Chapter 1

Introduction

1.1 Motivation

The last decade has witnessed the emergence and ubiquity of the Internet. This phenomenon has given rise to an ever-growing level of connectivity. The stress that exponential growth places on the network fabric can be observed in the expectations of users; namely, the ability to connect to the Internet at anytime and from anywhere, the constant availability of machines from which data is needed, the existence of bandwidth to carry data in both a reliable and timely manner, and the resilience of connections to faulty machines and links. Thus, the network is in dire need of solutions to support these demands. This thesis examines issues in the design of efficient, scalable network algorithms that are highly robust.

Much of the networking infrastructure for the Internet was designed under the assumption that machines communicate in a point-to-point fashion, i.e., communication occurs between pairs of machines. A sender addresses a message to a specific receiver and typically awaits explicit feedback to determine whether or not the information sent was properly received. Consequently many of the core Internet protocols were designed with a model that assumes direct and reliable exchanges between a single sender and a single receiver.

There are two aspects about this premise that have changed over time with regards to scalable group communication, by which we mean the ability for groups of communicating processes to grow very large in size. First, point-to-point messaging has given way to multipoint messaging. Various forms of efficient multipoint communication have evolved for use not only within the local area network (LAN), but also within the wide area network (WAN), facilitating multiway exchanges between groups of senders and receivers. Second, acknowledged messaging has been replaced by unacknowledged messaging. Depending on the application, messages sent to large groups of receivers may not require acknowledgment by the recipients. For example, an interactive conferencing application that distributes real-time audio does not retransmit missed speech segments. By the time the data would be retransmitted, the conference is well beyond the point where the audio is useful.

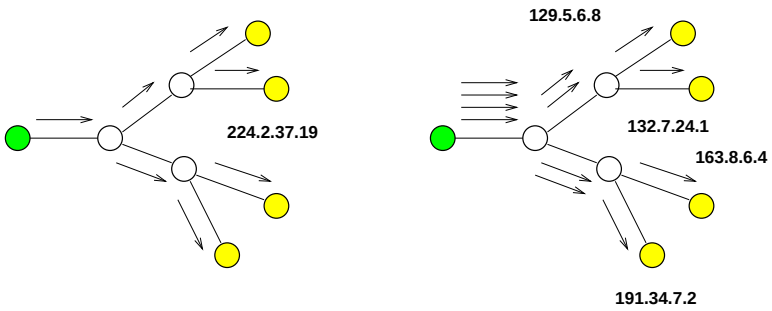

Figure 1.1: **Multicast vs. Unicast: Efficiency and Group Addressing.**

In this thesis, we examine the use of networking substrates that provide an efficient abstraction for multipoint communication, such as IP Multicast [20] as displayed in Figure 1.1. We use the terms multipoint and multicast interchangeably throughout this thesis. However, it is critical to note that our results, while assuming multipoint substrates, are independent of any specific implementation, and in particular independent of the choice of IP Multicast as the underlying implementation.

Multipoint substrates are interesting for at least two reasons. They provide an implementation of efficient broadcast. Copies of a message are made by routers at the branching points in the network, as the message is forwarded to a group of processes, members of which reside along different paths. Therefore, only one copy of the message is ever present on any link in the network. This is in stark contrast to replication at the source; an endpoint keeps track of the individual addresses for group members and disseminates an identical message to every participating process by initiating separate point-to-point interactions on a pair-wise basis. Multipoint substrates also provide the powerful abstraction of a group address. A process interested in communicating with all other group members simply uses the group address. A message sent to the group address is sent to all processes subscribed to the group at that point in time. Therefore multicast provides an efficient and thus scalable means to transmit data to groups of processes. Figure 1.1 shows these benefits by displaying the key differences between multicast and unicast communication: improved link usage and group addressing.

While multicast communication is inherently more scalable for groups of processes than its point-to-point counterpart, it still faces scalability challenges. For example, the traditional mechanism used to support reliable message passing in an unreliable network is for receivers to use acknowledgments (ACKs) to indicate when messages have been successfully received. If this mechanism is used to implement reliable group communication, then a sender must receive an ACK from each receiver participating in the group for each message sent. The number of ACKs grows with the number of receivers, potentially overwhelming the sender and thereby preventing scalability. This leads to the classic problem known as message implosion, not only at the sender but possibly at routers on the path back to sender (Figure 1.2).

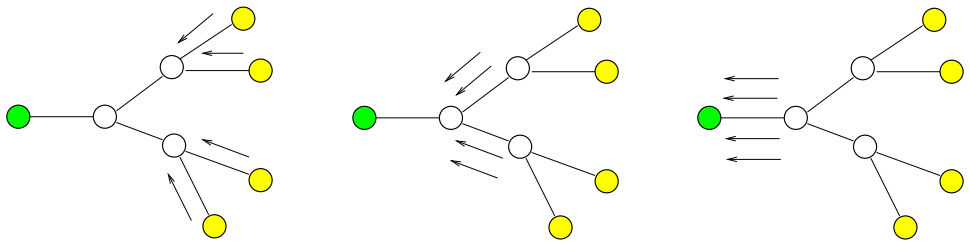

Figure 1.2: **Message Implosion Toward the Sender.**

Alternatively, receivers may send negative acknowledgments (NACKs) to indicate when messages have been lost. A NACK is usually triggered by the detection of a skipped sequence number. Although NACKs reduce the overall numbers of messages sent back to the sender, they do not avoid message implosion, which might occur when a whole branch of the multicast distribution tree loses the same message.

The protocols we examine in this thesis attempt to avoid message implosion by using completely unacknowledged messaging, which we refer to as announcements, for group communication. Messages are announced to a set of processes that passively wait to receive them. Of course, announcements in and of themselves are a form of unreliable communication. However, when an announcement is resent periodically ad infinitum, and the network is not permanently disconnected, the probability that the message reaches a receiver and eventually all receivers approaches 1.

Furthermore, when announcements are made periodically, each message serves to renew or to update local state information from the announcers. Such an approach is highly robust in the face of network faults, late process arrivals and process departures. Because incorrect state will be corrected by subsequent announcements, there is no need to perform any special sequence of events when errors occur. Thus, the system is designed to survive from the outset. Additionally, because all participating processes receive a copy of each message, all group members play an equal role in maintaining the state of the system and there are no single points of failure.

This thesis explores the use of periodic announcements as a replacement for acknowledged messaging. However, due to differences in transmission delays and message losses, the use of announcements may lead to inconsistent data among the different listening processes, as each participant may receive a different set of messages. As a result, this style of messaging commonly leads to the condition that each process is only able to approximate locally the global system state.

The tradeoff then in using announcements is that they avoid message implosion but may lead to inconsistency. We therefore are most interested in the class of applications that can tolerate transient inconsistency, so long as the property of eventual consistency is maintained. As evidence of these applications, a new class of protocols has emerged that are based on the principles of multipoint communication, often used in conjunction with announcement-style messaging. These protocols are

efficient, scalable, and highly robust as a result.

These protocols reside at a variety of levels in the protocol stack and provide a range of critical Internet services. The protocols range from algorithms for routing (IGMP [13], PIM [18]), to those for quality of service (RSVP [61]), real-time transport (RTP [54]), reliable multicast (SRM [27]), distributed directories (SAP [35], SIP [36], SRVLOC [34]), and more recently, even what we have come to think of as the Internet itself, the Domain Name System (DNS [58] [21]). These protocols represent both deployed and proposed Internet standards. Yet, the principles of multipoint and announcement-style communication have application beyond the Internet, in any context where distributed control arises.

We observed that many of these multipoint protocols rely on a small set of very powerful techniques, or micro-algorithms, in order to accomplish scalability. The main goal of this thesis, through analysis and simulation, is to evaluate several of these fundamental techniques so that designers can gauge their usefulness over a range of operating conditions. An analysis of these techniques provides insight not only into their behavior, but also into the behavior of more complex algorithms that rely upon them.

1.2 Techniques for Scalability

In this thesis, we focus on three of the techniques that are repeatedly used for scalable group communication: Suppression (SUP), Announce-Listen (AL), and Leader Election (LE).

Suppression. Suppression is the technique whereby a process inserts a random delay before message transmission. In the delay interval, if the local process receives a message from another process, then the local process suppresses the transmission of its own message. If not, the announcement is sent as intended.

Suppression is an important tool to employ when processes arrive into a system concurrently, as might happen when processes reboot after a failure, or when all processes respond to the same message received. Suppression takes what would have been a group of simultaneous transmissions and spreads them out over the delay interval, simultaneously making message implosion less likely and reducing the number of messages ultimately sent, since earlier messages will suppress later messages.

Announce-Listen. In Announce-Listen, a sender process disseminates information to a group of processes by issuing periodic multicast announcements, and receivers passively listen for these announcements. A listener process infers information about the global state of a system from the periodic receipt or loss of messages from announcer processes.

AL is used in lieu of acknowledged messaging in situations where there is little or no back-channel bandwidth, i.e., there is limited bandwidth in the direction from receivers to senders. As a result, AL is beneficial in those instances where usage of traditional ACKs or NACKs coming back from receivers would inundate a sender. In addition, AL is useful to employ when information changes happen at a high enough rate to warrant continual updates. Thus, AL messaging is resilient to faults in the network because state is continually replenished. AL is well-suited for scenarios where group membership is constantly in flux, as late-joiners will quickly learn the current state of the system.

Leader Election. Leader Election is an algorithm that identifies a single process from among a group of processes. The criteria for a process becoming leader is agreed upon apriori by all participating processes (e.g., largest process identifier, lowest processing load, highest degree of connectivity, et cetera). By listening to announcements, each group member independently determines who the leader is and independently detects when the leader has departed. Thus, leader election occurs through loose coordination of the processes rather than strict consensus, which might require several rounds of exchanges among all processes in order to converge to a single leader.

LE is similar in its effect to SUP in that it suppresses the number of messages generated by a group of processes. LE, however, more formally identifies a single "leader" process to act on behalf of the other processes. LE keeps the overhead of group messaging low by reducing the numbers of group members that need to issue messages. As a result, LE helps a group of processes to behave as if it were one process.

1.3 Network Model

In the protocols we study, their analysis, and the accompanying simulations, we assume that the network supports a multipoint abstraction. Any message sent to a group address is forwarded to all group members subscribed to the address at that time. There is also the assumption that all messages issued are announcements, in that messages sent do not trigger acknowledgments or negative acknowledgments in response.

Communication groups may change in size over their lifetime [4]. Our analysis primarily concentrates on the scenario where all processes arrive simultaneously into the system, and subsequently when steady state is achieved, i.e., the group membership is steady. We also discuss the impact of perturbations to the membership, where applicable.

We assume message loss can occur. As such, we analyze each algorithm using a probabilistic failure model. Traditional analysis for distributed systems provides hard guarantees for conditions that always hold (safety properties) or conditions that are guaranteed eventually (progress properties). The safety and progress properties that hold in the presence of unacknowledged messaging

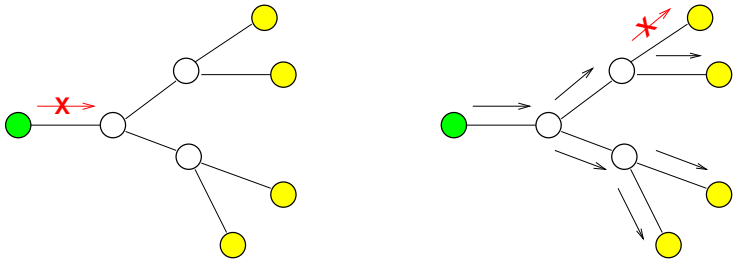

Figure 1.3: **Correlated vs. Uncorrelated Loss: Message Loss Near vs. Far from Sender.**

are very weak because, in the worst case, messages can be lost. Therefore, in this thesis, we devise probabilistic metrics to reason about the performance of each scalability technique.

Messages experience a fixed transmission delay across the network regardless of the location of the sender and receiver. For our calculations, the transmission delay actually encompasses transmission and processing delay, in that it captures the time that it takes for the receiver process to receive the announcement at the application level. We study fixed transmission delays as a prerequisite to understanding variable delays. The simplifying assumption that there are fixed delays makes the analysis tractable. However, the performance metrics that we derive for each scalability technique are still applicable in the variable delay case; provided we know the maximum (or minimum) delay between any pair of processes within the group, we are able to use the performance metrics to provide an upper (or lower) bound on performance for the group.

One challenge in modeling multicast traffic is that multicast distribution trees may lead to loss correlation (Figure 1.3). A message dropped close to the sender leads to correlated loss, as all receivers experience the same loss. A message dropped close to a receiver leads to uncorrelated loss, as it may or may not have been dropped at other receivers. A message dropped somewhere within the multicast distribution tree affects all downstream receivers. We evaluate the impact of both correlated and uncorrelated loss on protocol performance. We study the extreme cases where messages are dropped next to the sender or next to the receiver, in order to understand performance bounds. In the remainder of this thesis, we refer to these extremes simply as correlated or uncorrelated loss, respectively. In addition, we study the impact of multiple losses versus single losses on the behavior of the algorithms.

Another goal of this research is to expose key parameters that impact each algorithm and that have not been examined previously or completely. Consequently, we are able to study regions of the operating space that have not been investigated fully. This is accomplished in part by focusing on a multipoint context (as in the case of Announce-Listen) and in part by supporting more complex loss models (in the case of Suppression). Instead of looking at explicit topologies (as is often the case for Leader Election algorithms), we consider topologies in terms of the delay bounds they produce among processes.

By identifying a small but consistent set of parameters that are used in the mathematical formulae that describe each scalability technique, we enable protocol designers to properly tune their algorithms. For example, understanding the parameters allows us to answer typical questions such as: What is an appropriate timer value? How much delay is incurred when N processes participate? How much messaging overhead is generated when the loss probability is p? How long before there is a consistent copy of the data at all processes? Will an algorithm function properly when placed in a new context where critical network attributes are substantially different (e.g., using a protocol on a wireless network when it was designed originally for the wired realm)?

Bandwidth usage does not appear directly in the model. However, given a time interval T at which announcement messages are sent periodically, a message loss probability p, a network bandwidth capacity c, and a message size of s bytes, then we can define l to be an adjusted loss parameter that indirectly reflects bandwidth limitations. When the amount of traffic remains below c, the simple loss function p is maintained. When the bandwidth capacity is exceeded, only a fraction of the successful traffic gets through, thereby increasing the effective loss rate.

$$l = \begin{cases} p & \text{if } (s/T) \leq c \\ 1 - (1-p)(cT/s) & \text{otherwise} \end{cases}$$

Throughout this thesis, we primarily discuss the impact of p, a simple message loss probability, on the algorithms. However, in Chapter 5, we provide an example of the impact of congestion on p, which results in an effective loss rate that is much higher.

Not only do we aim to model and to analyze scalability techniques for group communication, but we use simulation to validate our results. All of the simulations presented in this thesis used the network simulator `ns` that was developed collaboratively by researchers at UC Berkeley, Lawrence Berkeley National Laboratories, USC Information Sciences Institute, Xerox PARC, and elsewhere [6].

1.4 Related Work

The use of probabilistic models builds on Chandy and Misra's work that explores the idea of process knowledge [15]. They formalize how to reason about what processes learn and assert that knowledge is only gained through message transmission. In later work [14], Chandy and Schooler show that process knowledge is too strong a concept for distributed applications in faulty environments and suggest a weaker but more appropriate concept of estimation. The algorithms we analyze in this thesis are part of the class of distributed problems that are best described with estimation probabilities. However, we analyze these algorithms using a more realistic loss model. We consider both single and multiple message losses, as well as the impact of correlated and uncorrelated losses on

the group of communicating processes.

In [10] and [39], reliable multicast is analyzed under probabilistic failure models. Birman et al design a multicast protocol with a bimodal delivery guarantee; when a message is sent to a multicast group, almost all or almost none of the destinations receive the message with high probability. A key departure from the multicast techniques we analyze is that they add synchronization and ordering properties on top of the basic multicast announcements.

The idea of composition, that smaller protocols can be composed into larger protocols, is related to several systems. ISIS [11] is a toolkit of functions that can be combined to create group communication protocols. ISIS provides a rich set of semantic building blocks that focus on tightly-coupled communication, where senders and receivers have acknowledged message exchanges. However, ISIS only offers a fixed set of semantic properties that can be combined. In response to this limitation, software libraries such as Horus [57] and the research of Bhatti et al [9] have decomposed network algorithms into finer grain micro-protocols. Both allow flexibility in the kinds of properties that can be composed into larger protocols. The Ensemble system [38] takes these design principles one step further by using formal methods to reason about the correctness of the resulting protocols. Nonetheless, these efforts have a slightly different emphasis than our work, which is aimed at providing a probabilistic analysis of the behavior of the component algorithms.

In a similar vein, the Reliable Multicast Working Group of the Internet Research Task Force has begun to examine a building block approach to designing reliable multicast transport protocols [40]. Because there is sufficient commonality between the many existing reliable multicast protocols, it should be possible to define a small number of components out of which the others can be constructed. The kinds of building blocks that they propose are at a higher level than the techniques we study in this thesis. Additionally, our focus is on the analysis rather than functional attributes of the components. However, we expect that many of the building blocks that emerge for reliable multicast transport will rely on the techniques for scalability that we study here.

In addition to identifying the key components to build larger protocols, we must pose the question of equivalence; can we evaluate an algorithm by evaluating the components out of which it is built? Hayden touches on this in his work on formal methods that allow him to reason about the correct operation of constructed algorithms [38], given the properties of the component algorithms. Here, we ask the related question of whether or not the performance analysis of the components has any bearing on the performance analysis of the combined algorithms.

In the area of complexity theory there is the related strategy of using a series of transformations to translate one algorithm into another. This technique is used to decide if an algorithm of unknown complexity can be transformed into an algorithm already known to be NP-complete [32]. In our research, we ask if the performance bounds of the algorithmic building blocks are indicative of the bounds for the complex algorithms that rely upon them.

For the specific scalability techniques that we study, we review related work in each of the chapters in which they are discussed.

1.5 Overview

In Chapters 2 - 5 we derive a series of performance metrics for each scalability technique. These metrics provide probabilistic bounds for the behavior of the algorithms under various network conditions. The metrics focus on the expected response time, network usage, memory overhead, consistency attainable, and convergence time. Given the metrics, a designer is better equipped to parameterize the algorithms and to make informed tradeoffs. Moreover, the performance of an algorithm can be predicted, given that specific operating conditions exist (e.g., a certain level of loss or delay among group members). Broadly speaking, these chapters identify the key parameters that affect this class of loosely-coupled multipoint algorithms, and establish a general methodology for the analysis of these and other scalability techniques.

In Chapter 2, we evaluate the Suppression algorithm initially with no loss in the network. We deliberately scrutinize its performance separate from any particular context, e.g., reliable multicast. As a stand-alone entity, the algorithm is not subject to assumptions about how it may or may not be used in subsequent iterations. We show clearly that Suppression's response metric is in delicate balance with its overhead metric, and examine a new timer distribution function that raises the issue of "designer" distribution functions in general. Given a target range for parameter values, and a metric of choice, we establish that a distribution function can be created to optimize the performance of the algorithm.

In Chapter 3, we revisit Suppression under lossy conditions. We re-examine the metrics established in Chapter 2, but updated to reflect message loss. We present and analyze several new metrics that are important for Suppression, both as a stand-alone algorithm and as an algorithm combined with other techniques: the completion or halting time of the algorithm, as well as the number of extra versus required messages generated. Most importantly, we model loss more realistically than previous work on Suppression [27] [50] [45], allowing multiple losses and studying the effects of correlated and uncorrelated loss.

Chapter 4 presents the Announce-Listen algorithm, focusing on an analysis of its consistency model. We propose an alternate model for analysis than existing models [53] [49]. The new model has several advantages. It exposes essential parameters for evaluation: timers not only for announcement and update periodicity, but also for cache expiration; transmission delay, which can be compared with timer intervals; and group size, which allows a more accurate assessment of messaging overhead. Finally, unlike previous models, our model is directly implementable in simulation, allowing us to validate the analysis. With validation, the analysis has stronger predictive value.

In Chapter 5, we examine the Leader Election algorithm, exploring the idea that metrics for simple components can be used to bound the performance of the more complex algorithms that rely upon them. Specifically, we investigate the idea that Leader Election is composed from Suppression and Announce-Listen. In the optimal case, there is a direct correlation between the performance of LE and the performance of SUP and AL, but under non-optimal conditions it diverges considerably. Nonetheless, the Leader Election technique offers insight into the importance of equivalence in compositional algorithms. In addition, our examination of Leader Election is a departure from the norm, in that we do not tie it to a particular topology. Moreover, we analyze its judicious use of Suppression to scale beyond conventional LE algorithms and explore the use of an alternate leader election criteria to improve convergence. As with other techniques, we study the algorithm in the context of both correlated and uncorrelated loss models, to more accurately reflect the range of loss conditions that groups of processes may encounter.

In Chapter 6, we summarize our results, contributions and future directions.