAST) with target adjustment (Test 4s).

Performance is considered in terms of how fast the pub-sub streams are sent throughout the network and in terms of how much of the published information is received by the average subscriber. In particular, the most critical information must still arrive possibly at the expense of less critical information streams. Less critical information streams will be proportionately lowered when less bandwidth is available or increased if more bandwidth is available. As such, the experiments will measure key factors such as how well agents are able to realize what their available bandwidth is, how well they are able to change their data stream subscriptions to match their predefined interests, and how quickly the adjustments can be achieved.

In terms experimental metrics, this is basically how a protocol recovers the observed-to-target ratio. The observed-to-target ratio is the ratio of the observed infectivity to the target quality. To extend earlier definitions: the observed infectivity is the actual probability a *Cast agent perceives in being able to send out an update for a particular group, and the target quality is the pre-configured interest a group level has. The ratio is basically an expression of the observed probability a node can send out a message that will be received by a peer, and target quality is the expected probability a node can send out a message that will be received by a peer. As Hopkinson described observed infectivity[13], the observed infectivity is the result of AGWCast's weight algorithm executing a least squares calculation on susceptibility measurements taken at nodes. This means that the actual probability that a node will send an update is based off of the probability that a node will receive an update, which is based off of whether an agent has successfully kept up with gossip rounds; the ratio of the observed infectivity indirectly monitors the bandwidth by noting how successful a node might be in infecting a peer with an update. The observed infectivity, when compared to the target ratio, should ideally be at 100%, indicating that the target quality set for that particular agent is utilizing the available bandwidth perfectly. If the target quality is less than 100%, then an agent is only receiving a percentage of the updates it expects. The lower the percentage value of the ratio, the less efficiently the agent is using the bandwidth for sending and receiving updates for a given application group.

Therefore, the observed-to-target ratio captures the success of the algorithms in the experiments. The higher the observed-to-target ratio, the better the protocol adapted

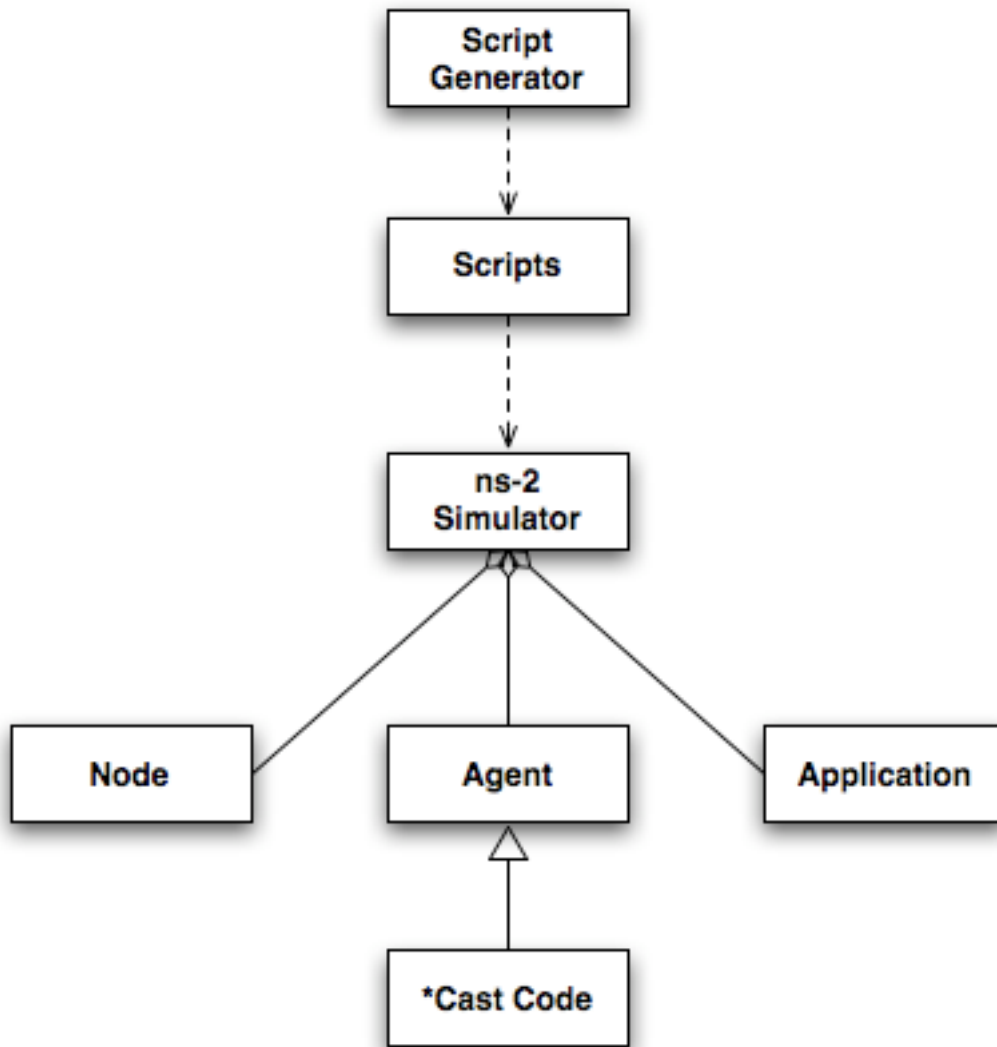


Figure 3.8: Script Generator Program

to shrinking or expanding bandwidth as it is receiving the majority of the updates it expected to receive in the first place. The lower observed-to-target ratio, the worse the protocol is doing in terms of adapting to shrinking or expanding bandwidth. In the former case, the protocol is successfully “backing off” and preserving or reserving bandwidth for higher priority data streams; in the latter the case, the protocol is unable to sense changes in the network and is continuing to adversely affect the bandwidth available.

These experiments will meet the objectives because they demonstrate the adaptability of the protocol under changing conditions. They will illustrate how SBCCast might be used as a feedback system for higher-level agents, and compare how that feedback system can provide improvements over traditional methods of non-deterministic message passing.

3.7 Procedures

Following the Table 3.33 above, there are four total tests to accomplish, and with ten runs and three conditions, a total of 120 runs to accomplish. They are automated using a script that builds the scenario, submits it to ns-2, executes and then records the results to a file. Each test runs one at a time, and consists of initializing a topology with the specific *Cast agent (PBCast, GWCast, AGWCast or SBCCast), configuring with a specific network traffic condition, generating a randomly generated seed, and executing ns-2 on the newly created output. Another script automates the interpretation of the collected data. The entire process is illustrated in Figure 3.8.

To explain more precisely from Figure 3.8: the script generator program creates a script for one of the tests. It embeds a randomly generated number to seed the random generators in the *Cast agents to vary the repeated tests. Then, it executes that script for that test. When that script is executed, it launches ns-2 simulator and sends its information (the topology that is described in the script) to ns-2. The simulator then executes the scenario described by the script, creating node, agent and application objects and executing their actions. For example, the *Cast code is a specialized agent object within ns-2,

and is called and used by the simulator as required by the script. Hence, the script generator creates different traffic conditions, and the ns-2 simulator utilizes the different types of *Cast agents. This procedure is repeated 120 times; once for each test required.

3.8 Summary

This chapter discussed the problem to be explored, the scenario to specify the problem, described SB*Cast, the protocol developed for testing, outlined the experiments to test *Cast protocols, and the procedures to accomplish the experiments.

IV. Analysis and Results

4.1 Chapter Overview

This section reviews and evaluates the results of the experiments described in the previous chapter. The first part of this chapter describes and displays the graphs of the results. The second part discusses their significance and to the key investigative questions posed at the beginning of this paper.

4.2 Results of Simulation

4.2.1 *Observed-to-Target Graphs*

The key question answered was relating to how well the protocol adapted to adverse conditions. Because each scenario was run ten times (ten trials) with each of the three conditions (three experiments), there are three graphs for each of the three experiments. Each graph records the performance of all four protocols within one experiment as an x-y scatter plot of the observed-to-target ratio to time. The adverse conditions that affected the protocols consisted of the traffic generators that were turned on, and so each graph also records the number of active CBRs to time as well.

Before the main graphs are fully explained, it is important to note the activity of the CBRs and what they are doing in the main graph, because the number of CBRs active is the variable that differentiates the experiments from each other. Before they are plotted to the main graphs for comparison with the observed-to-target measurements, it is helpful to view them on a single graph, in Figure 4.1.

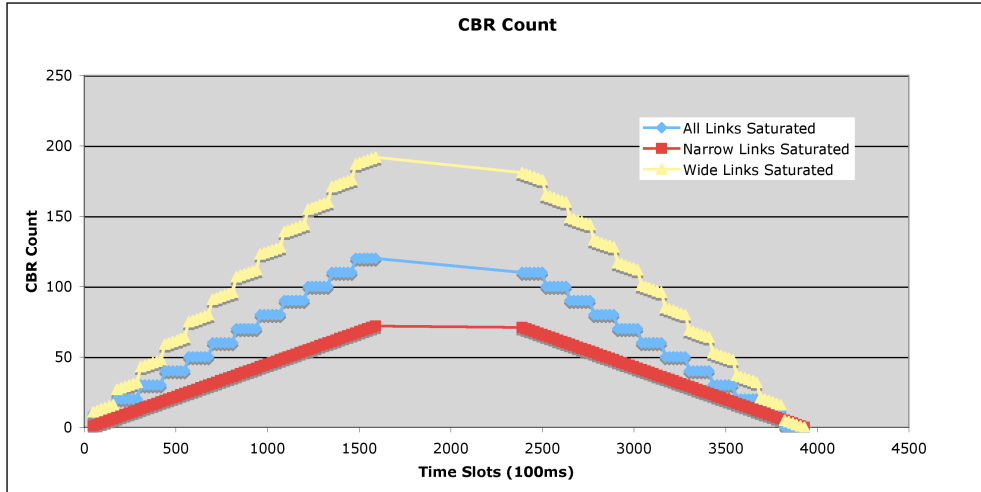


Figure 4.1: CBR Counts For All Experiments

This graph has three x-y scatter plots per different network conditions. The x-axis are in *time slots* of 100 ms. The y-axis is the number of CBRs that are turned on at a time slot. These graphs will be transposed directly onto the main graphs (see below).

Like the graphs in Figure 4.1, in the main graphs, the x-axis in each graph represents time in 100 ms time slots. The y-axis on the left of each graph is the observed-to-target ratio, and scales from 0.00 to 1.00. The y-axis on the right is the number of CBRs active, and is the same y-axis that appears in Figure 4.1. The CBR graph specific to that experiment appears in the pertinent graph. The values for observed-to-target ratios are geometric mean values, and these are found using the geometric mean formula of:

$$\left(\prod_{i=1}^n a_i \right)^{1/n} = \sqrt[n]{a_1 \cdot a_2 \dots a_n}$$

Where n is the total number of agents in the topology, and a is the value of observed-to-target ratio observed at an agent during that time slot. This mean was used because of the diversity of agents in terms of locations, trials and interest levels set. The graphs of the results follow over the next three pages, as Figures 4.2, 4.3 and 4.4.

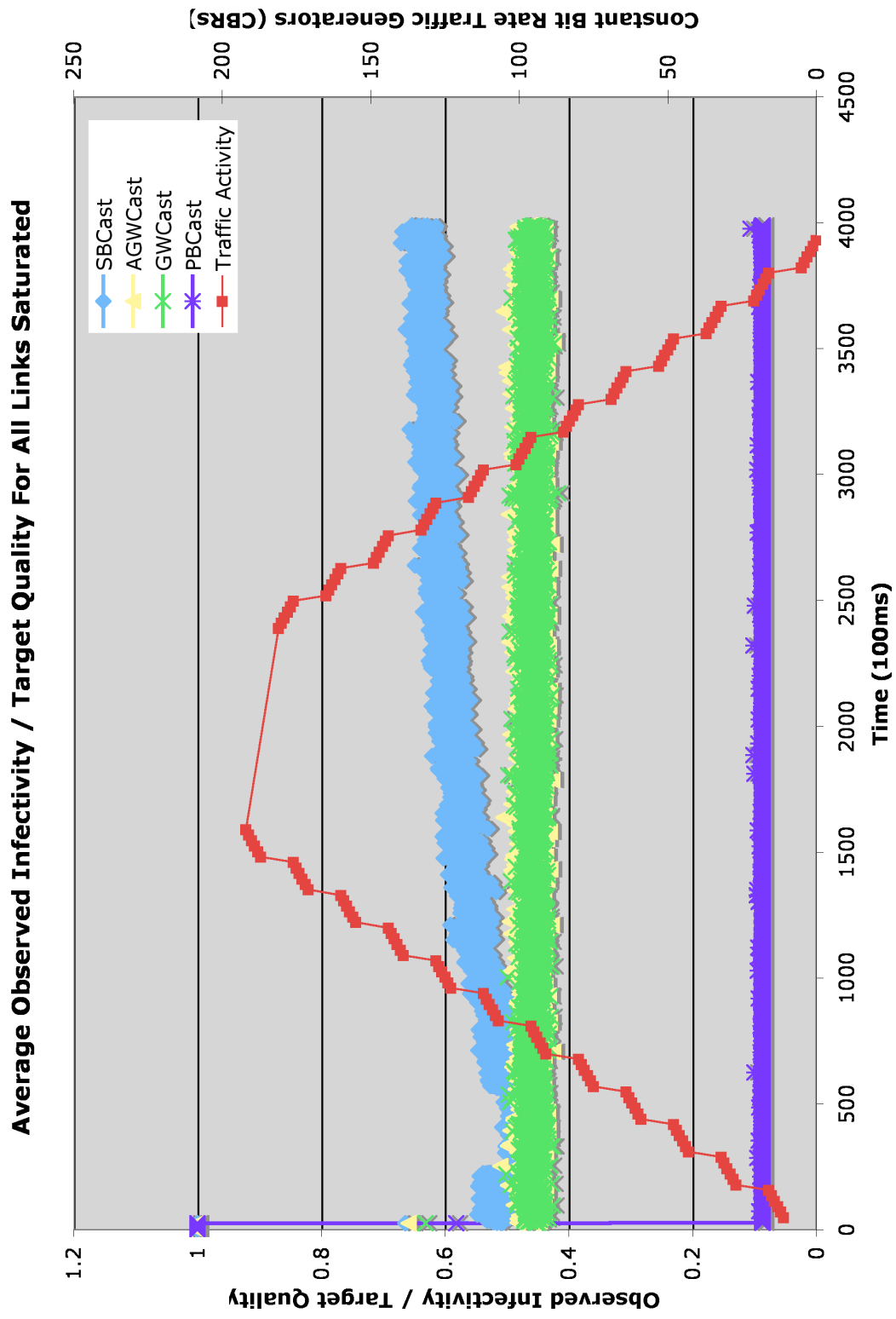


Figure 4.2: Test 1 Results (All Links Saturated)

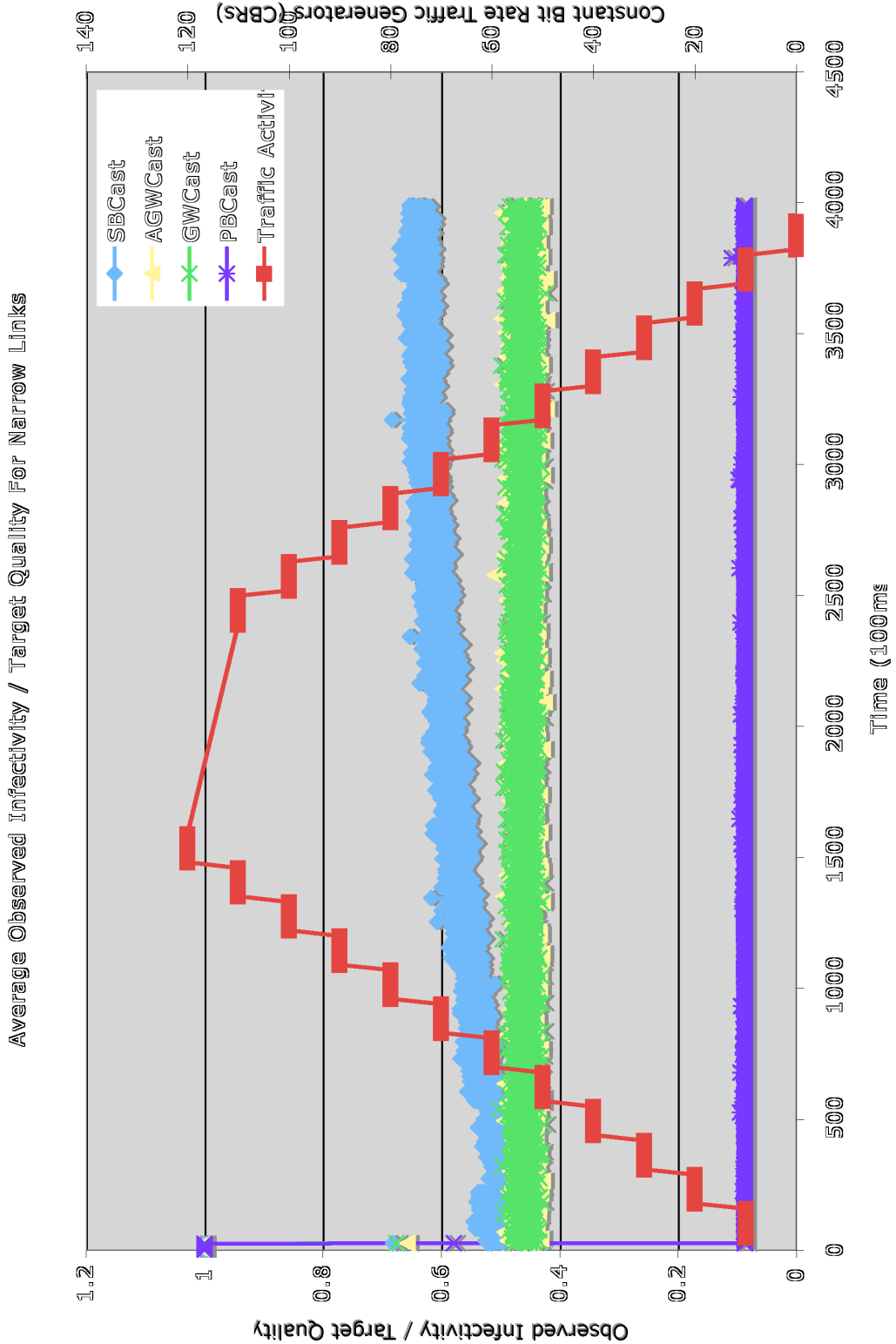


Figure 4.3: Test 2 Results (Narrow Links Saturated)

Average Observed Infectivity / Target Quality For Wide Links Saturated

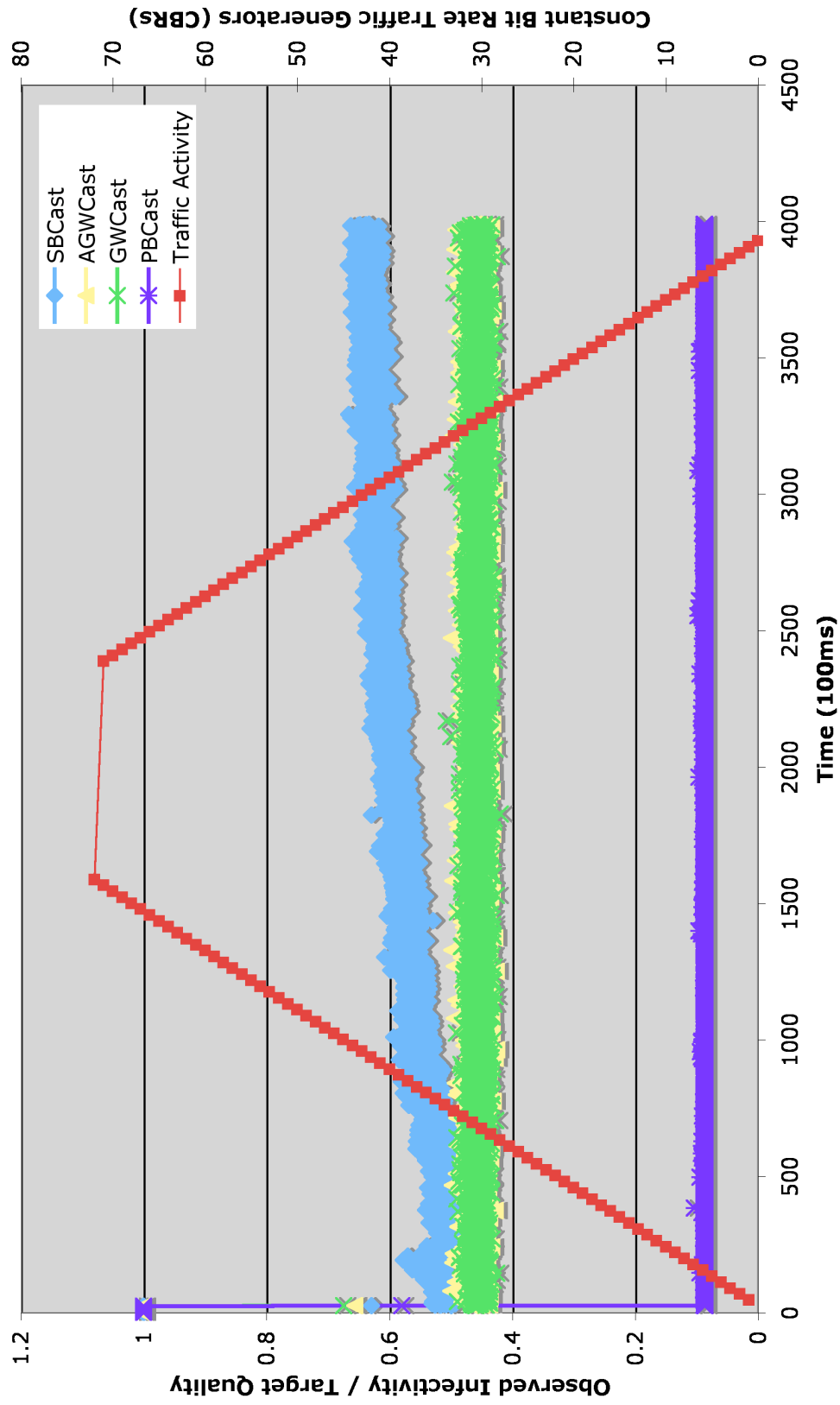


Figure 4.4: Test 3 Results (Wide Links Saturated)

4.2.2 *Standard Deviation Graphs*

The standard deviation graphs appear over the next six pages as Figures 4.5 to 4.16. Figures 4.5 to 4.8 are the standard deviation graphs for the first experiment, “All Links Saturated.” The second set of four (Figures 4.9 to 4.13) are for the experiment “Narrow Links Saturated” and the third set (Figures 4.13 to 4.16) are for the “Wide Links Saturated” experiment. Each graph is similar to the Figures 4.2 to 4.4, with the x-axis as time slots and the y-axis as the observed-to-target ratio, but the CBR counts have been omitted. Each graph contains the original x-y scatter plot for the observed-to-target for the specified protocol, and a bar, in the positive and negative directions, at each data point for the standard deviation of the observed-to-target ratio in the y-axis. The magnitude of the bars is the geometric mean of the standard deviation of the observed-to-target ratio calculated at that point. The standard deviation was calculated for the standard deviation of a sample, whose formula is:

$$S = \sqrt{\frac{\sum (x - \bar{x})^2}{(n - 1)}}$$

And the resulting standard deviation for each node was then put into the geometric mean calculation stated in the previous section.

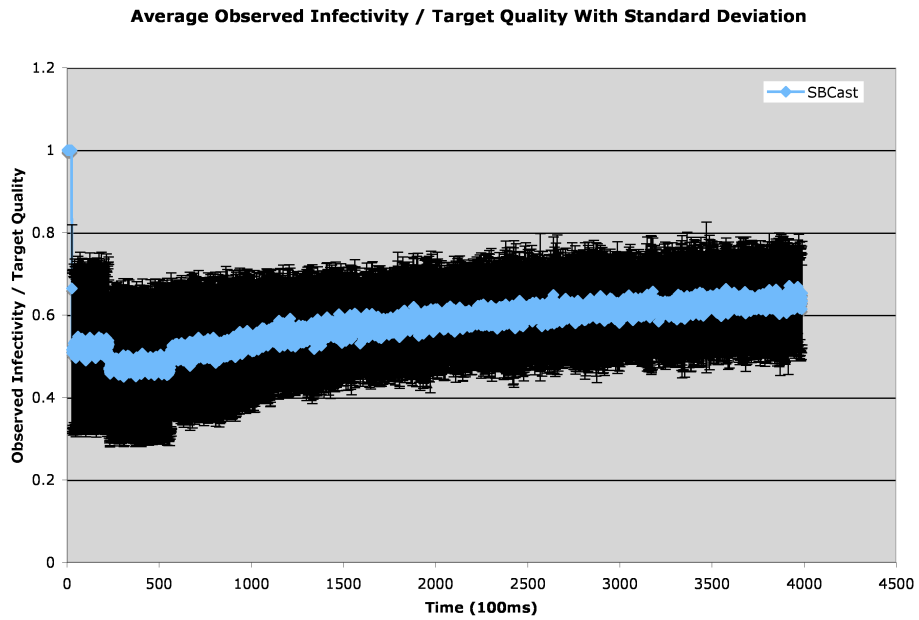


Figure 4.5: SBCast Standard Deviation Graph for All Links Saturated

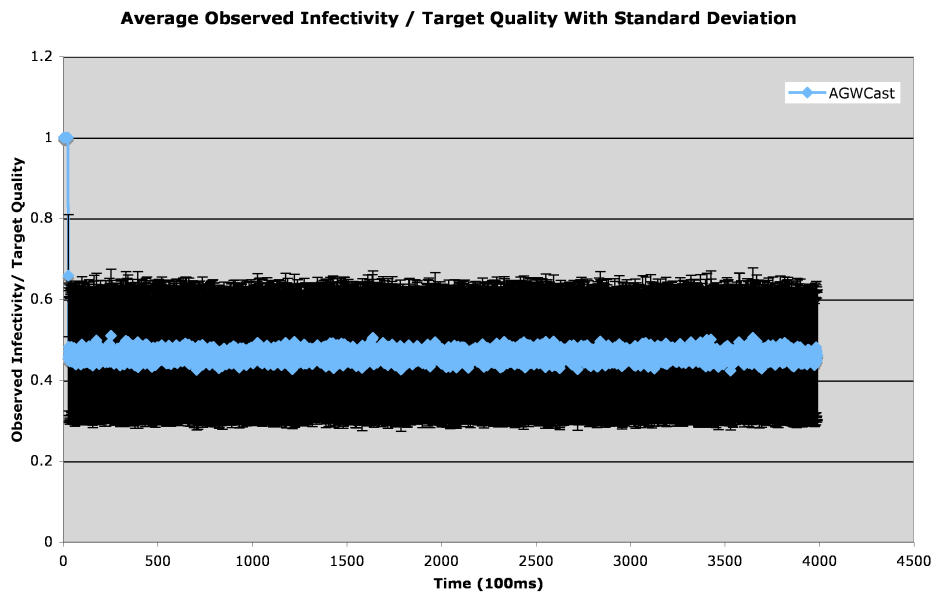


Figure 4.6: AGWCast Standard Deviation Graph for All Links Saturated

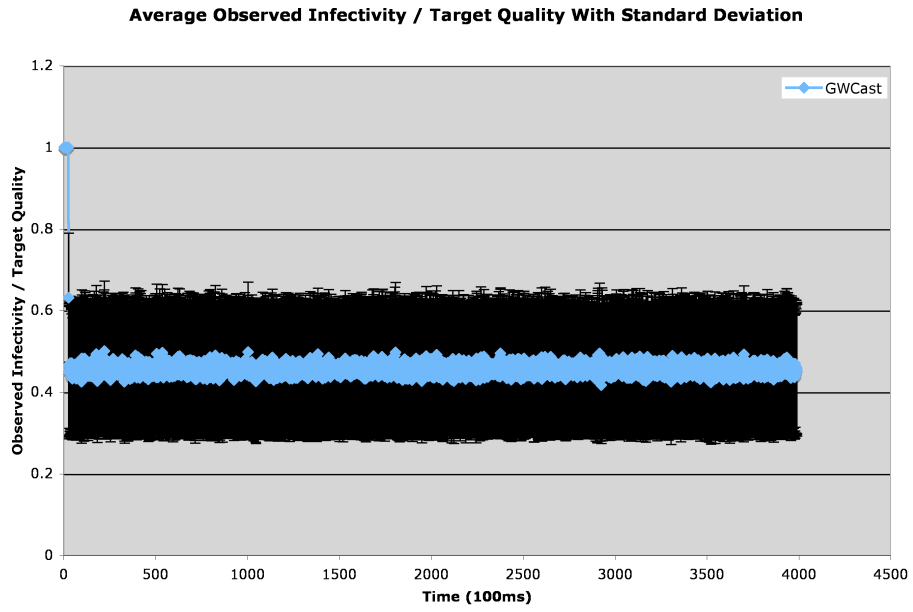


Figure 4.6: GWCast Standard Deviation Graph for All Links Saturated

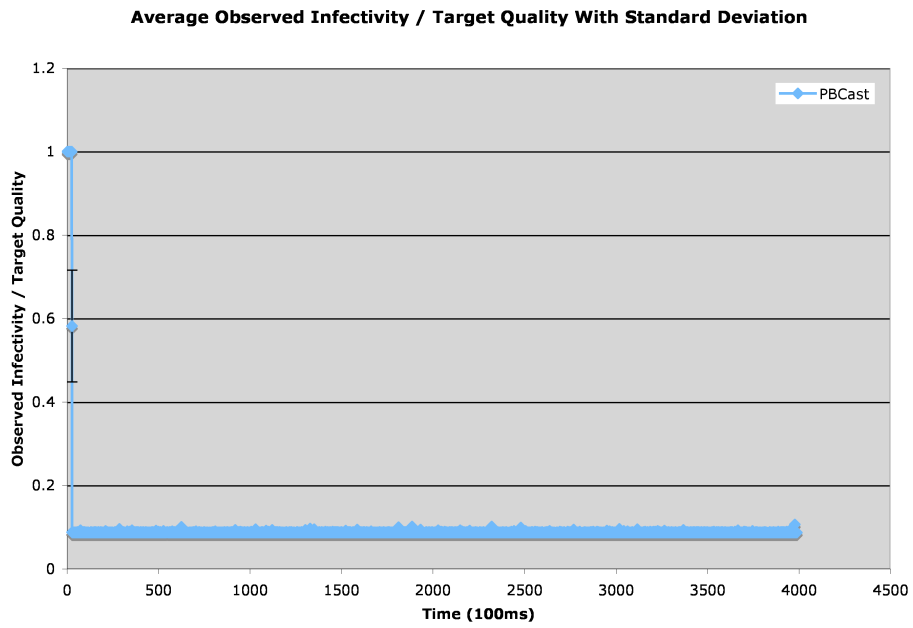


Figure 4.7: PBCast Standard Deviation for All Links Saturated

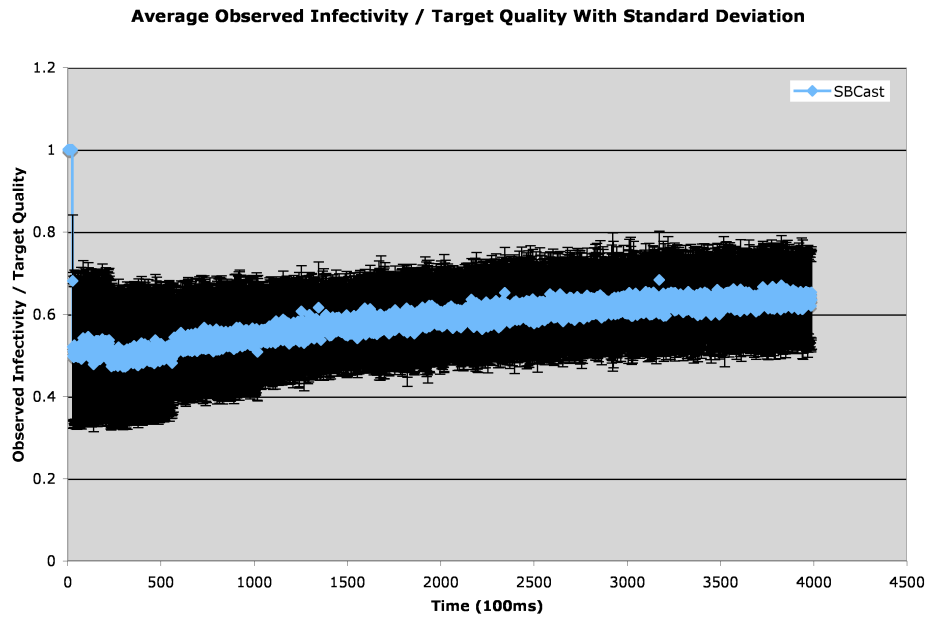


Figure 4.8: SBCast Standard Deviation for Narrow Links Saturated

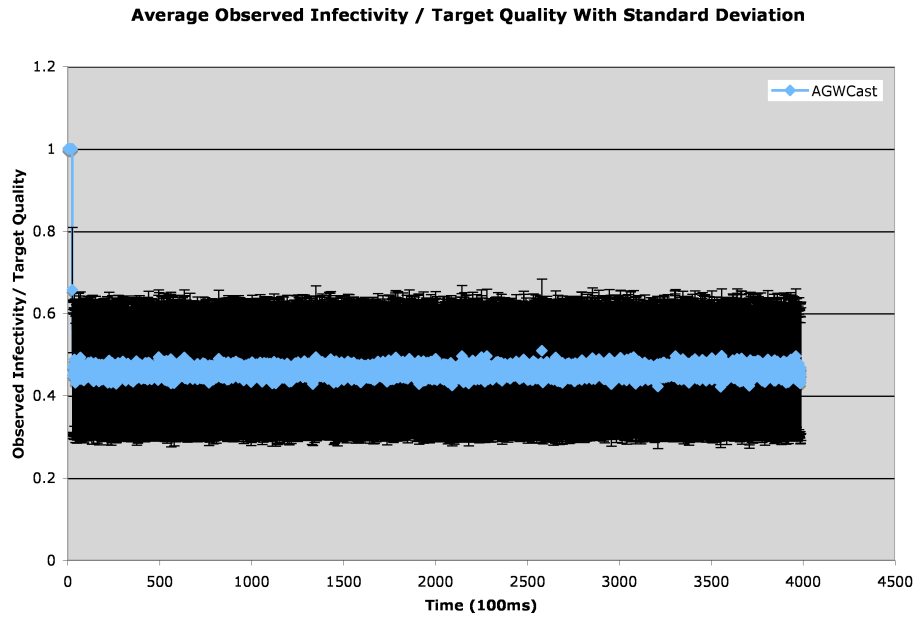


Figure 4.9: AGWCast Standard Deviation for Narrow Links Saturated

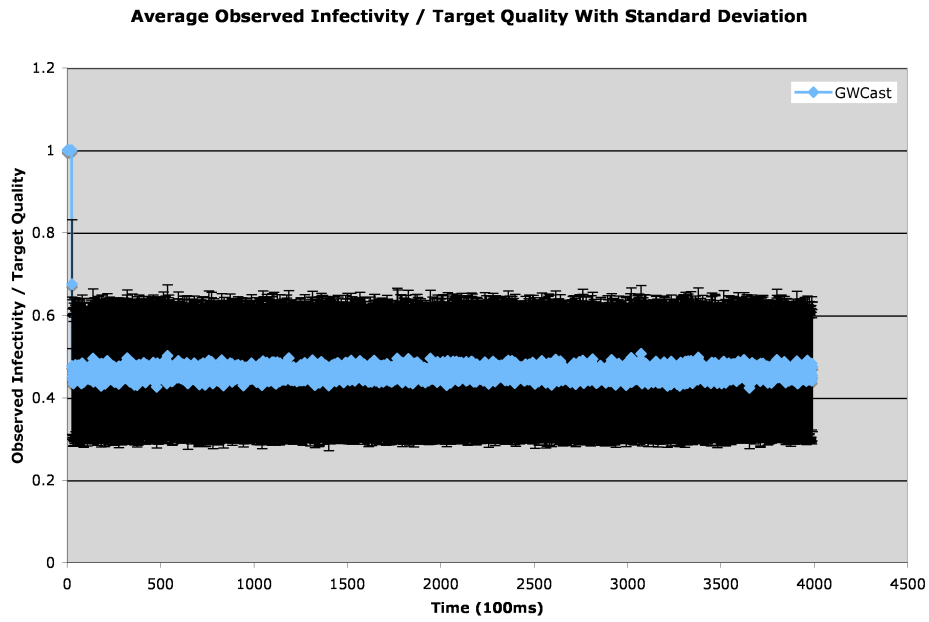


Figure 4.10: GWCast Standard Deviation for Narrow Links Saturated

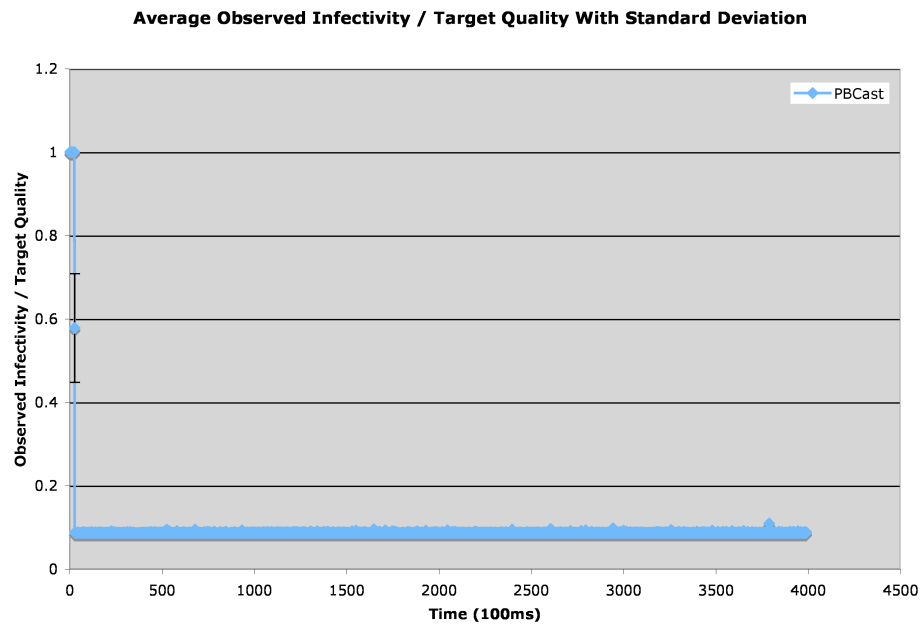


Figure 4.11: PBCast Standard Deviation for Narrow Links Saturated

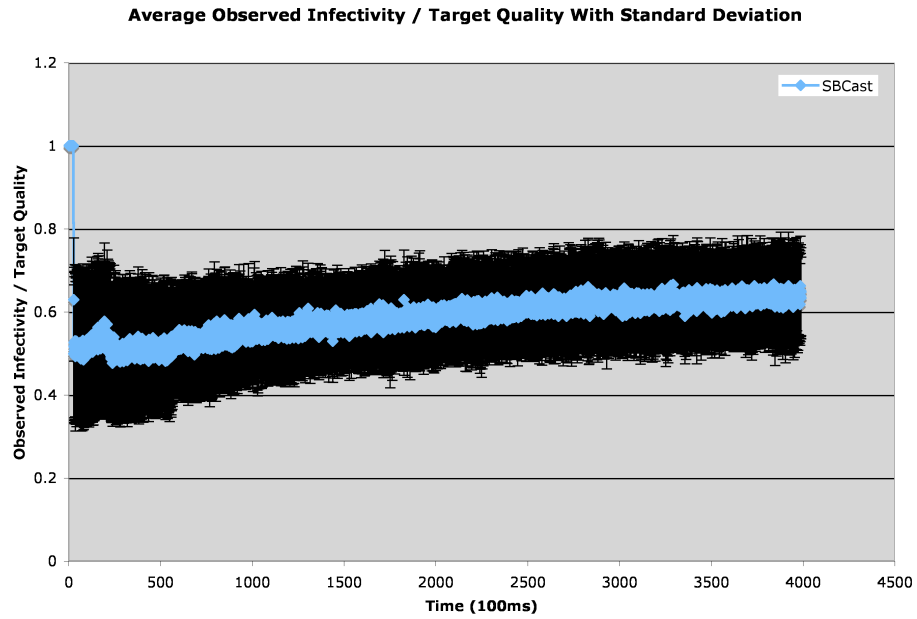


Figure 4.12: SBCast Standard Deviation for Wide Links Saturated

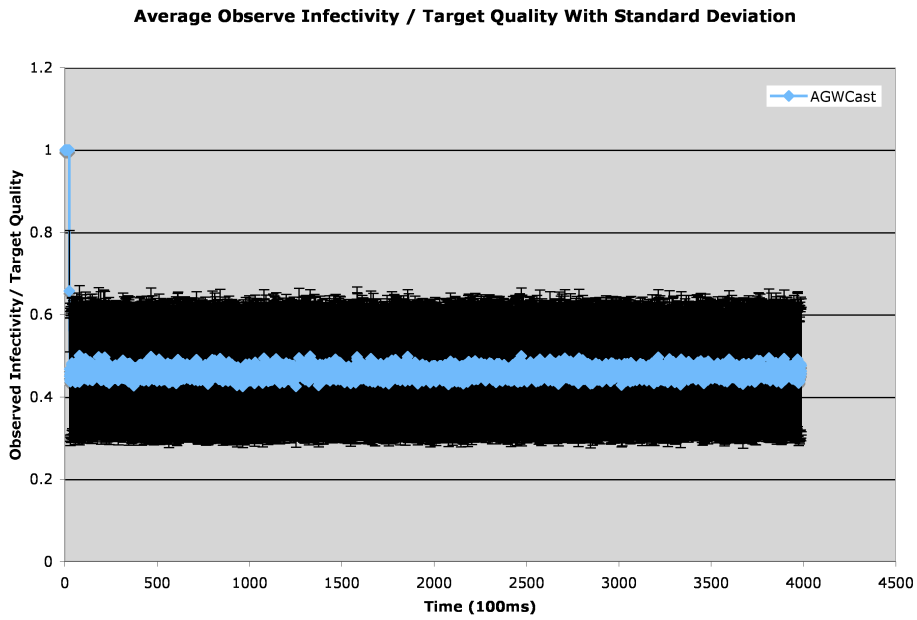


Figure 4.13: AGWCast Standard Deviation for Wide Links Saturated

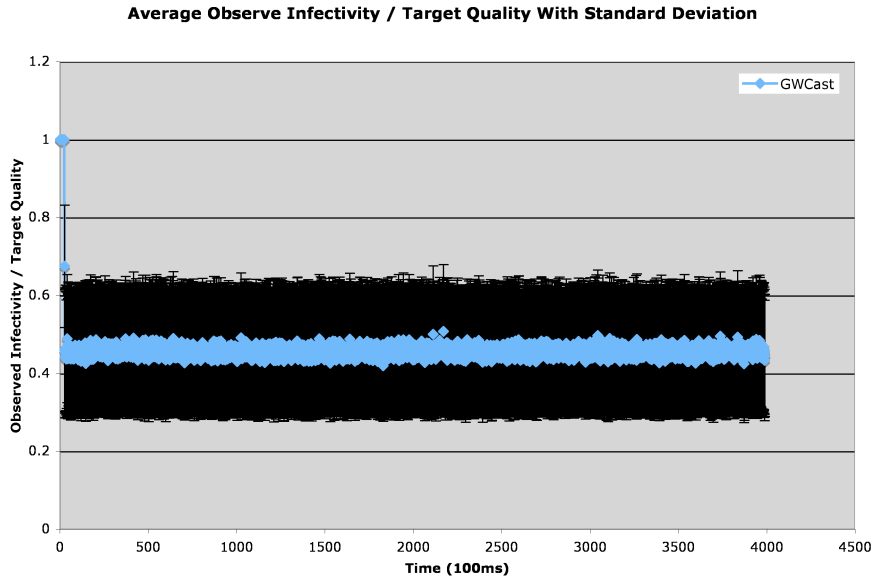


Figure 4.14: GWCast Standard Deviation for Wide Links Saturated

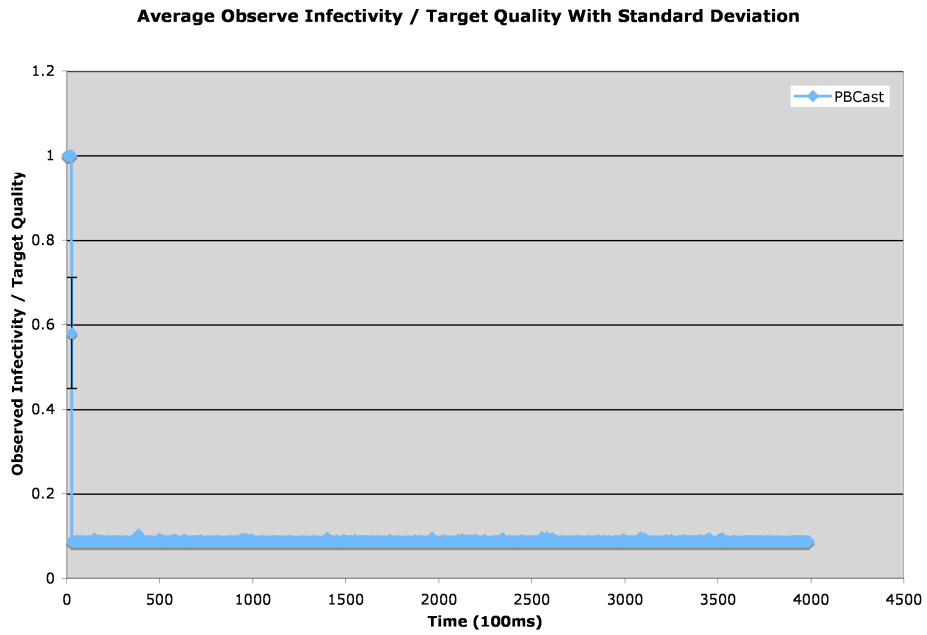


Figure 4.15: PBCast Standard Deviation for Wide Links Saturated

4.3 Investigative Questions Answered

The primary objective was determining the applicability of SBCast, an adaptive gravitational gossip technique, to a pub-sub architecture such as a JBI, and whether or not SBCast could maintain or improve the reliability, scalability and adaptability of a JBI. Towards this objective, the key investigative questions was how well SBCast perform in a JBI, given a battlefield scenario. How would SBCast manage the bandwidth by controlling or restricting updates? These results demonstrate how SBCast would perform, regarding its ability to recapture an observed-to-target ratio of 100%, and dealing with increased bandwidth in this experiment

The primary objective of these experiments was to determine the applicability of SBCast to JBIs, and whether or not SBCast could maintain or improve the reliability, scalability and adaptability of a JBI. Towards this objective, the experiments simulated a battlefield scenario that applied JBI and also modeled background traffic. The results are in. Given this problem domain, how would SBCast manage the bandwidth by controlling or restricting updates? Or, more specifically, from Chapter I, the key investigative questions to answer are:

- Can SBCast be applied to an operational JBI scenario?
- How would SBCast perform as a controller component in JBI middle-ware, specifically in regards to scalability, reliability and adaptability?

This section answers each question individually, based on the graphed results. Each question is given its own subsection. The second question actually contains three questions, so each sub-question is regarded separately.

4.3.1 Can SBCCast be applied to an operational JBI scenario?

Yes, SBCCast is applicable to an operational JBI. In transforming the original *Cast experiments so that they fit a scenario that modeled a battlefield in which a JBI would be employed, the system of gravitational weights and target qualities was shown to work well with the stated requirements of JBIs. From the main graphs (Figures 4.2, 4.3 and 4.4) one can tell from the maintenance of the observed-to-target ratio that any of the *Cast protocols can act as a controller can assure some expected rate of updates. At first, when there is no traffic from either the CBRs or the agents, the observed-to-target ratio is 100%, but when traffic from both the CBRs and the agents start, each protocol drops to around some percentage of the observed-to-target ratio. For PBCast, its x-y plot resided around 20%; for AGWCast and GWCast this was around 45%. SBCCast, however, sets itself apart from the others because its slacking technique recovers some of the observed-to-target ratio, from 45% to 65%

This specific requirement that this question is addressing is the ability of SBCCast to handle differing priorities of different subscribers. In generating the results for Figures 4.2, 4.3 and 4.4, the standard deviations for SBCCast, AGWCast and GWCast varied 10-15% around the geometric mean, whereas PBCast had near 0%, as shown in their respective standard deviation graphs (Figures 4.5-4.16). This variety of readings for SBCCast,

GWCast and AGWcast shows that the observed infectivity varied from node to node, meaning that nodes in those protocols' trials viewed varied levels of infectivity and thus had different levels of bandwidth for different groups it could receive updates for. This implies that there was control over the quality of updates different nodes received from different groups. The opposite case would be that the standard deviation for the observed-to-target ratio at each node would be around 0%, as all agents in that network would send or receive around the same amount of messages. The only trials that have that standard deviation are the PBCast trials, and the PBCast trials are the only trials that do not use interest levels as well. More conclusive evidence would be to graph the observed-to-target ratios at each node, but for the purposes of this paper these geometric mean values for observed-to-target ratio with large standard deviations indicate that SBCast, GWCast and AGWCast are modifying bandwidth availability according to the interest levels set.

Again, what sets SBCast apart from GWCast and AGWCast is that it recovers some of the observed-to-target ratio. But what is interesting with the case of SBCast is even as it adjusts the target qualities, it maintains nearly the same standard deviation range as AGWCast and GWCast as it increases the observed-to-target ratio. Inspecting graph 4.5, 4.9 and 4.13, one can see that SBCast appears to behave similarly to GWCast and AGWCast for the first forty time slots, or four seconds. This corresponds to the first ten adaptive rounds that SBCast must wait before it starts adjusting targets. An adaptive round is when the adaptive mechanisms in AGWCast and GWCast adjust the gravitational weights, which occur every 400 ms, or four time slots [13]. This relates to the call

to SBCast's target adjustment algorithm, which, as remarked by the pseudo-code from Figure 3.6, is placed right before AGWCast's weight adjustment. Once the SBCast target adjustment strategy is able to start, then the observed-to-target values begin to grow back towards 100%. During that growth, the standard deviation varies by roughly the same range as AGWCast and GWCast, 10-15%. This indicates that the ability to prioritize streams is maintained even as target qualities are modified. Therefore, SBCast fits the requirement for applicability to the JBI, and that it successfully manages different priorities of subscribers to specified application streams.

Unfortunately, the graph of SBCast in any of the scenarios did not recover the observed-to-target ratio back or close to 100%. This means that SBCast is not yet at ideal for JBI usage. The objective of this question is whether SBCast can be applied as a controller to JBI; the next question deals with to what degree SBCast is useful at this point in its development, and will also address the issue of why SBCast does not recover to the 100% of the desired observed-to-infect ratio.

4.3.2 How would SBCast perform as a middleware controller in JBIs?

The answer to this investigative question is broken into three sections, corresponding to the evaluation of SBCast in regards to scalability, reliability and adaptability. Although each section evaluates the graphs separately, there is one factor in the results that prevents reliability and adaptability from being accurately assessed, and that factor is the effect of the CBRs.

After evaluating the results data and generating the scatter plots graphs that would become Figures 4.2 to 4.16, it was discovered that the CBRs appeared to have no effect. The example expected behavior for a gossip protocol is portrayed in the hypothetical graph in Figure 4.17: as more CBRs were turned on, each of the gossip protocols would

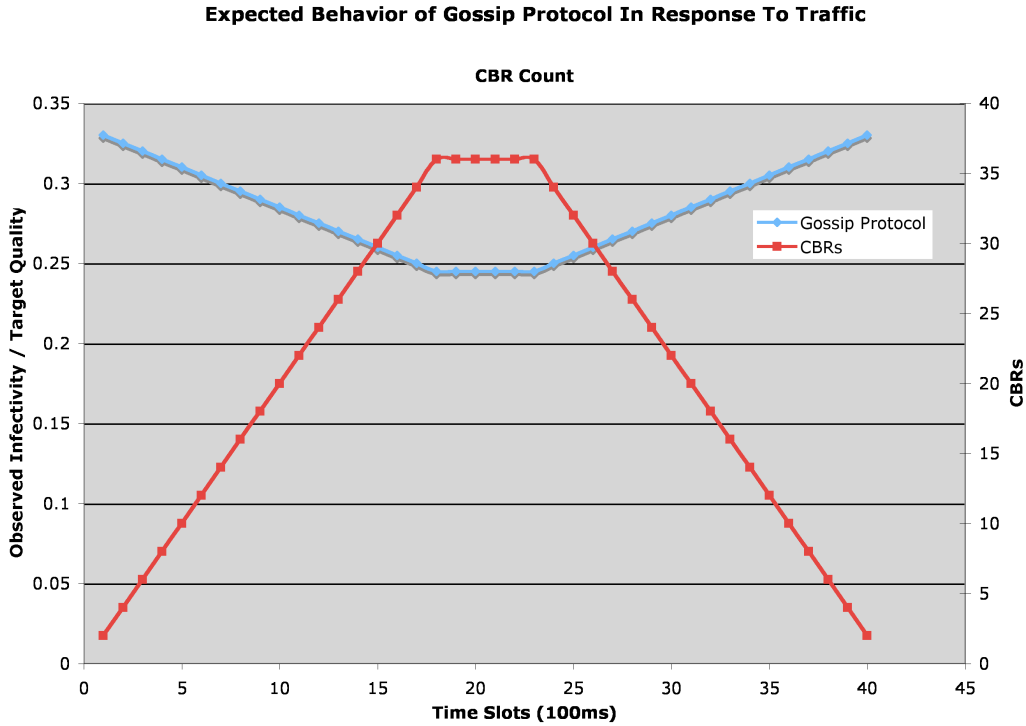


Figure 4.17: An Example of Expected Behavior In Response To Traffic

“bow” or “morph” at first, but then would adjust back. However, in examining all three collected graphs (Figures 4.2-4.4) it was discovered that the observed-to-target ratios were all the same for all protocols, and did not vary in response to more or less CBRs being turned on by the scenario script as the time slots increased. After re-examining the configuration scripts for the experiments, calculations were made considering the scenario link’s capacity values and the configuration of the CBRs. This estimation of the

size of the background verified that the CBRs would have little effect when compared to the *Cast traffic. A fix would be to re-run the tests using CBRs that would output more intense levels of traffic. Due to time constraints re-testing will be addressed in future work.

One uncertainty that this development brings is the comparison between the GWCast and AGWCast graphs. AGWCast differs from GWCast in the employment of the adaptive mechanism, geared towards adjusting the gravitational weights in response to background traffic. Without the CBRs producing significant background traffic, there would be no visible difference between the observed-to-target ratio graphs for either GWCast and AGWCast, because the adaptive mechanism might not be reacting properly in AGWCast. Most likely, with the adaptive mechanism, AGWCast (and ultimately SBCast, which inherits the adaptive mechanism as well) would show a graph that is more aggressively reacting to the cumulative traffic than the one that appears in the Figure 4.17.

As a result, reliability and adaptability cannot be accurately assessed at this time, although the answer to the question could be reasonably investigated, as well as scalability. Both can be interpreted without consideration for background traffic. Each section, then, will interpret the results separately, and also in regards to the absence of the CBRs.

4.3.3 Scalability

The observed-to-target ratio can be considered a direct measure of scalability. Although traditional probabilistic techniques have been proven to be scalable [4], they have

a reputation for being “noisy”. This noisiness means that the more gossip agents we place in a graph, the “noisier” it will be and so there is a problem with scale. This problem with scale is not directly tied to the number of nodes or agents, but rather a problem of the number of nodes or agents in the network medium chosen; one such example is using gossip in a WAN environment as tested by Rodriguez with NEEM [21]. With the *Cast family, to some degree this can be demonstrated with Figures 4.2-4.4. The observed-to-target ratio hovers at a certain level and, with some variance, maintains roughly that level. PBCast remains at around 20%, GWCast and AGWCast maintain roughly 45%, while SBCast rises from 40% to 65%. Ideally, this should be 100%. However, if none of these protocols was scalable, one would observe, over time goes by, that the observed-to-target ratio would deteriorate as agents would send more and more updates. These updates would accumulate over time, and the results would be a lower observed-to-target ratio as a network becomes clogged with messages and the observed infectivity goes down. However, as seen in Figures 4.2-4.4, this is not the case and the plots of PBCast, GWCast and AGWCast maintain roughly the aforementioned observed-to-target ratios. Although a more direct test would be to successively increase the number of agents, this actually demonstrates an attribute of *Cast agents: that through a timeout setting for a message, an update is considered too old to accept and would be dropped [3]. In the gravitational gossip protocols, this becomes a factor in the susceptibility of an agent with respect to a specific message, and thus a factor in the weight calculation.

That distinction results in an observation: that gravitational *Cast are more scalable than PBCast, a traditional gossip protocol. The evidence of this is in the observed-to-

target ratio for the gravitational protocols versus the non-gravitational protocols: the observed-to-target ratios are much higher. The higher the ratio, the higher percentage of the messages agents are able to send out. This shows that, even with the same number of agents, the agents' updates are not interfering as much with each other in the three gravitational trials as the non-gravitational. Because SBCast shares the gravitational characteristics of GWCast and AGWCast, SBCast can add scalability qualities to JBIs.

4.3.4 Reliability

A perfect observed-to-target ratio of 100% means that all messages that are sent are also received. However, for each of the *Cast protocols none have achieved that perfect 100%, or close to it. In the case of PBCast, only 20% of messages were received in all experiments; in AGWCast and GWCast, only 45% of messages were received. For SBCast, the protocol recovered from 45% of messages received to 65% of messages received. An obvious interpretation is this: because none of the *Cast protocols, including SBCast, can keep a perfect or near-perfect observed-to-target ratio, then SBCast cannot provide reliability. On the other hand, does a perfectly 100% reliable protocol exist? If one did exist, then protocol research would never be required again. In light of the observed-to-target ratios interpreted from Figures 4.2-4.4, it could be said that in general PBCast is not as reliable as AGWCast or GWCast, and GWCast or AGWCast is not as reliable as SBCast. Likewise, as SBCast recovers the observed-to-target ratio, then SBCast is not at first reliable, but may recover to nearly 100% reliability over time.

But even this explanation does not satisfy the absence of CBRs, or answer the question as to why SBCast does not fully recover the observed-to-target ratio to achieve reliability. One requirement for reliability for JBIs would be that the protocol reacts to background traffic and recovers completely. Since the CBRs did not have a significant impact, then the graph for SBCast recovers gradually but not completely, without reacting to any background traffic. This indicates that SBCast does not fully recover from traffic it creates, and would most likely not recover from background traffic. This also indicates that even without additional interference, SBCast does not reach the ideal 100% needed to improve reliability in JBIs.

By contrast, the other *Cast protocols do not recover any of the observed-to-target ratio. Because the other *Cast protocols do not have a target adjustment strategy, the observed-to-target ratio is static throughout the entire scenario and would never improve. Therefore, the fact that SBCast partially recovers is an improvement upon its relatives. This minor improvement leads to an explanation as to why the observed-to-target ratio of SBCast did not improve fast enough: it is not aggressive enough in adjusting the target.

Referring to the pseudo-code for the historical target strategy in Figure 3.7, there are several possible factors that contribute to SBCast's conservative behavior. The first is the length of the history buffer, which is ten adaptive rounds long. If a time slot is 100 ms, an adaptive round takes 4 slots, and the historical strategy waits ten adaptive rounds, then the historical target strategy is initially waiting 40 seconds before it can begin to react to network activity. Additionally, because of the historical strategy's length, the network conditions could actually be improving but not meet the algorithms thresholds. The

thresholds of the historical strategy are that: 1) it looks for eight of ten rounds of higher observed infectivity readings and 2) it looks for at least five rounds of decreasing susceptibility. By the time the algorithm is able to react to what it perceives to be improving conditions, the conditions could become worse, and conversely by the time the strategy reacts to worsening conditions, conditions might actually improve.

Another possible conservative factor is the improvements made to the quality target. In one rare case, the target quality is restored to its original value only if the susceptibility is close to 0, a highly unlikely event. A susceptibility of 0 means that all agents have received all their updates, a nearly impossible occurrence. This is the most aggressive restoration of the target quality, where the other increments added to the target quality are mere fractions of the difference between the original target and the current target. With the less aggressive improvements to restore the target quality, the target quality improves too slowly, and results in affecting the gravitational weights gradually, which in turn might prevent network conditions from improving.

The last possible factor to discuss is the value by which the target quality is reduced. The historical target locates the best observed infectivity in the history buffer and sets it as the new target quality. If this max value is actually an outlier and not close to the average across the recorded observed infectivities, this could result in the target quality remaining still too high for the agent to keep up with. As a result, SBCast will continue to fail in meeting the target. This could continue and SBCast would re-adjust the target quality with the best observed infectivity. The common case would be that the value of the best infectivity would also drop severely, but it is still possible that another outlier with

the same or similar value. This would result in SBCast lowering the target qualities slightly and perpetually missing them.

At this point, the results do not demonstrate whether or not SBCast would make JBI's more reliable. However, what can be inferred is that the target adjustment strategy of SBCast must be improved to deal with background traffic and traffic from its own agents, so as to move closer to the ideal 100% object-to-target ratio that would benefit JBI's.

4.3.5 Adaptability

Adaptability is related to reliability, and deals with the protocol's ability to maintain an expected level of updates despite bandwidth changes. As noted previously, the CBRs did not appear to make a significant impact on the network. As a result, neither can adaptability be verified with other traffic. However, from the observed-to-target ratio graphs (Figures 4.2-4.4), it is evident that the *Cast protocol traffic affects itself, which is why the observed-to-target ratios are never fully 100%. Some of the possible behavior with additional traffic generators could be inferred, and with SBCast's improvement of the observed-to-target ratio, SBCast is adapting.

If any of the *Cast protocols truly were adapting, what would the graph look like? As mentioned previously, the expected behavior (Figure 4.17) characterizes this possible form. To reiterate: the graph should at first be affected when CBR activity builds, but then the graph should recover to either its original observed-to-target ratio or towards 100%. If SBCast were adapting, its graph would dip momentarily in response to the buildup of traffic, and then re-adjust back to its original value. There is no apparent CBR

activity in the graphs SBCast in Figures 4.2-4.4, but from any of those three graphs SBCast starts out at 100%, then drops immediately when its own agents start to send updates. Despite the interference caused by its own updates, the target strategy activates and begins to raise the observed-to-target ratio closer to 100%. This behavior correlates to the expected behavior for a gossip protocol in adapting to background traffic: that the traffic would occur, but the protocol should adjust in response. However, actually having significant test traffic would verify this. Therefore, even with this self-inflicted traffic, SBCast still recovers closer towards 100%, indicating that SBCast is indeed reacting to traffic and adapting.

4.4 Summary

This chapter displayed the results of running the various experiments outlined in chapter three and interpreted their direct meaning to the investigative questions posed in Chapter I. Although the complete recovery of the observed-to-target ratio was not achieved within the duration of the scenario, the technique of “slacking”, or dialing up or down the target qualities, does show some potential for lowering the rate of updates for lesser priority data streams and preserving bandwidth available overall.

V. Conclusions and Recommendations

5.1 Chapter Overview

This chapter covers the conclusions that arose from the experiments and the analysis of those experiments, recommendations for actions and recommendations for future research. The conclusions discusses the direct results and what these entail for employing SBCast or *Cast to JBI implementations. Recommendations for action discusses some of the issues during experimentation and development and how these could be remedied. In essences, Recommendations for Action explains how these experiments could be better re-accomplished in the near term. Recommendations for Future Research discusses possible avenues for extending this work, to include possibly implementing a prototype version of *Cast for inclusion in the current JBI suite.

5.2 Conclusions of Research

Regarding the resulting graphs, certain overall conclusions can be drawn regarding SBCast and any of the *Cast's inclusion in the JBI. These include the expected quality of updates, dealing with the expected quality of updates, and its implementation.

A strictly probabilistic approach agrees with prior research. The results demonstrate Rodriguez conclusion, that gossip protocols like PBCast tend to be “noisy” and are inefficient users of bandwidth [21]. The graph for PBCast in all three experiments (Figures 4.2, 4.3 and 4.4) demonstrates this fact. In all three cases, bandwidth utilization inferred by the observed-to-target ratio was below 20%. That is, PBCast agents were receiving less than twenty percent of their expected messages. Since this is both the base-

line for the *Cast family, there is room for improvement, the improvements made to PBCast's successors, GWCast [3] and AGWCast [13] from Hopkinson's research certainly enhance the observed-to-target ratio.

Expected quality of updates. It is apparent from these graphs that transmissions from *Cast nodes overall maintain the same consistency for the entire period. Although some traffic was introduced, the overall geometric means remained at the same level for PBCast, GWCast and AGWCast. The effects of the traffic generators were actually more noticeable (if at all) towards the second half of the scenario, when all CBRs were active and began to drop off. The standard deviations across levels varied more, although not noticeably. However, overall the observed-to-target ratios are less than 50%. Signifying that half the expected messages are not received, this fact still highlights the need for a technique for raising/lowering target quality to raise the observed-to-target ratio and thus use available bandwidth more easily.

The SBCast technique of backing off target qualities and then recovering shows promise. Although more work could be accomplished towards both providing more challenging traffic or extending the length of the experiments, the SBCast graph shows the protocol beginning to recover towards the perfect 100% observed-to-target ratio. If given more time, then perhaps SBCast could have restored to over 90%, as the graph shows a gradual recovery. As discussed in the previous chapter, the target strategy used by SBCast in the experiments is not aggressive enough in adjusting the target qualities.

Applicability to Joint Battlespace Infosphere. From the perspective of these experiments, a scenario that involved or resembled units involved in a military operation

successfully used SBCast agents to send and receive updates. The scenario yielded results that showed that SBCast agents (or any *Cast agent) can maintain some expected level of quality of transmissions. With SBCast, there is a mechanism in place that can “back off” under conditions where bandwidth may not be available. This ability to give “slack” to the target qualities could allow a probabilistic protocol to function in an intermittent environment, such as wireless links. As a result, *Cast protocols can be applied as a middle-ware layer for JBIs.

5.3 Recommendations for Action

Like most research, there areas that were either not explored or areas that were not explored fully. These directions might lead to better insight into using SBCast or deriving another *Cast variant in making JBIs more scalable and reliable. These include upgrading the realism of the simulated traffic generated within the scenarios, improving the target adjustment strategy and re-engineering the *Cast family for better maintainability and performance.

Upgrading the realism of the traffic in the scenario. Although great pains were taken in bending and grafting the scenarios to reflect more closely a possibly real-world JBI implementation, there was one specific deficiency. The traffic was modeled as a series of constant bitrate generators to abstract the randomness of traffic, but the expected levels of traffic were not achieved. Currently, the CBRs introduced some traffic, but was not enough to make a dramatic effect. Additionally, rather than following a single, rolling wave approach that yields a steady rise and fall of the traffic, a single rise during the first

third of the traffic and then a constant and consistent level of traffic for longer periods might also induce more positive reactions from SBCast. Basically, making it larger and constant, instead of the rolling approach taken by these experiments, could enhance the realism of the traffic.

Multiple senders. This experiment and previous *Cast experiments all utilize one sender, which is unrealistic in real world operational use. The use of the JBI would consist of different units being able to push updates via pub-sub to all subscribers. Senders introducing updates produces affects on the traffic that should be accounted for. To better model this, multiple senders should exist in a scenario to make the experiment more realistic to possible use. The decision to continue with one sender was based on performance issues; per some of the limitations noted in the point for improving performance, having more than one sender multiplied the runtime of an experiment by the number of senders. Therefore, this recommendation is to increase performance to support the use of multiple senders.

Improving the target adjustment strategy. Currently, SBCast's adjustment strategy is not very aggressive in recovering. Specifically, it only adjusts the target quality back gradually. As mentioned in the conclusions section, given more time the protocol could have recovered the observed-to-target ratio significantly. A good future experiment would be to modify the target adjustment strategy to more or less aggressiveness, and then measure the effects. A comparison of how quickly the protocol recovers would demonstrate the strengths of an aggressive or gradual approach for SBCast to adjust the target qualities. In fact, the historical target adjustment strategy could be easily replaced with a

more aggressive strategy, due to Liskov's substitution principle [17] and the utilization of the strategy design pattern[12] in implementing SBCast. One feasible path of research would be to create a whole series of target adjustment strategies, and compare their performance to each other.

*Re-engineering *Cast protocols for maintainability and speed performance.* This covers quick modifications that can be accomplished in relatively short time. An example of a software engineering enhancement would be similar to the use of a strategy design pattern as described in the implementation of SBCast's target quality adjustment routine, except applied to another part of the *Cast base code. One likely candidate would be the adaptive mechanism. This would allow opportunities to explore enhancements to the gravitational gossip mechanism.

An example of a performance enhancement would be the use of calls to MatLab's libraries for `lsqcurvefit` routines. Currently, the implementation of the adaptive mechanism relies on having an instance of Matlab running. Having a separate instance of Matlab in order to execute the `lsqcurvefit` routines slows down the overall runtime of the simulations, because processing must be diverted to Matlab. The method of process communication with the Matlab instance is time intensive. By implementing a direct library call, the execution of the *Cast protocol would bypass the overhead produced by running and communicating with a separate instance of Matlab. Tests could be run within hours, instead of days.

By implementing the smaller software engineering enhancements, the *Cast source code would both have a smaller learning curve, be easier to translate for use in

JBIs and simpler to maintain or modify. By implementing the performance enhancement, experiments utilizing *Cast code could be executed more quickly and yield more immediate results, so that the algorithms could be evaluated faster.

5.4 Recommendations for Future Research

Recommendations for future work follow in similar vein for recommendations for action. These focus on re-engineering *Cast code for better maintainability and improving performance. However, the difference is that the future research could yield *Cast variants that might better perform in JBIs and also would be more straightforward and applicable to real implementation.

*Modularize *Cast.* One specific software engineering issue with the *Cast code is that it does not cleanly delineate methods or functionalities with each other. If the various components and modules were more apparent, modifying them would be more straightforward. The use of design patterns, such as the strategy pattern used to implement the target adjustment strategy for SB*Cast, would help in delineating the different methods from each other, or perhaps moving the methods to the best locations within classes for higher cohesion and lower coupling. Higher cohesion and lower coupling are two of nine *General Responsibility Assignment Software Patterns*, defined by Larman, a software engineering researcher [16]. Higher cohesion means that methods of modules have tight, focused functionality and lower coupling means that there is less dependence on modules having to know internals of each other to accomplish their functions. Once modularized, the components can be adjusted and tested for maximum effect (as measured in the ex-

periments in the future) and also can be translated directly into a real world implementation of a JBI.

*Distribution of *Cast's data structure.* Currently, the coordination among *Cast agents takes place within a large, multidimensional C/C++ arrays. This design decision speeds up the ns-2 simulation and abstracts away the coordination overhead a distributed systems protocol might produce, but also creates an inflexible agent that is difficult to execute or describe for ns-2. One glaring limitation of the use of this data structure is that varying numbers of senders and groups are not implementable; since the multidimensional array is required to have an element, some groups or senders cannot be omitted, limiting certain topology configurations. By modifying the underlying data structure, greater flexibility in creating scenarios could be achieved, resulting in scenarios of greater scale, variety and realism.

*Real implementation of SB*Cast.* To fully answer the investigative question as to how well SB*Cast (or any *Cast) agent would work in practice would be to construct a prototype. This prototype could also be matched to prototypes of IESTs to work out how the components would inter-operate. This could be done by using (hopefully modularized) *Cast source code and translating from the ns-2 model into an actual language.

5.5 Summary

This paper explored the use of probabilistic protocols in JBI. In particular, it explored the need for scalability/reliability, and utilized a novel technique for adjusting the

target qualities (the expected rate of updates). Adding this technique to the *Cast family of probabilistic protocols, this protocol, dubbed SBCast, was tested over three experiments with varying network conditions. The results showed that probabilistic protocols could apply to providing middleware services for monitoring the Joint Battlespace Infosphere.

Appendix A: Development of Ruby Cast: The *Cast Script Generator

A.1 Appendix Overview

One significant problem that was not directly tied to the investigative questions but was instrumental in completing this thesis was bringing the original *Cast experiments from the academic environment into a military operational scenario. Initially, the estimates for the time needed to develop a workable scenario were optimistically brief. Hopkinson, Birman and others had implemented the original *Cast and iteratively evolved it over time [13], and the expectation was that the scenarios involved in Hopkinson's experiments would easily translate into a model of JBIs. However, the original *Cast experiments were not originally designed for a military scenarios, and proved much more difficult in adapting for use as a model for JBI. This appendix section discusses the original experiment and requirements to model the JBI, and the source code modification and development needed to meet these requirements.

A.2 From Academic Exercise to Operational Environment

The original *Cast scenarios supported an unconventional topology that highlighted the best performance features of the *Cast protocols. Although these topologies were simple and relatively straightforward, they tended towards unusual forms that did not resemble actual networks used in industry or defense. Their simplicity was geared towards testing the protocol's abilities, and did not necessarily reflect an actual "real world" application. This approach is pragmatic and efficient, and focuses on the goals of

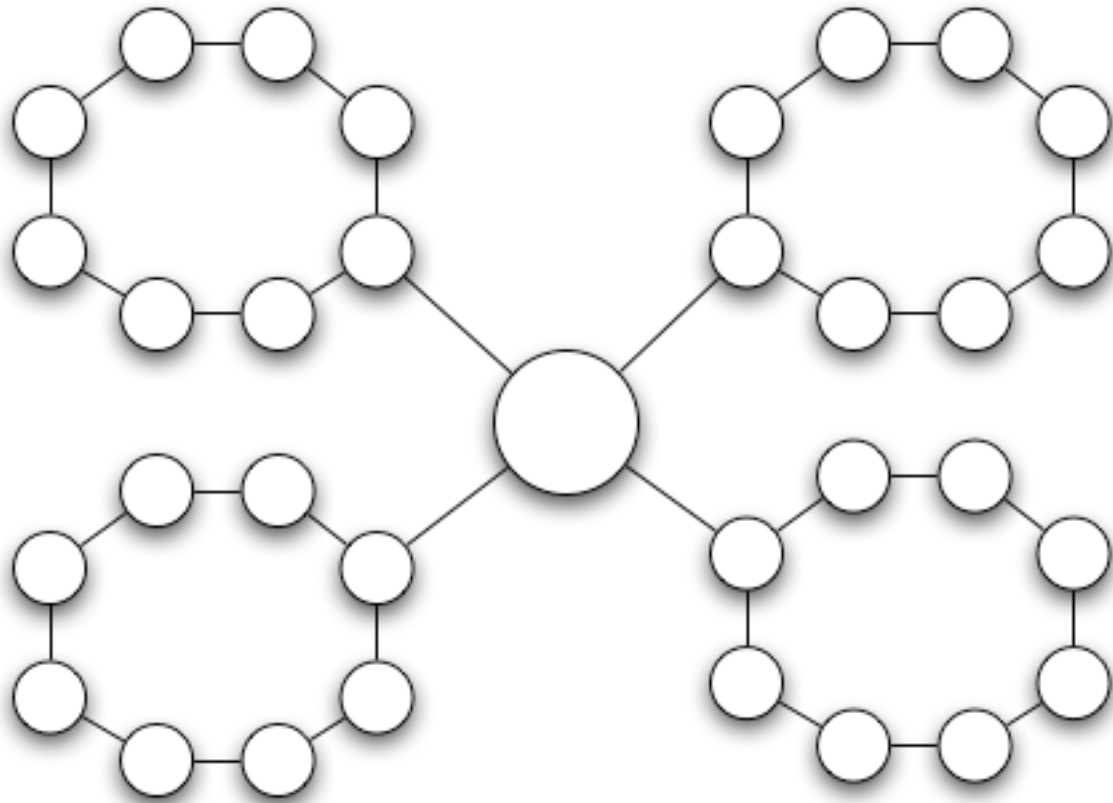


Figure A.1: Typical *Cast Topology

*Cast research in the academic environment. A typical scenario topology appears in Figure A.1.

The topology consists of one or more rings branched together off of a central node in a star configuration. The central node served as the sender, and each ring served as a group level, as described in Chapter III. The links were simple duplex links, and modeled cable lines in a fixed facility. This differs from what was required to model a JBI environment, described in the scenario section in Chapter III. Contrasted with the topology in

Figure 3.1 (or Figure B.1 for more complete representation), the required topology is a much more complicated hybrid of star and mesh topologies with wireless links.

The groups in the original *Cast scenarios were non-specific, as they did not need to represent any real data stream. Consequently, all agents subscribed to all groups with the same interest. Contrasted with the table of interests used in the experiments in this paper (Table 3.2), the much simpler table of interests appears below (Figure A.1):

Table A.1: Typical Interest In Original *Cast Experiments

Nodes	Group Levels	Group 1	Group 2	Group 3	Group 4
1	1	80	75	50	25
2	1	80	75	50	25
3	1	80	75	50	25
4	2	80	75	50	25
5	2	80	75	50	25
6	2	80	75	50	25
...					

The agents that appeared in the original *Cast settings were uniform and did not vary between group levels. The interest ratings for nodes needed to expanded to support the concept of multiple units with varying levels of interests in different application streams for different group levels. To this end, different node types would have to be created with separate, customized interest ratings that would characterize a unit in the battlefield. Achieving these customized interest ratings was necessary to simulate JBI's

scheme for prioritization of pub-sub streams. Again, this need for a complex interest structure is delineated in Table 3.2.

From a conceptual viewpoint, the translation from an academic environment to a military scenario appears straightforward. The conceptual shifts involve directly translating nodes in the original scenarios to troop units in the JBI scenarios, grouping like troop units into group levels, and expanding the interest tables to support a wider variety of units and prioritization for application streams. From a development standpoint, there were a number of software engineering challenges to overcome.

A.3 Software Development

The primary tasks involved with molding the original scenarios into the one required for JBIs were:

- Reconfigure the topology from the academic environment to the military environment, with associate unit types and group levels
- Support the implementation of multiple node types with customized interest levels

To achieve these goals, the current software architecture for executing experiments was evaluated, and then actions were taken to modify or replace components.

A.4 Original *Cast Scenario Code

The process of executing experiments was reviewed, and flows similarly to the process outlined in the Process section in Chapter III. An illustration of this process appears in Figure A.2.

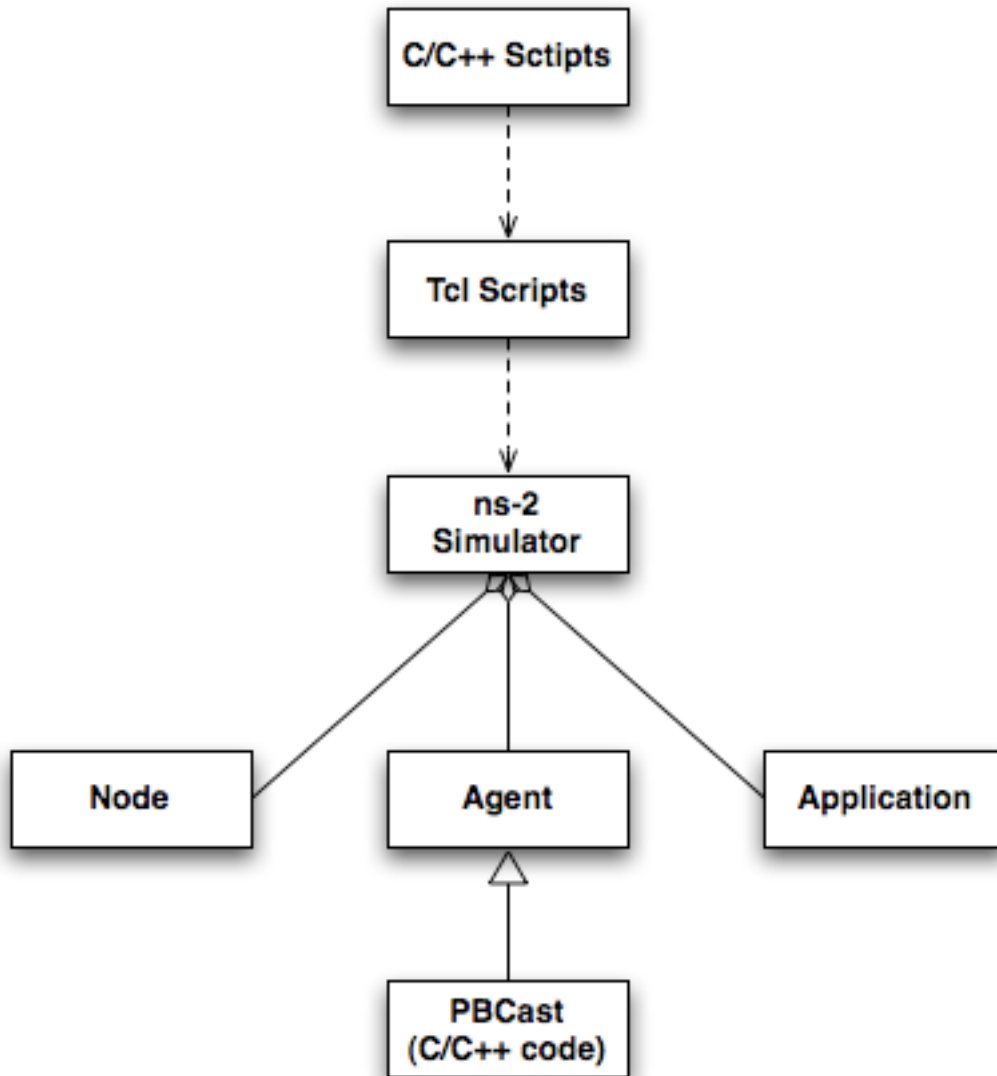


Figure A.2: Original *Cast Scenario Procedure

The boxes marked “C/C++ Scripts” and “PBCast C/C++ code” are the primary differences between the new process used in Chapter III and the original process used. It was determined that these were the two components that were to be modified for the experiments. The latter component, the PBCast code, was modified into SBCast, and those changes are documented in the SBCast section Chapter III. The former component, the C/C++ scripts, is the subject of this section. The C/C++ Script was responsible for creation

of the topology, group levels and events, so it was targeted for modification in order to accomplish the two tasks required to translate the scenarios into JBI scenarios.

The C/C++ Script was the *script generator*, and it accomplished its tasks of generating ns-2 topologies, group levels and events by outputting expressions of the topology in *Tcl*, a programming language that the ns-2 simulator understands. Tcl stands for “tool command language”, and is a popular language in use by Unix applications. Following the explanation of the experimentation process outlined in Chapter III, the C/C++ Script contained information about the configuration of the scenario, and wrote Tcl scripts for each experiment run based on this information. To refer to Tcl used specifically for ns-2, this paper uses the term ns-2/Tcl. It is possible to write ns-2/Tcl code by hand, and execute a scenario directly by manipulating ns-2 manually. This approach becomes unfeasible when scenarios become more complex and multiple trials are needed. This C/C++ Script is helpful by allowing the execution of large volumes of trials with complex scenarios.

The initial “easy fit” would be to simply modify the existing C/C++ script to describe the JBI scenario. For purposes of utilizing the code, an initial survey was done to interpret the code and figure out what precisely it does, and the resulting UML diagram appears in Figure A.3, below.

This survey revealed that the code, although written in an object-oriented language, was highly monolithic and procedural. More precisely, the program consisted of three classes, one of which had multiple responsibilities and no clear order or schema of their functions. The class in particular, the GenScript class had *low cohesion*, meaning

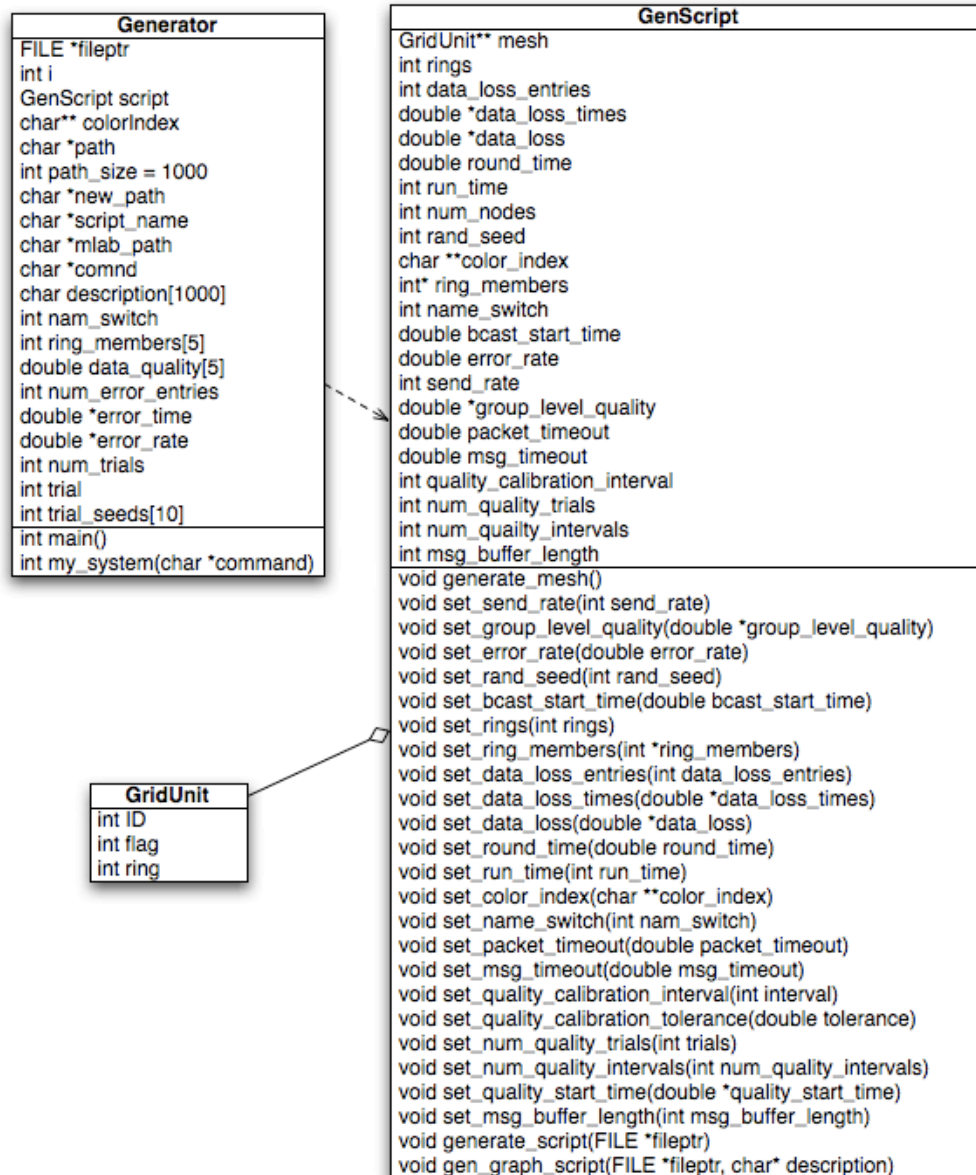


Figure A.3: UML Diagram for C/C++ Script Generator

that it implemented too many methods that were possibly not directly related to the class’ original purpose [16]. The evidence of this low cohesion is in GenScript’s long list of methods or attributes. This resulted in the GenScript class becoming so large as to encapsulate the majority of functions to define the topology.

This was a daunting obstacle in the face of generating the JBI topology and defining group level interests, as the design obfuscated the definition and creation of the agents, links and interest settings. The form of the code made the script non-modular, and distinct components could not be added, removed or modified discretely. This presented challenges to future modification. If the requirements for the scenario were to ever change (which it did frequently during the course of this thesis), modifications would be tedious and take days to accomplish.

After examining the C/C++ Script generator, the decision was made to generate a new script generator.

A.5 Ruby Cast Scenario Script Generator

This section steps through the process by which the new script generator was created. It outlines the thoughts and considerations that went into designing the script generator to create a more flexible network model to test SBCast within JBIs.

It was determined that the best way to model the agents in the network would be to make them first class objects in an object-oriented design (OOD) schema. Interpreting the ns-2 documentation and example ns-2/Tcl scenario scripts, it was observed that the overall schema for ns-2 was actually object oriented, and utilized inheritance to define objects from classes and super-classes[8]. Declarations in ns-2/Tcl view nodes, links and other network infrastructure in this same way, so this object-oriented approach is a good fit. The original C/C++ script, by comparison, iteratively loops and constructs groups of nodes and links procedurally. In an object oriented design, the nodes, links and other ns-2 objects would simply construct themselves. Following an object oriented design schema would also help in breaking up the larger, low cohesion Generator class in the C/C++ script into highly cohesive, simpler ns-2 objects to create modularity for easier maintenance and modification. Considering the basic model for ns-2, the first model for the script generator looked like the UML diagram in Figure A.4.

This model expresses the basic components of an ns-2 simulation, as well as defining an ns-2 *superclass* that all ns-2 objects inherited basic functionality from. Here, the basic components of an ns-2 simulation are: a node, which can represent a computer system in a network; an agent, which can represent some client software on a computer system in a network; a link, which can represent any communication medium, from ethernet

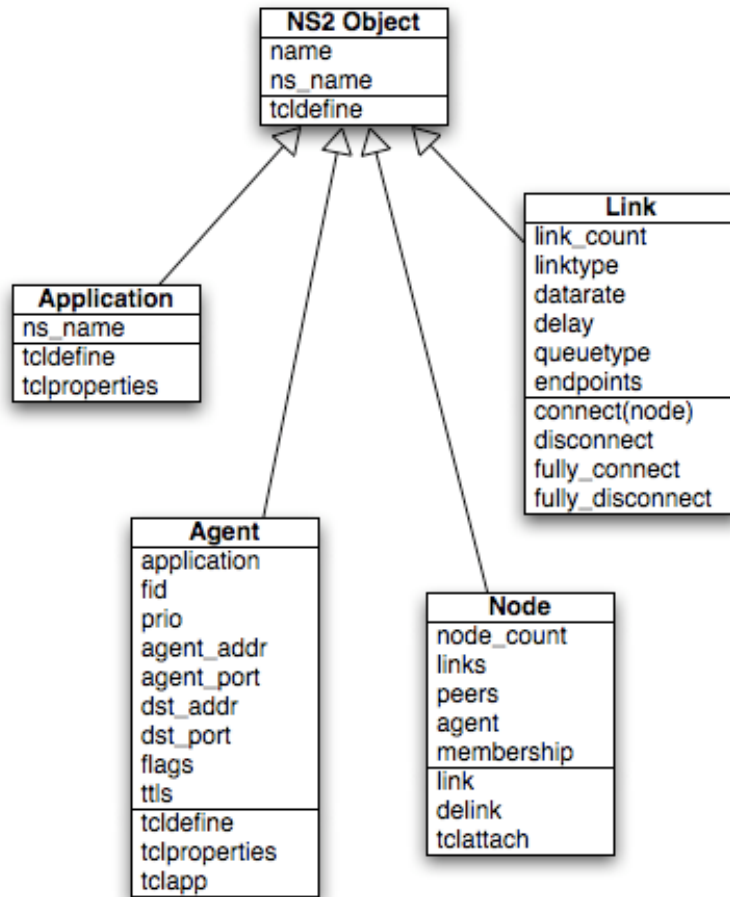


Figure A.4: Basic ns-2 Object Model

cables to satellite shots; and an application, which can represent a server or service available on a network. It is also important to note that nodes are the basic ns-2 objects that make up networks, and links connect nodes, and that one or more applications or agents are *attached* to nodes. In addition, all ns-2 objects possess unique ns-2 attributes and functions, so their subclass definitions were also populated with appropriate attributes and methods that expressed corresponding ns-2 attributes and functions. The core mechanism that translated the models into the ns-2/Tcl code that would be interpreted by the ns-2 simulator was the `tcldefine` method, which was inherited by all subclasses of the

ns-2 object. This method enabled all the ns-2 objects to “write themselves” into the ns-2/Tcl script.

Ruby was chosen as the programming language to implement the code. Ruby is an object-oriented, high-level scripting language that combines the object-oriented features of Java and Smalltalk programming languages with the fluid, scripting nature of Perl[24]. It allows for rapid development and at the same time can easily employ object-oriented design principles, such as the use of design patterns and *general responsibility assignment software principles* (GRASP). Design patterns, defined by Alexander, are a programming paradigm that helps improve the maintainability and reusability of source code [2]. GRASP principles are object oriented design methodologies make object-oriented code more efficient by ensuring objects and classes are assigned the correct responsibilities or functions [16]. Ruby also contains a number of built-in, useful utilities such as file input/output libraries and data structure manipulation tools. Ruby provides a good fit for modeling JBIs because it allows the use of software engineering design paradigms and advanced data structures, two factors that helped overcome the lack of flexibility and maintainability found in the original C/C++ script generator. To denote its use of the Ruby programming language and its use in generating *Cast experiments, the new script generator is referred to as *Ruby Cast*.

To use more than the basic nodes, agents, links or applications, ns-2 and Tcl syntax typically uses subclasses, and any ns-2 object is often a subclass of another. Therefore, this script would also follow this same design rule, and so the specific links, nodes

and agents necessary were implemented as subclasses to the basic types defined above.

This leads to this less general, more specific UML model in Figure A.5.

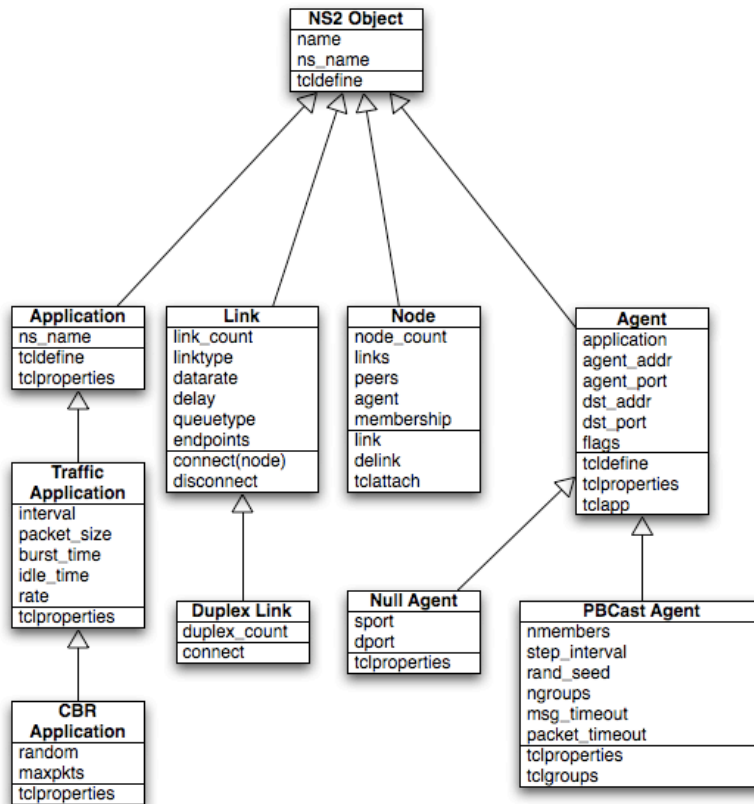


Figure A.5: Intermediate ns-2 Object Model

The important takeaway from this figure is that the PBCast agent, which would implement the code of all the *Cast protocols in the ns-2 simulation, is a subclass of the Agent class, which in turn is a subclass of the ns-2 objects. This is the custom agent that the PBCast source code generates. In addition, the Null Agents and the CBRs also appear in this version of the model, as well as the duplex links. Although any ns-2 object can be modeled in the script generator in this way, the pragmatic decision was made to implement only the subclasses that were needed to express the JBI scenario. If any additional ns-2 object were needed for the scenario, its code could be easily constructed and added

to the model later. This power and flexibility came from the object-oriented design of the script up to this point.

One mechanism that Tcl/ns-2 lacks is a facility for modeling topologies, such as a star, hub, ring or mesh. Such a mechanism is necessary to link all the separate nodes, links and agents. This was probably not incorporated by the VINT group because topologies are so wide and varied that it would not be trivial to implement. However, this design decision is what makes using Tcl/ns-2 code to make complex topologies for ns-2 simulations difficult and tedious. It forces programmers to come up with large, complex iterations in order to link nodes together to form the topologies. The best example is the C/C++ script generator; referring to the diagram in Figure A.2, in the method `generate_meshes`, the script utilizes a series of multi-dimensional arrays in multiple passes to construct nodes, configure their attributes and link them. In order to escape the ring topologies established in the original *Cast scenarios and use the meshes needed by the JBI scenario, the use of the C/C++ script generator would require non-trivial manipulations of the double-nested arrays in `generate_meshes`. Rather than follow this approach, it was decided to actually implement topologies, but do so in a way that the VINT group had not anticipated: the topologies would be implemented using a *composite pattern*, a design pattern that consists of treating member objects and the group those objects belong to uniformly [9]. Basically, this allows objects to be of the same class and *contain* objects of that same class. In this case, a topology super class, that was also a subclass to the ns-2 class, was created, and various subclasses for each of the topologies was created: a star subclass, a ring subclass, a mesh subclass, and so forth. Using this technique, a sce-

nario could be created with a topology that could contain either nodes or other topologies, which in turn might contain their own nodes and topologies, *ad infinitum*. An example of this appears in Figure A.6.

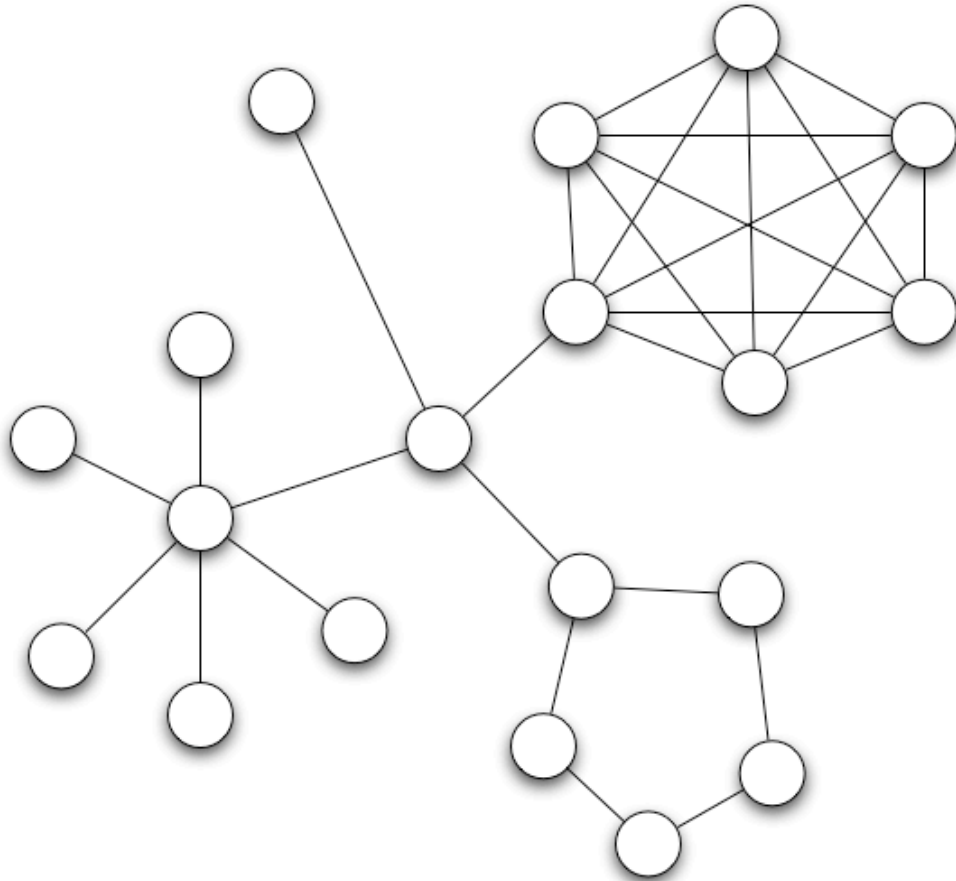


Figure A.6: Composite Topology

In Figure A.6, there is a star topology at the center. One of its spokes contains a node, and the other three contain a mesh, a ring and another star. The UML for this configuration, added to the earlier UML diagrams, appears in Figure A.7

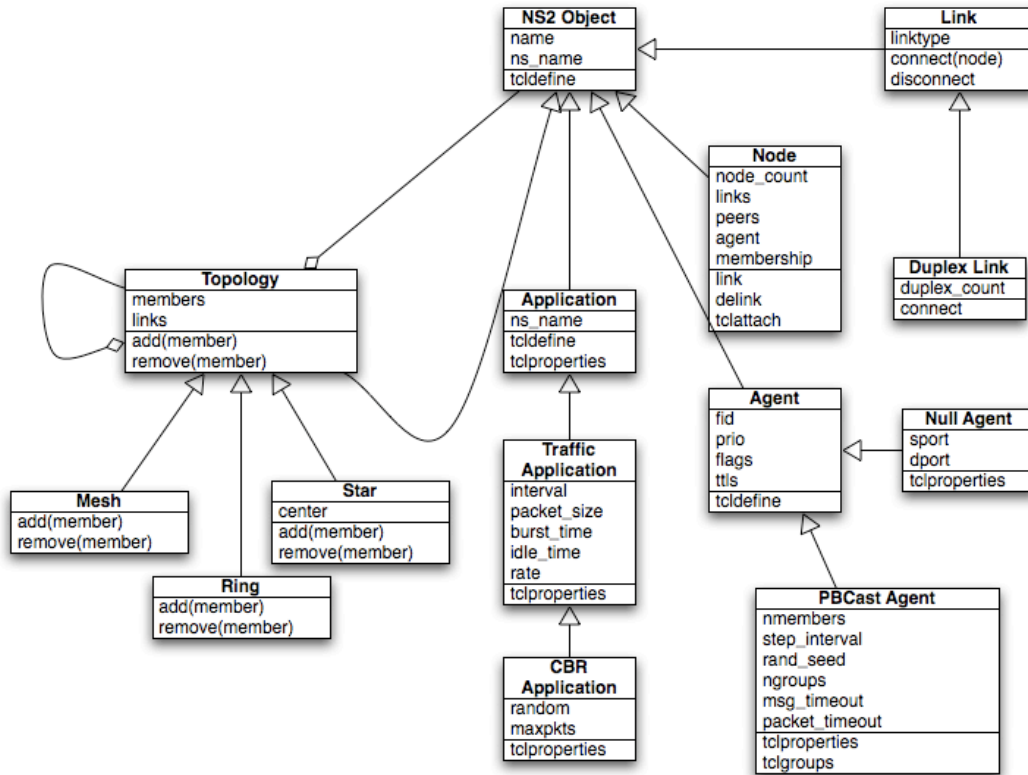


Figure A.7: Intermediate ns-2 Object Model With Composite Topology

Continuing with the overall theme of allowing the ns-2 objects to define themselves, a topology would call its `tcldefine` method, which in turn calls the `tcldefine` method for the objects it owns, all the way down the chain. Using the composite design pattern also created another hook into the *Cast mechanisms: as a result of nodes being grouped into topologies, that same topology could be considered a group level. As a result, one requirement of the scenario in Figure 3.1 in Chapter III was met: associated units, meshed physically in the network, become a group level.

At this point, the model had achieved the goal of bringing the scenarios created in the original *Cast experiments into a military environment. The second task to accomplish was to add support for multiple node types with customized interest levels. This means being able to model specific units with specific priorities for specific JBI applications. For example, one desired result would be generating a representation of an F-22 that has higher priorities, or interest, for receiving a data stream involving the Air Tasking Order (ATO). These units and interests are tabulated in Table 3.2 in Chapter III. This problem contained two parts: the first part was to define the multiple node types with customized interest levels, and the second part was to define the group applications that the nodes would subscribe to.

For the first part of the problem, the objective was to figure out where to encapsulate the specific interest for a specific unit. There was no clear fit, as the way the original PBCast code implements interest is not in the same way that ns-2 implements objects. An extremely procedural approach, interest levels are implemented as a massive multidimensional array where the coordinates identify the unit that possesses that interest in a particular

application stream. As a result, the class that would contain the interest needed to translate between the object oriented design of ns-2 and the procedural nature of the PBCast code. The first choice was to implement a subclass to the Node class. However, it was determined that solution was too complex and error prone, because the *Cast agent attached to the node would be the interface for *Cast traffic, data would have to be sent back and forth from a node to the attached agent. The next choice was placing the interests in a subclass of the *Cast agent. This approach would be too difficult to implement, as it requires multiple subclasses of the *Cast agent. Considering that there are four *Cast protocols being tested and seven different unit types, this would be labor intensive because twenty-eight subclasses would need to be created. Additionally, should more unit types be introduced into the scenario, a corresponding subclass for each *Cast protocol would also need to be created.

The best solution was found by applying the strategy design pattern [12]. By analyzing that the only component that varied were the set of interest levels, it was discovered that a better solution would be to implement *interest strategies*. Interest strategies would be fabricated objects that would encapsulate the interest levels for a particular unit type and would be added to a *Cast agent as that agent is being defined. When the script generator builds that particular agent into the Tcl script, it would reference the interest strategy for the interest level values and allocate that interest value into the multidimensional array in the PBCast code. The UML for this design is shown in Figure A.8.

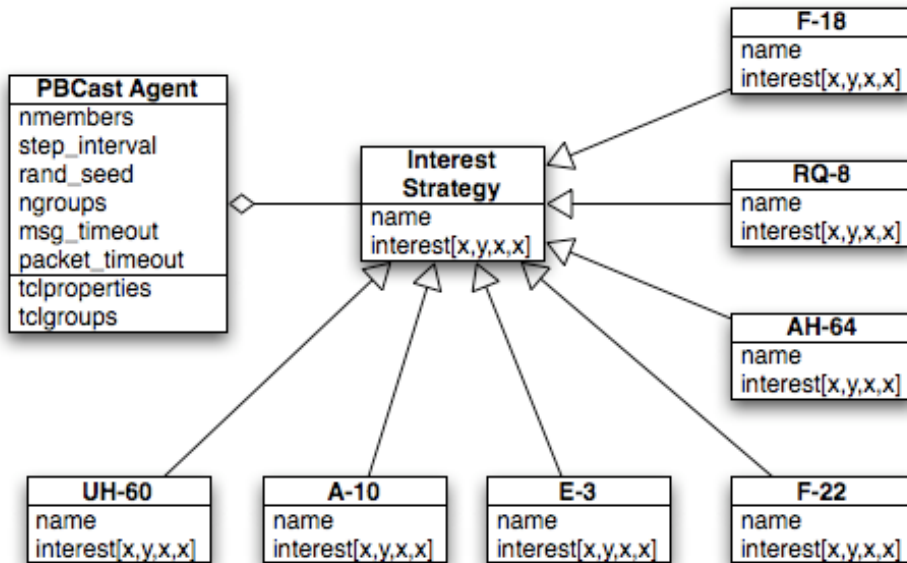


Figure A.8: Interest Strategy Model

Use of interest strategies allows the identification of units without the need for multiple node or agent classes, and provides access to interest data without having to create new subclasses for *Cast agents.

The second part of the problem, the objective involves defining the group in the scenario. Like the interest levels, the PBCast code implements groups in a non-intuitive, procedural way: groups are a collection of multidimensional arrays, whose coordinates define the group and its subscribers. Groups also do not have an equivalent ns-2 object, either. Basically, a group is an array that contains data for subscribers. The task is to decide what class would best encapsulate this data.

Unfortunately, no currently existing class in the model would fit the group data, and the best solution was to fabricate a completely new class. Since the group is not an actual ns-2 object, there was no need to make it a subclass of ns-2 objects. The new. The

class would be utilized by group levels, which are basically the topologies. From these considerations, the result is the class that hangs off of the topology, in Figure A.9.

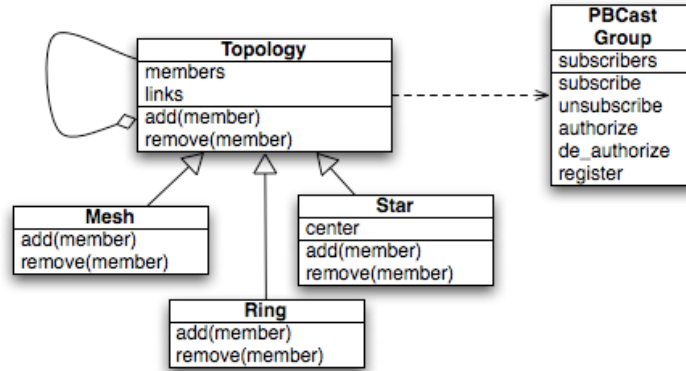


Figure A.9: PBCast Group

The additions of the interest strategies and the PBCastGroup class, the collection of classes with which to define the JBI model is complete (Figure A.10).

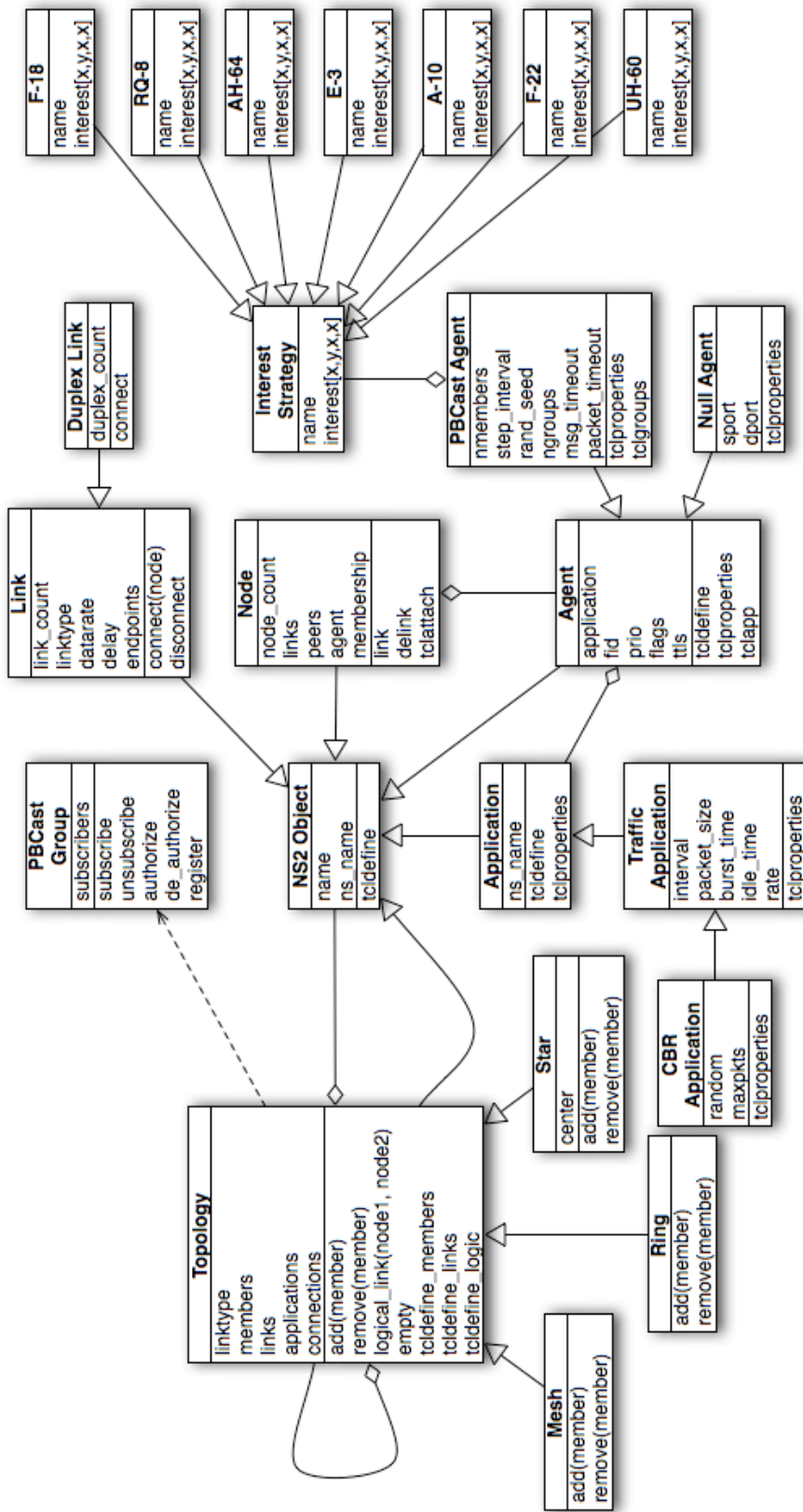


Figure A.10: Ruby Cast: Complete Model

With the completion of Ruby Cast, the capability now exists to generate scenarios that portray a JBI environment that incorporates multiple types of nodes with different priority levels for different application streams.

A.6 Summary

This chapter discussed the transformation of the original *Cast testing environment from an academic environment into a military environment. The process started with an evaluation of the existing scenario script generator and resulted in the development of a new script generator, Ruby Cast. The result is a modification of the process as it appears Figure 3.10 in Chapter III.

In the end, this appears to be a long and complicated task that results in code that does what previous versions of the code accomplished. This begets the question, what benefits are derived from this intense re-engineering effort? Comparing the UML diagram for Ruby Cast (Figures A.9) to the original C/C++ script generator (Figure A.3) the complexity seems to have increased with the number of objects. The difference is the complexity is spread over multiple classes that represent concepts in the problem (ns-2 objects and the JBI model), and the result is that the identity of classes and the functionality is more easily understood and is more easily modified and extended. The benefit is that not only can the script generator generate scenarios for the current *Cast experiment, but also can be reconfigured quickly and rapidly to support any changes in any *Cast experiment in the future.

Appendix B. Supplementary Diagrams

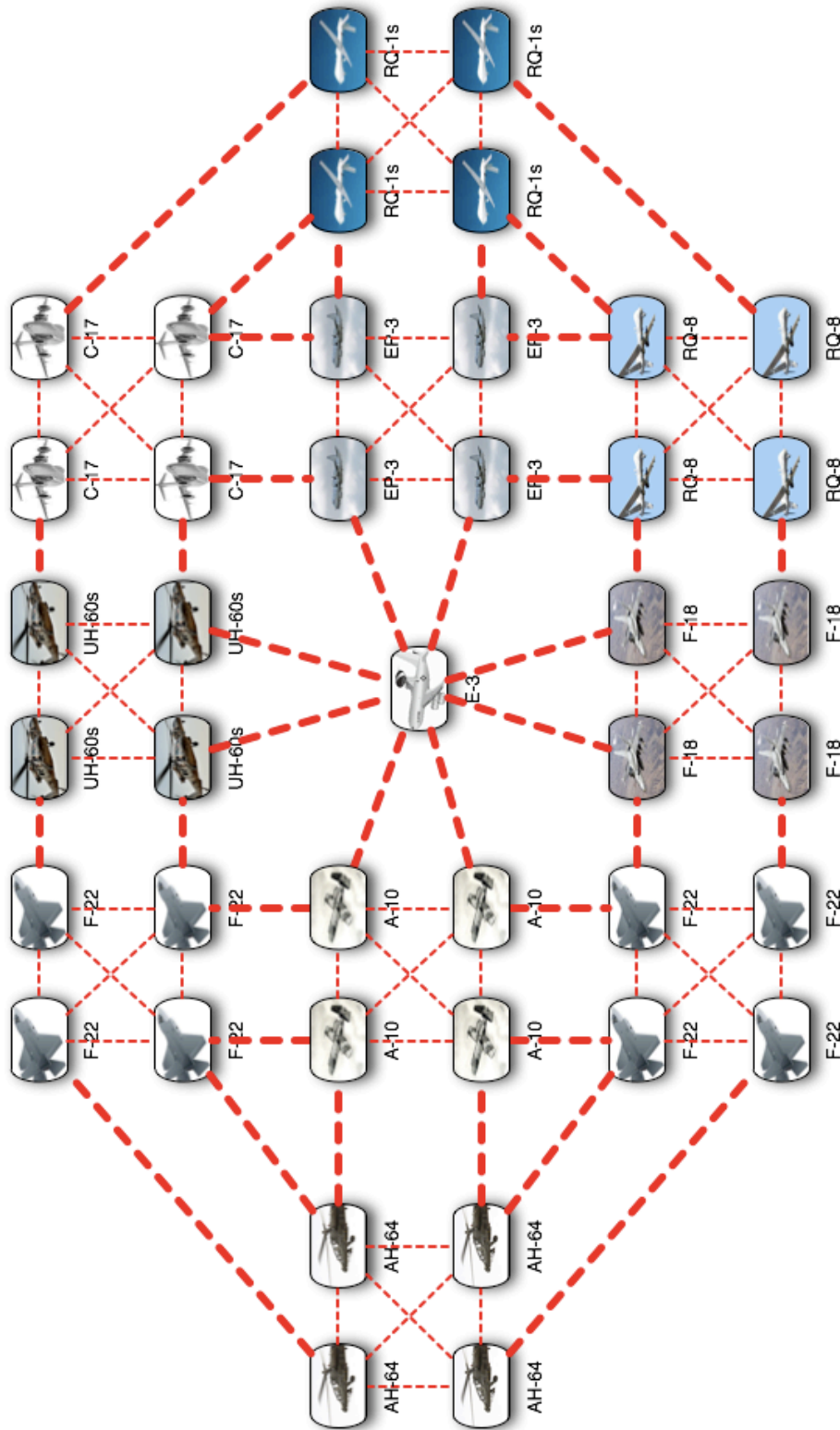


Figure B.1 Scenario Topology, Expanded

Bibliography

1. nsnam.isi.edu/nsnam/index.php/Main_Page
2. Alexander, Christopher, Sara Ishikawa, Murray Silverstein, Max Jacobson, Ingrid Fiksdahl-King, and Shlomo Angel. *A Pattern Language*. Oxford University Press, New York, 1977.
3. Birman, Kenneth, Ken Hopkinson and Kate Jenkins. *A Gossip Protocol For Subgroup Multicast*, Department of Computer Science, Cornell University (Upson Hall), Ithaca, NY 14853, 1999.
4. Birman, Kenneth, Mark Hayden, Yaron Minsky, and Zhen Xiao. "Bimodal Multicast," *ACM Transactions on Computer Systems*, 17,2:41-88, May 1999.
5. Brown, Alan and M. Kolberg. "Internet-Draft Tools for P2P Network Simulation." Internet draft submitted to IETF.
<http://tools.ietf.org/group/irtf/draft-irtf-p2prg-core-simulators-00.txt> , 27 January, 2006.
6. Combs, Vaughn T., and Dr. Mark Linderman. "A Jini-based Publish and Subscribe Capability," *Proceedings for the International Society for Optical Engineering (SPIE)*, 4863:59-69, 2002.
7. Demers, A., D. Greene, C. Hauser, W. Irish, and J. Larson, "Epidemic Algorithms for Replicated Database Maintenance," *Proceedings of the ACM Symposium on Principles of Distributed Computing*, 1987.
8. Fall, Kevin and Kanna Varadhan. *The ns Manual*, Computer Science Division, University of California, 387 Soda Hall, Berkeley, CA 94720-1776.
9. Gamma, Erich, Richard Helm, Ralph Johnson and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, Boston, MA 1995.
10. Garlan, David and Mary Shaw. "An Introduction to Software Architecture." School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213-3890, January 1994.

11. Gjermundrød, K. Harald, Ioanna Dionysiou, David Bakken, Carl Hauser and Anjan Bose. "Flexible and Robust Status Dissemination Middleware," School of Electrical Engineering and Computer Science, Washington State University, Pullman, WA 99164-2752, 25 September 2003.
12. Halbert, Daniel C. and Patrick D. O'Brian. "Object-oriented Development," *IEEE Software*, 4(5):71-79, September 1987.
13. Hopkinson, Kenneth, Kate Jenkins, Kenneth Birman, James Thorp and Manu Parashar. *Adaptive Gravitational Gossip: A Gossip-based Communication Protocol with User-selectable Rates for Use in Monitoring the Electric Power Grid*, Department of Computer Science, Cornell University (Upson Hall), Ithaca, NY 14853, 30 May 2005.
14. Jelasity, Márk, Rachid Guerraoui, Anne-Marie Kermarrec and Maarten Van Steen. "Peer Sampling Service: Experimental Evaluation of Unstructured Gossip-Based Implementations," *Proceedings of the 5th Association of Computing Machinery (ACM)/International Federation for Information Processing (IFIP)/Advanced Computing Technical Association (USENIX) International Conference on Middleware*, 78:79-98. 2004.
15. Ji, Chuanyi and Anwar Elwalid. "Measurement-Based Network Monitoring and Inference: Scalability and Missing Information." *IEEE Journal on Selected Areas in Communications*, 20,4:714-725. May 2002.
16. Larman, Craig. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development*, Prentice Hall PTR, Upper Saddle River, NJ 07458, 2005.
17. Liskov, Barbara. "Data Abstraction and Hierarchy," *SIGPLAN Notices*, 23:5, May 1988.
18. Loyall, Joseph, Jamie Lawson and Gary Duzan. "Issues in Providing Quality of Service in a Joint Battlespace Infosphere," *Proceedings of the 10th IEEE International Workshop on Object-Oriented Real-Time Dependable Systems (WORDS)*, February 2005.
19. Marmelstein, Robert E., Lt Col. "Force Templates: A Blueprint for Coalition Interaction Within An Infosphere." *IEEE Intelligent Systems*, 17,3:2-7, May/June 2002.

20. Pereira, José, Luis Rodrigues, Alexandre S. Pinto, and Rui Carlos Oliveria. "Low Latency Probabilistic Broadcast in Wide Area Networks," *Proceedings of the 23rd IEEE International Symposium on Reliable Distributed Systems (SRDS04)*, 1:299-308. (18-20 October 2004).
21. Pereira, José, Luis Rodrigues, M. João Monteiro, Rui Carlos Oliveria, and Anne-Marie Kermarrec. "NEEM: Network-friendly Epidemic Multicast." *Proceedings on Reliable Distributed Systems*, 1:15-24. 6-18 October 2003.
22. Rodrigues, Luis, Sidath Handurkande, José Pereira, Rachid Guerraoui, and Anne-Marie Kermarrec. "Adaptive Gossip-Based Broadcast," *Proceedings, 2003 International Conference on Dependable Systems and Networks*, 1:47-56. 22-25 June 2003.
23. Satterthwaite, Charles P., Timothy W. Blocher, Dr. Davide E. Corman, Thomas S. Herm and Eric J. Martens. "IEST Force Template Technology Provides A Key Capability For Connecting Tactical Platforms to the Global Information Grid," *Digital Avionics Systems Conference*, 23,2: 11.B.2-11.1-9, October 2004.
24. Thomas, Dave. *Programming Ruby: The Pragmatic Programmer's Guide*. The Pragmatic Programmers, LLC, Raleigh, NC, October 2006.
25. US Air Force Scientific Advisory Board, *Report on Information Management to Support the Warrior*, SAB-TR-98-02. Washington: HQ USAF, December 17 2000.
26. Van Renesse, Robert, Kenneth Birman and W. Vogels. "Astrolabe: A Robust and Scalable Technology for Distributed System Monitoring, Management, and Data Mining," *ACM Transactions on Computer Systems*, 21,2:164-206, May 2003.
27. Zhang, Qi and Dharma P. Agarwal. "Dynamic Probabilistic Broadcasting in MANETs." *Journal of Parallel and Distributed Computing*, 65:220-233, 6 November 2004.

Vita

Capt E. Descartes Aban graduated from Ontario High School in Ontario, Ohio. He journeyed to Chicago, Illinois to endure the rigors of Illinois Institute of Technology (IIT) with an Air Force ROTC scholarship, and in December 1999 he was granted a Bachelors of Science in Computer Engineering as compensation for the days spent hunched over computers.

Upon graduation, he was commissioned through Detachment 195 AFROTC at IIT and was assigned to Scope Network/Global Communications Directorate of the Air Force Communications Agency at Scott AFB,IL, where he served as the Lead Information Assurance/Network Engineer,. In February 2002, he was assigned to the 33rd Information Operations Squadron, Lackland AFB, TX, where he conducted information operations in multiple roles: as ops crew commander, incident responder, and systems engineer. In September 2004, he deployed to US CENTAF headquarters, Qatar, to serve as the Chief of Information Assurance for the entire theater of operations. After returning from the desert, he entered the the Graduate School of Engineering and Management at the Air Force Institute of Technology, where he has toiled ceaselessly in pursuit of a masters degree in computer science. Scheduled to graduate in March 2007, he will be assigned to the 83rd Communications Squadron at Langley AFB, VA.

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE				Form Approved OMB No. 074-0188	
<p>The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of the collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.</p> <p>PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.</p>					
1. REPORT DATE (DD-MM-YYYY)		2. REPORT TYPE		3. DATES COVERED (From – To)	
22-03-2007		Master's Thesis		August 2005 – March 2007	
4. TITLE AND SUBTITLE Adaptive Gravitational Gossip in Monitoring the Joint Battlespace Infosphere				5a. CONTRACT NUMBER	
				5b. GRANT NUMBER	
				5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S) Aban, Edmund D, USAF				5d. PROJECT NUMBER	
				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 Hobson Way WPAFB OH 45433-7765				8. PERFORMING ORGANIZATION REPORT NUMBER AFIT/GCS/ENG/07-01	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Dr. Bob Herklotz (703) 696-6207 AFOSR 875 North Randolph Street Arlington, VA 22203-1768 robert.herklotz@afosr.af.mil				10. SPONSOR/MONITOR'S ACRONYM(S)	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION/AVAILABILITY STATEMENT APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT Future USAF operations will be heavily dependent on having the “right information” at the “right time”, and Joint Battlespace Infospheres (JBIs) are poised to fill that role. To do this, JBIs must be ubiquitous—always accessible, secure and responsive. Of all the literature written regarding JBIs, the most important problem to solve in order to make JBIs work in mobile scenarios are scalability, reliability and adaptability to changing battlefield conditions. This paper explores the use of SBCast, a novel adaptive probabilistic protocol and a delivery mechanism for JBI updates and as a possible solution towards guaranteeing these qualities. It documents tests of SBCast within a simulation environment configured with parameters based on actual military field operations. From these tests, the paper examines SBCast as an enhancer to JBI's ability for overcoming transient network failures while managing different classes of subscribers by available bandwidth and priorities. By using the feedback from SBCast as a middleware layer controller, JBIs would be able to “dial up” traffic for parts of the network and “dial down” traffic in others based on dynamic changes in network congestion or traffic demands.					
15. SUBJECT TERMS TCP, buffering, network layer, transport layer, congestion control, OPNET, channel loss					
16. SECURITY CLASSIFICATION OF:		17. LIMITATION OF ABSTRACT		18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE	UU	120	Dr. Kenneth Hopkinson
U	U	U			19b. TELEPHONE NUMBER (Include area code) (937) 255-3636 x4579 kenneth.hopkinson@afit.edu