2012

P2P Protest: Practical Adaptations of Epidemic Routing for Mobile Devices

Curtis Carmony
Bard College

P2P Protest: Practical Adaptations of Epidemic Routing for Mobile Devices

A Senior Project submitted to
The Division of Science, Mathematics, and Computing
of
Bard College

by
Curtis Carmony

Annandale-on-Hudson, New York
May, 2012

Abstract

A routing protocol known as Epidemic Routing has been proposed as a means to transmit information between nodes in partially connected mobile ad hoc networks (MANET's). Many different aspects of the Epidemic Routing protocol such as energy efficiency and message delivery latency have been examined in terms of military and scientific applications. This research examines the adaptation of the Epidemic Routing protocol and its variants as a possible means of communicating in urban protest type environments where traditional means of technological communication are easily disrupted by malicious parties. With this application in mind, the Epidemic Routing protocol was evaluated in a series of simulations to show that it is surprisingly capable of efficiently disseminating information in this type of situation and of withstanding tampering from malicious parties. Lastly the feasibility of adapting the Epidemic Routing protocol to a real world application both for the purpose of disseminating information in an urban protest environment and in general is examined given current technological limitations. The lack of a suitable wireless networking technology is identified as the primary deterrent in creating a real world adaptation of an Epidemic Routing protocol and the general requirements needed for a successful adaptation are outlined.

Contents

List of Figures

Dedication

To my loving family who have always supported me and my technological endeavors, no matter how extravagant, even though I didn't always know what I was doing. Not many parents would give their sixth grader a credit card to "build a computer", but I'm glad mine did. Thanks mom and dad, you now officially have a very expensive dedicated technical support staff. I promise I'll get that server rack out of the shop one day.

Acknowledgments

The Bard CS department has been instrumental in this project, namely in teaching me Computer Science. Specifically I would like to thank my adviser Keith O'Hara for seeing this project through, Sven Anderson for his advice and inspiration, and Hap Tivey for teaching me something I loved other than computer science. More importantly I would like to thank all three for proving to be close friends and valuable mentors over my four years at Bard.

1 Introduction

1.1 Overview

Starting in mid December 2010, a series of revolutions, uprisings, and protests swept through the Arab world. Aside from the tremendous political importance these movements had, they demonstrated for the first time on a large scale the impact technology, especially social media, can have on the interaction and organization of people within political movements. The extreme proliferation of commodity priced Internet ready computers along with social media networks such as Facebook, Twitter, and their international variants provided a robust platform for the rapid dissemination of political and organizational information. Where Paul Revere had to notify his fellow revolutionaries of troop movements with a midnight ride and lanterns, members of the Arab Spring were able to disseminate important information to millions of interested individuals in seconds with a certain degree of anonymity[4]. How significant this capability actually was is perhaps best evidenced by how dangerous the incumbent governments perceived it to be. Aside from attempting to identify the users of these technologies, these governments went to great lengths to prevent the dissemination of this information, be it through censoring the information communicated through these technologies or by shutting them down completely.

This ability of a government to censor, or "turn off" these technologies is arguably their most significant flaw when used in this application. Regardless of how close or far away a user is to his or her intended recipients, the information must pass through centralized gateways. Especially in developing nations where there are relatively few Internet connections coming in and out of the country, it is not a difficult task for the incumbent government to monitor, alter, or shutdown these connections. When this happens, these technologies either stop working (they are shutdown), they are no longer useful (the information has been censored), or they are dangerous (the integrity of the technology has been compromised and users are identifiable or misinformation is being disseminated).

Given the extent to which these technologies have been developed, it is somewhat surprising how easily they can be thwarted and that communications between individuals must pass through a remote, centralized location that is, or can easily become, under government control. Even in the United States, Bay Area Rapid Transit officers were able to shut down cell phone service on train platforms in an attempt to quell a potential protest without any judicial oversight or approval[3]. It seems mind-boggling that two people within a few feet of each other with several hundreds of dollars of the most recent technology are completely prevented from communicating digitally at the flip of a switch which they have no control over. What's needed is someway of transmitting information between users, even with lower reliability and efficiency, without transmitting data through these critical points. Peer-to-peer (P2P) networking protocols address the issue partially by allowing users to bypass centralized servers, such as those that run Facebook or Twitter, but still transmit information through the Internet and are still susceptible to any governmental control that might be over the few critical connections to it.

What is really needed is a distributed P2P networking protocol where users pass information *to* and *through* each other directly, without having to pass any information through centralized servers or the Internet. With such a protocol it should be theoretically possible for people in these types of situations to communicate with one another and disseminate information even if a malicious authority has taken control of a country's access to the Internet. This is a study of how exactly this would be theoretically possible, what new implications would arise in terms of the stability and security of the network, and an assessment of whether or not such a networking protocol is possible using current technology.

1.2 Epidemic Routing

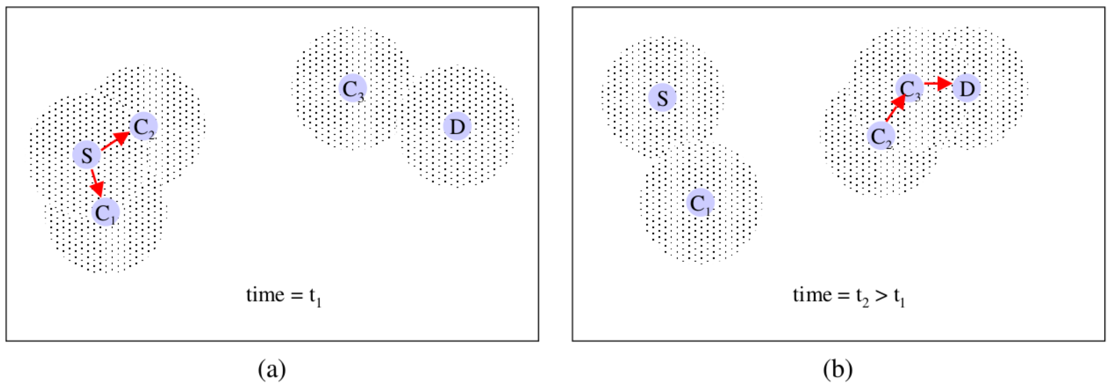


Figure 1.2.1. Becker and Vahdat's representation of Epidemic Routing [14]. Sender S sends a message to Recipient D through nodes C_2 and C_3.

Epidemic Routing was designed by David Becker and Amin Vahdat in 2000[14] to overcome the limitations of existing ad-hoc networking protocols which fail to deliver a message between two nodes in a network where there is no connection between them. Their new protocol would deliver "application data with high probability even when there is never a fully connected path between source and destination." The novel alteration they made to existing protocols was to transfer data between nodes regardless of whether or not the receiving node is the final recipient of the data, or whether or not it is connected to the final recipient, directly or through other nodes. Data spreads through the network from node to node like a virus until the destination node is reached, hence the name Epidemic Routing (Figure 1.2.1). Beyond the ordinary concerns in designing a network protocol with efficiency and speed, designing the Epidemic Routing protocol and its variants has the additional concern of a message consuming a significant amount of resources without it ever being delivered. To address these concerns Epidemic Routing was designed to "i) efficiently distribute messages through partially connected ad-hoc networks in a probabilistic fashion, ii) minimize the amount of resources consumed in delivering any single message, and iii) maximize the percentage of messages that are eventually delivered to their destination."[14].

To examine the effectiveness of their protocol in meeting these goals Becker and Vahdat conducted a series of experiments in which they measured the average latency and percentage of message delivery in comparison to the amount of resources dedicated to the protocol in a given experiment. These three design goals address the performance and reliability of the protocol, but Becker and Vahdat also acknowledge that any serious real world implementation would have to consider security in its design and as a metric of its performance. The Epidemic Routing protocol largely ignores the issue of security and its metric for the sake of simplicity, but other researchers have taken it upon themselves to explore how the protocol can be adapted and expanded to address them, looking into the issues of anonymity[5], considering the implications on stagnant messages in the network[7], maintaining the privacy of the node's physical location[8], and energy conservation in the protocol[15].

The adaptations proposed and evaluated in the research of Epidemic Routing and its variants are heavily dependent on the assumed applications of the protocol and most researchers have tended to focus their adaptations onto one constraint that they feel significantly impacts their proposed application of the protocol. Becker and Vahdat describe four possible applications of their original Epidemic Routing protocol, which sums up most of the research in the area: mobile sensor networks, smart dust, disaster recovery, and military deployment. In all instances, power consumption is considered to be the most limiting factor in terms of the routing protocol's performance and reliability as nodes in these applications are assumed to be placed over a large geographic area without any direct power source making their battery life the most limiting factor of the network.

2
Protocol Recreation and Results

2.1 Algorithm Description

In the case where two nodes in a network are not directly connected, they must depend on some sort of routing protocol to transmit information. In traditional routing protocols a connection between the two nodes is made indirectly by going through other nodes and/or gateways. The Epidemic Routing protocol aims to deliver messages between sender and recipient when there isn't even an indirect connection between them. In order to do this the Epidemic Routing protocol relies on other nodes in the network to carry and deliver messages. Messages are distributed throughout a network by the Epidemic Routing protocol like viruses. Every time two nodes come into contact they exchange messages, copying the messages that exist in one node's buffer but not the other. Messages spread throughout the network in this way, hopefully reaching its destination before it is replaced by newer messages. The algorithm behind the original Epidemic Routing protocol can be described in three stages: neighbor discovery, the anti-entropy session, and message delivery. Neighbor discovery is the process in which a node in the network comes within communication range of other nodes and selects the nodes which it is going to interact with. In the original Epidemic Routing protocol the node first requests its neighbors' unique ID's and looks them up in an internal table to see if they have interacted before, and if so when that last was. Each node has a parameter which as been labeled *neighbor time* (this parameter is described but not explicitly named by Vadhat and Becker) which determines the minimum amount of time required before the node will interact with a node again. If the node decides to interact with one of its neighbors it begins what Becker and Vahdat call an *anti-entropy session*, as seen in the *newNeighbors* function in Figure2.1.1.

The anti-entropy session is a process in which the two nodes communicate which messages they currently are in possession of. The anti-entropy session begins with the node with the smaller identifier sending a bit vector called a *summary vector* to the other node. The other node sends its own summary vector, and then each node computes which messages it would like to receive from the other node, this is seen in the *receiveSummaryVector* function in Figure 2.1.1. In the original Epidemic Routing protocol, no criteria exists for determining if a message is desired and so all nodes simply ask for all of the messages that the other communicating node has that they don't. Becker and Vahdat do mention however, that future adaptations of the protocol should allow the recipient node to decide which messages it would like to carry.

In the final stage of the Epidemic Routing protocol each node sends a message request to the other, and then sends the messages the other node requested. When receiving messages each node looks at each message's *hop count*, a parameter set on message creation that determines the the number of nodes a message can be received by before it is delivered to its destination, seen in the *receiveMessages* function in Figure 2.1.1. A message created with a hop count of one for example, can only be given to its recipient directly and cannot be transmitted through any other nodes. Each node stores a hash-table of the messages it is currently carrying, along with two buffers which contain lists of the messages the node it is carrying for itself and which messages it is carrying for others. In Becker and Vahdat's implementation messages are removed from the buffers and the hash-table when a buffer becomes full on a FIFO basis. When a node receives a message for which it is the listed recipient it does not notify any of the other nodes that the message has been delivered, and the message continues to spread through the network. What keeps messages from being forwarded indefinitely is the hop count parameter.

```
function newNeighbors($neighbors$)
    for Neighbor $n$ in $neighbors$ do
        if currentTime$-$n.lastSeen$\leq$ $neighborTime$ then
            sendSummaryVector($n$)
        end if
    end for
end function

function receiveSummaryVector($sv$)
    KeyList $keys$ = new KeyList()
    for MessageKey $mk$ in $sv$ do
        if !$messagTable$.contains($mk$) then
            $keys$.add($mk$)
        end if
    end for
    requestMessages($keys$)
end function

function receiveMessages($ml$)
    for Message $m$ in $ml$ do
        if $m$.hops$\leq$ $m$.maxHops then
            $m$.hops= $m$.hops1
            $messageTable$.add($m$)
        end if
    end for
end function
```

Figure 2.1.1. Psuedocode for some functions in the Epidemic Routing algorithm which take protocol parameters such as hop count and neighbor time into account.

Becker and Vahdat's experimental results show that the basic Epidemic Routing protocol is largely successful in achieving these goals. In comparison to existing ad hoc networking protocols that expect a fully connected path between source and destination nodes, Epidemic Routing provides a significant performance increase in terms of the percentage of messages delivered in networks without fully connected paths between source and destination nodes. Their results also demonstrate the protocol's success in largely fulfilling the three primary goals. By altering protocol parameters such as radio connection range, hop count, and buffer size they demonstrate the trade off between resource consumption and performance.

2.2 Implementation

The Epidemic Routing protocol as outlined by Becker and Vahdat was reimplemented to recreate their published results. By reimplementing the protocol this research is able to better understand the protocol and to create a foundational implementation upon which it can experiment and adapt. The implementation is separated into three levels in order to facilitate modular alteration of the protocol: the node, the router, and the network interface. Each of these is designed encapsulate a certain function in the protocol without being too dependent on the technicalities of the implementation. With the implementation constructed this way it is easier to alter aspects of the protocol in a modular way without having to re-write the entire thing. This eases the comparison between the impact various alterations have on performance in comparison to the original protocol. Java was selected as the language for development in the hopes that it would be easier to port to different platforms.

On top of the implementation of the protocol a basic simulation using the Processing environment was developed named NetSim. Figure 2.2.1 shows the NetSim simulation in action. The graphical display of the simulation is optional and was written to better visualize the spread of messages throughout the network. It can be disabled, allowing for faster simulations, which was done in creating all of the presented data. The simulation models the motion of the nodes within a limited physical space using the same random way-point motion model used in the original research. It does not take into account the subtleties

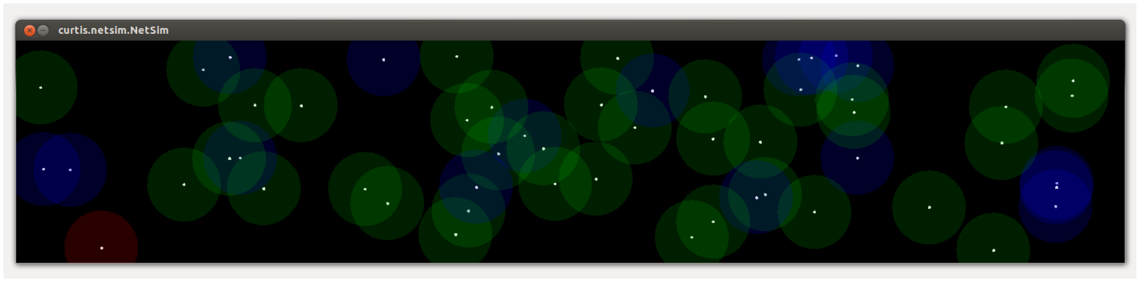


Figure 2.2.1. A screen shot of the $NetSim$ simulation. The white dot is the node's center, the colored circle its communication radius. The communication radius colors shows the node's most recent action. Green indicates a message sent, blue a message accepted to carry, pink a message delivered.

of wireless technologies, or simulate the transmission of data through a real networking protocol. Becker and Vahdat used the Monarch simulator to test their implementation, which does simulate the network actions on the lowest level. These differences in the simulations definitely have an impact on the performance of the algorithm, but not in any overly significant way. The published results, for example, have many more "artifacts" in the data that could be attributed to the complexity of the simulation, while the results produced in this research follow the same general trends but appear much more regular.

The results produced in this research is also sometimes significantly better than those published, which again most likely results from the simplicity of NetSim simulation. It is also important to note that Becker and Vahdat state that they use the 802.11 (Wi-Fi) networking protocol, but do not describe how they circumvent its prohibitive aspects. In other words, the 802.11 protocol could not actually be used in a real world implementation of their simulation do to the high connection establishment time.

In recreating Becker and Vahdat's experiments the parameters for their experiments were copied to the extent that they were described. There are many different parameters listed by Becker and Vahdat in their description of the protocol, a full list can be seen in Figure2.2.4, but in describing their experiments they neglected to say what values they used for many of them. For these unknown parameters reasonable values were selected or the reimplemented algorithm was altered so that they can be ignored in an experiment so as to not impede the performance of the protocol.

For example, in describing the algorithm they mention two message buffers, one containing only messages which originated with the node, and the other only containing messages which the node is carrying for others. In describing the parameters for the experiments they ran, they mention only one buffer size. Whether they are describing the size of each buffer, or if they are splitting the size between the two buffers is unclear and so the decision to give each buffer this size was made.

Aside from these parameters, all of these experiments take place in a 300 by 1500 meter space, with 50 nodes, a communication radius of 50 meters, with 1980 messages being delivered. The network parameters explored in the next two experiments are the message hop limit and the node's buffer size, respectively. Unless otherwise noted all of the original experimental data is the average of 20 different simulations run with different random seeds. Becker and Vahdat do not mention how many simulations their results represent. The 1980 messages are created by a message generation mechanism devised by Becker and Vahdat in which 45 of the 50 nodes are randomly selected to send messages. Each of these nodes randomly selects 44 nodes to whom they send a message. A single message in the entire simulation is sent every second. In the recreated implementation, all of these messages are generated before the simulation begins and are added to a buffer. Every second the simulation picks the head off of the buffer and then gives the message to the associated node to be sent. Since 1980 messages are generated in total, message sending is not complete until after 1980 seconds.

In the first experiment, the message buffer size was set to 2000 for both the internal and external buffers since in this case that makes the buffers essentially "infinite" as there are only 1980 messages in the simulation. Becker and Vahdat did the same as the intention is to demonstrate the effect of the message hop limit on the performance of the protocol. The

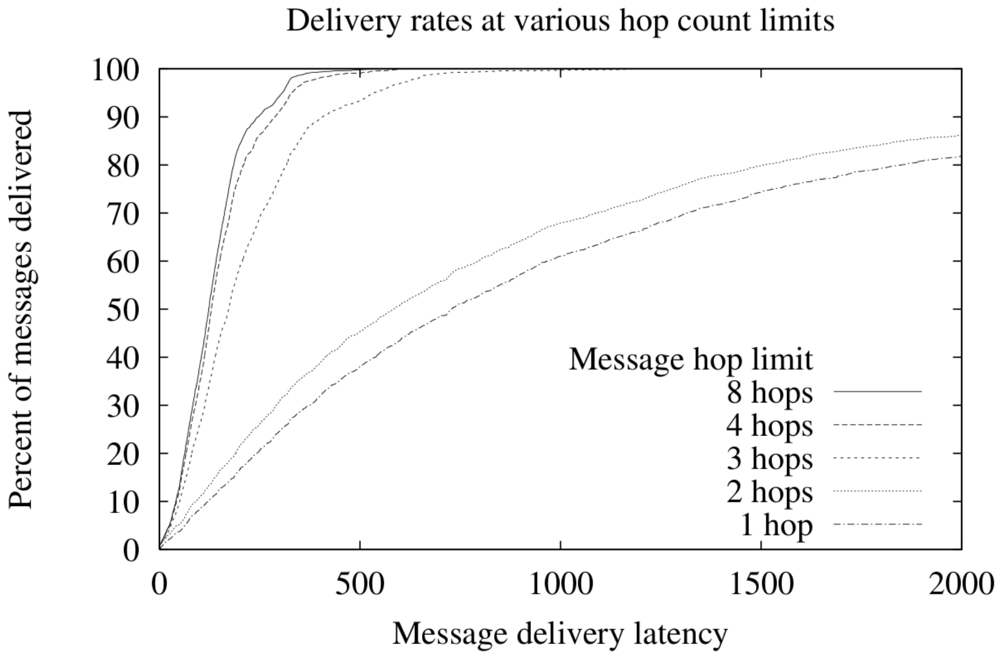


Figure 2.2.2. Becker and Vahdat's published results, showing protocol performance for varying hop counts with a buffer size of 2000[14].

second experiment uses a hop count of 4 to demonstrate the performance of the protocol with varying buffer sizes. In all of the experiments presented below, the graphed data is the percentage of messages delivered within a certain latency which demonstrates the efficiency of the protocol.

The first experiment compares performance of the Epidemic Routing protocol with different hop counts. The intent of this experiment is to show that if the message hop limit is set sufficiently high, a majority of the messages in the simulation can be delivered with a reasonable latency. The results seen in Figure 2.2.2 and Figure 2.2.3 show Becker and Vahdat's results and the recreated results, respectively. As can be seen, they both demonstrate the same general trend. As the message hop limit decreases, the average latency of message delivery increases. Decreasing the message hop limit decreases the number of copies of a message within the system, as when the hop limit is reached the

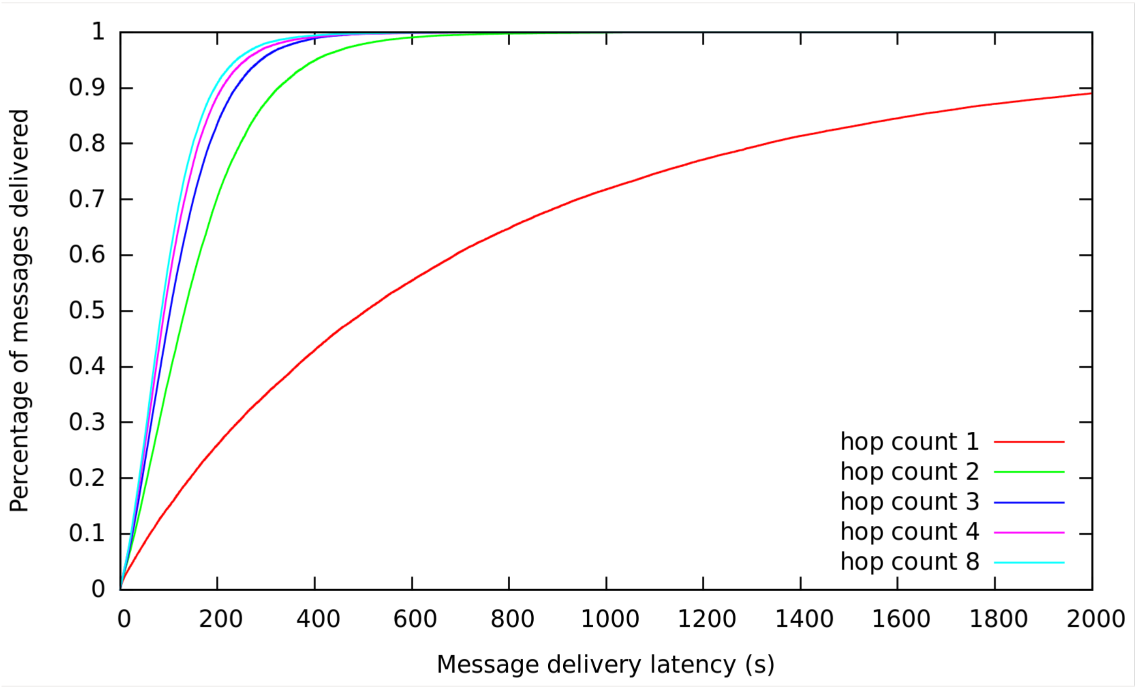


Figure 2.2.3. Recreated results, showing protocol performance for varying hop counts with a buffer size of 2000.

message is not permitted to be delivered to another node until it reaches its destination. For example, with a message hop limit of 1, there are at most 3960 messages since copies of the message can only exist with the message sender and the message recipient. On the whole, the results demonstrate what Becker and Vahdat argue, namely that it the Epidemic Routing protocol is capable of producing reasonable performance without extremely high resource consumption.

As previously mentioned the results produced in this research are much "smoother" which is believed to be caused by the simplicity of the NetSim simulation in comparison to Monarch. This could also be caused by the number of times the simulation was run and how their results were averaged. The most unsettling difference between the two sets of results is the instance in which the message hop limit is 2. Unlike Becker and Vahdat's results, the protocol's performance in the NetSim simulation with a message hop limit of 2 is much closer to that of a message hop limit of 1. It had first been considered that there

- Max Hop Count - The max number of “hops” a message can take.
- Internal Buffer Size - The size of buffer for messages a node is sending.
- External Buffer Size - The size of the buffer for messages a node is carrying.
- Neighbor Time - The minimum interval between connection reestablishment for two nodes.
- Communication Radius - The radius of the simulated wireless signal.
- Number of Nodes - The number of nodes in the simulation.
- Width - The width of the simulation.
- Height - The height of the simulation.
- Average Velocity - The average velocity of the nodes.
- Average Wait Time - The average time between reaching a way-point and picking a new one.

Figure 2.2.4. List of parameters that can be set in the NetSim simulation when simulating the Epidemic Routing protocol.

was a bug in this research’s implementation of the protocol or in the NetSim simulation, but after careful review it is thought that the cause of the discrepancy is the previously mentioned fact that the NetSim simulation does not take into account the time it takes to establish a connection and deliver a message against the amount of time two nodes are within communication range. This discrepancy would become much larger as the hop count decreased, as not only would the sending node have to come into direct contact with the recipient node, but they would have to be in contact long enough for the message to be delivered.

The second experiment focuses on determining protocol performance with varying buffer sizes. Their results demonstrate what’s expected; as the buffer size is decreased the message delivery latency increases. Like the message hop limit, the buffer size effects the total number of message copies that can exist in the network. As described previously, in the original protocol when the message buffer becomes full messages are removed on a First-In First-Out (FIFO) basis. This means basically that message copies are eventually removed from the system. Looking at Becker and Vahdat’s results the message buffer size appears to have a much more significant impact on the protocol’s performance than the message

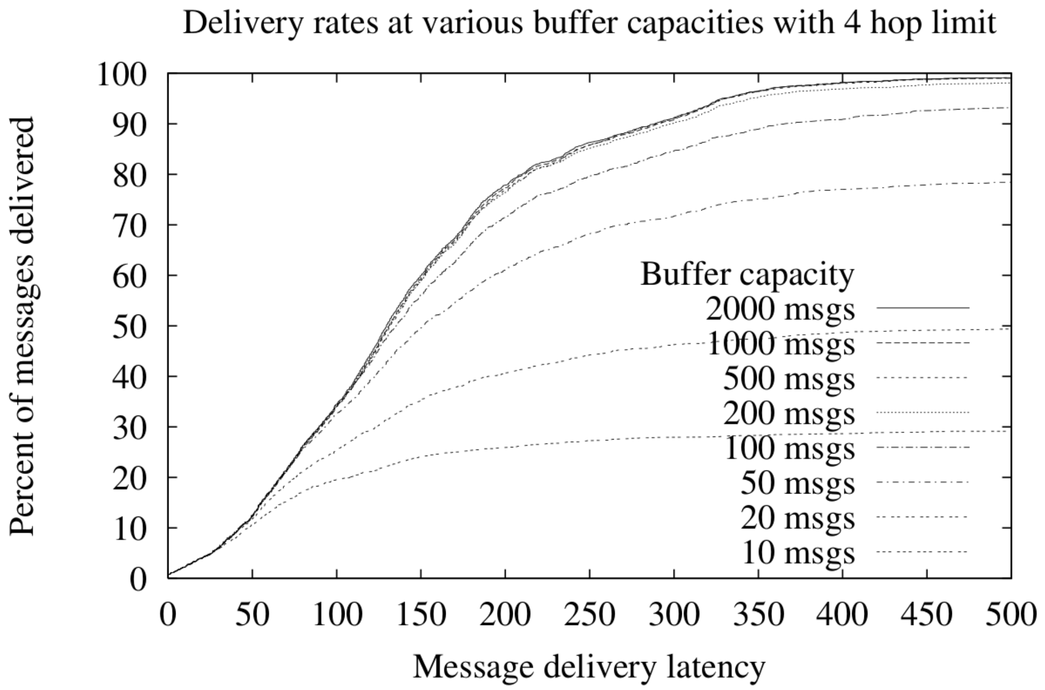


Figure 2.2.5. Becker and Vahdat's published results, showing protocol performance for varying buffer sizes with a hop count of 4[14].

hop limit. It's important to note that based on the results of the previous experiment, they selected a hop count of 4 as a reasonable medium between protocol performance and consumption of the nodes' resources. A hop count of 8, for example consumes twice as many resources as a hop count of 4, but delivers only a minor increase in performance (keep in mind that the original protocol does not have a mechanism for removing delivered messages from the network; even though a message might have been delivered, the nodes will keep sharing it with others). In this case a buffer size of 1000, roughly fifty percent of the messages in the simulation, seems like the ideal buffer size in terms of the tradeoff between performance and resource consumption. Like the first experiment, this research's results do not differ significantly than Becker and Vahdat's. Both demonstrate, as expected, that as buffer size is decreased performance of the protocol decreases as well.

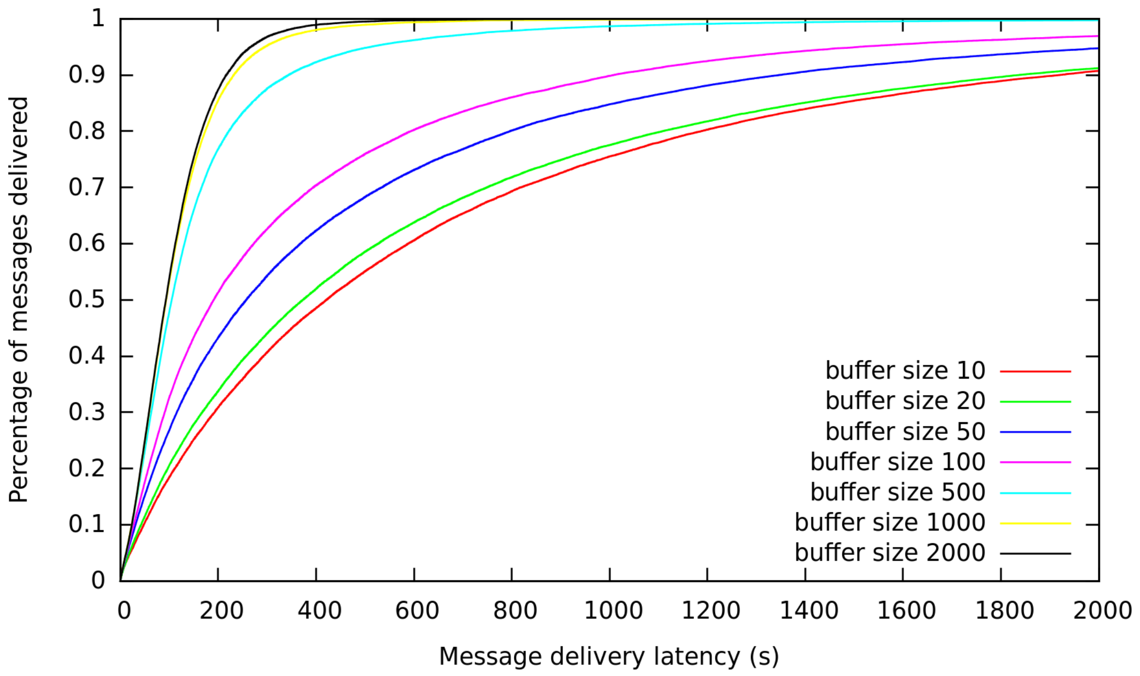


Figure 2.2.6. Recreated results, showing protocol performance for varying buffer sizes with a hop count of 4.

It should be noted that the relationship between buffer size and realistic resource constraints is not explored by Becker and Vahdat. In their simulations, they used messages sized 1 KB[14]. The largest buffer size they test is 2000 messages which requires about 35% more space then the standard 3 1/2 inch floppy disk, a comparison which they do not make. Depending on the message size, what a realistic buffer size in a real world application varies greatly. If the proposed application of distributing small text based messages is considered, it can be assumed that message size will be similar to the standard 140 byte SMS message. A thousand of these messages would be less than 14 KB. With current smart-phones having permanent storage in the range of gigabytes the difference between a buffer size of 1000 and 2000 is trivial. On the other hand, if much larger messages are considered, say the an MP3 file of size 4 MB, the difference between a buffer size of 1000 and 2000 messages becomes much more significant, 3.9 GB and 7.8 GB respectively. Such a file size, given the limitations of current wireless networking technology is of course unreasonable, but it demonstrates that for anything short of multimedia the buffer size is not a limiting factor on smart-phones.

3
Malicious Behavior

3.1 Overview

Aside from issues of performance, the most significant concerns in adapting Epidemic Routing to a real world application center around the ability of the network to resist tampering by malicious parties. Considering an application like the one in Chapter 1, malicious parties can be expected to attempt to degrade the performance of the network (completely if possible), to intercept, alter, or censor the information being transmitted, and to identify both the identities and geographic locations of nodes in the network so that they can be stopped through other means. Any such malicious behavior within the network to achieve these goals can be broken into to two basic categories: passive and active.

3.1.1 Passive Malicious Behavior

Passive malicious behavior can be defined as observation and information gathering. It might involve participating in the protocol inasmuch as would be required to collect data about the messages and the nodes that are transmitting them in that the malicious party might send/receive summary vectors, receive messages, etc, but would not actively attempt to impede the performance of the network by participating in it. The reasons for wanting this information would depend on the malicious party and the situation, but its not hard to imagine it being valuable for various political reasons. Determining the identity, location, and importance of nodes in the network would also be useful in disrupting the network by removing the node from the network with physical means.

How such a malicious party would be able to capture this information and use it to identify the location of an individual node participating in an Epidemic Routing protocol has been studied and various solutions proposed and evaluated[5, 8]. In short the approaches that could be used by a malicious party to gather this information involve gathering large amounts of data from one or more locations and then analyzing this information to extract the desired information. The proposed solutions involve making this process as difficult as possible by further obscuring information in the network. For example Lu et al. propose sending messages in parts over time to make it more difficult for malicious parties to determine exactly when, where and from whom a message originated [8].

Protecting a node's privacy is a difficult task due to the subtleties of the protocol and the nature of networks comprised of mobile nodes. The extent to which information can be extracted from these networks is in part evidenced by the Epidemic Routing based protocols which exploit this information to increase their efficiency and message delivery rates [2, 6]. These protocols are able to keep track of a node's movement patterns and the number and type of interactions they have with other nodes and use this information to determine when and to whom to send messages. While the tools used to gather this information are integrated into a protocol at the lowest level, a malicious party with enough nodes could be able to reconstruct a large amount of that information from a protocol such as the original Epidemic Routing that does not collect or use it.

Any implementation of an Epidemic Routing protocol would also have to seriously consider the issue of data security.Message confidentiality and sender identity verification are common cybersecurity issues that the original Epidemic Routing protocol and its theoretical variants ignore for the sake if simplicity. Spoofing identity or altering or censoring message contents would be trivial in most of these protocols. When one node interacts with another, it simply asks that node what it's ID (a string of some kind) is, letting the queried node respond with whatever it likes leaving the asking node with no way to verify whether or not that information is true. The same is also true for message contents. There is nothing in the current protocol to prevent a node from viewing and/or modifying the contents of a message for which it is not the recipient. A malicious node could for example accept a message from another node, read the contents, "censor" them, and then distribute the message while maintaining the appearance of authenticity. These issues are not unique to the Epidemic Routing protocol, and much research has been done on protecting data integrity and identity verification. Which techniques can be used and how well they would work is beyond the scope of this research as cybersecurity is an extremely large field of research on its own. Suffice it to say there are many different approaches to security that could be applied to the Epidemic Routing protocol to give it a reasonable level of security.

3.1.2 Active Malicious Behavior

Arguably the simplest active attack that could be made is a distributed denial of service (DDoS) type attack. Unlike the tradition DDoS attack made against web servers in which a large number of connections are made against the server repeatedly with the intention of preventing legitimate access to the server, a DDoS attack against the Epidemic Routing protocol would likely aim to flood the network with messages containing fake or malicious information. Such tampering is hard to prevent, as evidenced by the continual presence of DDoS attacks that continue to plague the Internet today. The most significant difference between the traditional DDoS attack and a flooding attack in an Epidemic Routing network is that a DDoS attack is a coordinated attack made by numerous attackers against a single target, where as a flooding attack would be a few malicious parties attacking all of the other nodes in the network. That is, it is unlikely that in an Epidemic Routing protocol there will ever be more malicious attackers than legitimate targets making it more difficult for a few malicious nodes to completely disrupt the delivery of messages within the network.

Aside from the obvious impact of reducing network efficiency by consuming unnecessary resources a flooding attack could significantly effect the network by pushing some legitimate messages out of a node's buffers, and in the worst case pushing a message out of all of the buffers in the network except for the sender's internal buffer. Again, due to the distributed nature of the Epidemic Routing protocol it does not seem likely that a malicious party could completely prevent the delivery of a message, as the sender would continue to distribute the message to other nodes. However, a malicious node would be able to significantly increase the latency of message delivery, especially for messages with lower hop counts. By pushing a message out of other nodes buffers, a malicious node is effectively able to decrease a messages hop count, as the sending node is required to more directly interact with the recipient.

The original Epidemic Routing protocol is especially susceptible to this type of attack because no mechanism currently exists for rejecting certain messages or connections from certain nodes other than hop count and neighbor time, both of which were not designed to protect the network from malicious nodes but to increase performance of the protocol in general. The neighbor time parameter could protect the network to a limited extent in that it would prevent nodes from connecting to the same malicious node with high frequency, but it does nothing to restrict the messages that are sent between the nodes once a connection is established which is the real issue, as malicious messages will be forwarded by legitimate nodes. In the case of hop count, a node can simply set a message's hop count abnormally high to ensure that it is continuously distributed across the network.

As was previously seen, if a node's buffer size is smaller than the number of messages within the network, not all messages can be delivered within a reasonable amount of time. By creating numerous messages and by setting their message counts abnormally high a malicious node can attempt to make the majority of the messages that are delivered its own. The malicious node could also accept messages for transportation but not actually pass those messages on to any other nodes so that its presence and participation in the network would have no positive impact on the delivery of legitimate messages.

3.1.3 Metrics

In the case of passive malicious behaviors it is difficult to measure their ability to gather information about the network and its participants. In the case of intercepting information the original Epidemic Routing protocol is completely susceptible, and is not worth examining until some sort of message encryption is put into place, which as previously mentioned is outside of the scope of this research. Node localization is also a very complex task, to which the Epidemic Routing protocol has already proven susceptible and extensively examined[8]. The proposed application of an urban protest type scenario also makes localization less of a threat since presumably the nodes have already self-identified themselves as prospective targets to malicious parties. There is no mystery in unraveling their location since a protest by nature draws attention to location instead of obscuring it. Because of these reasons this research elects to not measure the effects of passive malicious behavior on the Epidemic Routing protocol and to instead focus on the effects of active malicious behavior.

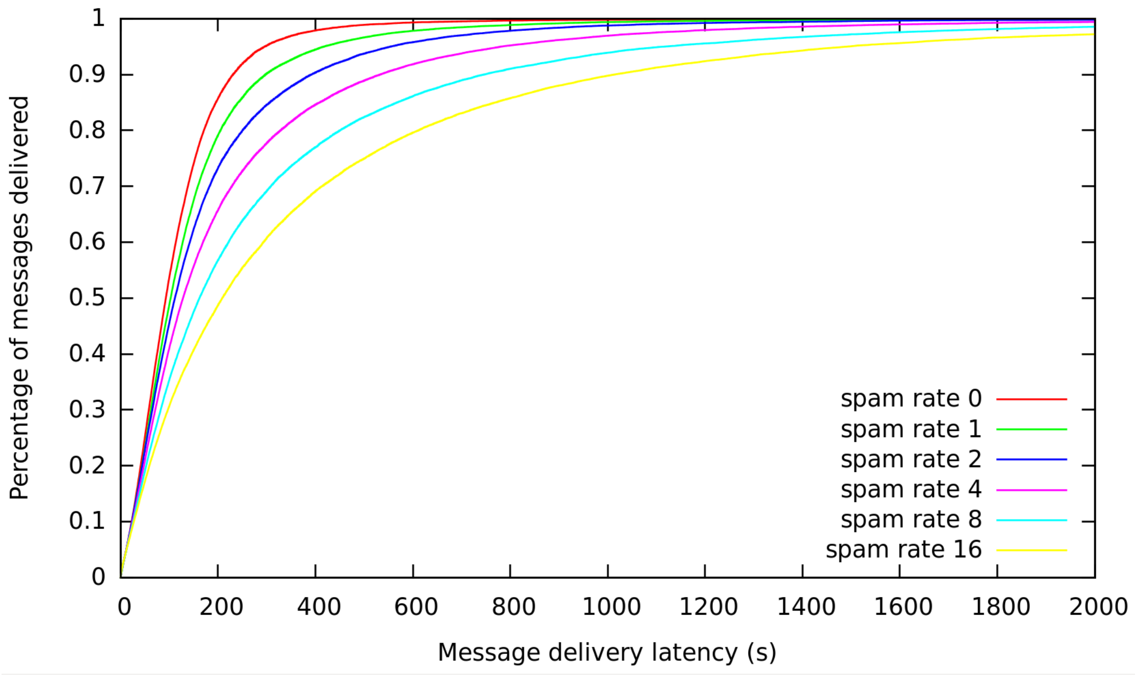
In measuring the impact a malicious node might have on the performance on an Epidemic Routing protocol, the same metrics of latency, delivery rate, and resource consumption are again used. It is also important to examine how many malicious nodes are required to have a significant impact on the performance of the network and how extreme their behaviors need to be. While it might be possible for large number of malicious nodes acting with an extreme behavior, it might not be realistic in the proposed application. Only if a relatively few number of malicious nodes acting reasonably inconspicuously can it be presumed that malicious nodes can realistically have a significant impact on the network.

A tamper resistant protocol would have to model the effect of active malicious nodes on the protocol, develop mechanisms to prevent that interference, and then evaluate the effectiveness of those mechanisms. A serious look would have to be made at the information the protocol discloses about nodes and their interactions, and how that information might be minimized. Certainly these issues are numerous, and history has certainly demonstrated that in terms of cybersecurity there are almost certainly more flaws to be exploited. It would be impossible to discover and address every security issue, but it is certainly possible to isolate what appear be the most significant security issues and related design concerns. The two issues that are the most immediately apparent are spamming (DDOS) attacks and message confidentiality/identification verification.

3.2 Malicious Hop Model

Malicious Hop was the malicious variant of the Epidemic routing protocol that was developed. It aims to decrease the performance of the network, measured in message delivery rates and average message delivery latency by altering the hop counts of messages that it sends and receives to carry and by flooding the network with spam.

This variant of the Epidemic Routing protocol has three additional parameters, the internal buffer modification factor, the external buffer modification factor, and the spam rate as measured in messages per second. The buffer modification values determine the amount by which the hop counts of messages sent from the malicious node and messages received for transportation are modified, respectively. If a negative value is used for the external buffer modification factor, than the node does not pass along external messages at all. To the sending node the message looks as if the message has been accepted for transportation, but the malicious node will never pass that message along to any other node. Without sending spam malicious nodes have no negative impact on the network since internal buffer modification factor has no impact on messages that are sent and any positive external buffer modification factor can only have a positive impact on the performance of the network. It is also important to note that like legitimate nodes, malicious nodes are not aware of each other and act individually.



Number of Malicious Nodes	Messages Delivered	Delivery Rate (%)	Average Latency (s)	Standard Deviation (s)
0	1980	100.00	5.89	116.09
1	1980	100.00	7.23	160.94
2	1979.75	99.99	8.61	212.82
4	1979.55	99.98	10.78	288.14
8	1979.15	99.96	13.64	395.86
16	1979.35	99.97	16.53	526.14

Figure 3.2.1. Protocol performance in the modified simulation with varying numbers of malicious nodes, internal and external modification factors are 4 and -1 respectively. Internal and external buffers have a size of 1000. Hop count of 4. Malicious nodes send one spam message per second.

Figure 3.2.1 shows the performance of the Epidemic Routing protocol with the presence of a varying number of malicious nodes that are using the Malicious Hop protocol and also sending a spam message once a second. The results show that even the introduction of one malicious node exhibiting this behavior can have a significant negative impact on the performance of the network by increasing the average latency. With only a couple of malicious nodes it is sometimes possible to prevent some messages from being delivered within the cut-off (2000 seconds). Given that the original message generation mechanism sends one legitimate message a second, it might seem unrealistic that each added malicious node sends a spam messages every second. But it is much easier to create fake or false information than real information. While the legitimate user might spend minutes composing a message to be delivered through the network, a malicious user can generate several thousand messages containing no real information in less than a second.

4
Simulating a Novel Scenario for Epidemic Routing

Given that this research aims to study the effectiveness of the Epidemic Routing protocol as an information dissemination technique in urban protest environments, it becomes necessary to depart from the scenario outlined by Becker and Vahdat. Firstly, the physical restraints originally used in the simulation are unrealistic if attempting to model human behavior. The original simulation uses fifty nodes moving at an average of 10m/s in a rectangular box sized 1500m by 300m. In terms of simulating human behavior in an urban environment, admittedly simple but more realistic constraints could be a significantly larger number of nodes, 100, moving within a large square of 1000m by 1000m at a much slower, and realistic, speed at an average of 2 m/s.

All of the same tests were run in the new protest type environment as run in the original. There are a couple of differences between these simulations and the originals aside from the mentions physical differences. Because the number of nodes has increased and Becker and Vahdat's original message generation scheme is used there is an increase in the number of messages generated going from 1980 to 8010. To compensate for the increase in the number of messages generated additional buffer sizes of 4000 and 1000 were tested along side those originally used. Predictably buffer sizes that worked well in the original simulation do not work as well in the modified one as seen in Figure 4.0.1.

4.1 Broadcast Messages

The proposed application of distributing information in an urban protest environment also differs from Becker and Vahdat's simulation in that in the protest environment it is more likely that people are trying to spread information to as many of their peers as possible instead of trying to send a message to a specific recipient. Using social networking services such as Twitter and Facebook as an example, the user tweets or posts information to his or her wall or to a group with the intention of it being view by as many as possible. Just as in Twitter and Facebook not all of the broadcast messages received by a user are important. In Twitter especially a few users generate messages which are the most valued as evidenced by the fact that most users have less than ten followers [10]. Somehow these users have developed a certain reputation which allows their messages to stand out from all of the rest. In other words, they are the signal to the noise. That is not to say that all users are incapable of generating important or desired information, just at a smaller rate.

To simulate this kind of behavior a different message generation mechanism has to be used. Messages are classified either as important or noise. The nodes in the simulation are divided in to two separate groups as well, important and normal. The only difference between these groups is the rate at which they send important and noise broadcast messages. Both groups send both types of broadcasts, but the important nodes send noise messages at a lower rate and the normal nodes send noise messages at a higher rate and important messages at a much lower rate.

Since the broadcast messages do not have a specified recipient message delivery is counted as the number of nodes who have received a particular broadcast message. That is, 100% message delivery is when all of the nodes other then the sender receive the message. Because the messages can be pushed out of the buffers it is possible for a node to receive a broadcast message more than once, only the first delivery is recorded.

The new broadcast message generation mechanism relies on a Poisson processes. An exponential distribution is used to generate the interval between the sending of different messages. This distribution is parametrized by the average rate of messaging generation as measured in messages per seconds. Figure 4.1.1 shows the cumulative distribution function for some of these rates. This graph shows the probability of an interval being selected for different message generation rates. What is important to note here is that the range of possible intervals between message generation varies greatly for each rate. An average message generation rate of 1/60s for example still produces intervals of several hundred seconds, albeit with much lower frequency. Before the simulation begins every node samples the exponential probability distribution for a given message generation rate which returns the number of frames before the next message is sent. The message is then put into a priority queue and sorted based on the difference between the time of its occurrence. For every frame, this priority queue is checked to see if any messages are ready for delivery. If there are any, the message is passed to the node for sending when that node's status is updated. Once a message is sent, a new message is generated in the same way. Only messages that are sent and not just generated are logged.

Unlike Becker and Vahdat's message generation mechanism, the broadcast message generation mechanism does not generate a fixed number of messages. Rather it continues to generate messages throughout the duration of the simulation. Because of the continual message generation it is impossible for the routing protocol to deliver all of the messages, explaining why in none of the simulations reach a delivery rate of 100%. While both both important and noise messages are generated and delivered, only the delivery of important messages is considered relevant. Because the messages are generated continually there is no benefit in in running the simulation multiple times and averaging the results, as was done with the simulations which used Becker and Vahdat's message generation mechanism. Instead, simulations using the broadcast message generation mechanism were run for a much longer period of time, 250,000 frames or roughly seven hours, allowing for the generation of more messages and thus more stable results. Both the original hop and buffer count tests that were used in Becker and Vahdat's simulations were used against the new protest like environment and broadcast message generation mechanism.

For the hop count experiment a buffer size of 1000 is used for both the internal and external buffers. For the buffer size experiment, a hop count of 4 is used. For these initial tests no noise messages were sent, normal nodes sent important messages at an average rate of once an hour, and important nodes sent important messages at an average of once every ten minutes. Five of the nodes were selected as being important. Increasing either of these parameters increases the number of important messages generated and has no significant impact on the performance on the protocol, so long as the number of messages generated is small enough in relation to the buffer size as to prevent messages from being pushed out of them.

As seen in Figures 4.1.2 and 4.1.3 these results are very similar to those previously seen but do have some differences. As previously mentioned, it is impossible for all of the messages to be delivered as the broadcast message generation continues to the very end of the simulation causing the visible upper bound on the percentage of messages delivered. Secondly the number of messages sent varies from simulation to simulation due to the random nature of the broadcast message generation mechanism. Predictably the percentage of messages delivered increases and and average latency decreases when the hop count and buffer size is increased.

With these baseline results the next parameter that was tested was the rate of noise broadcast generation. Unlike the important broadcast generation rate, the noise generation rate can have a significant impact on the performance of the network in that if it is sufficiently high it can force messages out of the nodes' buffers. For this test a buffer size of 5000 and a hop count of 4 was used, because the previous simulations proved them to be a reasonable tradeoff between resource consumption. For this test, the noise message generation rate for important nodes was kept constant at 1/1800s since due to their small number, varying the rate would have a relatively small impact on the performance of the network and since it seems intuitive that important nodes should always send noise broadcasts at a lower rate.

Figure 4.1.4 shows the effect varying the noise rate has on the performance of the network, and as can be seen, demonstrates that the noise rate has to be relatively high in order to have a large negative impact on the performance of the network. There are some slight anomalies in this graph, namely that the simulation run with a noise rate of 1/600 s (once every ten minutes) outperforms the simulations run with the noise rates of 1/1800s (once every 30 minutes) and 1/3600s (once every hour). Given that difference between the percentage of important broadcasts delivered between these simulations is less than once percent, this discrepancy is likely due to the random nature of the simulation. However, it is possible that something much more complex is occurring in which the noise broadcast generation rate is syncing up with some other aspect of the simulation, such as the important broadcast rate or the motion model, to create better results. The larger noise rates had the predictable negative impact on the performance of the network, but arguably less than would be expected. Even with normal nodes sending noise messages on the average of once a second, more than 60% of the messages are delivered, though with a fairly large delivery latency. Such a high rate of noise broadcast generation seems a little unreasonable; these messages aren't intended to be like the spam messages used in the Malicious Hop protocol, rather to simulate the noise that exists in current social media networks.

Given these results and the proposed application of an urban protest type environment, a noise rate of 1/60s seems like a reasonable compromise between trying to realistically simulate user behavior and exploring the weaknesses of the protocol. It does not seem unreasonable for a user to send a noise message on the average of once a minute, and such a noise rate has an apparent impact on the performance of the protocol. As previously stated, this negative impact is closely tied to the buffer size. The more often important messages are pushed out of a buffer, the more direct the contact between the sender and the recipient needs to be, increasing the latency of message delivery and decreasing the percentage of messages delivered overall.

As seen in Figure 4.1.5, with a buffer size of 4000, and no noise broadcasts, the Epidemic Routing protocol was able to deliver 97.63% of the messages with an average latency of 608.15 seconds. When the noise rate is set to 1/60s the same buffer size achieves a message delivery rate of 86.81% with an average latency of 1421.76 seconds. Even though The percentage of total messages delivered drops a little under 11%, the average latency more than doubles when the noise broadcasts are introduced into the simulation, going from about ten minutes to about twenty three. While this certainly is a significant change in the performance of the network it is important to keep these results in perspective. 879 important messages were delivered at a rate of close to 87% despite the fact that almost 40,000 noise broadcasts were sent. If the buffer size is increased to hold 1000 messages, nearly 92% of the messages are delivered and the average latencies are brought down much closer to what they were before the introduction of the noise rate.

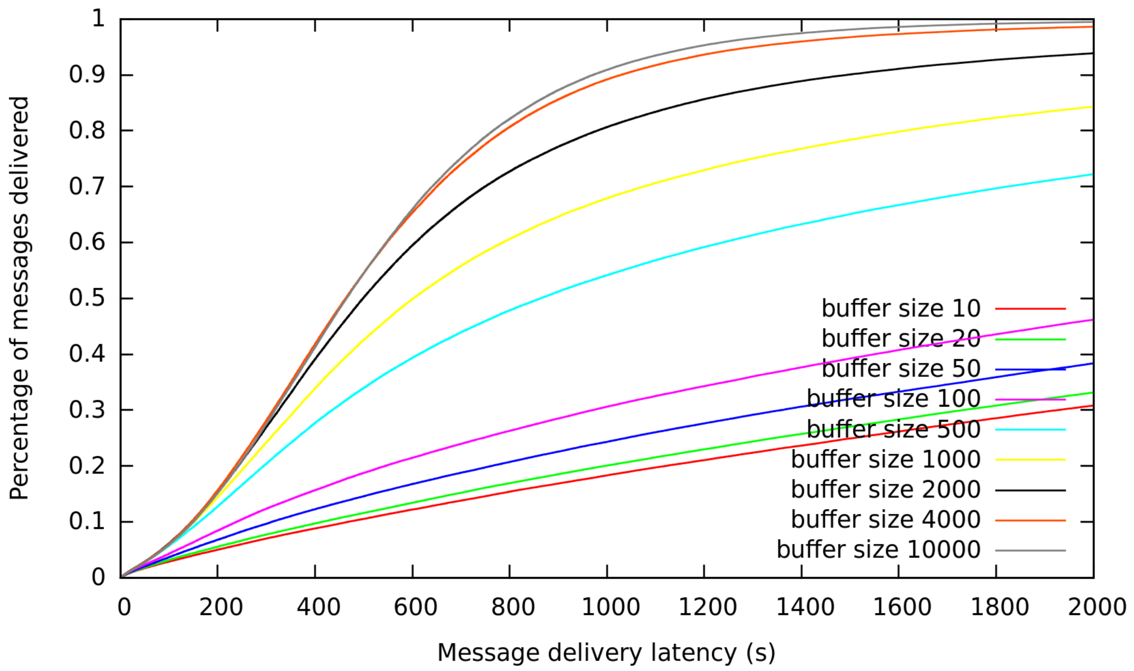
Again selecting a buffer size of 5000 as a reasonable compromise between network performance and resource consumption, the effect of the hop count on the network performance was again examined. Figure **??** shows that as seen in the previous experiments increasing the hop count increases the percentage of messages delivered and decreases the average latency of message delivery. As also seen the in previous experiments, there is a point at which increasing the hop count produces diminishing returns. Doubling the hop count from 4 to 8 increases the percentage of message delivery by a little more than a point and decreases the average latency by about 3 minutes.

4.2 Malicious Hop Redux

Thus far the original Epidemic Routing protocol has proven itself effective in delivering important messages in a different physical environment, with a different messaging paradigm, against a relatively large noise rate. It's ability to deliver the message broadcasts at a reasonable percentage of delivery and average latency in spite of limited resources and fairly significant level of noise demonstrates its ability to resist what can be thought of malignant behavior. Noise broadcasts are not thought of being generated and sent with the intention of negatively impacting the performance of the network, but rather a necessary but detrimental behavior that is the product of its application. Twitter does not try to delivery only the tweets it thinks are important.

That being said there is still the possibility of a malicious behavior existing in this modified simulation. Modifying the hop counts of both messages sent and carried had proved to have little negative impact on the performance of the network in the original simulation, but sending out large quantities of spam did. The effect of this behavior is somewhat mitigated by the fact that it seems unreasonable for there ever to be a large number of malicious nodes interacting with the legitimate nodes in this type of environment for an extended amount of time. Again making a compromise between simulating realistic behavior and exploring the weaknesses of the protocol the number of additional malicious nodes was selected to be 5.

Figure 4.2.1 shows that even with extremely high spam rates, the Epidemic Routing protocol fairs pretty well. Even when malicious nodes are generating spam messages on an average of every tenth of a second 84.43% of the important broadcast messages are delivered and the average latency is not dramatically increased. Compared to the results of varying the noise generation rate, the effect of the malicious nodes seems negligible. This is largely caused by the fact that there are so few malicious nodes in comparison to the legitimate ones. Regardless of how fast spam broadcasts are sent, malicious nodes still need to compete with the noise from the legitimate nodes for buffer space, and the malicious nodes are at a disadvantage because their messages are more concentrated. While a malicious node might be able to completely wipe out the buffers of the nodes in their nearby vicinity there are still numerous nodes in the network which still retain useful information who are waiting to replace that spam with broadcasts both important and noise alike. This can be seen in this simple comparison: In the instance where malicious nodes are generating spam every tenth of a second on average, normal nodes are generating noise every minute on average, 790481 spam broadcasts were sent, 39574 noise broadcasts were sent, and 845 important broadcasts were sent. The important broadcasts reached 85% of the network.



Buffer Size	Messages Sent	Messages Delivered	Delivery Rate (%)	Average Latency (s)	Standard Deviation (s)
10000	800	77413	97.74	655.73	159.06
4000	895	86507	97.63	608.15	144.86
2000	845	79747	95.33	681.70	168.58
1000	879	85042	97.73	664.68	164.89
500	851	81280	96.48	672.21	179.86
200	888	80715	91.81	979.13	353.17
100	845	74942	89.58	1249.22	467.38
50	873	69768	80.72	1834.75	701.91
20	905	68948	76.96	2309.11	828.43
10	888	67753	77.07	2419.45	853.25

Figure 4.0.1. Protocol performance in the modified simulation with varying buffer sizes. Hop count of 4.

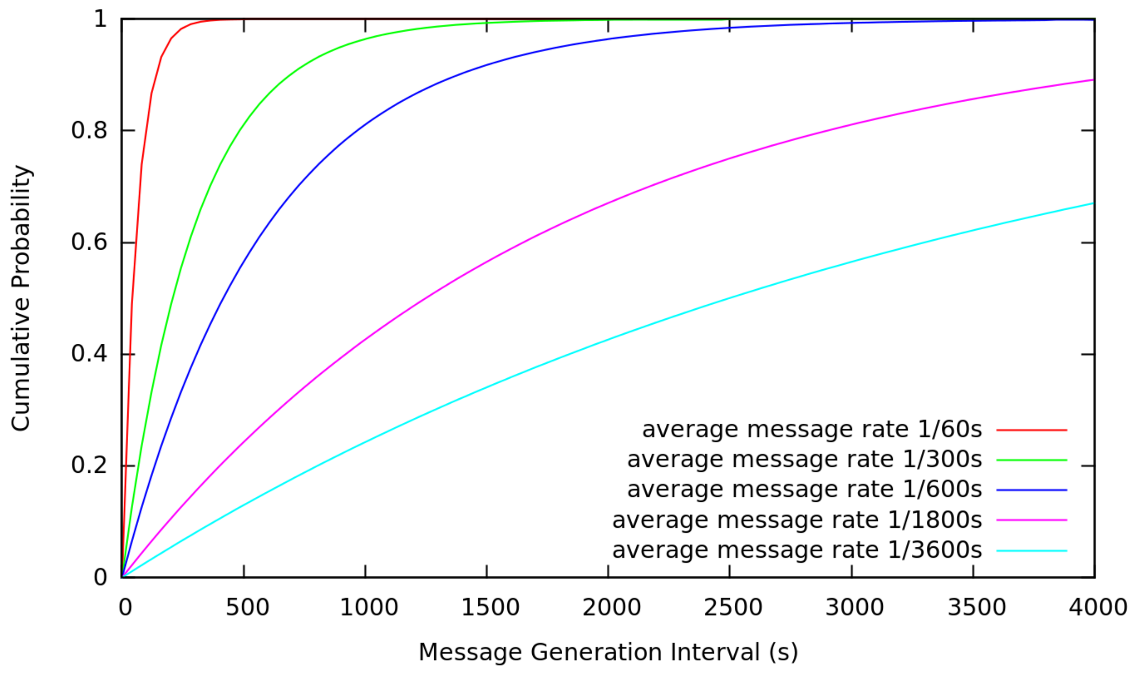
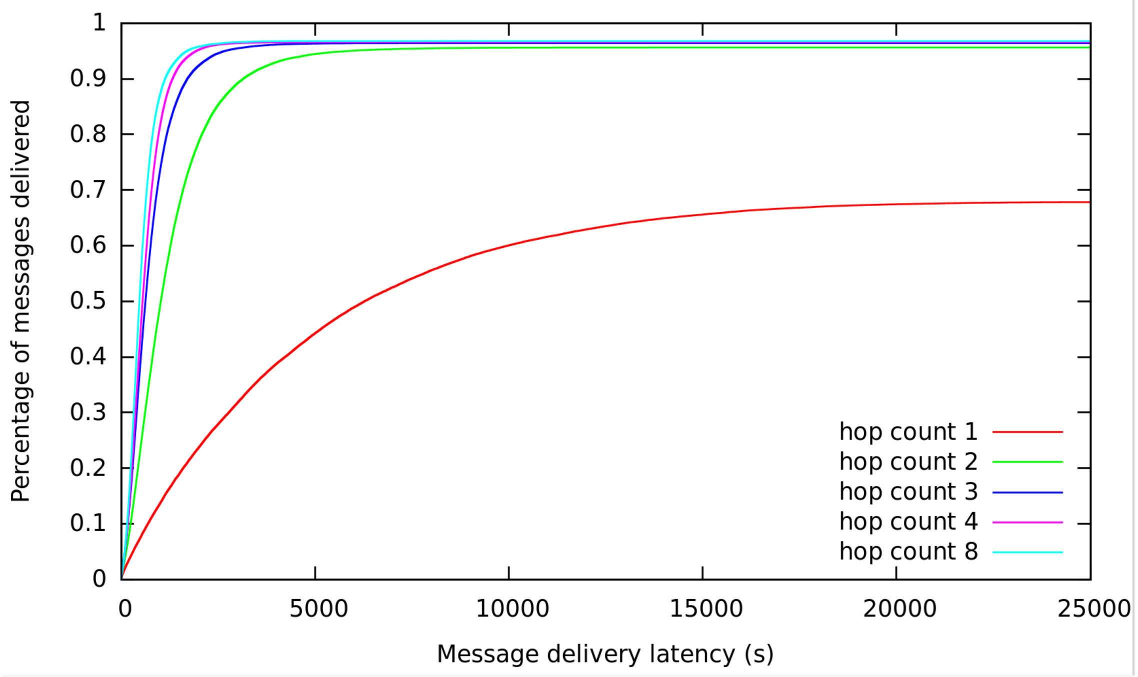
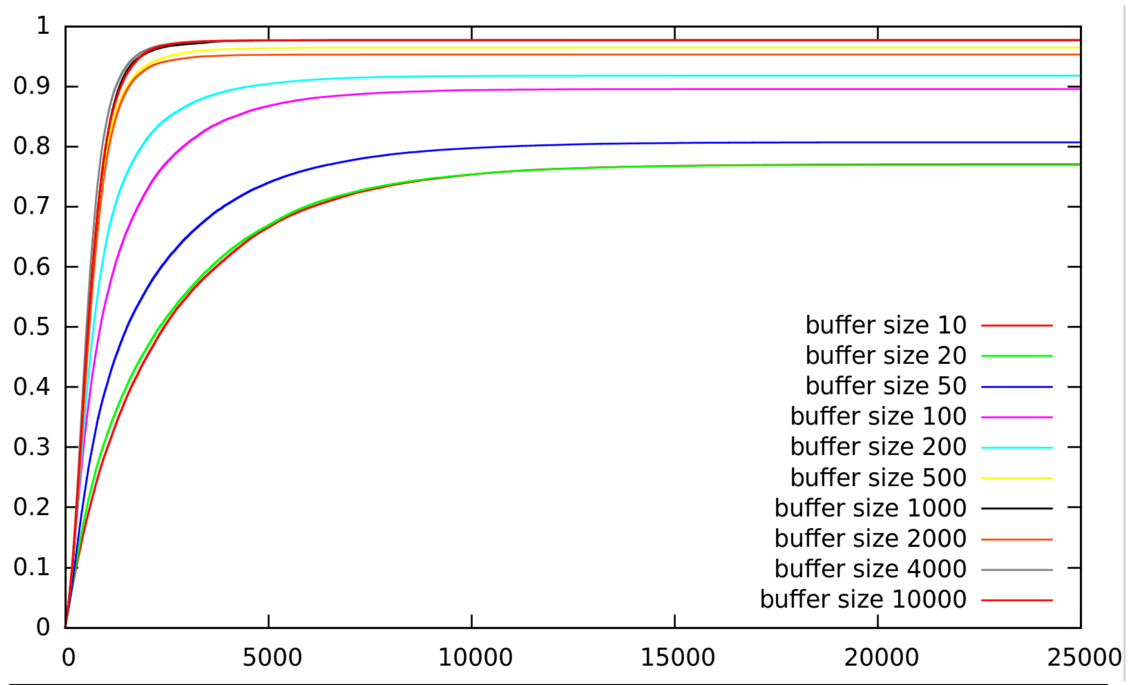


Figure 4.1.1. The cumulative distribution function for various average message generation rates. This graph shows the probability that a particular message generation interval will be selected for a particular average message generation rate.



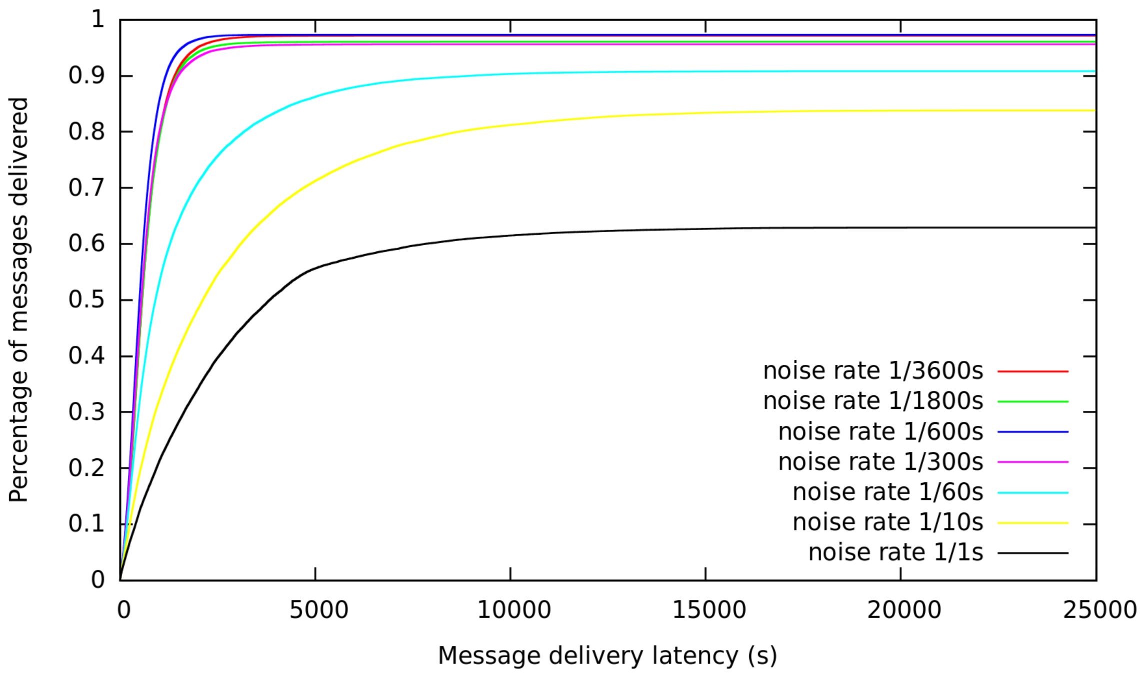
Hop Count	Messages Sent	Messages Delivered	Delivery Rate (%)	Average Latency (s)	Standard Deviation (s)
8	876	83930	96.78	537.43	127.84
4	960	91792	96.58	621.73	142.68
3	853	81419	96.41	743.99	194.69
2	882	83502	95.63	1250.62	350.89
1	866	58153	67.83	4547.92	1429.67

Figure 4.1.2. Protocol performance in the modified simulation with the broadcast message generation technique for various hop counts. 5 important nodes and 100 normal nodes were used, with average important message sending rates of once every ten minutes and once every hour respectively.



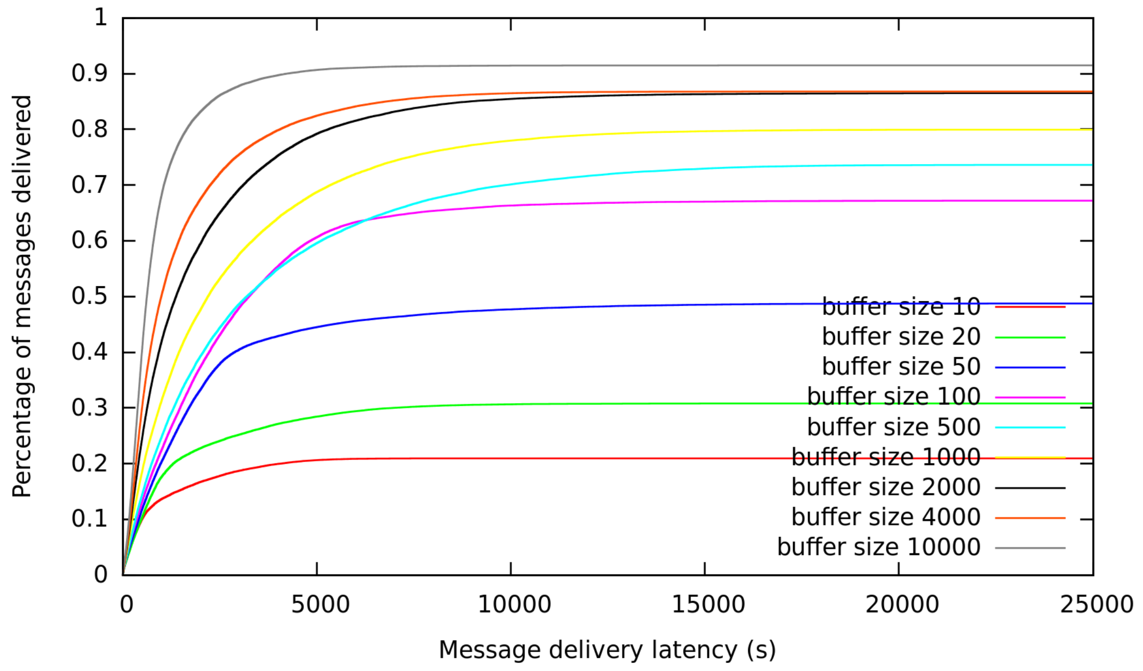
Buffer Size	Messages Sent	Messages Delivered	Delivery Rate (%)	Average Latency (s)	Standard Deviation (s)
10000	800	77413	97.74	655.73	159.06
4000	895	86507	97.63	608.15	144.86
2000	845	79747	95.33	681.70	168.58
1000	879	85042	97.73	664.68	164.89
500	851	81280	96.48	672.21	179.86
200	888	80715	91.81	979.13	353.17
100	845	74942	89.58	1249.22	467.38
50	873	69768	80.72	1834.75	701.91
20	905	68948	76.96	2309.11	828.43
10	888	67753	77.07	2419.45	853.25

Figure 4.1.3. Protocol performance in the modified simulation with the broadcast message generation technique for various buffer sizes. 5 important nodes and 100 normal nodes were used, with average important message sending rates of once every ten minutes and once every hour respectively.



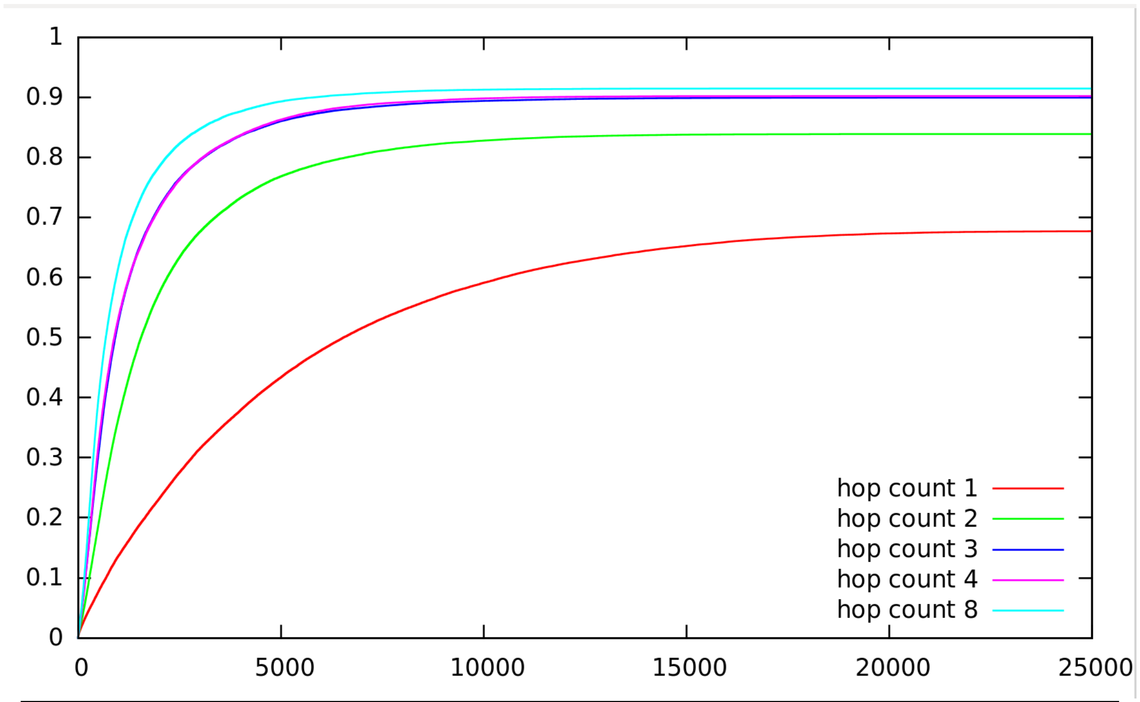
Noise Rate (Messages/Seconds)	Messages Sent	Messages Delivered	Delivery Rate (%)	Average Latency (s)	Standard Deviation (s)
1/3600	860	82725	97.16	661.16	157.40
1/1800	882	83891	96.08	652.30	156.86
1/600	892	85936	97.31	564.86	127.08
1/300	873	82654	95.63	638.78	164.37
1/60	871	78312	90.82	1423.69	562.26
1/10	854	70857	83.81	2531.94	910.65
1/1	728	45362	62.94	2478.76	862.25

Figure 4.1.4. Protocol performance with varying noise rates, buffer size 5000, hop count 4. 5 important nodes and 100 normal nodes with used, with average important message sending rates of once every ten minutes and once every hour respectively.



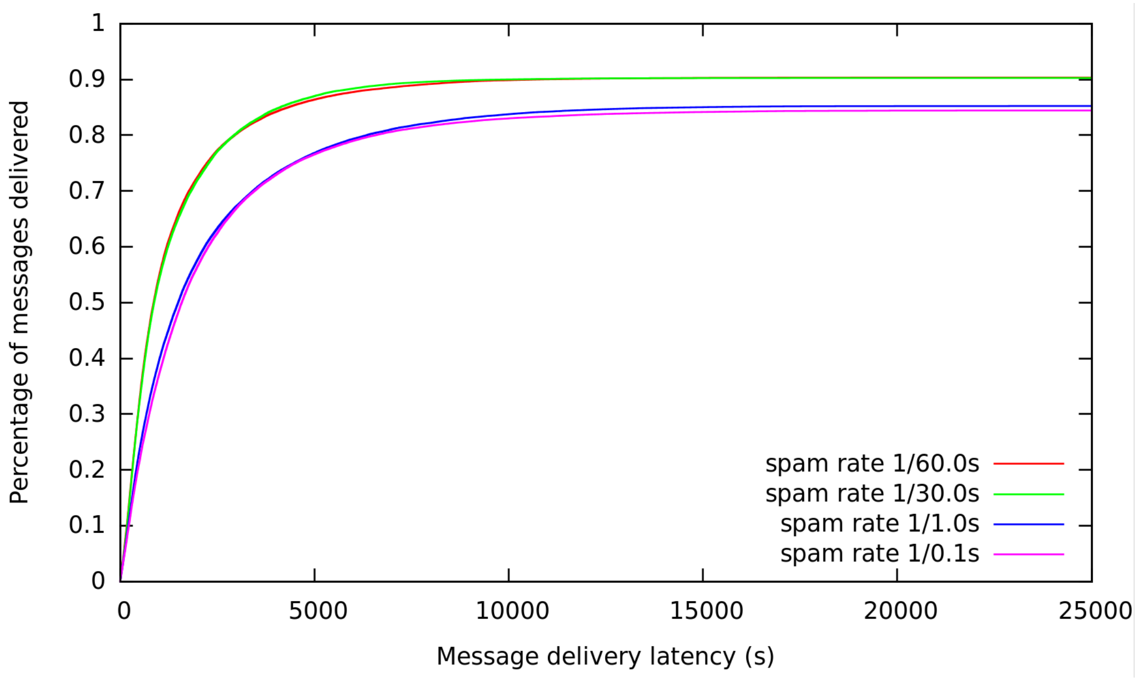
Noise Rate (Messages/Seconds)	Messages Sent	Messages Delivered	Delivery Rate (%)	Average Latency (s)	Standard Deviation (s)
1/3600	860	82725	97.16	661.16	157.40
1/1800	882	83891	96.08	652.30	156.86
1/600	892	85936	97.31	564.86	127.08
1/300	873	82654	95.63	638.78	164.37
1/60	871	78312	90.82	1423.69	562.26
1/10	854	70857	83.81	2531.94	910.65
1/1	728	45362	62.94	2478.76	862.25

Figure 4.1.5. Protocol performance with a noise rate of 1/60s, hop count 4, for various buffer sizes.



Hop Count	Messages Sent	Messages Delivered	Delivery Rate (%)	Average Latency (s)	Standard Deviation (s)
8	873	79058	91.47	1066.76	431.48
4	864	77174	90.22	1367.06	533.44
3	876	78012	89.95	1377.28	551.32
2	888	73755	83.90	1934.90	698.60
1	820	54965	67.71	4684.16	1468.38

Figure 4.1.6. Protocol performance in the modified simulation with the broadcast a broadcast noise rate of 1/60s for various hop counts. Buffer size 5000.



Spam Rate (Messages/Seconds)	Messages Sent	Messages Delivered	Delivery Rate (%)	Average Latency (s)	Standard Deviation (s)
1/60	856	76562	90.35	1336.10	546.93
1/30	843	75329	90.26	1300.46	490.59
1/1	844	71220	85.24	1986.31	765.40
1/.1	845	70627	84.43	1998.58	756.20

Figure 4.2.1. Protocol performance with a noise rate of 1/60s, buffer size 5000, hop count 4, for various spam rates for the 5 additional malicious nodes

5
Discussion

5.1 Limitations of the NetSim simulator

Despite the best of intentions, the NetSim simulator is admittedly simplistic. While certain aspects of the simulation are more realistic than the simulations run by Becker and Vahdat for the proposed application, there is one area in which the Monarch simulator which they used is undeniably more realistic. As previously mentioned the simulations Becker and Vahdat ran data between nodes was transmitted through a simulated 802.11 connection. While their original research does not go into any detail as to what this means in terms of transmission speed, latency, or time required for connection establishment, it is a restriction that would exist for any real world application which the NetSim simulator does not take into account. Aside from the desire to maintain simplicity, this was done because there isn't currently any wireless data transmission technology that could realistically be used in an Epidemic Routing type protocol as outlined below.

Another aspect of the simulation which has a significant impact on the performance of the protocol is the model used to generate the motion of the nodes. This model, known as a mobility model, should in theory accurately model the realistic motion of the agents of the proposed application. Becker and Vahdat's research is largely a proof of concept, and as such there isn't a clear behavior that should be modeled. In their original simulation, they use a mobility model commonly used in the simulation of MANET's called random way-point [1]. The random way-point mobility model is fairly simple, a node randomly selects a location within the confines of the simulation, and then moves towards it at a random velocity. When the node comes within a certain distance of the way-point, it stops for a random amount of time, and the process is repeated. This mobility model is good testing for Epidemic Routing as a proof of concept because it effectively simulates a "random" behavior, but in testing the effectiveness of the protocol in real world applications a more realistic mobility model would produce more accurate results.

The most straightforward approach would be to use data from a real world situation, in other words, the problem of simulating how people move in an urban environment during a protest type situation is solved if there is data taken from people moving in a urban environment in a protest type situation. Although there are a couple of data sets that have been collected for the purposes of studying MANET's, both of them are based off of academic type environments, such as conferences or student interactions on a college campus [8]. These data sets would be more accurate than the random way-point motion model for studying the effectiveness of Epidemic Routing networking protocols, but they do not necessarily apply to the proposed application, namely in that the collected data sets describe the behavior of a few hundred people in a reasonably large geographic area over the span of at least a few days. For the proposed application it is more interesting to see how a larger number of people interact in a more confined space over a shorter amount of time, for example, can one individual share information the entire participating population within the course of a few hours.

While there are many more sophisticated mobility models that could be implemented in such a way as to more accurately simulate user behavior in this type of environment, the purpose of this study, like Becker and Vahdat's, is not to determine whether or not an Epidemic Routing protocol is practically feasible for the proposed application, because currently it is not, but to determine whether or not an Epidemic Routing type protocol is theoretically suited for this type of application. Given the complexities and uncertainties of more advanced mobility models and the theoretical nature of this research it is considered sufficient to have adapted the original random way-point model to a larger space with a more realistic average speed.

5.2 Practical Considerations

Existing research into Epidemic Routing has focused on the currently widespread technologies of Bluetooth and Wi-Fi [11–14]. Though these technologies have proved useful in many applications, they are realistically unable to support the real world application of an Epidemic Routing type protocol. The Bluetooth V3 specifications, for example, lists a maximum connection establishment time of six seconds. The alternative of creating an ad-hoc 802.11 wireless connection is a complex process which requires user input and can hardly establish a connection in less time. Existing research into Epidemic Protocols has not taken such connection establishment times into account for the simple reason that they make such research infeasible. Becker and Vahdat for example report an average message delivery latency of .2 seconds under some protocol parameters in their simulation [14], a feat clearly not currently attainable in a real-world application.

There are a couple of upcoming technologies that look promising in terms of a successful Epidemic Routing protocol application however. The Near Field Communication (NFC) protocol claims to have a connection setup time of less that a tenth of a second, and Bluetooth V4's Bluetooth Low Energy (BLE) protocol claims a connection setup time of less that three milliseconds. Both of these protocols have a relatively short communication radius (.2m and 50m respectively) and low throughput (424kbit/s and 200kbit/s respectively), but could conceivably be used with Epidemic Routing in an effective way in the right type of application. If it were possible to make connections at these speeds and at these distances without direct user interaction it might be possible to create an Epidemic Routing based protocol centered around urban protest environments where there are frequent and rapid interactions at close ranges. Though the throughputs of these technologies are relatively low, they would be sufficient to support the transmission of text based communications. The 140 character Twitter or SMS length message, which has recently proven itself extremely popular, if not useful, for example, could conceivably be sent in under a second including the connection overhead. NFC's connection radius seems absurdly small, but consider how often someone is within 20 cm of someone else for more than a second, especially in a crowded urban environment. Both of these technologies also promise to be extremely power efficient, BLE claiming to use as little as a hundredth of the power used by Bluetooth v3.

The protocol itself is not very CPU intensive, and significant advances have been made recently in decreasing the power consumption of CPU's, especially in mobile devices. Solid state storage has also become increasingly large and inexpensive, replacing the hard disk in most mobile or embedded applications, which also provides significant advantages in terms of decreasing power consumption. What remains energy expensive is data transmission. Constantly using the wireless radio when no nodes or maybe even a few nodes are within transmission range is not currently realistic in terms of energy consumption. Attempts have been made to alter the Epidemic Routing protocol so that is it is more aware of this energy issue. Yong et al. for example outline an energy conscious variant of the Epidemic Routing protocol that only transmits information to nodes when there are more then a set number of nodes in the host's communication radius, their logic being that the wireless radio should only be used when there is a minimum benefit in doing so[15]. It is unclear what wireless technology they envision being used with the protocol, but traditional wireless technologies such as Bluetooth and Wi-Fi still use a considerable amount of energy even on detecting if there are devices nearby. Turning the wireless radio on and detecting nearby devices incurs a significant energy cost for both of these protocols, a cost which has been minimized, but not realistically considered.

Due to the increasing computational power, decreasing price and increasing ubiquity of consumer level smart-phones, they have been considered by many to be an ideal platform for an Epidemic Routing type protocol. The Haggle project was the most notable attempt to bring Epidemic Routing to the real world through an Epidemic Routing protocol using smart-phones as one of its core platforms[11–13]. The project it seems has lost steam, the latest work bringing a sometimes functional demonstration of the protocol through an image sharing application to the Android Market. The lack of a suitable wireless technology and the breadth of its goal seem to have hindered the development of Haggle into a practical application. In its defense, Haggle was designed to be independent of the wireless technology used and should be adaptable to new wireless communication technologies. Integrating the recently developed Near Field Communication or Bluetooth Low Energy wireless communication protocols into the Haggle networking stack, however, would be a monstrous task, owing largely to its near complete lack of documentation and the size of its codebase. These new technologies aren't really suited for Haggle's goal of being a viable replacement of TCP/IP either, given the high bandwidth of current uses of the protocol, and the low bandwidths supported by even the newest technologies. Eventhough Haggle has been designed to be independent of the wireless radio technology that actually transmits data, its real world application have suffered from the combined inadequacy of the current technologies and its overambitious goals.

5.3 Unresolved Issues

NFC or Bluetooth v4 might make a real world application of an Epidemic Routing type protocol feasible, but the issues of performance, reliability, and security raised by Vahdat and Becker remain. The original Epidemic Routing protocol and all of its published variations have aimed for high performance, have evaluated different protocol parameters, and have studied various trade offs such as buffer size versus message delivery latency. Significant advances have certainly been made in terms of optimizing the protocol, but many of these optimizations have significant impacts on other aspects of the protocol. For example, adaptations of the protocol that take into account a node's pattern of movement to increase the number of messages delivered have been researched, but such adaptations might have significant impacts on the privacy of the node[2, 6].

Solutions proposed to protect the privacy of the node include the use of cryptography, breaking messages into segments, and selectively sending segments over time and physical differences. While they provide significant benefits in terms of security, they also have significant costs in terms of performance and reliability. Cryptographic functions are CPU intensive and again raise the issue of energy consumption on a mobile platform. Selectively sending messages or fragmenting them necessarily decreases reliability since the probability of a message being delivered in full decreases[5, 8].

At some point a trade-off must be struck between protocol performance, node privacy, security, and resource consumption. What this tradeoff is depends heavily on the proposed application and platform of the protocol. Up until this point most of the research done in the area has been theoretical, in part because a feasible platform does not exist. The platform that meets the network connectivity requirements might end up having twice the CPU or half the power consumption of today's mobile devices so it is foolish to attempt put concrete restraints on the protocol.

That being said, a more comprehensive evaluation of how exactly an Epidemic Routing protocol could be used and how effective it would be for a specific real world application and what its general requirements need to be is still useful in steering the development of further research and practical applications. Vahdat and Becker outlined an ingenious idea, but in an intentionally abstract implementation. The applications of the Epidemic Routing protocol are limitless, but it needs its own "killer app" to drive further development and real world adaptation.

5.4 Epidemic Routing Protocol in the Urban Protest Environment

Despite all of the technological limitations that currently prevent the Epidemic Routing protocol from being implemented in a real world application, on a more theoretical level it appears to be very well suited for the proposed application. The lowest message delivery rate seen was well above 50% which is tolerable considering it replaces a situation in which there is a 0% delivery rate. The limiting factor in these instances is buffer size, and as explained above, by the time a sufficient wireless technology exists the storage available on these devices is certain to increase. It might not be unheard of for these devices to store a million SMS sized messages and to sort through them efficiently.

Vahdat and Becker's originally proposed application sends messages between two specific nodes in the network, with a lot of the nodes receiving that information in between. But when the paradigm is shifted to broadcast delivery, the protocol becomes much more efficient because where every time a node accepted a message to be carried, it accepts a broadcast for delivery. This paradigm also suits the Epidemic Routing protocol because it essentially becomes impossible for the protocol to completely fail in delivering a broadcast. Even if it doesn't deliver the broadcast to all of the nodes, it delivers the broadcast to at least some of them. Where Vahdat and Becker talk about a high probability that a message is delivered to its recipient, one can talk about the high probability that almost everyone in the network receives a broadcast or the probability a certain amount of nodes receive the broadcast within a certain amount of time.

There might be a day where these kind of networking protocols replace the deeply entrenched standard of TCP/IP, and the individual might, as the Haggle project envisions, be able to send an e-mail to a coworker or send a picture to a friend through an Epidemic Routing type protocol. That future is a long ways away, but that doesn't mean there is no place for the Epidemic Routing protocol in the meantime. Assuming that the technology did exist, at least to the extent that it supports the proposed application of delivering small text based messages, the question then becomes how resistant is the protocol to tampering. As previously mentioned, nothing it seems in the world of technology is tamper resistant. A lot of these cybersecurity issues such as message integrity and identity verification are not specific to the Epidemic Routing protocol, and discussing the ways in which they might be addressed is as theoretical as is discussing what kind of wireless protocol might support such a network. There are, however, tampering issues specific to Epidemic Routing type protocols which in this research have examined.

The Epidemic Routing protocol proved surprisingly resilient, but that's not to say it is flawless. The type of attack most closely examined was a variant of the DDoS attack. The examined variant of the DDoS attack is not very distributed at all, since in relation to the attackers, the targets are much more numerous. There are two paradigms at play here, and it is likely that the attacks that work well against one won't work well against the other. That is, because the Epidemic Routing protocol is not extremely susceptible to a denial of service attack does not mean that it is resilient to all attacks. In the same way that the original architects of the Internet would have a hard time envisioning the cybersecurity issues that plague it today, its hard to look forward and envision what type of attacks might bring an Epidemic Routing network to its knees.

But that is another area in which the proposed application excels. If anything can be taken from the spontaneous, grass-roots protests which inspired this research its that the masses are surprisingly clever and that entrenched bureaucracies have a slow moving and dim-witted understanding of technology. When the BART officers disabled cellphone service, they might have been successfully thwarted a protest but the resulting national coverage certainly drew more attention to the protesters' cause than their protest ever would have. When the Egyptian government severed their country's Internet connection, they might have prevented Egyptians from communicating with one another, but their actions drew such a level of international scrutiny that its hard to imagine them foreseeing it. The Internet has developed a term for this phenomenon, its called the Streisand effect, lovingly named after the media firestorm that resulted from Barbara Striesand's attempt to remove relatively unknown pictures of her residence from the Internet[9]. Because of the numerous forms of communication that technology provides us, when the few attempt to censor a bit of information the masses are able to react by making it more popular. No existing implementable technology is centered around this phenomenon more than Epidemic Routing, because no existing technology is so decentralized and user dependent.

In designing the tests to simulate the distribution of broadcasts through an urban protest environment this research was careful to make the protocol ignorant of the importance of the messages. The protocol's purpose is not to determine which information is important; the protocol's purpose is to transmit information. Important, noise, and spam broadcasts alike are all treated the same way, as it should. To do otherwise would be integrating a censorship mechanism into the protocol itself. In spite of all of the noise, malicious or not, the Epidemic Routing protocol is surprisingly effective in distributing the few important broadcasts that were sent. Five important nodes sending important messages once every ten minutes were able to distribute that information to more than half of the network with an average latency of about 30 minutes in spite of 95 normal nodes sending noise broadcasts once a minute, and 5 malicious nodes sending 10 spam broadcasts a second. What hasn't been considered is the intent of these "important" messages. Up until now it has been assumed that "malicious" parties would attempt to to disrupt the performance of the network. It's just as possible, and likely, that they would use the Epidemic Routing protocol to transmit their own malicious information; the protocol is just as useful to the peaceful protester as the terrorist. Epidemic Routing, like all technologies, can be used for bad and good. It is not up to the technology to determine the morality of its use; that responsibility is left for its users. Just as the rise of the Internet raised a myriad of ethical and legal questions which modern society is just now beginning to comprehensively deal with, a successful adaptation of an Epidemic Routing protocol will certainly create its own issues. For better or worse, Epidemic Routing truly does spread information like a virus.

Bibliography

[1] Fan Bai and Ahmed Helmy, *Chapter 1 A SURVEY OF MOBILITY MODELS in Wireless Adhoc Networks*, 2006.

[2] Murat Ali Bayir and Murat Demirbas, *PRO: A Profile-Based Routing Protocol for Pocket Switched Networks*, GLOBECOM, 2010, pp. 1–5.

[3] Michael Cabanatuan, *BART cell phone shutdown rules adopted*, SFGate (2011December).

[4] A. Dunn, *THE ARAB SPRING: REVOLUTION AND SHIFTING GEOPOLITICS: Unplugging a Nation: State Media Strategy During Egypt's January 25 Uprising*, Fletcher F. World Aff. **35** (2011), 15–25.

[5] Xiaoyan Hong, Jiejun Kong, and Mario Gerla, *Mobility changes anonymity: new passive threats in mobile ad hoc networks*, Wireless Communications and Mobile Computing **6** (2006), 281–293.

[6] Pan Hui, Jon Crowcroft, and Eiko Yoneki, *Bubble rap: social-based forwarding in delay tolerant networks*, MobiHoc, 2008, pp. 241–250.

[7] Richard La, *An Analytical Model of Epidemic Routing with Immunity for Disruption Tolerant Networks*, University of Maryland, 2010.

[8] Xiaofeng Lu, Pan Hui, Don Towsley, Juahua Pu, and Zhang Xiong, *Anti-localization anonymous routing for Delay Tolerant Network*, Computer Networks **54** (2010), no. 11, 1899–1910.

[9] Mike Masnick, *Since When Is It Illegal To Just Mention A Trademark Online?*, TechDirt (January 2005).

[10] Erick Schonfeld, *On Twitter, Most People Are Sheep: 80 Percent Of Accounts Have Fewer Than 10 Followers*, Tech Crunch (June 2009).

[11] James Scott, Jon Crowcroft, Pan Hui, and Christophe Diot, *Haggle: a Networking Architecture Designed Around Mobile Users*, HAL - CCSd - CNRS, 2006.

[12] Jing Su, James Scott, Pan Hui, Jon Crowcroft, Eyal de Lara, Christophe Diot, Ashvin Goel, Menghow Lim, and Eben Upton, *Haggle: Seamless Networking for Mobile Applications*, UbiComp, 2007, pp. 391–408.

[13] Jing Su, James Scott, Pan Hui, Eben Upton, Meng How Lim, Christophe Diot, Jon Crowcroft, Ashvin Goel, and Eyal de Lara, *Haggle: Clean-slate networking for mobile devices*, Technical Report UCAM-CL-TR-680, University of Cambridge, Computer Laboratory, 2007.

[14] Amin Vahdat and David Becker, *Epidemic Routing for Partially-Connected Ad Hoc Networks*, 2000 (en).

[15] Li Yong, Jiang Yurong, Jin Depeng, Su Li, Zeng Lieguang, and Wu Dapeng (Oliver), *Energy-Efficient Optimal Opportunistic Forwarding for Delay-Tolerant Networks.*, IEEE Transactions on Vehicular Technology **59** (2010), no. 9, 4500–4512.