Multicast Survivability in Hierarchical Broadcast Networks

Azad Azadmanesh
University of Nebraska
Computer Science Department
Omaha, Nebraska 68182
USA
azad@unomaha.edu

Axel Krings
University of Idaho
Computer Science Department
Moscow, Idaho 83844
USA
krings@uidaho.edu

Daryush Laqab
University of Nebraska
Computer Science Department
Omaha, Nebraska 68182
USA
dlaqab@gmail.com

Abstract: Despite the increasing number of applications that benefit from multicasting, most reliable multicast protocols consider omission faults only. This research is concerned with survivability of multicast communication where the communication medium is shared among the hosts. The approach presented specifies that a network with N nodes is resistant against network omissions and malicious nodes if $N \geq 2t + b + 2$, where t and b are the number of malicious and benign faulty hosts, respectively. The simulation results show that the network traffic is dynamically adapted to the rate of faults and that the network performs well under omission and malicious faults, unless the rate of faults is high. Furthermore, it is shown that each additional network segment requires $(2t_g + 1)$ gateways, where t_g indicates the number of faulty gateways.

Key–Words: Byzantine agreement, Hybrid fault models, Multicast, Peer-to-Peer networks, Ad-hoc networks

1 Introduction

In multicast communication, a message is sent to a group of receivers. Some applications of multicasting are in multiplayer computer games, webcasting, audio/video-conferencing, process coordination, and routing in ad-hoc networks [3, 10, 14, 17, 21, 22, 24]. Multicast communication has been used over the Internet where the interconnection is inherently point-to-point. It has also been used in multi-point communication media such as Ethernet. Although, there exists a fair amount of research in the area of multicasting, most of the research is focused on omission faults. An omission occurs when a node expecting a message does not receive it, e.g., due to link failure, crash of the transmitting node, or if the message is corrupted during transmission and not correctable by the ECC component. As the research with respect to malicious behavior or a mixture of failure types is limited, the main goal is to ensure the multicast protocol designed not only tolerates omission faults, but it also tolerates malicious behavior. Different failure modes and their impact on network fault tolerance and network performance will be investigated. As this research is concerned with general broadcasts, the principles discussed here are applicable to all broadcast environments, including wireless networking such as ad-hoc and sensor networks.

A multicast is considered reliable when all correctly operating nodes either deliver a message multicasted or none of them deliver it if at least one node fails to deliver the message. Here a distinction is made between receiving and delivering a message. A node might receive a message but not deliver the message or there might be a delay between receiving and delivering a message. This delay is caused by the receiving overhead such as queuing time, ECC process, and the multicasting overhead to ensure multicast properties are met.

This research uses the concept of acknowledgment(ACK) and negative acknowledgment (NACK) handshaking by nodes, such as in the Trans protocol [15]. In the Trans protocol, each node communicates with others through broadcasting messages. Once a node receives a message, it piggybacks an ACK for the received message to its next outgoing message. Should a node determine it has not received a message, the node adds a NACK for the missing message to the very next outgoing message. This enables the network to be robust against omission faults. The nodes can also construct empty messages just for the purpose of sending a list of ACKs or NACKs.

The Trans protocol is essentially an error detection and error recovery protocol. It depends on the physical layer to put the outgoing messages on the bus and to pick up the incoming messages from it. The protocol does not deal with delivering of messages to the application layer. Therefore, it is primarily a data link layer protocol intended for detecting and recovering from omission faults over a bus medium.

The authors of Trans have also designed another protocol, called Total protocol [16], which places a

total order on the delivery of messages. The Total protocol was later enhanced to handle Byzantine failures [18]. A *Byzantine* fault is not constrained by any assumption regarding its behavior. For example, a Byzantine faulty node is capable of transmitting conflicting messages to the receiving nodes. To remove the impact of Byzantine behavior, the protocol assumes the underlying multicast protocol uses digital signatures to identify the source of a message and to distinguish among different messages.

The rest of this study is organized into the following sections. Section 2 touches upon some background and introductory information. Sections 3 and 4 are concerned with a single shared medium, such as an Ethernet segment. Section 3 focuses on the approach and design of a reliable multicast protocol. The section also discusses the fault types that can be endured by the protocol. Section 4 shows simulation results based on the protocol design discussed in the previous section. Section 5 sheds some light on the requirements to extend a single segment network into multi-level multicast segments. A summary of the work is given in Section 6.

2 Background and Definitions

An efficient multicast technique conserves bandwidth by eliminating unnecessary duplication of messages. Messages branch-out at routers only when it is needed to reach receivers on different paths. Multicast protocols can be categorized into inter-domain and intra-domain protocols. Inter-domain multicast protocols are used when the receivers are not on the same network. Therefore, inter-domain multicast requires the assistance of the network layer of intermediate routers to forward the messages. Multicast Backbone (MBone) [8] is an inter-domain multicast protocol that provides delivery of messages to a group of receivers across an inter-network. MBone is commonly used for broadcasting video across the Internet. Border Gate Protocol (BGP), which is in use widely, is a robust inter-domain routing protocol responsible for exchanging routing information. Multicast BGP (MBGP) [4] is an extension of BGP to enable inter-domain multicasting and multicast routing policy. Intra-domain protocols are designed for LAN protocols, such as Ethernet, that operate at the data link layer. The Internet authority has reserved the addresses in the range 224.0.0.0 to 224.0.0.255 for multicasting on a LAN segment. No router would forward these messages with such addresses, as the Time-to-Live (TTL) is set to 1. The addresses 224.0.1.0 through 238.255.255.255 are reserved as global multicast addresses used for message multicasting across the Internet. One major challenge with implementing a protocol is managing the multicast group membership. During the multicast process, it is possible for a given set of nodes to join the multicast group, or to leave the group. Internet Group Management Protocol (IGMP) is a protocol that allows routers to keep track of group membership [9]

2.1 Fault Tolerance

Fault Tolerance is referred to the ability of a given system to function properly and accurately in the presence of faults. This is possible when some redundancy is introduced to the system. The degree of redundancy, which can be a mixture of hardware, software, and time, is a design time decision based on factors such as the number and types of faults. The masking or detection of failure is usually done by increasing the amount of communication among the nodes and imposing an agreement algorithm within the system. The objective of the agreement algorithm is to ensure that every non-faulty node agrees on the value of the message that is going to be delivered to the application (user). The best known agreement algorithm is the Byzantine General Problem (BGP) [11, 20], wherein all the receiving nodes need to decide on a value broadcast by a single node. The authors demonstrate that agreement is guaranteed if the number of faulty nodes t that behave in Byzantine manner is less than one-third of the total number of nodes N, i.e., $N > 3t$.

Although there are many different types of faults, the behavior of a fault can be generalized into one of the following categories [1, 23, 25]:

- *Benign* - These faults share the attribute of being obvious and self incrementing to every node. For example, when a value that is broadcasted is outside of an expected range. Often, this class of faults is the least complex to accommodate for.
- *Symmetric* - In contrast to benign faults, the errors caused by symmetric faults may not be distinguished from legitimate ones. These faults can be in transmissive or omissive form. More specifically, in a *transmissive symmetric* (TS) behavior, the same erroneous message is received by all receivers, e.g., a faulty node broadcasts a wrong time-of-day. In the *omissive symmetric* (OS) behavior, no message is received. For example, a message becomes corrupted in transmission in such a way that it is detectable and thus discarded by every correct node. These faults are more common in multi-point networks.
- *Asymmetric* - These faults are not constrained by any assumption regarding their behavior.

These faults can be partitioned into *multi-valued asymmetric* (MVA), *strictly omissive asymmetric* (SOA), and *single error asymmetric* (SEA). In a MVA fault, also called a Byzantine fault, the receiving nodes perceive conflicting messages with regard to the same message. These faults are common in point-to-point networks and difficult to achieve in multi-point networks [2]. When a SOA fault occurs, each correct node receives either a single correct message or no message at all. This behavior often occurs in multi-point networks, where a single correct message broadcasted is not received by some nodes. Finally, in SEA behavior, the same dubious message transmitted is received by at least one node but not by all of them. This mode of behavior, for example, can occur in a broadcast environment where a malicious message is lost in transit and so it is not received by some receivers.

2.2 Multicast Communication

Approaches to multicast protocols can be viewed as: 1) distributed versus centralized, 2) tree-based, or 3) ring-based. These protocols are not necessarily exclusive of each other. In a centralized control, a node has the responsibility of making sure the messages are received by all the group members. The group members send their messages to the central node before being broadcast to all members. In a distributed approach, all members share the responsibility of delivery and making sure everyone receives the broadcast message. The advantage of the distributed approach is the fault tolerance aspect of the protocol in that no single point of failure exists. In the tree-based protocols, the hosts are divided into groups, and each group is headed by a single leader. This way, the group members are likely to be leaders of other groups. The main advantage of such protocols is the reduction in message acknowledgments sent by the receivers, as each group leader is responsible only for one group and not for the entire receivers. Finally, the premise of ring-based protocols is to support atomic and total ordering for messages transmitted. For instance a site that has the token is allowed to send its messages to the group members, and is responsible for making sure all group members receive its messages. By having the token pass through the group members in a predetermined order provides total message order [6, 10, 13].

In general, there are four categories of multicast protocols with regards to message ordering [5]:

- *Unordered delivery*, in which the multicast protocol does not make any guarantees about the order in which the messages are accepted by different hosts.
- *FIFO-ordered* delivery requires the multicast protocol to deliver incoming messages from a given host in the same order in which the host originally sent the messages.
- *Causally-ordered delivery* means that the messages are delivered to the upper layers while preserving the causality between the messages. In other words, if in a given multicast group, message m_1 causes (precedes) m_2, and m_2 causes m_3, all receiving hosts will deliver m_1 before m_2, and m_2 before m_3.
- *Totally-ordered delivery* means that the messages are delivered in the same order to the upper layers by all nodes.

3 Network Design and Protocol

At times, broadcast and multicast of messages can be used interchangeably. However when needed, distinction between the two forms of message transmission will be made to remove any confusion. We assume the following:

A1. *Unordered-delivery of messages is assumed.* Each message consists of a header and a payload. The header contains information such as an identifier consisting of the node identifier and the sequence number of the message. The sequence numbers distinguish among messages coming from the same node. To further distinguish among messages with the same sequence numbers being transmitted from different nodes, each sequence number is pre-pended with the node identifier. The payload contains information like data, ACK list, NACK list, and rebroadcast information to distinguish the message from a message broadcast for the first time.

A2. *A message broadcast by a non-faulty node is received by at least t other non-faulty nodes.* As no trust can be placed on the faulty nodes, a non-faulty node must leave a trace of the message among other non-faulty nodes. This will be further explained in Section 4.

A3. *A node cannot impersonate another node.* That is, a faulty node cannot create a message and pretend that it originated from a different node. Furthermore, the sequence numbers for new messages are generated in hardware, so that a node would not be able to generate two new messages with the same sequence number.

A4. *A receiving node can not determine whether a message broadcast for the first time is from a faulty or a non-faulty node*.

The following conditions are referred to as Reliability Conditions (RCs):

- *Agreement:* A message delivered by a correct node is eventually delivered by every correct node.
- *Integrity:* A message is delivered only once by every correct node.
- *Validity:* A message broadcast by a correct node will eventually be delivered by the node.

The inclusion of the Validity condition may seem unnecessary as a node always delivers its own message. However, it is necessary to ensure that this condition in combination with the Agreement condition guarantees the delivery of the correct messages by all correct nodes.

3.1 The Fault Model

Most multicast protocols assume message omissions are due to accidental failures. However, a faulty node can create confusion among hosts, or even behave asymmetrically. The following shows a series of scenarios that a malicious host can take advantage of:

A. *A malicious host sends a NACK for a message that it has already received*. In this scenario, a malicious node adds to its negative acknowledgment list a NACK for a message that it has already received. The faulty node will then append the list to the next outgoing message. This scenario impersonates a legitimate message omission, as it will cause the non-faulty nodes to rebroadcast the message that they think is not received by a node. If it happens frequently, it can degrade network performance, causing a situation similar to the "denial of service" attack. The non-faulty nodes would then spend more time on broadcasting unnecessary messages.

B. *A malicious host sends a NACK for a message that has not yet been sent*. This scenario will cause the non-faulty nodes to also NACK the unsent message. As the message was never sent, it will never be received by any of the non-faulty nodes. These negative acknowledgments will cause the non-faulty nodes to continuously NACK the message. For this scenario to occur, the message sequence number must be higher than that of the last message sent by the same node. Consequently the unsent message would be NACKed until a message with that sequence number is transmitted. Since a NACK identifies the original source of the message, it is unwise to ask the originator to intervene, as the originator might be one of the faulty nodes in the system.

C. *A malicious host sends an ACK for a message that it has already acknowledged*. This scenario will not cause harm to the correct operation of the network, since these ACKs will be ignored by all non-faulty nodes that have received the message.

D. *A malicious host sends an ACK for a message that has not yet been sent*. This scenario occurs when a malicious node adds to its positive acknowledgment list an ACK for a message that was never sent previously. The fallacious positive ACK will then be piggybacked to the next outgoing message sent by the malicious node. This scenario causes the non-faulty nodes to continuously NACK the unsent message, which might degrade the network performance if there is a large gap between the sequence number of the last message sent and the sequence number of the unsent message for which an ACK is transmitted by the faulty node.

E. *A malicious host alters a message before it rebroadcasts the message*. This scenario can occur when a malicious node receives a NACK. Whether the faulty node has received the message, it can broadcast a falsified message in response to the NACK. Consequently, the non-faulty node that sent the NACK cannot determine which copy of the messages rebroadcast by other nodes is the original message.

Scenario C would not violate the RC. The impact of scenarios B and D are the same, in that the non-faulty nodes will continuously NACK the unsent message. Therefore, items B and D require the same solution, and a solution to either item is a solution to the other one. Capitalizing on different malicious behavior discussed so far, the scenarios A - E can be condensed into the following cases:

M1. *A malicious node sends a NACK continuously for a message that has already been sent, with the intent to flood the network with messages, causing performance breakdown*. This will address scenario A discussed above.

M2. *A malicious node NACKs a message that was never sent, with the intent to cause other nodes to also negatively acknowledge the unsent message*

continuously. This will address scenarios B and D discussed above.

M3. *A malicious node rebroadcasts a modified message or a manufactured message, pretending that the message was received from another node, with the intent of causing the hosts to accept the wrong message or creating discrepancies in message content received by different hosts*. This is scenario E discussed above.

In addition to the three categories of malicious behavior discussed above (M1-M3), a Byzantine node can behave omissively, i.e., it can behave OS by not sending messages at times. A faulty node can be perceived to have behaved in an SEA manner as well, in that some of its messages are lost in transmission. Additionally, the transmission medium might be able to behave in SOA. For instance, if a message transmitted by a correct node is not received by some nodes.

For a MVA error to occur, a Byzantine node must be able to transmit a message in more than one form to multiple receivers, or a message must be transformed into multiple, undetectable forms during transmission, due to noise or hardware failure. Both faulty behaviors are difficult to achieve, if not impossible, in a shared medium such as Ethernet [2], and thus are not considered in this research. However, these faults need serious consideration in ultra-dependable systems, such as flight-control systems, which have a failure rate requirement in the order of at least 10^{-9} [1, 7, 19].

We assume, however, that Byzantine failure at the protocol level is still a possibility. A faulty node can behave as in M3 at different times, to create inconsistency among different nodes with respect to a specific message. For example, a faulty node can broadcast a different form of a message any time it receives a NACK for the same message identifier.

Based on the previous discussion, the behavior of the t faulty nodes can be perceived as: TS, OS, SEA, or MVA at the protocol level. In addition, the benign faulty nodes are recognized globally and thus are removed beforehand. Therefore, $n = N - b$ is the number of nodes that participate in the multicast process, where b is the total number of Benign faulty nodes. It will be shown that the RCs are guaranteed if $N \geq 2t+b+2$, where N is the number of nodes in the system and t is the maximum number of faulty nodes. Furthermore, it will be shown that the RCs are preserved under scenarios M1-M3, which in turn imply that scenarios A - E are covered.

3.2 The Protocol Approach

The following rules guarantee the RCs:

1. *A non-faulty node accepts a message transmitted for the first time*. Hence, a node receiving a message with a new identifier can not determine whether the message originated from a malicious node (See A4). A corrupted, not correctable message is simply discarded.

2. *In response to NACKing a message, a non-faulty node accepts a copy of the message if* $(t+1)$ *identical rebroadcasts of the message are received from different nodes*. The $(t + 1)$ identical messages ensure that at least one of the rebroadcasts has emanated from a non-faulty node. This further ensures that NACKing by a legitimate node in response to an ACK or NACK by a faulty node does not result into accepting a bogus message. This addresses cases M2 and M3.

3. *A message is ignored if* $(n - t - 1)$ *NACKs from different nodes are received for the same message*. Therefore, if faulty nodes refrain from providing a NACK, a non-faulty node needs to receive a NACK from every other non-faulty node. To further limit the overhead caused by a malicious behavior, a bound can be placed on the number of rebroadcasts for each node or a message. This covers case M1.

If a message is received by some correct nodes, every non-faulty node will eventually receive $(t + 1)$ identical rebroadcasts of the message. Therefore, the Agreement and Validity conditions will be guaranteed. As each new message contains a different sequence number and each correct node can keep track of the identifier of the last message delivered, the Integrity condition will also be preserved. Furthermore, the requirement of $(n - t - 1)$ NACKs attempts to minimize network traffic and in the worst case ensures network liveness, by preventing the non-faulty nodes to continuously engage in transmitting a NACK message. A node will stop sending a NACK if it either receives $(t + 1)$ identical messages, thus accepting the message, or it will ignore the message if it receives $(n - t - 1)$ NACKs from different nodes.

Implicit in the above discussion is the fact that no non-faulty node will relay a message to another node, unless it accepts the message first, as the result of receiving $(t + 1)$ identical messages. Otherwise, a non-faulty node might relay a dubious message, and thus contribute to the total $(t + 1)$ needed by another non-faulty node to accept the message.

In response to a NACK, since it is possible that all $(t+1)$ rebroadcasts are from the non-faulty nodes and the t faulty nodes refrain from broadcasting, the total number of nodes in the system must be $n \geq (t+1)+t$. Counting the number of benign faults and the node

itself receiving $(t+1)$ identical messages from other nodes, the lower bound on the total number of nodes is $N \geq 2t+b+2$. This also implies that there must be at least two non-faulty nodes in the system to ensure the RCs are meaningful.

4 Protocol Simulation

A simulation program has been developed to test the protocol and examine the network operation performance [12]. The program is able to handle M1 - M3, through the behaviors in the form of SO, SOA, TS, and the protocol level TA as described in Subsection 3.1. The simulation runs show that the RCs can not be guaranteed if $n \geq 2t+2$ is not satisfied.

If a node does not receive a message due to an omission, it is highly likely that the message will be received during one of the rebroadcasts of the message. Obviously, the lower the omission probability, the sooner the message will be received. If the omission probability for each node is p, the probability that a message is not received by a specific node after r broadcasts is p^r. Fig. 1 displays this probability for r ranging from 1 to 20, and p ranging from 0.1 to 0.9.

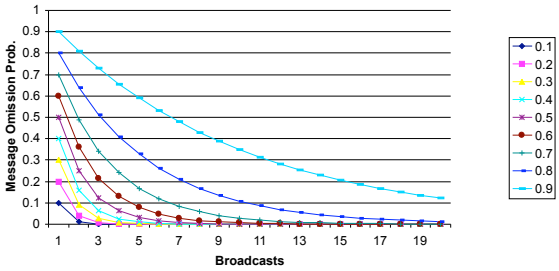

Fig. 1: Probability of a message not received after b broadcasts.

As the figure shows, the number of rebroadcasts decreases sharply as the omission probably is reduced. At around 0.50 omission probability, a message is received within the first 10 rebroadcasts. To better understand the effect of omissions, the simulation program is run with 10 nodes, 10 messages per node, ranging omission probabilities from 10% to 90%, and m broadcasts of each message, ranging m from 1 to 30. For each value of m, one broadcast and $(m-1)$ rebroadcasts are done to all nodes. An original message is marked as "omitted" if there exists a node that does not receive the message in m attempts. Therefore, the minimum and maximum number of original messages omitted can be 0 and 100 respectively. For each combination of omission probability and broadcasts, the program is run 5 times. The total number of original messages omitted are then averaged over the 5 runs. Although the results of one simulation might be somewhat different from another, Fig. 2 shows consistency with Fig. 1 in the fast tendency of messages being received when the omission probability is low or moderate. Hence, there would not be a real need to create extra overhead caused by an agreement process for each new message, which can cumulatively be too high. Each node can independently deliver a message received, knowing that other nodes would eventually do the same. Note that Fig. 1 is with respect to a single particular node, whereas the messages not received in Fig. 2 is with respect to 10 nodes.

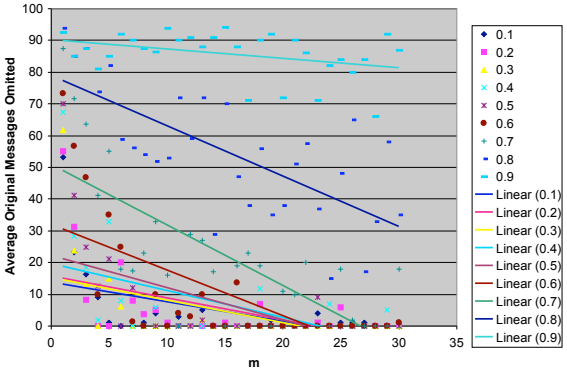

Fig. 2: Downward trend of messages not received.

The next simulation run shows the average number of non-faulty nodes that receive the original broadcast of a message. The program is run with t set to 2 for a network of 10 nodes. Each node is set to broadcast 10 messages, and the omission probability is ranged from 0.10 to 0.90. For each omission probability, the program is executed 10 times. As a result, a total of 1000 messages is collected for each value of omission probability. The total of messages are then averaged to show the average number of nodes receiving a message on the first broadcast. Fig. 3 shows, for

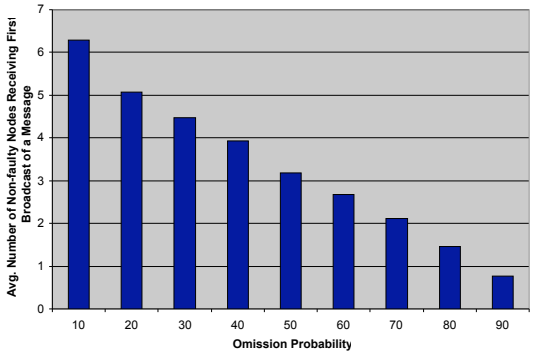

Fig. 3: Average Number of non-faulty nodes receiving first broadcast of a message.

omission probability as high as 0.80, that at least one non-faulty node receives a message. For 0.90 omission probability, the average number of non-faulty

nodes receiving a message is 0.76. Although, the condition that a message broadcast for the first time must be received by at least one non-faulty is necessary, the following discussion shows that this condition is not sufficient to guarantee the RCs.

An interesting question is to obtain the total number of broadcasts to deliver the messages originated from the network nodes in the presence of omission and malicious faults. A related question is to find the overhead in broadcast per message. A simulation is run for a network of 20 nodes, each broadcasting 10 messages. The number of malicious nodes, which are capable of modifying messages, ranges from 0 to 9 and the omission probability of the network is set to 0.10. Thus, there is a total of 200 original messages for each configuration. The program is then run 10 times, for a total of 2000 messages. Fig. 4 shows the total number of messages sent between the nodes until all non-faulty nodes accept and deliver all the messages.

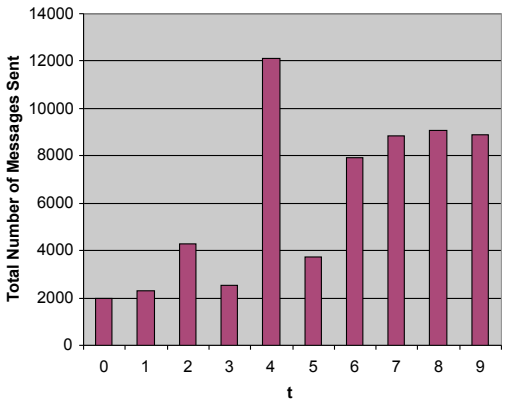

Fig. 4: Total number of messages sent to deliver 2000 messages.

Fig. 5 shows the average message overhead per original message for the previous experiment. Assume T is the total number of messages sent, and M is the number of original messages in the network. Then:

$$\text{Overhead per message} = \frac{T-M}{M} = \frac{T}{M} - 1$$

This figure shows that the maximum average overhead occurs when $t = 4$, with $T \approx 12000$ and $M = 2000$. The average message overhead is about $(0+5)/2 = 2.5$. Furthermore, the figure shows that message overhead is not much when the total number of faulty nodes is moderate, that is when the number of faulty nodes is less than half of what the network can endure to operate properly. However, one should observe that this is a simulation, so every run might yield a different result. To be close to reality, the average overhead was found over 10 different configurations.

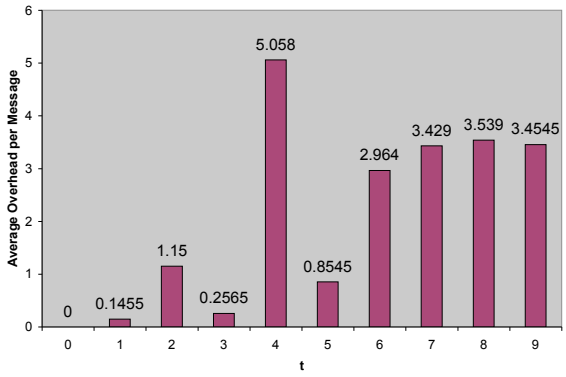

Fig. 5: Average number of extra messages broadcast to deliver an original message.

Consider a situation where $n = 10$, $t = 2$, and a message broadcast by a non-faulty node A is received by only one other non-faulty node B. Node B will ACK the message causing other non-faulty nodes to NACK the message. At this point, other non-faulty nodes would not be able to receive $(t+1) = 3$ identical messages, as there are only two non-faulty nodes having a copy of the message. Additionally, a non-faulty node C that did not receive the message might not be able to receive $(n-t-1) = 7$ NACKs to discard the message. As there are 6 non-faulty nodes that did not receive the message, if the faulty nodes decide not to send in a NACK, then C would receive only 6 NACKs. As a result, node C would not be able to accept the message, and would not be able to discard the message either. A similar problem might occur if the message is originally broadcast by a faulty node and received by less than $(t+1)$ non-faulty nodes. Consequently, to avoid this situation, it is assumed that a new broadcast must be received by at least $(t+1)$ non-faulty nodes. Fig. 3 depicts the result of an experiment with the same value of $n = 10$ and $t = 2$. The figure shows that with omission probability as high as 0.5, on average at least $(t+1) = 3$ non-faulty nodes receive the original messages.

Consider a non-faulty node that broadcasts a message that needs to be received by at least t nodes of the remaining $(n-t-1)$ non-faulty nodes. Assume the omission probability is p. The probability that exactly

j non-faulty nodes do not receive the message is:

$$\binom{n-t-1}{j} p^j (1-p)^{(n-t-1)-j}$$

Therefore, the probability that at least t non-faulty nodes will receive the message is:

$$\sum_{i=0}^{(n-t-1)-t} \binom{n-t-1}{i} p^i (1-p)^{(n-t-1)-i}$$

Fig. 6 shows this probability graphically for p ranging from 0.1 to 0.5, with t set to 2 and increasing n starting at the minimum value 6. The figure signifies that the probability of at least t non-faulty nodes receiving a broadcast approaches 1 quickly as number of nodes is increased, especially when the omission probability p is not high. Also the graph shows consistency with the simulation results in Fig. 3 at $n = 10$.

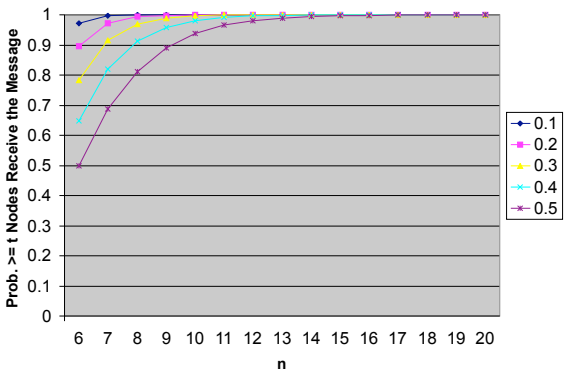

Fig. 6: The probability that at least t non-faulty nodes receive the first broadcast of a message.

5 Hierarchical Multicasting

Up to this point the multicasting paradigm was limited to only include nodes within a single network. Now we consider multiple networks, connected by gateways.

5.1 Simple connected networks

Fig. 7 shows a network consisting of two subnetworks A and B that are connected by gateways. There are a total of n_g gateways directly connected to the two neighboring subnetworks. As in the previous discussion, each subnetwork uses a single shared medium. Two multicast scenarios are now possible. Either the multicast spans over a single subnet or it spans over both subnets, i.e., the multicast includes nodes in networks A and B. Since single networks have been already discussed in the previous sections, the focus will be on the latter scenario.

Let's consider the connectivity of the two subnets. If $n_g = 1$, then all communication between A and B passes though the same gateway. In configurations with $n_g > 1$, the gateways operate in a redundant mode where each gateway forwards messages to all subnetworks that are directly connected to and have nodes participating in the multicast. Note that this is different from standard network configurations where only one gateway usually forwards packets, i.e., the configuration builds a spanning tree and multiple gateways play only a role if an active gateway in the spanning tree fails.

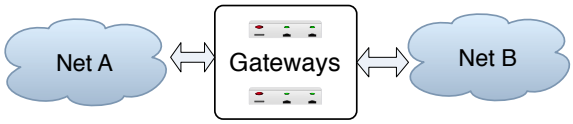

Fig. 7: Configuration with two networks

Consider the case where the subnetworks A and B communicate via a single gateway, i.e., $n_g = 1$. Furthermore assume that a message broadcast originates on A and the multicast membership includes nodes in both subnets. Since the gateway is the only point of connection between the two subnetworks, the gateway will rebroadcast the message onto subnet B. However, the nodes in subnet B will not be able to receive the required $(t + 1)$ identical copies of the same message from different nodes to accept the message. Additionally, the gateway can not forward the message onto B as a new one, since the message did not originate from the gateway. Furthermore, if the gateways are treated as standard network nodes, the requirement to receive $(t + 1)$ identical copies would require the number of gateways to be at least $(2t + 1)$. However, this large number of gateways most likely is not practical for $t > 2$.

Therefore, to incorporate multiple subnetworks connected by gateways, the fault model needs to be adapted. Up to this point we considered t to be the number of faulty nodes. However, it seems reasonable to treat standard network nodes and gateways differently with respect to failure probability. It can be argued that a gateway is more reliable, or less prone to failure, due to the fact that it is not used as a general purpose computer and it is a special piece of hardware with dedicated software.

Let n_a and n_b denote the number of nodes in network segment A and B that participate in the multicast respectively, i.e., $n = n_a + n_b$. The n nodes are in addition to the n_g gateways. Furthermore, let t_n denote the number of faulty nodes in all subnetworks

and let t_g be the number of faulty gateways between the two subnets, with the failure probability of a gateway to be much less than that of a network node, i.e., $t_g \ll t_n$.

It can be shown that to tolerate the failure of t_n nodes and t_g gateways, one needs

$$n \geq 2t_n + 2$$

nodes and

$$n_g \geq 2t_g + 1$$

gateways, where no assumption is made about the distribution of the faulty nodes in the different subnetworks. Hence, the bound for N is the same as that in the single network case, i.e., $N = n_a + n_b + b$.

The criterion for accepting a message via the broadcast paradigm across the gateways is different. Since authentication is assumed, i.e., no gateway can impersonate to be a different node, each receiving node can recognize whether a message is broadcast by a gateway. Therefore, a simple majority message forwards by the non-faulty gateways is sufficient. Consequently, a receiving node can vote on a correct message using a $(2t_g + 1)$ majority of received messages. This voting effectively constitutes gateway fault masking and can be viewed as a restoring organ.

The overhead resulting from sending a single message to another network is of factor n_g plus the overhead associated with voting. Note that this only applies to messages sent across subnetworks and not to messages sent to the same subnetwork that contains the originating node. Furthermore, since the failure rate of a gateways is assumed significantly lower than that of a network node, t_g and thus n_g will be small. For example, t_g is likely to be equal to 1 or 2 resulting in 3 or 5 gateways, respectively.

A multicast network can be represented by a special kind of network graph in which the vertices are nodes or gateways and the edges are *logical* network connections. Given two graphs G_i and G_j with respective vertex sets V_i and V_j and edge sets E_i and E_j, the *union* $G = G_i \cup G_j$ has $V = V_i \cup V_j$ and $E = E_i \cup E_j$. The *join* $G = G_i + G_j$ consists of $G_i \cup G_j$ together with all edges joining V_i and V_j, i.e., $\forall v_p \in V_i$ and $\forall v_q \in V_j, e_{p,q} \in E$, the edge set of the join graph. Fig. 8 shows an example of a join graph. The edges defining the join operation between G_1 and G_2 are shown as dashed lines. Note that the edge connectivity is logical, i.e., in the context of broadcasting, if a vertex emits a message, all connected vertices receive the message (in the error-free case). This should not be confused with point-to-point messages, where vertices emit messages to neighboring vertices on one edge at a time.

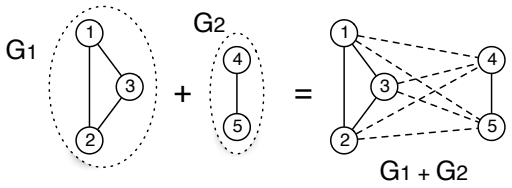

Fig. 8: Join operation (+) of two graphs

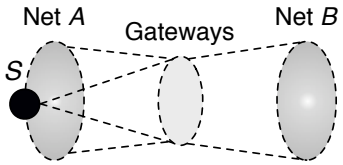

Fig. 9: General join graph of two subnetworks

Let's revisit the requirement that a new broadcast must be received by at least t_n other non-faulty nodes in the context of multiple subnetworks. The fact that these nodes may now span over two neighboring networks has no consequence other than the overhead associated with sending $(2t_g + 1)$ messages for each message that needs to cross gateways. As long as no more than t_g gateways are affected by failure or malicious act, each node which receives forwarding messages from gateways will be able to determine whether it should accept the message, i.e., if $(t_n + 1)$ identical messages are received, or discard the message, i.e., if $(n - t_n - 1)$ NACKs are received.

The join graph associated with Fig. 7 is shown in Fig. 9. Node S broadcasts to the nodes in the two subnetworks, shown in dark gray, and the gateways are depicted in the light-gray shaded oval. Formally, node S broadcasts on subnetwork A and to the gateways. The gateways, by the nature of the join operation, forward the messages to all participating nodes in B, which vote on the messages received from the gateways. The additional overhead associated with the gateways is captured by the join operation of the two subnetworks, when a new message is broadcasted for the first time. Specifically, a new message broadcasted onto subnetwork A needs to be rebroadcasted by at most n_g gateways. But, the message overhead due to rebroadcasts crossing the gateways is less. This is because the number of gateways is small, and thus $(t_g + 1)$ identical copies are needed to accept a message in comparison to $(t_n + 1)$ needed by the nodes on the originating subnetwork. For example, three gateways can mask one failure, which is a relatively small price to pay.

5.2 General hierarchical multicasting

Whereas the previous subsection considered the special case of two subnetworks connected by gateways, this subsection is concerned with general network configurations. Consider a scenario with n_s network segments that are interconnected with gateways. Fig. 10 shows a configuration with $n_s = 4$ and depicts the possible scenarios in which subnetworks can be connected with gateways. The segments A and B are connected via a group of gateways, which are not connected to any other networks. On the other hand, networks A, C, and D share gateways. In fact the gateways connecting A and D are a subset of the gateways connecting A and C. The network model of the previous subsection can now be extended to multiple subnetworks. The following cases can be enumerated, where the associated network interconnections are indicated in parenthesis:

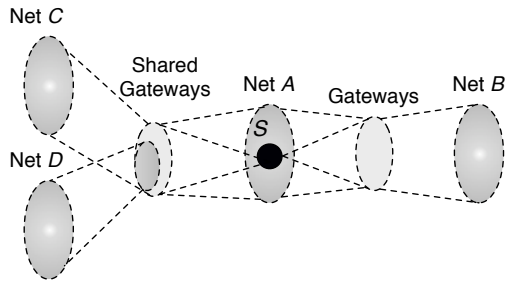

Fig. 10: General join graph with multiple networks

1. *Two subnetworks are interconnected by non-shared gateways*. This is the scenario for subnetwork (A, B), where the number of non-shared gateways is $n_{g(A,B)}$. In order to mask $t_{g(A,B)}$ gateway faults, at least $(2t_{g(A,B)} + 1)$ gateways are needed.

2. *Multiple subnetworks share gateways*. This is the case involving subnetworks A, C, and D. To mask $t_{g(A,C)}$ gateway faults, a total of $n_{g(A,C)} \geq 2t_{g(A,C)} + 1$ gateways are needed. Similarly, a total of $n_{g(A,D)} \geq 2t_{g(A,D)} + 1$ gateways are needed to mask $t_{g(A,D)}$ gateway faults. However, with respect to $n_{g(A,C)}$ no more than $t_{g(A,D)}$ of the shared gateways (depicted by the darker shaded oval) may fail, and the remaining $(t_{g(A,C)} - t_{g(A,D)})$ faults must occur on non-shared gateways (drawn in a lighter shade of gray in the oval).

The discussion above assumed subnetworks to be only one hop away from the initial sending node. Multiple hops are considered in Fig. 11. There is no assumption about the distribution of the t_n faulty nodes, e.g., they could be concentrated in specific subnets or distributed randomly over all subnets. In the first case, multi-hop messages are forwarded from one group of gateways to the next and due to the majority argument of $(2t_g + 1)$ messages, there will always be a majority of correct messages relayed to the next hop.

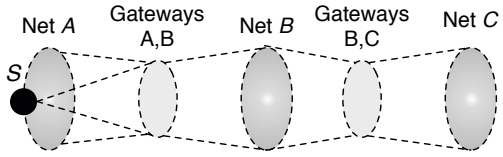

Fig. 11: General join graph with multiple hops

Considering n_g gateways between any two neighboring subnetworks and n_s segments, so that $n = n_1 + n_2 + \ldots + n_{n_s}$, the bounds in the previous subsection regarding the two subnetworks still hold for the n_s segments, i.e., $N \geq 2t_n + b + 2$ and $n_g \geq 2t_g + 1$. Similarly, the overhead associated with a multicast spanning over multiple n_s subnets is induced by the message traffic of the redundant gateways, i.e., $(2t_g + 1)$ messages per subnet hop, and it is bound by a total of $n_s n_g = n_s(2t_g + 1)$ gateways. If gateways are shared, as shown in Fig. 10 for networks A, C, and D, the number of gateways is reduced by the number of shared gateways.

6 Conclusion

Using a broadcast medium, it is difficult or unlikely that messages can be received asymmetrically, but it has been shown that an asymmetric behavior is still possible by allowing a faulty node to broadcast different messages for a given original message at different times. It has been shown that $(t_n + 1)$ identical messages are needed to deliver a message if the message is not received in the first broadcast, and $(n - t_n - 1)$ NACKs are needed to ignore a message. The research did not discuss transmission of messages by the use of digital signatures. If messages are digitally signed, the receipt of only one digitally signed message is sufficient to accept the message, instead of $(t_n + 1)$ identical messages. But there would not be any changes in $(n - t_n - 1)$ NACKs that are needed to address M1.

The protocol assumes that a faulty node can behave in OS, TS, and TA manners at the protocol level. In addition, it is assumed that the network medium can temporarily experience omissions in the form of OS or SOA, due to different scenarios such as loss of

a message or not being able to correct errors.

It has been shown that to tolerate the failure of t_n nodes and t_g gateways, one needs $N \geq 2t_n + b + 2$ nodes, possibly spread over n_s subnets, and $n_g \geq 2t_g + 1$ gateways, connecting any two adjacent subnets.

One of the requirements in achieving reliable multicasting is reaching agreement in delivering a message. Normally, a Byzantine agreement algorithm is needed to ensure that every non-faulty node agrees on delivering a message. The process of reaching this agreement for every message will become very prohibitive if new messages are transmitted continuously, or if a point-to-point topology were used. The proposed protocol reduces message complexity, as each node can independently determine the safety of delivering a message to the user. Furthermore, simulation has shown message complexity is dynamically adjusted according to the level of network fault condition, whilst the correct operation of the network is maintained.

References:

[1] M.H. Azadmanesh, R.M. Kieckhafer, Exploiting Omissive Faults in Synchronous Approximate Agreement, *IEEE Transactions on Computers*, Vol.49, No.1010, 2000, pp. 1031-1042.

[2] O. Babaoglu, R. Drummond, Streets of Byzantium: Network Architectures for Fast Reliable Broadcasts, *IEEE Transactions on Software Engineering*, Vol.SE-11, No.6, 1985, pp. 546-554.

[3] F. Barsotti, A. Caruso, S. Chessa, The Localized Vehicular Multiast Middleware: A Framework for Ad Hoc Inter-Vehicles Multiast Communications, *WSEAS Transactions on Communications*, Vol.5, No.9, 2006, pp. 1763-1768.

[4] T. Bates, Multiprotocol Extensions for BGP-4, *Network Working Group*, RFC 2858, http://faqs.org/rfcs/rfc2858.html, 2000.

[5] K. Birman, T. Joasph, "Communication Support for Reliable Distributed Computing", *Lecture Notes in Computer Science*, Vol.448, 1987, pp. 124-137.

[6] X. Defago, A. Schiper, P. Urban, Total Order Broadcast and Multiast Algorithms, *ACM Computing Surveys*, Vol.36, No.4, 2004, pp. 372-421.

[7] K. Driscoll, B. Hall, H. Sivencrona, P. Zumsteg, Byzantine Fault Tolerance, From Theory to Reality, *Lecture Notes in Computer Science* (LNCS), Computer Safety, Reliability, and Security, Vol.2788, 2003, pp. 235-248.

[8] H. Eriksson, MBone: The Multicast Backbone, *CACM*, Vol.37, No.8, 1994, pp. 54-60.

[9] W. Fenner, Internet Group Management Protocol, *Network Working Group*, RFC 2236, http://faqs.org/rfcs/rfc2236.html, 1997.

[10] K.P. Kihlstrom, The SecureRing Group Communication, *ACM Transactions on Information and System Security*, Vol.4, No.4, 2001, pp. 371-406.

[11] L. Lamport, et al, The Byzantine Generals Problem, *ACM TOPLAS*, Vol.4, No.3, 1982, pp. 382-401.

[12] D. Laqab, *Survivable Multicast Communication in Bus-based Networks*, MS Thesis, Computer Science Department, University of Nebraska-Omaha, 2006.

[13] B.N. Levine, J.J. Garcia-Luna-Aceves, A Comparison of Reliable Multicast Protocols, *Multimedia Systems*, Vol.6, No.5, 1998, pp. 334-348.

[14] X. Li, M.H. Ammar, S. Paul, Video Multicast over the Internet, *IEEE Network*, Vol.13, No.2, 1999, pp. 46-60.

[15] P.M. Melliar-Smith, L.E. Moser, Trans: A Reliable Broadcast Protocol, *IEE Proceedings*, Vol.140, No.6, 1993, pp. 481-493.

[16] P.M. Melliar-Smith, L.E. Moser, V. Agrawala, Broadcast Protocols for Distributed Systems, *IEEE Transactions on Distributed and Parallel Systems*, Vol.1, No.1, 1990, pp. 17-25.

[17] C.K. Miller, *Multicast Networking and Applications*, Addison-Wesley, 1999.

[18] L.E. Moser, P.M. Melliar-Smith, Byzantine-Resistant Total Ordering Algorithms, *Information and Computation*, Vol.150, No.1, 1999, pp. 75-111.

[19] M. Paulitsch, J. Morris, B. Hall, K. Driscoll, E. Latronico, P. Koopman, Coverage and the Use of Cyclic Redundancy Codes in Ultra-Dependable Systems, *Proceedings of the International Conference on Dependable Systems and Networks* (DSN), 2005, pp. 346-355.

[20] M. Pease, et al, Reaching Agreement in the Presence of Faults, *JACM*, Vol.27, No.2, 1980, pp. 228-234.

[21] C. Shih, T. Shih, Cluster-based Mulitcast Routing Protocol for MANET, *WSEAS Transactions on Computers*, Vol.6, No.3, 2007, pp. 566-572.

[22] H. Shin, K. Cho, A IP Multicast Technique for the IPTV Service, *WSEAS Transactions on Communications*, Vol.6, No.1, 2007, pp. 274-277.

[23] P.M. Thambidurai, Y.K. Park, Interactive Consistency with Multiple Failure Modes, *Proceedings of the 7th Reliable Distributed Systems Symposium*, 1988, pp. 93-100.

[24] B. Wang, C. Hou, A Survey on Multicast Routing and its QoS Extension: Problems, Algorithms, and Protocols, *IEEE Network*, Vol.14, No.1, 2000, pp. 22-36.

[25] P.J. Weber, Dynamic Reduction Algorithms for Fault Tolerant Convergent Voting with Hybrid Faults, PhD Dissertation, Electrical & Computer Engineering, Michigan Technological University, 2006.