Assessing the Vulnerabilities of Reliable Multicast Protocols

by

Lara M. Karbiner

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degrees of

Bachelor of Science in Computer Science and Engineering

and

Master of Engineering in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 1998

Author .....
Department of Electrical Engineering and Computer Science
May 22, 1998

Certified by .....
Dr. Clifford J. Weinstein
Group Leader, Information Systems Technology Group
Thesis Supervisor

Certified by .....
Dr. Thomas M. Parks
Technical Staff, Information Systems Technology Group
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Assessing the Vulnerabilities of Reliable Multicast Protocols

by

Lara M. Karbiner

Submitted to the Department of Electrical Engineering and Computer Science
on May 22, 1998, in partial fulfillment of the
requirements for the degrees of
Bachelor of Science in Computer Science and Engineering
and
Master of Engineering in Computer Science and Engineering

Abstract

In this thesis, I examine a protocol that is designed to provide reliable IP multicast. I evaluate the efficiency and reliability of the protocol in a friendly environment and in the face of several different types of malicious attacks. These attacks include general attacks on the network, which hamper any type of communication, as well as attacks aimed at specific weaknesses of the protocol. The protocol was evaluated to see how reliability and efficiency degrade in the face of such attacks.

Thesis Supervisor: Dr. Clifford J. Weinstein
Title: Group Leader, Information Systems Technology Group

Thesis Supervisor: Dr. Thomas M. Parks
Title: Technical Staff, Information Systems Technology Group

Acknowledgments

The implementation of the Reliable Multicast Protocol that I used for these tests was written by Todd Montgomery, Brian Whetten and John R. Callahan. David Kassay conceived of and helped implement some of the attacks against RMP. Chad Jones helped set up the test network configuration. Thomas Parks provided valuable feedback and guidance in the creation of this thesis. Clifford Weinstein supervised my work. I'd like to thank them all for their assistance.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Multicast is an extension to the regular protocol for sending messages between host nodes on the internet. Multicast describes the sending of messages from one node to a group of nodes. The sender may or may not know all of the recipients of the messages. Although mechanisms for reliably sending messages to one node or to a very small number of nodes are robust and well understood, mechanisms for multicasting messages are much newer and there are no commonly accepted standards for reliable multicast.

In this thesis, I will discuss several protocols which have been proposed and implemented to attempt to provide reliable multicast communications. I will also present attacks designed to interfere with these protocols. Then I will show measurements of the performance of these protocols, both with and without attacks and analyze the reliability of these protocols.

1.1 Motivations for developing reliable multicast

The effects of multicasting messages can be achieved by simply sending separate messages to each node in the group. This requires the sending node to send out a distinct message for each node that it's trying to send to. However, this often uses much more network bandwidth than is necessary, especially if many of the recipients of the message are on the same subnet. The traffic generated by this is shown in

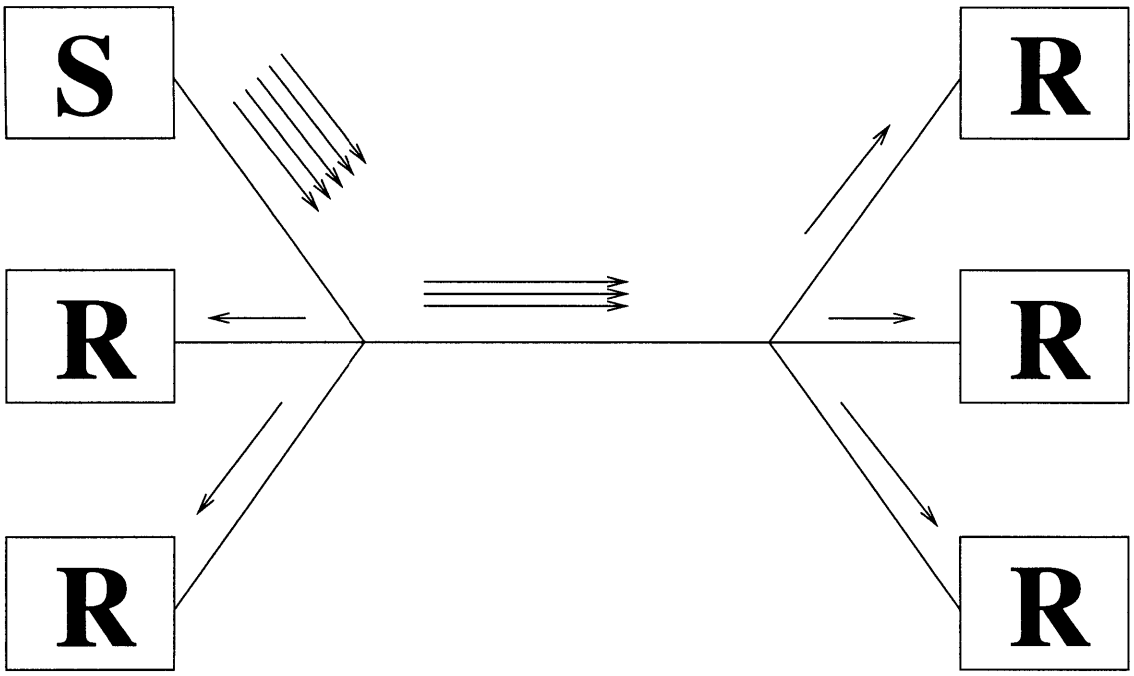

Figure 1-1: Network traffic with unicast communications.

figure 1-1.

This also requires the sending node to know in advance all of the nodes to which it is sending. For these reasons, protocols have been designed specifically for multicasting messages.

1.2 Unicast Communications

Reliable point to point communication over the internet is usually accomplished through the use of two standard protocols. The IP, or Internet Protocol[3], specifies how communication packets are routed. The TCP, or Transmission Control Protocol[4], ensures reliable delivery of packets.

The Internet Protocol specifies an addressing scheme for computers and a mechanism for fragmenting message packets that are too large to be sent over the network. It also defines a standard format for packet headers. It does nothing to ensure the delivery of packets.

The Transmission Control Protocol specifies a sequence of packets to be ex-

changed, via the Internet Protocol, between the sender and the receiver of the data. The TCP ensures that the receiver has received the data and the sender knows that the receiver has the data. This protocol will work, even if a significant fraction of the packets that are sent never arrive at their destination.

The Transmission Control Protocol also provides for flow control. The sender of the data gets feedback about how many of the packets it sends are never reaching their destination. Assuming that packet loss is due to network congestion, the sender can adjust the rate at which it sends packets to make full use of the available bandwidth while avoiding network congestion.

1.3 IP Multicast

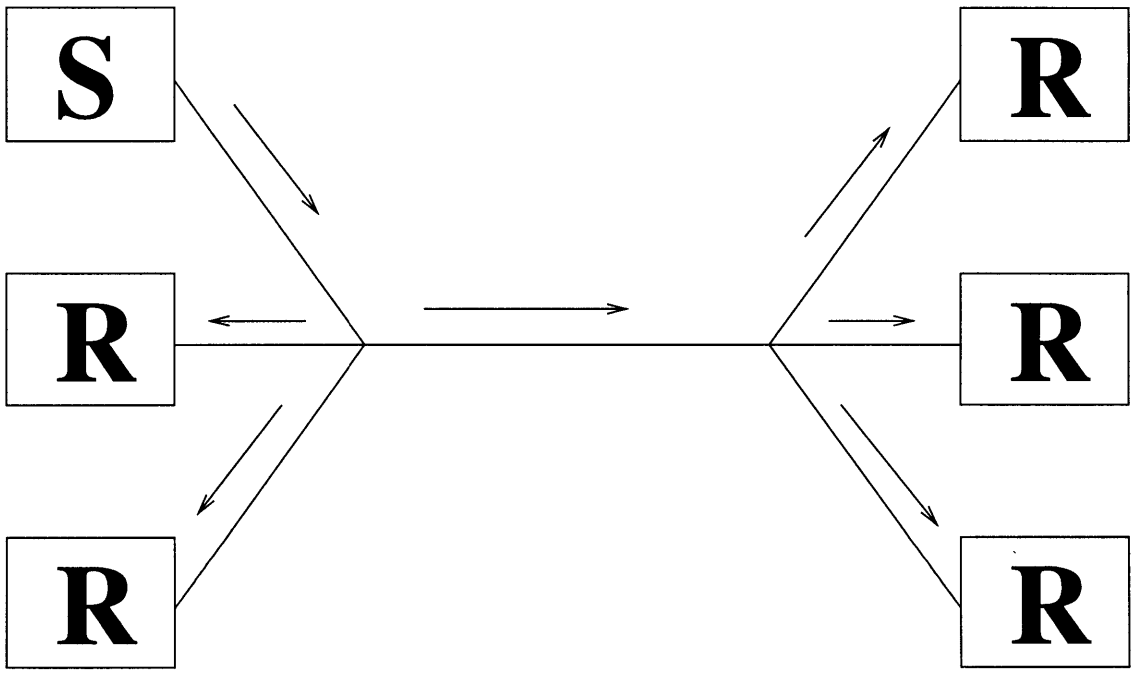

Figure 1-2: Network traffic with multicast communications.

The Internet Protocol has been extended to allow multicasting[1]. The extension specifies multicast group addresses. Hosts can subscribe to multicast groups, and senders can send to a group address rather than to an individual node. When a router receives a message addressed to a multicast group, it forwards it to a subnet if

and only if it knows of any nodes on the subnet subscribed to that group. This saves a lot of traffic, as shown in figure 1-2.

This protocol greatly reduces the communication complexity of sending a single message to multiple hosts. Under the original protocol, the number of packets sent would have had to equal the number of hosts receiving the message. The multicast extension guarantees that for each message packet sent to a multicast group, only one copy of the packet traverses each link of the network.

1.3.1 Difficulties in Achieving Reliability

The Internet Protocol extension does the same for multicast as the original IP did for unicast. However, there is no equivalent to TCP for multicast. Several protocols have been proposed and developed, but none is accepted as a standard for reliable multicast.

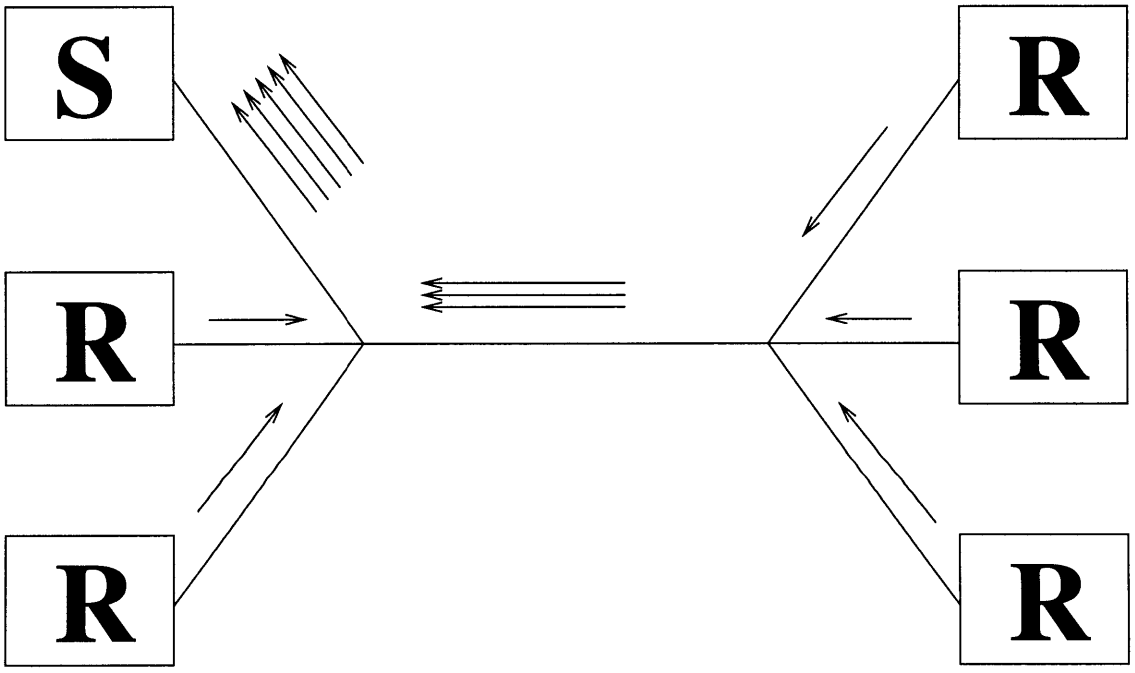

Figure 1-3: Acknowledgement implosion.

The method used in TCP would be impractical to transfer to multicast. In TCP, the receiver of the data sends an acknowledgement to the sender upon receipt of the data, and the sender resends the data if it fails to receive an acknowledgement. If

every subscriber to a multicast group were to send an acknowledgement packet to the sender, the efficiency benefits gained by the multicast protocol would be completely undone. This is called the ACK implosion problem, and is shown in figure 1-3.

Also, the sender would need to know every subscriber to the multicast group, or else it would have no way of knowing if all of the subscribers had sent acknowledgements. A way is needed to ensure delivery to all receivers without drastically increasing network traffic.

Chapter 2

Existing Reliable Multicast Protocols

There are several general strategies that have been proposed for ensuring reliable delivery of multicast messages. However, there is no protocol which is widely used and accepted as standard.

For this project, I am only going to look at one reliable multicast protocol. This protocols was chosen because source code for it was available, making modifications possible, and because it is sufficiently developed to be in a marketable form. This protocol is the Reliable Multicast Protocol, or RMP[5].

2.1 Classes of protocols

The existing reliable multicast protocols can be divided into three basic types. There are ring based protocols, tree based protocols and cloud based protocols[2].

Ring based protocols typically assign all of the multicast group members a place in a virtual ring. A token is held by one member of the ring and passed periodically. Typically the passing of the token is used to ensure delivery of all messages. When a host receives the token, it is guaranteed that every host has received all of the data sent before the last time the host received the token.

Tree based protocols divide the multicast group into a virtual tree. Each tree

node is responsible for making sure that all of its descendents have received all of the messages.

Cloud based protocols do not have any structure for assuring that all messages are delivered. Any node that does not receive a message simply requests the retransmission of that message until it receives it. This sort of negative acknowledgement system is generally less reliable but more efficient than a positive acknowledgement system.

2.2 RMP

The Reliable Multicast Protocol is a ring based protocol. There is a token which is passed around the ring. When a message is sent, the token holder sends an acknowledgement message to the sender. The acknowledgement includes the sequence number of the original message. The sender continues to resend each message until it receives an acknowledgement for that message from the token holder. Other nodes look for gaps in the sequence numbers of the messages they have received. Gaps indicate missing messages. If another node realizes it has not received a message, it requests a retransmission of the message.

Reliability is ensured through the passing of the token. Before the token holder passes the token to the next member in the group, it makes sure that that member has received every message up to that point. If the token is passed often enough, every group member is kept reasonably up to date.

Occasionally, network failures make the delivery of some messages impossible for a period of time. If some members of the group are unreachable for too long, the group dissolves and attempts to reform itself so that it contains only the members of the group that are actually reachable through the network. Processes in the group initiate successive reformations until one succeeds. A reformation is considered to have succeeded when all of the reachable group members agree upon the new membership list of the group.

The Reliable Multicast Protocol is reliable and reasonably efficient. However,

because of the ring configuration, scalability is limited. It also imposes the restriction, which is not present in IP multicast, that all the members of a multicast group are known at all times.

Chapter 3

Attacks

The intent of this project is to examine the chosen multicast protocol for performance. Two aspects of performance are efficiency and robustness. Multicasting a message to a large number of nodes should be considerably faster and require many fewer messages to be sent than sending the identical message to each node individually would be. The data that each receiver gets should be the same as what the sender sent. The sender should know if the receivers do not get all of the data for any reason.

The performance of a protocol can be measured on an unloaded network in optimal conditions. However, this will not give a very accurate impression of its performance in other types of conditions. To be useful practically, a protocol must be able to operate effectively over heavily loaded networks, unreliable network connections or environments which may include misconfigured or malicious hosts.

The purpose of evaluating the performance of protocols in the face of malicious attacks is to see how the protocols perform over a wide range of conditions, rather than just optimal conditions.

Some attacks affect performance. They may cause packets to be lost or delayed in transmission. The performance of a protocol in the face of such an attack should be similar to the performance of the protocol on a slow or lossy network. If a significant percentage of the traffic is still getting through, the protocol should still operate correctly. However, the rate at which the protocol degrades in performance under such an attack is an indication of how well the protocol is designed.

Other attacks involve transmitting packets specifically intended to disrupt the workings of the protocol. With the technology I am using in this thesis, which does not include encryption and verification of senders, it is impossible to guarantee that a protocol will behave correctly in the face of any attack. However, a well defined protocol should be able to still function in the face of some simple attacks.

3.1 Flooding Attacks

Flooding attacks are a very simple class of attacks which do not target any particular protocol. These attacks simply send extraneous packets at a very great rate to the multicast address that the protocol is using. The packets can contain anything, since the value of the attack lies in the volume of packets sent, not the contents of the packets.

The resulting network traffic can cause many packets to be delayed or to be lost completely due to network congestion. Some slowdown in the rate at which data is transfered is expected in the face of this attack.

A simple version of this attack was implemented for this thesis. The rate at which flood packets are sent in this attack is dependent only on the speed of the machine on which the attack is being run. The data that was sent in the packets was generated by simply allocating a memory buffer and sending its uninitialized contents.

3.2 Replay Attacks

Replay attacks are another general class of attacks. They involve listening to traffic on a network and sending exact duplicates of some or all of the packets sent over that network.

This sort of attack can also be used against protocols which employ encryption. Although the attacker cannot decrypt the packets, it can replay encrypted packets which the receivers can decrypt. Authentication techniques can provide protection against this attack at the cost of additional overhead in sending and receiving.

The implementation of this attack used for this thesis was a simple program which listened to the IP multicast address that the reliable multicast protocol used and replayed all packets sent to that address, except ones from the attacking node.

3.3 Dropped Packet Attacks

Dropped packet attacks are a very simple type of insider attacks. This type of attack requires the attacker to be able to pose as one of the receivers within the multicast protocol.

In the RMP protocol, each receiver must explicitly join the group of receivers. Each receiver must request permission to join, but in the implementation that was used for this thesis, there is no provision for excluding a receiver. If the membership of the group is limited in a secure way, this attack might become infeasible.

In this type of attack, the attacker intentionally drops some or all of the data packets received. This causes the attacker to request retransmissions of the dropped data, which increases the network traffic and delays the sending of new data packets.

This attack is also similar to the case where a sender is multicasting to a group of receivers, one of whom is on the other end of a slow, congested network link. In either case, one of the receivers is not getting all of the packets and must request retransmissions. The difference between this case and the implementation of the dropped packet attack that was used is that the attacking host only dropped data packets, whereas a host on a unreliable network would lose all types of packets with equal probability.

The algorithm used to determine which packets to drop involved two states. The network could either be "down", in which case packets were dropped, or "up". After each packet arrives, the network may or may not change states according to the packet loss parameter. The state diagram used to determine the network state is shown in figure /refdrop:fig1.

The algorithm I used had a scaling factor of one half. This means that if packets are dropped with overal probability $\mathcal{P}$ and the network is currently up, the probability

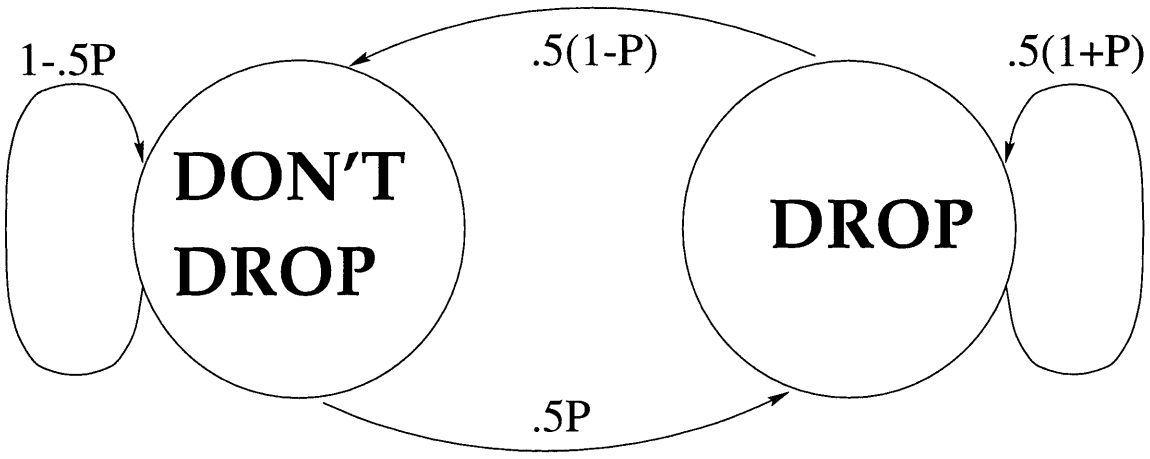

Figure 3-1: Simple state diagram for dropped packets.

of the network going down is one half of $\mathcal{P}$.

3.4 RMP Token Attack

Ring based protocols such as RMP depend on a token being passed in order to guarantee delivery to all of the receiving hosts. Therefore, a type of attack which is specific to ring based protocols is an attack on the token.

This type of attack does not prevent the sender from sending data or the receivers from receiving it. However, it removes the guarantee that all of the receivers have received all of the data. This sort of attack would probably be most effective in conjunction with another attack which prevents receivers from receiving data.

In one type of token attack, one receiver takes the token when it is passed to them. However, the receiver keeps the token, rather than passing it on. Like the dropped packet attack, this is another insider attack.

Chapter 4

Testing Procedure

All of the attacks described in chapter 3 were tested against an implementation of the RMP protocol. The implementation used was version 1.3 Beta of the implementation written by Todd Montgomer, Brian Whetten and John R. Callahan and available at `ftp://research.ivv.nasa.gov/pub/src/RMP/`.

An isolated network setup was used, to insure that outside traffic did not interfere with the measurements. This made the results obtained less representative of general use conditions. However, it decreased the variation in the network due to outside factors, making the results obtained more accurate and repeatable.

4.1 Network

The network used consisted of four computers connected via a single ethernet hub as shown in figure 4-1. All four computers used were running Solaris version 2.5.1. However, MASON and NOMAD have Sun Sparc processors and LEWIS and CLARK have Intel based x86 processors.

A simple file transfer program was used which took a file as input, joined the specified RMP group and sent the data in the file to the group via the RMP protocol. As soon as it had received confirmation that all of the group members had received the data, the program exited. The sending program did not know the identity of the intended receivers. It simply sent to all of the members of the RMP group at the

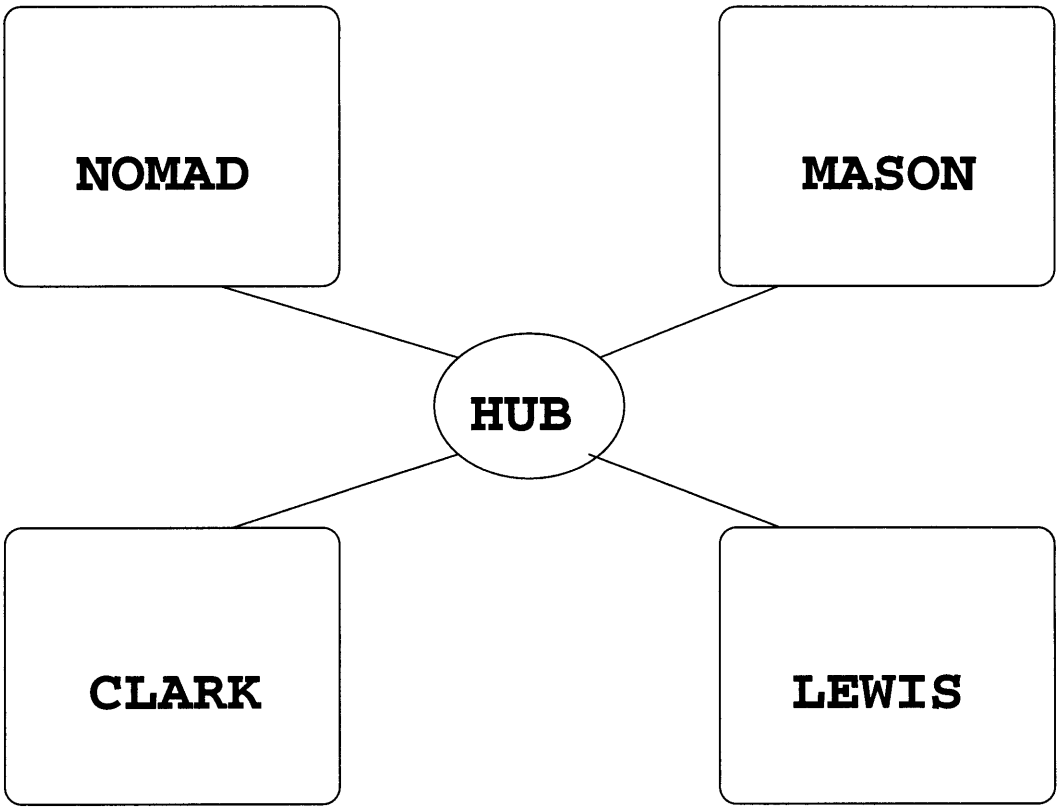

Figure 4-1: Test network configuration.

time it joined the group.

The receiver program that was used joined the specified RMP group at startup. It recorded everything sent to the RMP group while it was a member and printed it to a file. The receiver programs were all run before the sending program so that the sending program would see the correct group membership when it started.

In order to simulate a larger network and a larger group of receivers, multiple receive programs were run on each host. For the measurements done in this thesis, one host would run a send program and a receiver. The other hosts would run two receivers apiece. The attacking program was run on one of the machines with two receivers.

4.2 Gathering Data

Measurements on network traffic were collected using the `snoop` program. One of the hosts running two receiver programs also snooped all of the traffic on the network and recorded it for later analysis.

Snoop was chosen to gather data because it does not require statistics to be collected by the RMP protocol. Having the protocol keep statistics might have altered the results. The disadvantage of using snoop is that there is the possibility that snoop might not get all of the packets. To reduce this risk, MASON, which is the fastest of the four machines used, was used to collect the data.

The RMP send and receive programs were first run with no attacks against them and their performance was measured. This was done to provide a baseline with which to compare all subsequent test results.

The same send and receive programs were then run again with the flood attack, the replay attack and the dropped packet attack. The dropped packet attack was performed with 4 different levels of packet loss, as well as the baseline 0 percent loss. The levels measured were 25 percent, 50 percent, 75 percent and 95 percent packet loss on a single receiver.

For the purpose of all of the measurements described, a single 908 kilobyte data

file was multicast to the seven receivers in each test. The base configuration and each attack configuration was run ten times in order to find a range of possible performance responses as well as the average response.

The token attack was not tested. When this was implemented with the version of RMP used, the protocol detected the failure of the token pass. The response of RMP to a failure detection is to force a reformation of the group. However, in the version of RMP that was used for these experiments, the reformation function hasn't been implemented yet, so the process simply exited with an error. Therefore, no useful data could be gathered.

Chapter 5

Test Results

5.1 Measurements

The RMP file transfer program was run ten times with no attacks and ten times with each attack. A 908 kilobyte file was sent in each trial. The trace recorded from each trial was used to calculate a number of things, including the total time from when the sender first attempted to join the group until it left the group, the time actually spent transferring the data, the total number of data sent, both in terms of IP datagrams sent and in the total number of data packets sent. A single datagram could consist of multiple data packets if the datagram had to be broken down into a size that could be transmitted over the network.

In the cases where the file transfer did not complete succesfully, I measured the amount of data sent successfully to each receiver. I also verified whether the data that was received was correct and uncorrupted.

5.1.1 Base Case

First I ran file transfer program and took measurements with no attack running. This provided a base case with which to compare the performance of the protocol in the face of attacks.

In the trials run with no attacks, all of the data was successfully transmitted to

all of the receivers. The average amount of time for the data to be sent once the sender had joined the RMP group was 46.76 seconds. The average time it took the sender to send data, including the time from when it first attempted to join the group until it left was 82.44 seconds. More detailed numbers on all of the tests are given in appendix A.

5.1.2 Flooding Attack

For the flooding attack, the attacking machine flooded the network with garbage packets as fast as possible.

When a host joins an RMP group, it sends out a join message to the group address and waits for a response back. If it doesn't receive a response back within a certain period of time, it forms its own group.

The flooding attack caused many packets that hosts tried to send on the test network to be delayed or to not be sent. As a result, sometimes a host attempting to join an already existing RMP group would decide that the group didn't exist and form its own group with the same name and address.

When this occurred, there would be two or more distinct sets of hosts claiming to be the same group. Any hosts attempting to join the same RMP group subsequently would either end up joining one of the two groups or forming their own.

When this occured, the sender would join one of the groups, and only the receivers actually in that group would receive any data.

In six of the trials, all of the receivers got the data. However, in one trial the sender only sent to 4 hosts and in another trial it only sent to 2 hosts. In one trial the sender formed a group by itself and didn't send to any of the receivers.

In all of the these cases, all of the data was correctly transmitted to all of the receivers who got any data. The other receivers got no data and in some cases detected an error and exited. The sending program believed it had successfully transmitted the data to all of the receivers in the RMP group. However, the receivers it believed to be in the RMP group included only those receivers who actually got the data.

In one trial a failure was detected and the program exited before sending all of

the data. In this case, two of the receivers got 522.5 kilobytes of data before the error was detected. The rest of the receivers didn't receive any data.

In the trials where all of the receivers received all of the data, it took an average of 53.4 seconds to transfer the data, with a standard deviation of 6.9. The average time for the sender to join the group, send the data and leave the group was 80.9 seconds with a standard deviation of 12.

These times represent a slightly longer time to send data than the base case. However, this is not reflected in a correspondingly longer connect time.

In the trials where data was sent to only a subset of the receivers, the sending times were shorter.

5.1.3 Replay Attack

For this attack, a program on the host MASON listened to the multicast address and port being used for the multicast file transfer program. Whenever a packet that didn't come from the attacking program was sent to that address, the attacking program sent a packet with the exact same contents to the same address.

In nine of the ten trials, the sending program joined the group, sent the file and exited without detecting any error. In one of the trials the sender detected an error and exited after only 106 kilobytes had been sent.

In the nine trials that completed, the two receiving programs running on the machine MASON and the receiver running on the machine NOMAD received all of the data successfully. In one of the nine trials, the receivers running on the machine LEWIS received all of the data successfully.

In eight of the trials, the receivers on LEWIS received the same quantity of data as was sent. However, there were errors in the data received. Most of the file was the same as the file sent. However, some bytes differed. In the nine trials that completed, the same behavior was seen by the receivers on the machine CLARK.

The data received was not corrupted in the same way in different trials. Nor did different receivers necessarily have the same corrupted data on the same trial.

In the trials where the send completed, it took an average of 55.2 seconds to

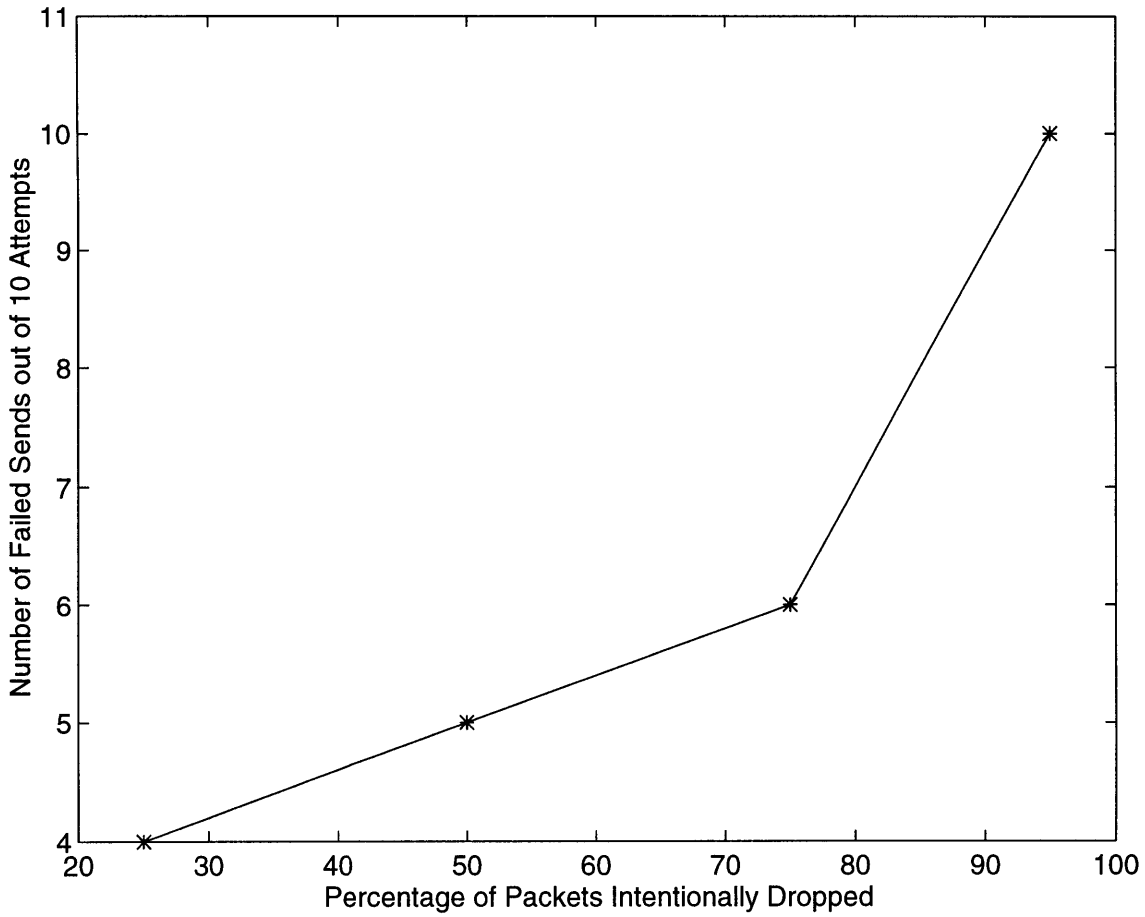

Figure 5-1: Packet drop rate vs. Number of Trials Which Failed to Complete

transfer the data, with a standard deviation of 6.1. The average time for the sender to join the group, send the data and leave the group was 83.6 seconds with a standard deviation of 13.9.

These times represent an increase in sending time over the base case. They are also slightly higher than the times obtained from the flood attack.

5.1.4 Dropped Packet Attack

I ran the dropped packet attack with packet drop rates from 25 percent to 95 percent. The results from these tests were compared against the baseline results.

As the the percentage of dropped packets increases, so does the likeihood that the send will fail to complete due to an error. This relationship is shown in figure 5-1.

The baseline results had a 0 percent failure rate, whereas none of the trials at the 95 percent packet drop rate completed successfully. Failures occurred when a group member perceived an error.

In this test, the apparent high packet loss rate to the attacking host made it appear as if the network was temporarily unreachable. The response that should result from detection of an error of that sort is to force a reformation. However, since the reformation code hadn't been implemented in the version of RMP used for these experiments, the host detecting the problem exited with an error instead.

In addition to happening more frequently, failure occurred sooner as the percentage of dropped packets increased. The relationship between the rate of packets dropped and the amount of data sent before a failure is detected is shown in figure 5-2. If a failure did not occur at any time during a trial, the amount of data successfully sent was 908 kilobytes.

In the trials where no error was detected and the send completed successfully, the amount of time required to send the data varied. As the rate of dropped packets increased, more retransmissions became necessary, slowing down the rate at which new data was sent.

The relationship between the percentage of packets dropped and the amount of time it took the sender to join the RMP group, send the data and leave the group is shown in figure 5-3.

The relationship between the percentage of packets dropped and the amount of time it took the sender to send the data, once the sender was already in the group, is shown in figure 5-4.

5.2 Interpretation

All of the attacks appeared to increase the time to send data by some amount. In addition, most of them caused other problems to appear as well.

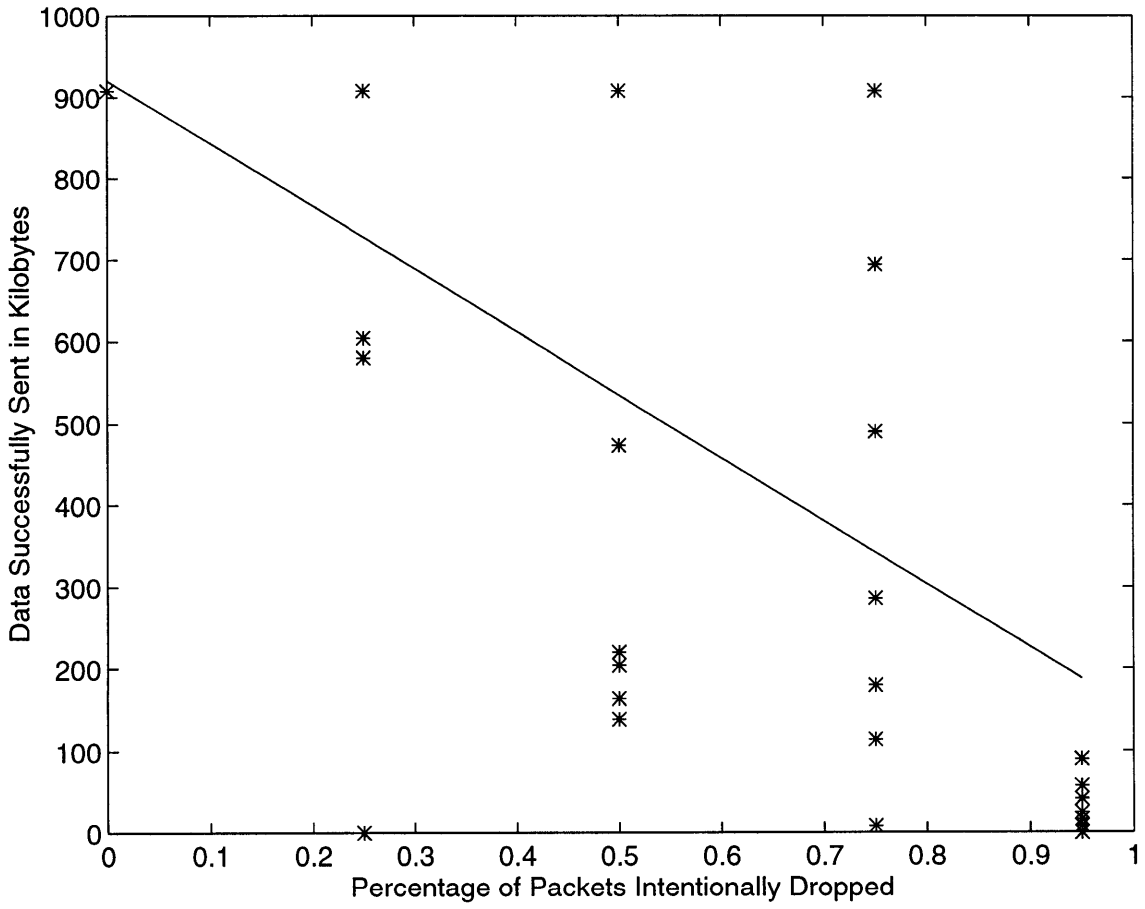

Figure 5-2: Packet drop rate vs. Data Successfully Sent Out of 908 Kilobytes Attempted

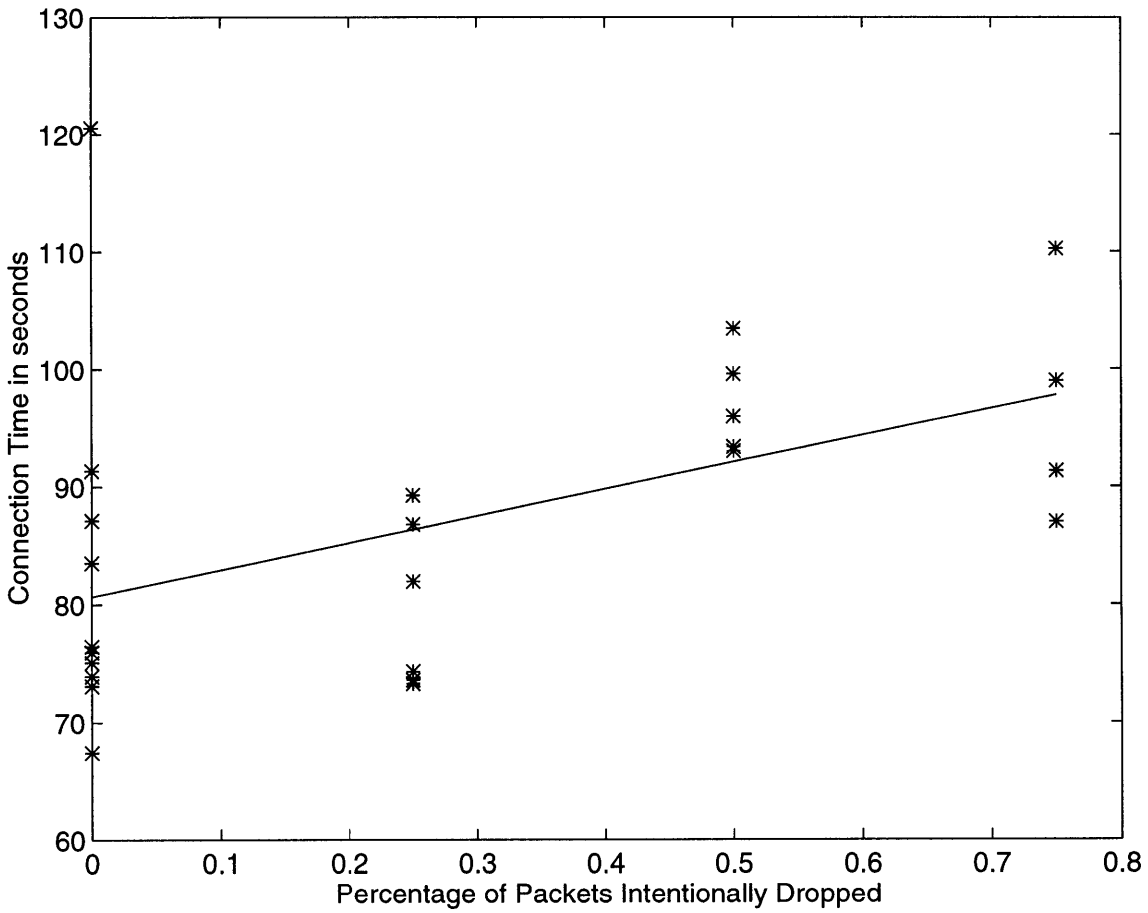

Figure 5-3: Packet drop rate vs. Total Time Spent by Sender Where the Send Completed

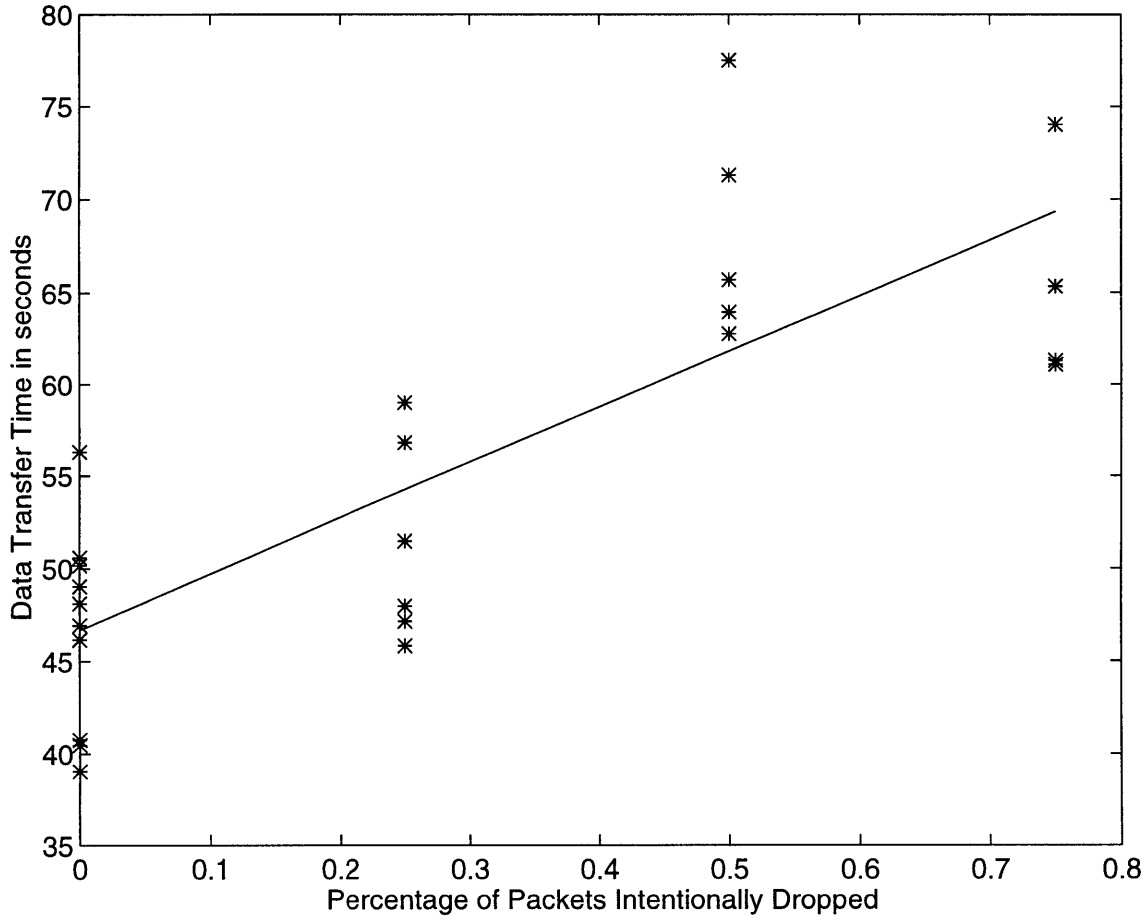

Figure 5-4: Packet Drop Rate vs. Time Spent Sending Data Where the Send Completed

5.2.1 Failures

Since the version of RMP that was used does not have reformations implemented, it is hard to guage how much failures detected in transmission would affect the transfer if reformations were an option in the face of failures.

It is probable that most if not all of the failures which caused the termination of the data transfer would have only caused a reformation to begin if that were implemented. In this case, if a successful reformation occurred such failures would cause a significant increase in transmission time, but not necessarily a fatal error.

As long as some data is successfully sent before a reformation occurs, progress is made. If the RMP group is always able to reform itself then the transfer will eventually complete. In all of the attacks that were implemented and tested, some data was sent at least some of the time before a failure occurred. Therefore, these attacks would delay transmission of data but not prevent it indefinitely if the group always successfully reformed itself.

There is the possibility that the group might fail to reform itself or that it might not include all of the original group members in the reformed group. If this were to happen, some or all of the receivers might not get the data. It is unknown what affect these attacks would have on the reformation process.

5.2.2 Incorrect Data

A more serious consequence of these attacks was incorrect data being received by some of the receivers without any of the group members detecting an error.

Both of the receivers who got incorrect data had Intel type processors running Solaris. It is unclear why only these two machines were affected. However, the problem appears to be spacific to that type of machine. If this is the case, it might be a problem with the specific implementation of the protocol on that platform and not a conceptual problem with the protocol.

Chapter 6

Conclusion

The results of the experiments described in this document show that the implementation of the Reliable Multicast Protocol which was used for these tests is vulnerable to attacks.

These attacks reliably slow down the rate at which data is sent or prevent the data from being transmitted altogether. In some circumstances these attacks can cause intended receivers to fail to be able to join the receiving group or data that was transmitted to be received incorrectly without either the sender or the receiver detecting the error.

However, the vulnerability of an implementation of a particular protocol to attacks does not necessarily imply that the protocol is vulnerable to those attacks. Such vulnerabilities can either be the fault of the protocol or of the implementation.

It is fairly clear that most if not all of the failures to transmit the data are a result of the particular implementation. The data corruption which was observed in the test of the replay attack also appears to be an artifact of this implementation.

On the other hand, the slowdown in the rate that data was sent to the receivers that was observed in response to several of the attacks is almost certainly a result of the protocol. The slowdown rate that was observed is also quite likely artificially low, since the rate was only calculated from the test trials which completed successfully. However, the trials upon which the attack had the greatest impact were the most likely to fail.

In addition to the observed slowdown, the protocol is particularly vulnerable to attacks that are aimed at reducing the speed at which data is transmitted because of the method the protocol uses to deal with errors.

If each of the failures observed represents a situation where a complete implementation of the protocol would initiate a reformation, then each of those failures represents a significant increase in the total time to transfer data. The sooner the transfer program detected an error and exited in the tests, the more times the attacker would have forced a reformation if the transfer program ran to completion. If a lot of reformations occur, this could represent a very significant slowdown.

The failure of some receivers to join the multicast group in the face of the flooding attack is also most likely a result of the protocol and not the particular implementation used. The protocol requires all receivers in the RMP protocol to explicitly join the RMP group and be acknowledged by the current group members.

The reliance of the protocol on knowing the group membership at all times is a point of failure. Any attack which prevents a receiver from communicating with the RMP group for a sufficient length of time can cause different receivers to have inconsistent views of the group membership.

6.1 Future Work

There are several areas where further research is called for. Experimenting with other implementations of the same protocol would provide more insight into what the limitations of the protocol are, and what are the limitations of this particular implementation. Also, there are many other attacks against RMP which could be implemented and tested.

Group reformations are processes which are particular to this protocol which would be particularly valuable to explore further. Experimenting with implementations which include reformations would provide many more opportunities to examine the response of this protocol to various attacks.

In addition, the existence of reformations provide a target for a whole class of

attacks. One can envision a class of attacks which seeks to prevent a reformation from ever successfully completing, thereby halting the transmission of data indefinitely once a reformation is forced. Some of these threats could be countered through the use of authentication to create a trusted group, but not all.

Appendix A

Test Results

	Kbytes Sent Successfully	IP Datagrams	Data Packets	Total Time	Data Transfer Time
1	908	280	1833	91.34	56.31
2	908	225	1497	120.55	39.04
3	908	266	1761	75.04	49.04
4	908	251	1633	83.54	48.12
5	908	246	1620	73.90	46.94
6	908	254	1711	73.08	46.17
7	908	226	1501	67.43	40.48
8	908	248	1651	76.43	50.17
9	908	263	1720	75.95	50.59
10	908	253	1664	87.13	40.76
Avg.	908	251	1659	82.44	46.76
St.Dev	0	16.9	105.72	15.18	5.38

Table A.1: RMP File Transfer Without Attacks

	Kbytes Sent Successfully	Successful Receivers	IP Datagrams	Data Packets	Total Time	Data Transfer Time
1	908	7	229	1607	98.05	58.71
2	908	4	195	1388	82.54	43.89
3	908	2	238	1656	75.38	43.55
4	908	7	237	1645	85.52	55.85
5	908	7	213	1475	61.05	39.93
6	908	7	209	1464	83.80	54.22
7	908	7	234	1563	78.15	53.81
8	908	7	231	1625	78.90	57.66
9	0	0	0	0	N/A	N/A
10	522.5	2	266	1822	110.03	62.49
Avg.	778.66	5	205.2	1424.5	83.71	52.23
St.Dev	299.21	2.75	74.63	515.06	13.87	7.85

Table A.2: RMP File Transfer With Flood Attack

	Kbytes Sent Successfully	Successful Receivers	Total Time	Data Transfer Time
1	908	3	83.76	62.08
2	908	3	110.93	59.69
3	908	3	79.27	58.31
4	908	3	67.21	42.48
5	908	3	76.04	54.42
6	106	3	82.58	54.04
7	908	3	73.52	49.76
8	908	3	78.47	52.43
9	908	3	101.48	58.91
10	908	5	81.76	59.01
Avg.	827.8	3.2	83.5	55.11
St.Dev	253.61	0.63	13.09	5.83

Table A.3: RMP File Transfer With Replay Attack

	Kbytes Sent Successfully	IP Datagrams	Data Packets	Total Time	Data Transfer Time
1	908	310	2021	73.58	47.14
2	908	325	2126	89.25	59.0
3	908	354	2259	73.27	47.96
4	908	279	1840	86.79	45.82
5	604	303	2059	84.30	40.93
6	580	271	1841	121.03	38.82
7	908	325	2086	74.27	51.45
8	908	327	2132	81.95	56.81
9	0	109	755	113.73	15.7
10	0	51	354	60.24	31.83
Avg.	663.2	265.4	1747.3	85.84	43.55
St.Dev	372.63	101.54	648.36	18.69	12.72
Successful Avg.	908	320	2077.3	79.85	51.36
Successful St.Dev	0	24.64	139.95	7.14	5.44

Table A.4: RMP File Transfer With 25% Packet Drop Rate

	Kbytes Sent Successfully	IP Datagrams	Data Packets	Total Time	Data Transfer Time
1	139	123	804	83.70	15.35
2	164	323	2138	99.41	49.85
3	908	485	3129	99.58	77.49
4	204	165	1109	72.76	33.23
5	220	161	1052	88.23	61.90
6	908	490	3230	95.94	71.29
7	908	427	2692	93.36	62.74
8	908	442	2834	93.00	65.69
9	908	428	2731	103.44	63.93
10	473.5	288	1871	109.41	41.15
Avg.	574.05	333.2	2159	93.88	54.26
St.Dev	363.23	142.02	906.90	10.45	19.24
Successful Avg.	908	454.4	2923.2	97.06	68.23
Successful St.Dev	0	30.84	242.30	4.43	6.13

Table A.5: RMP File Transfer With 50% Packet Drop Rate

	Kbytes Sent Successfully	IP Datagrams	Data Packets	Total Time	Data Transfer Time
1	908	567	3611	87.01	61.28
2	180	206	1392	75.72	24.91
3	908	629	3955	91.31	61.06
4	8	120	814	58.44	13.88
5	694	555	3604	124.61	70.37
6	908	599	3791	110.24	74.02
7	908	539	3533	98.99	65.33
8	114	161	1032	85.27	31.34
9	490	414	2653	105.46	67.79
10	286	258	1726	77.79	29.10
Avg.	540.4	404.8	2611.1	91.48	49.91
St.Dev	369.16	199.04	1248.5	19.11	22.39
Successful Avg.	908	583.5	3722.5	96.89	65.42
Successful St.Dev	0	39	188.94	10.19	6.06

Table A.6: RMP File Transfer With 75% Packet Drop Rate

	Kbytes Sent Successfully	IP Datagrams	Data Packets	Total Time	Data Transfer Time
1	24.5	215	1369	78.14	18.16
2	57	151	1014	74.29	15.42
3	8	128	822	56.05	15.12
4	41	193	1269	55.10	15.94
5	24.5	125	832	50.63	12.93
6	0	126	810	45.49	10.98
7	16	160	1021	65.07	13.33
8	16	116	787	49.5	9.76
9	8	107	714	46.18	8.68
10	90	206	1357	54.91	17.68
Avg.	28.5	152.7	999.5	57.54	13.8
St.Dev	27.38	39.34	249.39	11.37	3.24

Table A.7: RMP File Transfer With 95% Packet Drop Rate

Bibliography

[1] S. Deering. Host Extensions for IP Multicasting. Stanford University, Aug 1989. Internet RFC 1112.

[2] Brian Neil Levine and J.J. Garcia-Luna-Aceves. A Comparison of Reliable Multicast Protocols. *ACM Multimedia Systems Journal*, August 1998.

[3] DoD standard. Internet Protocol. USC/Information Sciences Institute Request for Comments, Jan 1980. Internet RFC 760, IEN 128.

[4] DoD standard. Transmission Control Protocol. USC/Information Sciences Institute Request for Comments, Jan 1980. Internet RFC 761, IEN 129.

[5] B. Whetten, T. Montgomery, and S. Kaplan. A High Performance Totally Ordered Multicast Protocol, Theory and Practice in Distributed Systems. *Springer Verlag LCNS*, 938.