AVCAST : NEW APPROACHES FOR IMPLEMENTING AVAILABILITY-DEPENDENT RELIABILITY FOR MULTICAST RECEIVERS

BY

THADPONG PONGTHAWORNKAMOL

B.Eng., Kasetsart University, 2003

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2006

Urbana, Illinois

Abstract

Today's large-scale distributed systems consist of collections of nodes that have highly variable availability — a phenomenon sometimes called *churn*. This availability variation is often a hindrance to achieving reliability and performance for distributed applications such as multicast. This paper looks into utilizing and leveraging availability information in order to provide availability-dependent message reliability for multicast receivers. An application (e.g., a publish-subscribe system) may want to scale the multicast message reliability at each receiver according to that receiver's availability (in terms of the fraction of time that receiver is online) — different options are that the reliability is independent of the availability, or proportional to it, or an arbitrary function of it. We propose several gossip-based algorithms to support several such predicates. These techniques rely on each node's availability being monitored in a distributed manner by a small group of other nodes in such a way that the monitoring load is evenly distributed in the system. Our techniques are light-weight, scalable, and are space- and time- efficient. We analyze our algorithms and evaluate them experimentally by injecting availability traces collected from real peer-to-peer systems.

To my beloved family

Acknowledgments

First, I would like to thank my advisor, Indranil Gupta. It was my pleasure and honour to have a chance to work under the guidance of such a dedicated and helpful advisor like him. Second, I would like to thank my family for letting me do what I dream of and always supporting me in every aspect of life with love and understanding. Third, I would like to thank the William J. Fulbright scholar program for providing me a chance to obtain the most valuable asset I ever imagined—education. Finally, I would like to thank all members of the Thai Student Association for making my life a fun and warm life during my graduate study.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Gossip-based protocols are useful information dissemination techniques for many large-scale distributed system applications such as publish-subscribe systems [48] (e.g., RSS [42]), multimedia streaming [23], multicast [46], peer-to-peer systems [29, 38], and grid computing [17, 18]. They exhibit several desired properties such as simplicity, scalability, and fault-tolerance [6, 26]. Moreover, gossip-based protocols are also resistant to churn — this is the dynamism due to highly variable availability of different nodes in the system. The term *node availability*, in the context of this paper, means the fraction of time a node has been available in the system so far.

Several churn-resistant and scalable gossip-based multicast algorithms have been proposed in the past, e.g., [4, 6, 15, 26, 37, 46]. However, to the best of our knowledge, none of these previous works provide support for availability-dependent reliability predicates, which is the capability to set the multicast reliability at multicast receivers based on the availability characteristics of the receivers. In the other words, we want to be able to specify and support a system-wide predicate that relates the reliability at each node to its availability.

There are several reasons why multicast reliability at each receiver should be related to the receiver's availability. The reasons are (1) *fairness* in reliability (high availability receivers get better reliability, but are not over-burdened), (2) *fighting freeloading* (hosts that have low availability and contribute nothing to the system, but get high multicast reliability), and (3) being an *incentive* for nodes to increase their own availability, which will result in a more reliable and resource-efficient system.

In order to address these issues, this paper presents *AVCast*, an availability-aware, gossip-based multicast protocol. AVCast currently allows a choice of two predicates that specify the availability-

reliability relationship. Using AVCast, we study the effects on multicast reliability of high-variance availability distributions across hosts. Note that the terms *node*, *host*, and *receiver* have the same meaning and will be used interchangeably throughout this paper.

1.1 Contributions

Extended from the work in [31], this paper has three main research contributions.

1. We propose a decentralized monitoring protocol for each member node to estimate the availability distribution of the system.

2. We create a generic framework to specify a range of availability-dependent reliability predicates.

3. We implement a variety of reliability predicates : *uniform reliability*, *availability-proportional reliability*, and *generalized availability-dependent reliability*.

1.2 System Model

This multicast protocol proposed in this work is based on the following assumptions.

Multiple-senders, multiple-receivers multicast systems : We assume a multiple-senders multicast system where each node can be a sender, a receiver, or both.

Asynchronous model : Each node performs its task autonomously without synchronization between nodes.

Crash-recovery model : Each multicast node can fail or leave the system silently. Such node is called an *offline* node. However, offline nodes can join the system (i.e., become *online*) again at later time.

Availability definition : The term *availability* of a node is defined as the fraction of the most recent time duration T that node is online in the system, where the length of T is pre-specified based on the system's characteristic. Setting T too short will make the availability calculation too sensitive to nodes' transient behavior while setting T too long will cause the availability calculation to rely too much on nodes' past behavior.

Reliability definition : Note that the conventional definition of multicast reliability is not appropriate for churned hosts that frequently switch their states between online and offline, since it is impossible for a node to receive a multicast message when that node is currently offline (this paper does not consider any repair mechanisms for each node to retrieve missing messages that were initiated during its offline period). Hence, we give a more appropriate definition of multicast reliability for a churned host as follows.

Definition – Multicast Reliability for a Churned Host: Consider a host x, and consider the number of multicast messages whose propagation time (i.e., from multicast initiation to multicast termination) completely overlaps with the available times for host x. Let f be the fraction of such messages received at host x. Then f is the multicast reliability for host x, denoted by r_x.

Notice that this definition of reliability is different and is more appropriate than the traditional definition, as it cleanly separates the unavailable times out of the reliability calculation.

In the analysis of AVCast protocol, however, we assume the following model assumptions in order to simplify the analysis. Note that the following assumptions do not have any effect on the algorithm of the protocol.

Synchronous multicast : We model a multicast as a synchronous process that operates in multiple rounds. Each round starts when a set of nodes forward its message and stops when all

forward messages are delivered at the receivers.

Atomic availability : In the analysis, we assume that any node that was online when a multicast was initiated will stay online during the *entire* period that the multicast is active. Compared to the continuous churn assumption, our assumption is more pessimistic, since we treat nodes that are offline and online several times during a multicast as offline nodes. Hence, the result of our analysis can be used as the upper bound of the practical result. With our assumption, we can discard offline nodes and focus on only online nodes in the analysis.

The rest of this paper is organized as follows. Section 2 discusses some related works on churn-fairness studies and multicast protocols. Section 3 presents basic concepts and components of the AVCast system. Section 4 gives a theoretical analysis of availability-aware reliability framework in AVCast. Section 5 proposes two reliability predicates supported in AVCast. Section 6 shows the experimental results. Finally, Section 7 concludes the paper.

Chapter 2

Related Work

2.1 Reliable Multicast Protocols

Group communication protocols, specifically multicast protocols, are essential components in many large-scale distributed systems in order to allow each participant to communicate with others in the systems reliably and efficiently. So far, there have been many works on designing reliable and efficient multicast protocols. Such works can be categorized into *Network-level* Multicast Protocols and *Application-level* Multicast Protocols.

2.1.1 Network-level Multicast Protocols

Network-level multicast protocols such as SRM [16], RMTP [30] and PIM [12] perform a multicast with an intervention from the routing fabric layer. By storing multicast membership information in network routers along the multicast path and forwarding multicast messages based on such information. network-level multicast protocols achieve good performance in terms of high bandwidth and low latency. However, network-level multicast protocols suffer from scalability problems since each router has to store the multicast information and then becomes a performance bottleneck when the multicast group is large. With such reason, the multicast functionalities have been disabled in many network routers.

2.1.2 Application-level Multicast Protocols

Since the performance of communication networks have been drastically improved over the past few years, the main considerations in multicast protocols are scalability and reliability. In order to increase scalability in distributed systems and networks, [41] also suggests that the core Internet layer should be designed to contain only core network functionalities, and hence all other features should be implemented in higher layers. Application-level multicast protocols follow such paradigm by implementing the multicast functionality in the application layer without the intervention from the network layer. In application-level multicast protocols, an overlay graph of multicast nodes is constructed. Each multicast node is then responsible for storing multicast membership information and forward multicast messages to other nodes via the overlay graph using normal services from the network layer. Since no additional overhead is incurred in routers, this approach is more scalable than traditional network-level multicast protocols.

Many application-level multicast protocols use tree-based graph topologies [8, 24, 34, 40, 50]. When a multicast is initiated, the multicast message will be flooded from the root node towards leaf nodes in the tree. Several tree-based multicast protocols exploit the routing and membership functionalities from distributed hash tables (DHTs) such as [33, 39, 45, 49] to create the multicast group and deliver messages to receivers. However, tree-based multicast protocols suffer from reliability problem under churns and node failures since failures or departures of root nodes of subtrees will cut all nodes in the subtrees off the multicast systems. Although such problem can be mitigated by the use of repair mechanisms in DHT membership and routing substrates, the problem still persists in the systems under heavy churns. Some works proposed other multicast overlay structure such as multiple trees [7] or direct acyclic graph [28] to solve the problem.

2.2 Gossiping Algorithms and Gossip-based Multicast Protocols

Recently, *gossip-based* protocols have become active areas of research in distributed systems. Originated from the theory of infectious diseases presented in [32], the concept of gossiping has been used in communication networks and distributed systems including database replication [13], membership monitoring protocol [11, 47], distributed hash table [21], distributed persistent storage [20] and multicast [4,15,37,46]. Unlike other deterministic multicast protocols, a gossip-based multicast protocol builds a random overlay structure of multicast nodes. Upon the receipt of a multicast message, the receiver node forward the multicast to its neighbor nodes randomly. Without the needs of static overlay structures and points of failures, gossip-based multicast protocols can achieve a good level of scalability and probabilistic reliability even in very dynamic systems. AVCast falls into this category. However, the purpose of AVCast is not achieving perfect reliability. Rather, AVCast aims to provide different multicast reliability levels for multicast nodes according to each node's availability.

Over the past few years, several gossip-based multicast protocols have been proposed [4, 15, 37, 46]. We examine some of them here.

Gocast [46] implemented a proximity-aware multicast protocol on top of Resilient Overlay Network(RON) [3] in order to achieve high throughput and low message delay. However, Gocast does not address the effect of churn to reliability of the multicast system. AVCast, on the other hand, focus on relating system multicast reliability to availability of the system itself. Such two works are orthogonal and thus a combination of the two approaches is possible.

DOS-Resistant Unforgeable Multicast(Drum) [4] addresses reliability of multicast protocol under malicious denial of service attacks. AVCast tries to achieve best-effort multicast reliability on systems under non-malicious churn. In AVCast, we consider the system model where nodes are not malicious, but can act selfishly by having low-availability.

2.3 Churn and Fairness Studies

There have been several works [2, 5, 43] addressing the characteristics of churn in large-scale distributed applications such as distributed file-sharing and multicast systems. The studies have shown that churn has effect on stability and performance of overlay networks and applications that are built on top of overlay networks. In order to solve such problems, [10, 35] have proposed techniques to build more churn-resistant overlay networks.

Besides the effect of churn on stability and performance problem, [1] also exposed the correlation between churn in global peer-to-peer applications and its effect to per-user fairness of quality of service. According to the study, most file-sharing systems consist of a significant portion of *free-riders*, the system users who exploit the benefit from the system without contributing to the system. Similarly, we quantitatively analyzes the effect of churn to stability and reliability of distributed systems, particularly in application-level, gossip-based multicast systems. Moreover, we presents a set of gossip-based multicast variation in order to ensure fairness in the term of multicast reliability.

Chapter 3

Basic Approach

AVCast consists of two components: 1) *monitoring and membership component* and 2) *availability-aware gossip-based multicast component*. In AVCast monitoring protocol, each AVCast node acts as a *pinging node* that monitors the availability of a few other nodes – in turn, each of these pinged nodes is called a *target node*. Each target node's availability is then monitored in a distributed fashion by a small group of pinging nodes. These pinging nodes are selected randomly but consistently for each target node, so that each node only monitors a small number of other target nodes. Note that the pinging and target node relationships are inverses of one another (a node x will be a pinging node of all x's target nodes). Once the availability information has been obtained by the membership component, each pinging node then uses the multicast component to forward multicast messages to a number of target nodes. How target nodes are selected to receive forwarded messages depends on the availability-dependent reliability predicate that is to be implemented.

This chapter details how the monitoring component and the multicast component operate. The mathematical analysis of AVCast's availability-dependent reliability predicates will then be presented in Section 4.

3.1 Monitoring and Membership Protocol

There are several simple methods to obtain the availability information of each node in the system (i.e., to select the pinging set for a given node). We discuss some of these approaches below, along with their disadvantages.

- Each node measures its availability by itself and reports its own availability to other nodes.

While this method is simple and straightforward, it allows a selfish node to cheat, i.e., to lie about its availability.

- Each target node's availability is measured by pinging nodes that are basically some of its neighbors in an application-defined *overlay network*. This method eliminates the above problem of cheating, but how pinging nodes are determined is specific to the type of overlay network. In power-law overlays, for instance, the high degree nodes would share a large pinging responsibility, thus causing load imbalance.

- Each node's availability is measured by neighbor nodes specified by randomization techniques (e.g., via a random walk). However, random walks can also make biased choices, e.g., in a star network [44], thus causing load imbalance.

- To overcome limitations of approaches mentioned above AVCast uses a *consistent randomization*, which is adapted from [22], to select the pinging set. AVCast uses a low-overhead, decentralized, *hash function-based* protocol to determine pinging nodes for each target node. Using a globally consistent hash function, denoted by H, each node can verify its pinging nodes and target nodes in a consistent manner across the system, thus eliminating problems of selfish nodes cheating and adversarial peers controlling the system. Further, the uniformly random nature of the function H ensures better load balance than the previous approaches above.

In the AVCast membership protocol, a node x maintains links pointing to two sets of nodes. The first set is called the *pinging set* of x (PS_x), which contains x's pinging nodes whose duties are to monitor x's availability and to probabilistically forward multicast messages to x. The second set of nodes is the target set of x (TS_x), which contains all nodes whose availability x is monitoring. Notice that $x \in PS_y$ if and only if $y \in TS_x$. A node x treats the availability distribution obtained from TS_x as a sample of system's global availability distribution. According to [19], having TS and PS of size $O(logN)$, where N is the approximate total number of offline and online nodes in

the system, is enough to provide accuracy in availability estimation and achieve good scalability in multicasting. Note that N is consistently known or approximated beforehand by using size estimation algorithms such as [27]([27] provides an estimated number of *online* nodes, denoted by n, which is different from N, However, we can calculate N from n and average system availability $E[a]$ with the equation $N = n/E[a]$). Another way to estimate N is by setting N to the power of 2 that is closest to the scale of the system. We do not envision this to be a hindrance since most peer-to-peer systems, in spite of churn, have stable system sizes [5].

The availability membership protocol is described in Figure 3.1 and Figure 3.2. Besides its own id number, each node x maintains two node-specific parameters used to determine sizes of pinging set and target set, K_{in}^x and K_{out}^x. A node x will add a node y to its target set TS_x (and y will add x to PS_y) if and only if

$$H(x, y) < \sqrt{K_{out}^x . K_{in}^y}/N \tag{3.1}$$

where H is a globally consistent hash function known to every node. H could be a SHA-1 [14] or a MD5 [36] hash function, but with the result normalized to the range $[0, 1]$. Initially, each node x sets its K_{in}^x and K_{out}^x values to a value of K (a known priori of all nodes set to a value that is $O(logN)$). Note that K is the expected size of PS and TS. However, if a node x finds its pinging set or target set smaller or larger than K, it can adjust its K_{in} and K_{out} parameters to balance the size of its TS and PS respectively. Such procedure is called *rebalancing operation*. The rebalancing operation will be discussed later in this chapter.

Now, we focus on the action during the monitoring process. When a node x joins the system for the first time, it sends a REQ (request) message to an arbitrary node z in the system (that node z then becomes x's contact node). The REQ message contains x's node id, and K_{in}^x, and K_{out}^x values. The contact node z then uses an ADV (advertise) message to forward x's request to all other nodes via a multicast. Any node y that receives the ADV message then evaluates the equation (3.1). If the condition is true, it will add x into its pinging set PS_y and sends a REP (reply) message back

procedure join()

1: **if** x joins for the first time **then**
2: $y \leftarrow$ random node
3: send $\langle \text{REQ}, x, K^x_{in}, K^x_{out} \rangle$ to y
4: **else**
5: $PS_x \leftarrow$ persistent storage
6: $TS_x \leftarrow$ persistent storage
7: $K^x_{in} \leftarrow$ persistent storage
8: $K^x_{out} \leftarrow$ persistent storage
9: **end if**

procedure update(y, K^y_{in}, K^y_{out})

1: **if** $H(x, y) < \sqrt{K^x_{out}.K^y_{in}}/N$ **then**
2: add y into TS_x
3: **end if**
4: **if** $H(y, x) < \sqrt{K^y_{out}.K^x_{in}}/N$ **then**
5: add y into PS_x
6: **end if**

Figure 3.1: Availability Membership Protocol

receive $\langle \text{REQ}, y, K^y_{in}, K^y_{out} \rangle$

1: multicast_send($\langle \text{ADV}, y, K^y_{in}, K^y_{out} \rangle$)

receive $\langle \text{ADV}, y, K^y_{in}, K^y_{out} \rangle$

1: update(y, K^y_{in}, K^y_{out})
2: **if** $y \in TS_x$ or $y \in PS_x$ **then**
3: send $\langle \text{REP}, x, K^x_{in}, K^x_{out} \rangle$ to y
4: **end if**

receive $\langle \text{REP}, y, K^y_{in}, K^y_{out} \rangle$

1: update(y, K^y_{in}, K^y_{out})

Figure 3.2: Membership Message Handlers

every period T_1

if $|PS_x| < (1-\alpha)K$ **then**
  $K_{in}^x \leftarrow (1+\beta)K_{in}^x$
end if
if $|PS_x| > (1+\alpha)K$ **then**
  $K_{in}^x \leftarrow (1-\beta)K_{in}^x$
end if
if $|TS_x| < (1-\alpha)K$ **then**
  $K_{out}^x \leftarrow (1+\beta)K_{out}^x$
end if
if $|TS_x| > (1+\alpha)K$ **then**
  $K_{out}^x \leftarrow (1-\beta)K_{out}^x$
end if
if K_{in}^x or K_{out}^x is changed **then**
  multicast_send($\langle$ADV$, x, K_{in}^x, K_{out}^x\rangle$)
end if

Figure 3.3: Rebalance Operation

every period T_2

for each node $y \in TS_x$ **do**
  send $\langle$PING$, x\rangle$ to y
  if receive $\langle$PONG$\rangle$ back from y before timeout T_0 **then**
    mark y as available
  else
    mark y as unavailable
  end if
  recalculate y's availability value a_y
end for

Figure 3.4: Membership Maintenance Procedure

procedure multicast_send($message$**)**

1: **for** $i = 1$ **to** C **do**
2: **send** $message$ **to a random *online* node** $y \in TS_x$
3: **end for**

receive multicast message from y

1: **if** $y \in PS_x$ **then**
2: **for** $i = 1$ **to** C **do**
3: **for each *online* node** $z \in TS_x$ **do**
4: **with probability** $= p(a_z)$**, forward message to** z
5: **end for**
6: **end for**
7: **end if**

Figure 3.5: Availability-aware No-wait Gossiping protocol at node x

to x. Similarly, node y evaluates (using the same equation (3.1)) whether node x should belong to TS_y. Upon receiving a REP message from y, x can verify the equation (3.1) and add y into its target set TS_x (or PS_x respectively). Note that the PS_x and TS_x lists are stored in x's persistent storage so that if x goes offline and joins the system again, it can retrieve the information without needing a contact node.

Rebalancing operation : In practice, the distribution of the hash space may not be uniform, resulting in the sizes of PS and TS at each node being different from K. To reduce the load variance across nodes, AVCast uses a rebalance procedure shown in Figure 3.3. This procedure adjusts K_{in} and K_{out} of individual nodes in order to keep the size of PS and TS as close to the expected size K as possible. The rebalance procedure defines two system-wide constants: α and β. α and β are preconfigured values ranging from 0 to 1. α can be considered as the level of tolerance of link degree invariance, while β defines how reactive the rebalance procedure is. A node x whose target set contains more than $(1 + \alpha)K$ members decreases its K^x_{out} value by the scale of $(1 - \beta)$. Similarly, if x's target set contains less than $(1 - \alpha)K$ members, x increases its K^x_{out} value by the scale of $(1 + \beta)$. Every time either K^x_{in} or K^x_{out} is recomputed, x re-advertises its new parameters. Each node x repeats the rebalancing procedure until $|TS_x|$ and $|PS_x|$ are within range $[(1-\alpha)K, (1+\alpha)K]$. The smaller α is, the less load invariance the system has and the more

control message overhead incurred for the rebalancing procedure.

The monitoring component operates in asynchronous protocol rounds (typically 5 to 10 seconds long) without synchronization between nodes. During each round, at a node x, it periodically sends PING messages to all target nodes in TS_x and waits for reply messages from them. Any target nodes that fail to send back reply messages before the next round will be considered unavailable during that round. Each node then stores its target nodes' raw availability traces in its persistent storage. To prevent excessive overhead, each pinging node uses the availability trace from T most recent rounds to calculate target nodes' availability value, where T is a globally defined constant for the system. The availability value of a target node y, denoted by a_y, measured at one of its pinging nodes x is calculated as the fraction of T most recent rounds that y responded to x. The availability traces older than T rounds can be either discarded or aggregated into a coarser-scale archive, depending on the implementation of the system.

3.2 Availability-aware Gossip-based Multicast Protocol

AVCast adopts an existing gossip-based multicast protocol called *no-wait gossiping*. This is not a new protocol, but was proposed in [46]. Figure 3.5 shows the availability-aware version of no-wait gossiping protocol used in AVCast. In AVCast, the sender node x initiates a multicast by sending a multicast message to C random *online* nodes in its target set TS_x, where C is a globally defined constant. Upon receiving a multicast message, a node immediately forwards C copies of the message, each one to each of C selected *online* target nodes. The way the online target nodes are picked up is not uniform — instead, a node y forwards the message to its online target node z with probability $p(a_z)$, where $p(.)$ is a global probability function of availability, and a_z denotes z's availability as observed by y. $p(.)$ is chosen depending on the global predicate that is to be satisfied (Section 5). The effect of choosing C and $p(.)$ will also be analyzed quantitatively in Section 4 and 5.

The no-wait gossiping protocol is completely stateless in the sense that each node forwards a

message to some of its target nodes *only once* and immediately after it receives the message. We believe that this stateless property makes the protocol appropriate to use in dynamic systems where nodes frequently go offline and online. Please note that AVCast also consists of stateful parts (i.e. the target set's availability information), but such stateful parts are not sensitive to dynamism of the system.

Chapter 4

Analysis of Gossip-based Multicast Protocol

In this chapter, we analyze several characteristics of AVCast availability-aware gossip-based multicast protocol introduced in Section 3.2. We show the effect of two global system parameters: the target selection probability function $p(.)$ and the number of copies each node forwards per message C. We also present the analysis of how different $p(.)$ and C affect the reliability predicate of the system.

We model a multicast as a synchronous process operating in multiple protocol rounds (The term *round* here is different from one in Section 3.1). The first round starts when the multicast is initiated and the last round ends when the multicast dies out. For a given multicast message, at any time, each *online* node falls into one of three categories: *virgin* nodes which have not yet received the message, *active* nodes which have just received the message but have not yet forwarded the message, and *inactive* nodes which have already received and forwarded the message. In each round, each active node gossips C copies of a message, one each to each of C selected online target nodes and then becomes inactive. Virgin nodes that receive the gossip message then become new active nodes in the next round. On the other hand, inactive nodes do nothing if they receive duplicates of the same message. The multicast process stops when there is no active node left in the system.

Consider a system of N nodes with an availability mass function f (i.e. the fraction of overall system nodes that have availability equal to a is $f(a)$). Hence, the average number of online nodes

in the system n at any time is

$$n = \sum_{\{a:f(a)\neq 0\}} (af(a)N) = E[a]N,$$

where $E[a]$ is the mean availability of all nodes in the system. We also define the *online* availability mass function $g(a)$ as the fraction of online nodes that have availability a. We can calculate $g(a)$ from $f(a)$ by the following equation:

$$g(a) = \frac{af(a)}{E[a]} \tag{4.1}$$

Now, for a given multicast, let x_t, y_t, and z_t be the number of online nodes (with respect to n) that are active, inactive, and virgin in the system at round t respectively ($x_t + y_t + z_t = n$ at any round t). Also, let $g_t(a)$ be the fraction of virgin nodes that have availability equal to a at protocol round t. Initially, a sender node initiates a multicast by sending the message to C online nodes randomly selected from its target set. In each of subsequent rounds, each active node forwards a constant number of copies C to its online target nodes using target probability function $p(a)$. Hence the number of messages forwarded in each round t is equal to Cx_t. Thus, the protocol model at round $t + 1$ can be described by the following set of equations..

$$
\begin{aligned}
x_{t+1} &= E[\text{\# virgin nodes got the message in round } t] \\
&= \sum_{\{a:g_t(a)\neq 0\}} (n g_t(a) \text{P[node gets the message]}) \\
&= \sum_{\{a:g_t(a)\neq 0\}} \left(n g_t(a)(1-(1-p(a))^{\frac{CK_{on}x_t}{n}}) \right) \\
&= \sum_{\{a:g_t(a)\neq 0\}} \left(n g_t(a)(1-(1-p(a))^{\frac{CKE[a]x_t}{n}}) \right) \\
y_{t+1} &= y_t + x_t \\
z_{t+1} &= n - x_{t+1} - y_{t+1} \\
g_{t+1}(a) &= g_t(a)((1-p(a))^{\frac{CKE[a]x_t}{n}})
\end{aligned}
$$

with the initial conditions as

$$x_0 = C, y_0 = 1, z_0 = n - x_0 - y_0$$

and

$$g_0(a) = g(a)(1 - \frac{C+1}{n})$$

Note that $g_t(a)$, the fraction of virgin nodes with availability a, keeps changing in each round t. The intuition behind this is that virgin nodes with different availability will be picked up to receive the message with different probability.

Given C, the number of copies forwarded per message, the availability distribution function $f(a)$, the total number of nodes in the system N, and the average size of pinging set and target set K as inputs, we can use the model mentioned above to analyze several characteristics of the multicast as described below.

4.1 System-wide Multicast Reliability and Message Propagation Delay

The system-wide multicast reliability is defined as the fraction of online nodes that receive at least one copy of multicast message. Thus, the system-wide multicast reliability R is

$$R = \frac{y_v}{n} = \frac{y_v}{NE[a]}$$

, where $v =$ the minimum value of t such that $x_t = 0$.

Hence, given the average size of the sender's target set K (usually, $K = O(\log n)$), the system availability mass function $f(a)$, and the target probability function $p(a)$, we can calculate the appropriate number of copies C in order to achieve desired system-wide multicast reliability. Calculating equations above can be locally performed at each multicast node using the availability distribution from its target set TS. Note that each node x in the system can estimate the availability mass function f as $f(a_y) = \frac{1}{|TS_x|}$ for each node $y \in TS_x$.

The message propagation delay is the number of protocol rounds in which the system contains a least one active node. Hence, the message propagation delay d can be defined as

$$d = v$$

where $v =$ the smallest t such that $x_t = 0$

4.2 Node-level Multicast Reliability

The node-level multicast reliability is defined as the probability that a node eventually receives at least one copy of multicast message from its pinging set (from the definition in Section 1, this probability is given that the node is available throughout the multicast period. According to the equations, the multicast reliability $r(a)$ at a node whose availability equal a can be estimated as

follows.

$$\begin{aligned} r(a) &= P[\text{node receives at least one copy}] \\ &= 1 - P[\text{node receives no copies}] \\ &= 1 - \prod_{t=1}^{\infty} \left[(1 - p(a))^{\frac{CKE[a]x_t}{n}} \right] \\ &= 1 - (1 - p(a))^{\frac{CKE[a]y_v}{n}} \\ &= 1 - (1 - p(a))^{CKE[a]R} \end{aligned} \tag{4.2}$$

where $v =$ the smallest t such that $x_t = 0$, and $R =$ system-wide multicast reliability

The analysis shows that the system-wide multicast reliability has an effect on node-level multicast reliability, no matter what node-level availability a or gossip target probability $p(a)$ are chosen. Also, the relation between expected global system-wide reliability R and local per-node reliability r can be described as

$$R = \frac{E[ra]}{E[a]} \tag{4.3}$$

where

$$E[ra] = \sum_{\{a:f(a)\neq 0\}} (af(a)r(a))$$

Chapter 5

Availability-dependent Reliability Predicates

In this section, we first introduce two basic reliable predicates supported in AVCast. The first predicate is *uniform reliability*, which all nodes receive roughly the same level of multicast reliability. The second predicate is *availability-proportional reliability*, where the reliability each node receives is equal to the availability value of the node itself. We then present the generalized availability-dependent reliability predicate, which can be used to implement *any* reliable predicate. Finally, we present the combination of the two basic predicates, resulting in *multi-class reliability* predicate.

5.1 Uniform Reliability (r = constant)

According to the model presented in the previous chapter, specifying target selection probability function $p(a) = \frac{1}{K_{on}}$, where K_{on} is the number of available nodes in the target set, will result in a naive uniform gossip-based multicast scheme where every available node in the pinging set is equally likely to be picked up as a message receiver.

The node-level multicast reliability $r(a)$ of a node with availability a in uniform gossip-based multicast can be expressed as the following equation.

$$\begin{aligned} r(a) &= 1-(1-p(a))^{CKRE[a]} \\ &= 1-(1-\frac{1}{K_{on}})^{CK_{on}R} \\ &\geq 1-e^{-RC} \end{aligned}$$

It can be seen that per-node reliability $r(a)$ value does not depend on the per-node availability a value. Thus, the quality of service each node obtains is equal to system-wide reliability. Moreover, the system-wide reliability can be expressed as $R = \frac{E[ra]}{E[a]} = \frac{E[r]E[a]}{E[a]} \geq 1 - e^{-RC}$ for this predicate.

5.2 Availability-proportional Reliability ($r = a$)

Although defining the target selection policy function as a constant results in equality of node-level multicast reliability at each node, such a policy does not provide the fairness property because high-availability nodes achieve the same level of multicast reliability as low-availability nodes. Since the fairness property is an important incentive for users in many large-scale peer-to-peer applications, one might want to construct a multicast infrastructure where multicast reliability at each node is *linearly proportional* to the availability of the node itself.

According to node-level reliability analysis (i.e., equation (4.2)),

$$r(a) = 1 - (1 - p(a))^{CKRE[a]}$$

However, we want $r(a)$ to be equal to a to satisfy the predicate. Replacing $r(a)$ with a in the above equation, the new equation is

$$a = 1 - (1 - p(a))^{CKRE[a]}$$

Also, since $r(a) = a$, global system-wide reliability R can be expressed as

$$R = \frac{E[ra]}{E[a]} = \frac{E[a^2]}{E[a]}$$

By replacing R and rearranging the equation, the target probability function can be expressed

as a function of availability as follows.

$$p(a) = 1 - (1 - a)^{\frac{1}{CKE[a^2]}}$$

Notice that $E[a^2]$ at node x can be calculated based on TS_x's availabilities. With the formula above, each node x can adjust its C parameter locally so that $\sum_{z \in TS_x} (p(a_z)) \leq 1$. The rest of the protocol is the same as the main protocol framework described in Section 3.2.

5.3 Generalized Availability-dependent Reliability (r = any function of a)

We now generalize the predicate presented in the previous section to support any arbitrary predicate $r(a)$. From the observation, we notice that equation (4.2) and equation (4.3) can be combined, resulting in the following equation

$$r(a) = 1 - (1 - p(a))^{CKE[ra]} \tag{5.1}$$

By rearranging equation (5.1), the generalized availability predicate equation is obtained as

$$p(a) = 1 - (1 - r(a))^{\frac{1}{CKE[ra]}} \tag{5.2}$$

Since $E[ra]$ can be computed by equation (4.3) once the availability distribution of the system is known, we can use equation 5.2 to implement *any* availability predicate in the same manner with the availability-proportional predicate. However, implementing some ideal predicates might cause each node to adjust its C parameter to an unrealistically high value. For example, in order to implement a predicate such that each node obtains perfect reliability (i.e. $r(a) = 1$ for all a), each node has to adjust its $C = \infty$, which is not practical.

With equation 5.2, we can see that the uniform reliability and availability-proportional reliabil-

Predicate	Tunable parameters	Property	$p(a)$
Uniform (copies)	number of copies C	$r(a) \geq 1 - e^{-RC}$	$\frac{1}{K_{on}}$
Uniform (reliability)	expected reliability R	$r(a) = R$	$1 - (1 - R)^{\frac{1}{CK_{on}R}}$
Availability-proportional	-	$r(a) = a$	$1 - (1 - a)^{\frac{1}{CKE[a^2]}}$
Generalized	reliability function $f(a)$	$r(a) = f(a)$	$1 - (1 - f(a))^{\frac{1}{CKE[fa]}}$

Table 5.1: Summary of basic availability-aware predicates

ity predicates are the specializations of the generalized reliability predicate where the forwarding probability $p(a)$ equal to $\frac{1}{K_{on}}$ and a repectively. Moreover, equation 5.2 allows the uniform reliability predicate to be adjusted by the expected reliability constant, which gives more flexibility in controlling multicast behavior than adjusting the number of copies forwarded per message. Table 5.3 gives the summary of all predicates proposed.

Based on basic predicates, one can construct more complex predicates as combinations of basic predicates. One complex predicate worth mentioning is *multi-class reliability* predicate. This predicate divides multicast nodes into several classes according to nodes' availability, and implements different reliability predicates in each class (i.e. nodes within the same class are treated with the same reliability predicate). This predicate can be viewed as a combination of different basic predicates in order to control system reliability to the desired level. For example, one might want to build an availability-proportional reliability multicast system with a minimum reliability guarantee for all nodes. Such a system can be achieved by splitting multicast nodes into two classes—nodes whose availability more than a prespecified threshold will be treated with the availability-proportional reliability predicate, and nodes whose availability less than the threshold will be treated with the uniform, minimum reliability predicate.

Chapter 6

Experimental Results

We evaluate the AVCast protocol via simulation. Our implementation of AVCast contains almost 3,000 lines of C++ code including the membership and the gossiping protocols. The availability distribution of nodes in the experiment is obtained from Overnet file-sharing system trace [5], which has average availability roughly equal to 0.3. In the simulation, the system consisting of 1442 nodes runs the AVCast protocol for 6,000 protocol rounds (each round lasts around 5 seconds in practical). For simplicity, round are synchronized throughout the system in the simulation. At the beginning of each round, each node randomly tosses a number between 0 and 1 to decide whether it is online or offline throughout that round (each node stays online if the number is less than its availability value). During the first 3,000 rounds, each peer runs the availability monitoring and view rebalance operations. During the last 3,000 rounds, a randomly selected online node initiates one multicast message to the system per round. Each multicast message's propagation is assumed to die out within a single round because nodes in the no-wait gossiping scheme forward a message all at once, resulting in a very quick multicast process. Hence, we can also assume that a node is either fully offline or fully online for a given message. Setting round duration to 5 seconds is reasonable since a multicast typically completes within 5 seconds while the monitoring process can achieves high accuracy. The average system reliability is the average fraction of online nodes that receives message in each of 3,000 rounds. The average node reliability of each node is measured by the number of rounds the node receives messages, divided by the number of rounds the node is online.

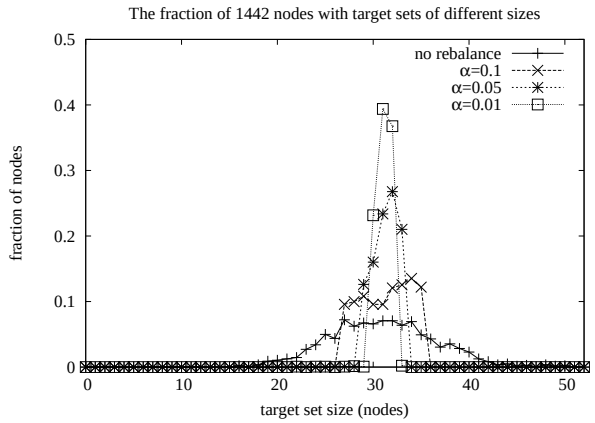
We first discuss the effectiveness of the membership protocol under different α (balance sensitivity) and K (expected size of TS and PS parameters) values. We then evaluate the availability-

aware gossiping protocol in a variety of reliability predicates with consistent knowledge availability assumption where each node has the global membership and availability information of the system. Finally, we evaluate the full AVCast gossiping protocol where each node uses the decentralized membership protocol to obtain the partial view and availability information. Our evaluation on the gossiping protocol is based on how well the node reliability distribution implements the predicates.

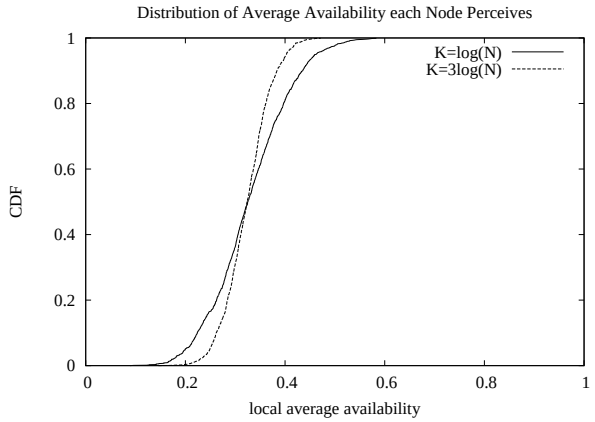
The Overnet trace was obtained in [5] by collecting the availability trace from hosts in the Overnet file-sharing system for a week. The trace was done with around a thousand of hosts by probing them every 20 minutes and recording their availability status. The Overnet trace revealed that the average availability of nodes in the experiment was roughly equal to 0.3. We believe that the Overnet availability trace cam also be applied to other large-scale distributed applications as well.

6.1 Availability Membership

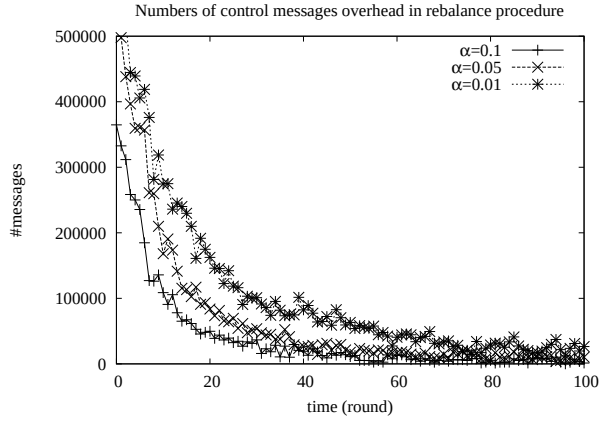
This section presents simulation results of the monitoring and membership protocol. The effectiveness of the membership protocol will be measured in terms of (1) how well the view size is balanced throughout all nodes, (2) how accurately the availability distribution each node perceives from its target set, and (3) the number of control message overhead incurred from the rebalancing operations. At time $t = 0$ in the simulation, all of 1,442 nodes were brought up online and thus each node knows all of its target set. During each t between $t = 0$ and $t = 3,000$, each node runs the view rebalance operation in order to keep its target size to $3logN = 3log(1,442) = 31$. In the rebalance operation, we vary α, the sensitivity factor, from 0.1 to 0.01 and fix β, the aggressiveness factor, at 0.1. At $t = 3,000$, we observe the size of target set at each node and the availability distribution each node has recorded from its target set.



(a) The size each node's target set



(b) The distribution of average local availability



(c) Control message overhead during rebalance period

Figure 6.1: The monitoring component

6.1.1 Local view balance

Figure 6.1(a) shows the distribution of sizes of the target set at each node at $t = 3,000$. Note that the results are also similar to the distribution of sizes of the pinging set. The smaller α is, the more consistent the sizes of the target set are. As the result also applies to the pinging set of each node, smaller α leads to a more balanced load across all nodes. However, setting α too small may cause oscillations and frequent view changes.

6.1.2 Accuracy

We use the average system availability each node observes from its target set as a measurement of how accurately each node perceives the availability condition of the system. Figure 6.1(b) shows the result of the system with different target set sizes. Notice that as the bigger the view size is, the more accurate system availability each node perceives. However, it seems not to have too much of a difference between $K = logN$ and $K = 3logN$.

6.1.3 Control Message Overhead

Figure 6.1(c) shows the control message overhead used in the rebalancing protocol during the first 100 rounds of the rebalancing operation. Message overhead was high at first since all nodes were adjusting its K_{out} and K_{in} values. After few rounds, traffic load dropped drastically since most nodes were satisfied with their settings. According to the result, the higher α is, the higher control message overhead incurred in the system. This is because more nodes will need to rebalance their views due to the more strict constraint.

6.2 Availability-aware Gossiping with Consistent Knowledge Availability Information

This section presents the result of the availability-aware gossiping *without* using the monitoring and membership protocol. Rather, the simulation is done with the assumption of consistent knowledge availability information where each node is assumed to know the global view of the system including 100%-accurate availability information. In the other word, each node's pinging set and target set are the entire set of all nodes in the system. While this assumption cannot be made possible in large-scale multicast systems, it is practical to make such assumption in small-scale or medium-scale multicast systems. For example, such assumption holds in systems with global membership protocols where each node monitors all other nodes in the system. Hence, the results in this section can be considered as upper bounds of the gossiping protocol's efficiency.

There are four reliability predicates observed in the simulation. Besides the uniform reliability and availability-proportional predicates, we experiment two specializations of the multi-class gossiping protocol called *bimodal gossiping* and *threshold-linear gossiping*. Bimodal gossiping is a two-class gossiping protocol where each class, separated by an availability threshold, uses uniform reliability predicate with different expected reliability values. Threshold-linear gossiping is a two-class gossiping protocol where one class containing nodes whose availability values exceed a prespecified threshold uses availability-proportional reliability predicate, and another class containing nodes whose availability values are less than the threshold uses uniform reliability predicate (minimum reliability guarantee).

During each protocol round in the simulation, a randomly selected node initiates a multicast by sending the message to other C *online* nodes which are randomly selected from the global list. The reliability each node receives is equal to the number of rounds each node receives at least one copy of message divided by the number of rounds such node is online. . We use two methods to measure how accurately the reliability distribution resembles the desired predicate : the root mean square method (RMS) and the standard deviation of error method (StdevErr). The

Predicate	RMS	StdevErr
Uniform ($copies = 1$)	0.0374	0.0328
Uniform ($copies = 3$)	0.0255	0.0220
Availability-proportional	0.0238	0.0179
Bimodal ($E[r_1] = 0.3$,$E[r_2] = 0.9$,Threshold=0.5)	0.0523	0.0451
Threshold-linear (Threshold=0.3)	0.0509	0.0435

Table 6.1: Root-mean-square and standard deviation of error of reliability distributions from each predicate of AVCast with consistent knowledge availability information

root mean square method is used to measure how accurate the reliability distribution follows the predicate. The standard deviation of errormethod is used to measure how uniformly each error occurs. Table 6.2 shows parameter, root-mean-square, and standard deviation or error values from the simulations with different reliability predicates.

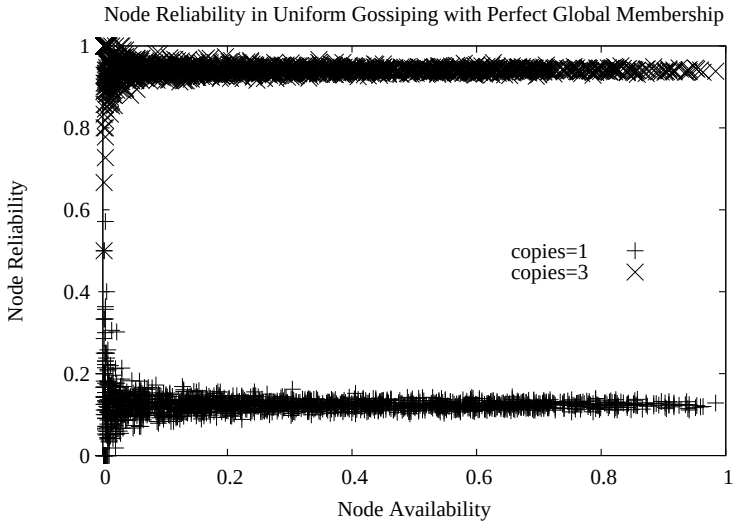


Figure 6.2: Consistent availability knowledge : Node-level reliability distribution in the uniform reliability predicate with perfect global membership

Figure 6.2, 6.3, 6.4, and 6.5 present the reliability distributions of the simulations with uniform, availability-proportional, bimodal, and threshold-linear reliability predicates respectively. The results from the figures, in conjunction with root-mean-square and standard deviation results in table 6.2, yield the effectiveness of the availability-aware gossiping—all the reliability distribution results closely and consistently resemble the desired reliability predicates.

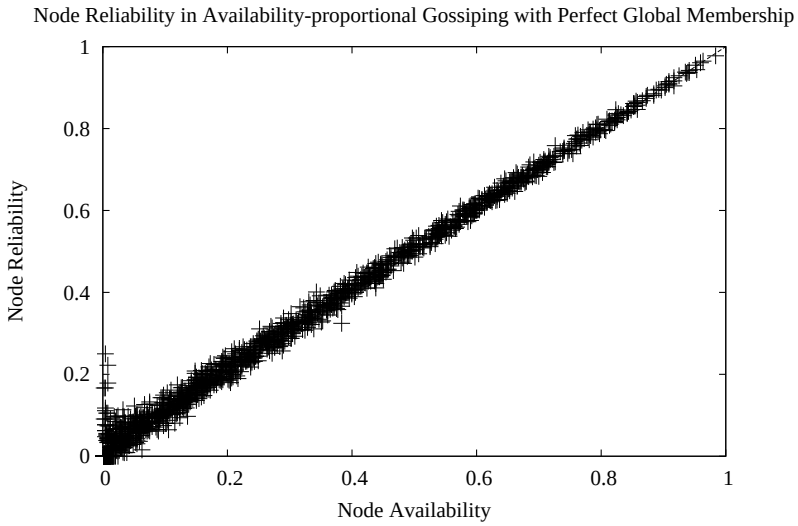


Figure 6.3: Consistent availability knowledge : Node-level reliability distribution in the availability-proportional reliability predicate with perfect global membership

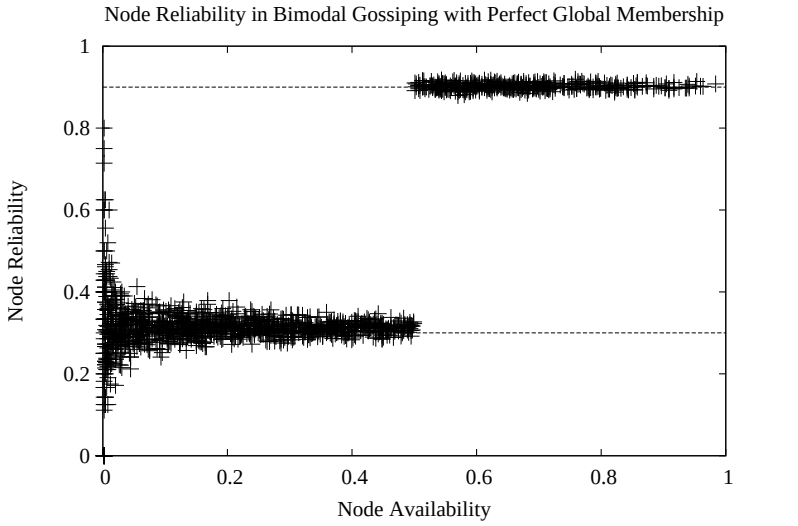


Figure 6.4: Consistent availability knowledge : Node-level reliability distribution in the bimodal reliability predicate with perfect global membership

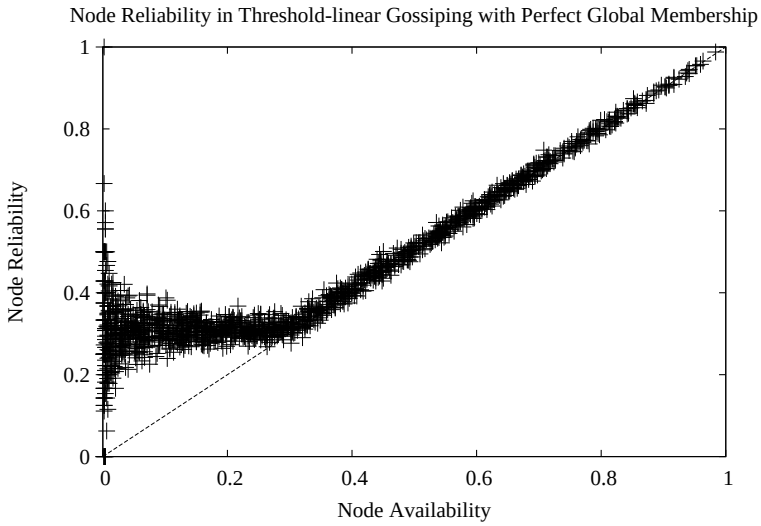


Figure 6.5: Consistent availability knowledge : Node-level reliability distribution in the threshold-linear reliability predicate with perfect global membership

6.3 Availability-aware Gossiping with Inconsistent Knowledge Availability Information

This section presents the result of the availability-aware gossiping where each multicast node obtains the membership list and availability information using the monitoring and membership component presented in section 3.1. The experiment is done with different expected view sizes K and different α parameters. The results are compared to the system where each node has global membership knowledge. We test each reliability predicate with different numbers of copies C and the average target set size K parameters. Unless explicitly stated, each experiment is done with the following default membership view parameter values: rebalance sensitivity $\alpha = 0.05$, balance aggressiveness $\beta = 0.1$, total number of nodes $N = 1,442$. All predicate parameters, along with root-mean-square and standard deviation values of reliability distributions are shown in Table 6.3.

6.3.1 Uniform Reliability

The results of the uniform gossip-based multicast simulation are shown in Figure 6.6. The overall conclusions are as follows. It can be seen that each node merely obtains the same node-level

Predicate	RMS	StdevErr
Uniform ($copies = 1$)	0.0418	0.0283
Uniform ($copies = 3$)	0.0723	0.0533
Availability-proportional	0.0696	0.0492
Bimodal ($E[r_1] = 0.3$,$E[r_2] = 0.9$,Threshold=0.5)	0.0994	0.0703
Threshold-linear (Threshold=0.3)	0.0866	0.0570

Table 6.2: Root-mean-square and standard deviation of error of reliability distributions from each predicate of AVCast with inconsistent knowledge availability information

reliability, regardless of its availability value. As the number of copies forwarded per message increases, the multicast reliability also increases. Also, the equations derived in Section 4 predict the system-wide reliability accurately when the average target size is more than $2logN$. Figure 6.6(a) displays the availability-reliability scatter plot where each point represents each node. There are three sets of plots in the graph, representing two experiments with two different C, the number of copies forwarded per message. Both experiments used the average target size $= 2logN$ and $\alpha = 0.05$.

Figure 6.6(b) demonstrates the average node reliability of system with different view sizes K and different C parameters without the rebalance procedure (simulations with rebalance procedure yield the similar result as ones without rebalance procedure). The figure shows the effectiveness of system in the sense that setting target view size more than or equal to $2logN$ suffices to have the same performance as setting each node to have the global membership knowledge. The performance difference between systems with different view sizes converges when C is increased. The figure concludes that for the system sizes considered, setting $K = 3logN$ and $C = 4$ results in good performance while incurring reasonable space and network overhead.

Another perspective to evaluate the constant reliability predicate is the consistency in the quality of service each node receives from the system. Figure 6.6(c) shows the standard deviation of node reliability in systems for $k = 3logN$ and different α values. The result is consistent with the result from figure 6.6(a) that the standard deviation increases as C increases from 1 to 2, but the standard deviation decreases as C increases beyond 2. Also, the smaller α is, the smaller the standard deviation in the system. Generally, the standard deviations of node reliability in local-view

systems are comparable to the one in the global-view system.

In conclusion, the simulation shows that the system's behavior in uniform gossiping follows the constant reliability predicate very well. In addition, setting $K = 3logN$, $C = 3$, and $\alpha = 0.1$ is an appropriate configuration.

6.3.2 Availability-proportional Reliability

This section presents the simulation result of the availability-proportional gossiping protocol. The experiment is done with view size $K = 3logN$ and $\alpha = 0.05$. Figure 6.7(a) shows the scatter plot between the availability and the reliability at each node. As shown in the figure, most nodes have multicast reliability at roughly the same level as its availability, which is consistent with the availability-proportional predicate.

Figure 6.7(b) shows the cumulative distribution of nodes whose reliability differs from their availability in different scales. As seen from the picture, around 60% of nodes in the system obtain multicast reliability that differs from their availability within a value of 0.05 or less. Around 80% of nodes in the system obtain multicast reliability that differs from their availability in the scale of 0.1. Only 2% of all nodes obtain the multicast reliability that differs from their availability more than 0.2.

In conclusion, the availability-proportional gossiping protocol performs well in the sense that each node receives the service with quality proportional to its contribution to the system. Only a few nodes receive a service with quality significantly different from their behavior.

6.3.3 Multi-class Reliability

The results of the two experiments (bimodal gossiping and threshold-linear gossiping) are shown in Figure 6.8(a) and 6.8(b) Both experiments are done with view size $K = 3logN$, $\alpha = 0.05$, and $\beta = 0.1$. Figure 6.8(a) presents the result of the bimodal gossiping protocol with two expected reliability values equal to 0.3 and 0.75 respectively. Figure 6.8(b) presents the result of the

Predicate	Max	Avg	Stdev
Availability-proportional	3	1.5062	0.4690
Bimodal	3	2.1354	0.2088
Mixed	3.25	1.6127	0.2411

Table 6.3: Numbers of copies forwarded per message received in each predicate

threshold-linear gossiping protocol with availability threshold equal to 0.3. Both results confirm the AVCast predicates correctly. Although the reliability distributions do not perfectly resemble the desired predicates, they follow the trends of the predicates.

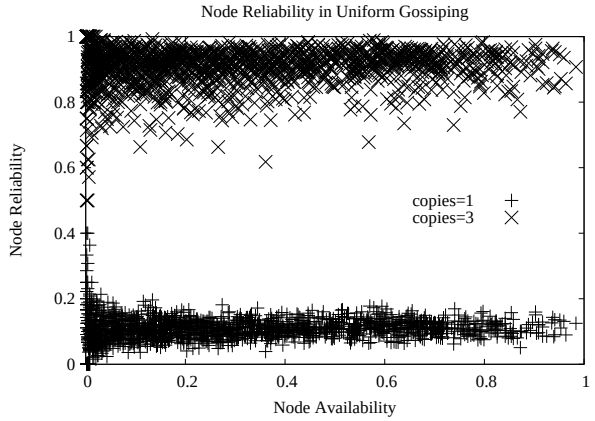
6.3.4 Number of Copies Forwarded Per Message

To measure how well loads are distributed across multicast nodes, Table 6.3.4 shows values of the maximum, the average, and the standard deviation of number of copies each node forwards per message received in each predicate (except uniform predicates in which numbers of copies forwarded are constants). As seen from the table, all predicates achieve a good level of load balancing. In fact, no node needs to forward more than 4 copies of message per message received. We can conclude from such results that the monitoring and membership protocol succeeds to provide the consistent view and load balancing for each multicast node.

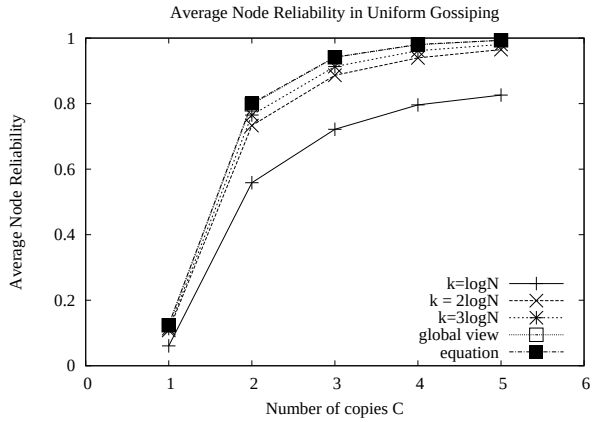
6.4 Resilence to Uncooperative Nodes

It is usually unavoidable for heterogeneous, lage-scale, distributed systems to contain a fraction of uncooperative or even malicious peers. Designing a distributed protocol that can tolerate a level of such ill-behaved nodes is an important research issue. Here, we present some initial results of how resilient AVCast protocol is under the presence of uncooperative peers. In the experiment, we consider only *lazy peers* (i.e., free-riders) which only receive messages without forwarding any copy to their target nodes. We study how the system behaves globally under some fractions of lazy peers.

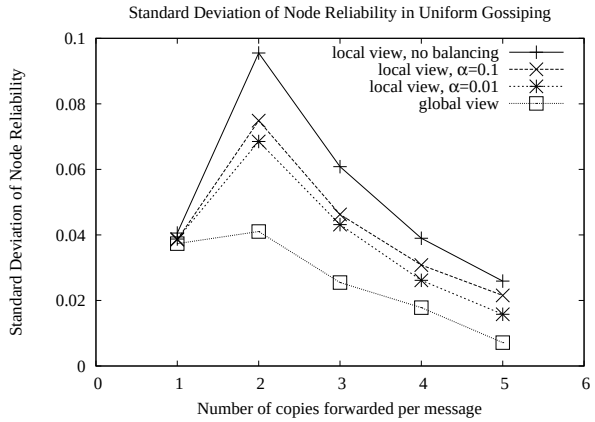
Figure 6.9 presents the results of consistent-knowledge uniform gossiping and proportional



(a) Node-level reliability distribution for $K = 2logn$ and $\alpha = 0.05$

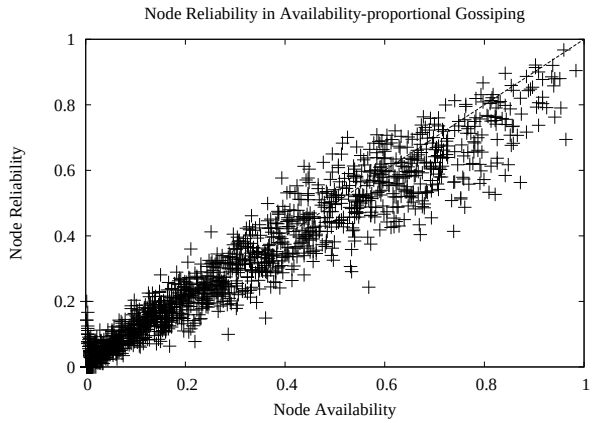


(b) Average node-level reliability

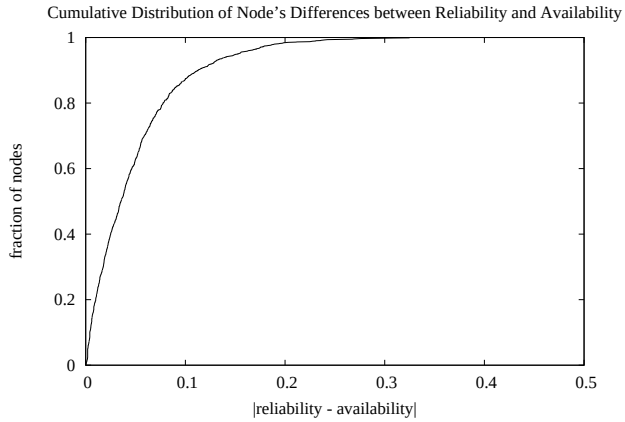


(c) Standard deviation of Node-level reliability with different α

Figure 6.6: Inconsistent availability knowledge : Uniform reliability predicate under churn (Overnet trace)

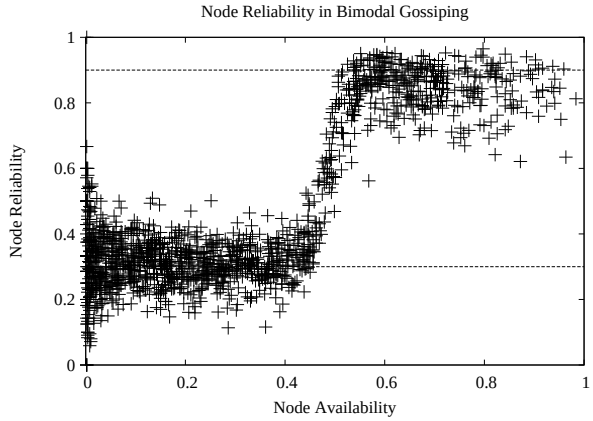


(a) Node-level reliability distribution

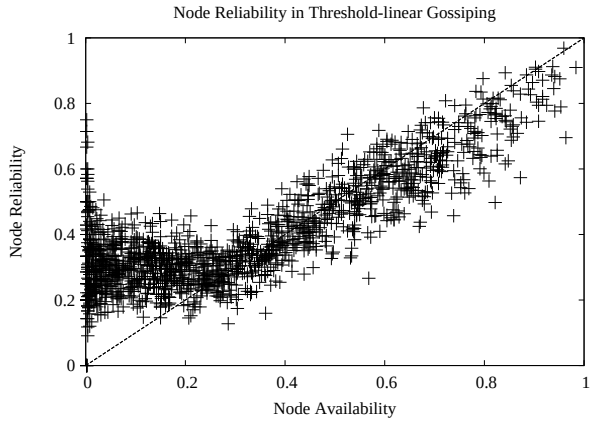


(b) CDF of nodes by differences between availability and reliability

Figure 6.7: Inconsistent availability knowledge : Availability-proportional reliability predicate under churn (Overnet trace)



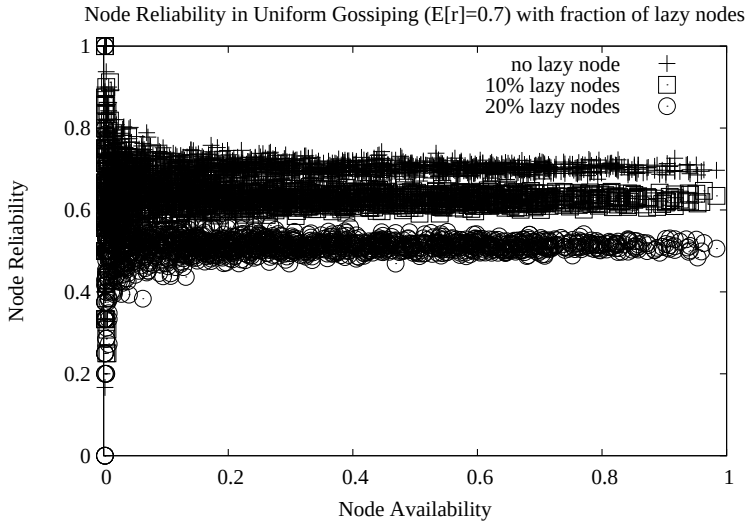
(a) Bimodal reliability predicate



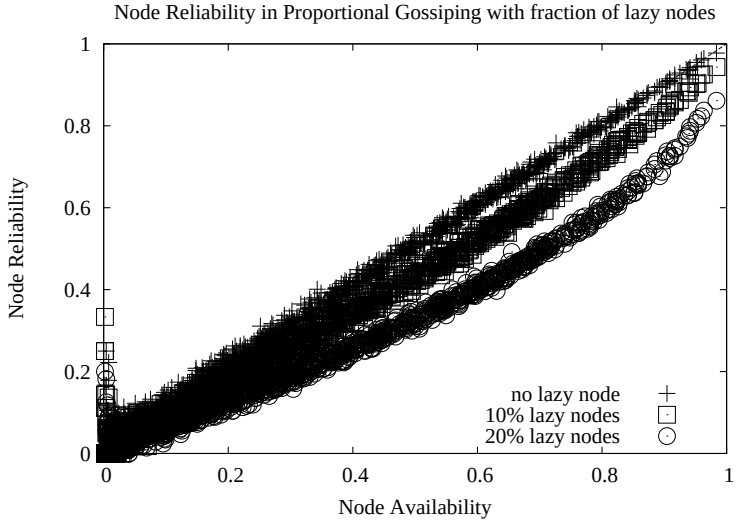
(b) Threshold-linear reliability predicate

Figure 6.8: Inconsistent availability knowledge : Node-level reliability distribution with multi-class reliability predicate under churn (Overnet trace)

gossiping with different fractions of lazy peers. It can be seen from the figure that AVCast can tolerate up to 20% of lazy peers. Although the average reliability of uniform gossiping decreases as the fraction of lazy peers increases, the system continues to maintain the predicate in the sense that each node still receives the same reliability. On the other hand, we observed that the reliability distribution of the system with proportional gossip starts to diverge from its desired predicate ($r(a) = a$) when the fraction of lazy nodes is more than 20%.



(a) Uniform reliability predicate with expected reliability=0.7



(b) Availability-proportional reliability predicate

Figure 6.9: Consistent availability knowledge : Node-level reliability distribution in the system with lazy nodes

Chapter 7

Conclusions and Future Directions

7.1 Conclusions

This work presented AVCast, an availability-aware membership management and multicast framework. AVCast provides an availability-monitoring service and an availability-aware gossip-based multicast service for each node in the system in a decentralized manner. The paper also presented a generic framework that allows an application to adjust AVCast parameters in order to implement a multicast system with the desired availability-dependent reliability predicate. Finally, the paper analyzed several reliability predicates that lead to system fairness or desired behavior—uniform per-node reliability, availability-proportional per-node reliability, and generalized availability-dependent reliability. The experimental results validated the correctness and applicability of the proposed schemes.

7.2 Future Directions

There are many issues that can lead to improvements of AVCast's performance and applicability. One interesting issue is how to reduce the effect of view inconsistency in multicast receivers. AVCast currently uses a decentralized availability monitoring protocol to obtain different target sets and pinging sets for different receivers. such different sets vary in size and availability distribution, leading to inaccurate reliability distribution—the reliability distribution does not perfectly resemble the desired predicate. A possible approach to solve such view variance is that each peer exchanges its neighbor lists (either target set or pinging set, or both) with a randomly chosen

neighbor in order to increase the view size and reduce the variance of calculating the expected availability of the system.

Another interesting issue is how to maintain the consistency and the efficiency of the predicate under the presence of uncooperative peers (e.g., freeriders). Although the problem of selfish nodes boasting their availability values in order to receive better service can be solved by consistent randomization technique in AVCast, other techniques are required to prevent uncooperative or even malicious peers from causing the system's performance to degrade or diverge from the desired predicate. For example, some selfish nodes may only receive messages without forwarding them to other peers. In another more extreme case, malicious nodes may forward messages in a way that does not conform to the desired predicate function, resulting in catastrophic global behavior. Such issue remains an open question in the field of research on security in distributed systems.

References

[1] E. Adar and B. A. Huberman, "Free Riding on Gnutella," *First Monday*, vol. 5, no. 10, 2000.

[2] K. C. Almeroth and M. H. Ammar, "Collecting and Modeling the Join/Leave Behavior of Multicast Group Members in the MBone," in *Proc. HPDC '96*. Washington, DC, USA: IEEE Computer Society, 1996, p. 209.

[3] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris, "Resilient Overlay Networks," in *Proc. SOSP '01*. New York, NY, USA: ACM Press, 2001, pp. 131–145.

[4] G. Badishi, I. Keidar, and A. Sasson, "Exposing and Eliminating Vulnerabilities to Denial of Service Attacks in Secure Gossip-Based Multicast," in *Proc. DSN '04*. IEEE Computer Society, 2004, pp. 223–232.

[5] R. Bhagwan, S. Savage, and G. M. Voelker, "Understanding Availability," in *Proc. IPTPS '03*, ser. Lecture Notes in Computer Science, M. F. Kaashoek and I. Stoica, Eds., vol. 2735. Springer, 2003, pp. 256–267.

[6] K. P. Birman, M. Hayden, O. Ozkasap, Z. Xiao, M. Budiu, and Y. Minsky, "Bimodal Multicast," *ACM Trans. Comput. Syst.*, vol. 17, no. 2, pp. 41–88, May 1999.

[7] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. I. T. Rowstron, and A. Singh, "Splitstream: High-Bandwidth Multicast in Cooperative Environments." in *Proc SOSP'03*, 2003, pp. 298–313.

[8] Y.-H. Chu, S. G. Rao, and H. Zhang, "A Case for End System Multicast." in *Proc. SIGMETRICS'00*, 2000, pp. 1–12.

[9] J. Crowcroft and M. Hofmann, Eds., *Networked Group Communication, Third International COST264 Workshop, NGC 2001, London, UK, November 7-9, 2001, Proceedings*, ser. Lecture Notes in Computer Science, vol. 2233. Springer, 2001.

[10] F. Dabek, J. Li, E. Sit, J. Robertson, M. F. Kaashoek, and R. Morris, "Designing a DHT for Low Latency and High Throughput," in *Proc. NSDI '04*. USENIX, 2004, pp. 85–98.

[11] A. Das, I. Gupta, and A. Motivala, "SWIM: Scalable Weakly-consistent Infection-style Process Group Membership Protocol." in *Proc. DSN'02*. IEEE Computer Society, 2002, pp. 303–312.

[12] S. E. Deering, D. Estrin, D. Farinacci, V. Jacobson, C.-G. Liu, and L. Wei, "The pim Architecture for Wide-area Multicast Routing." *IEEE/ACM Trans. Netw.*, vol. 4, no. 2, pp. 153–162, 1996.

[13] A. J. Demers, D. H. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. E. Sturgis, D. C. Swinehart, and D. B. Terry, "Epidemic Algorithms for Replicated Database Maintenance." in *PODC'87*, 1987, pp. 1–12.

[14] D. Eastlake 3rd and P. Jones, "US Secure Hash Algorithm 1 (SHA1)," RFC 3174, Sept. 2001.

[15] P. T. Eugster, R. Guerraoui, S. B. Handurukande, P. Kouznetsov, and A.-M. Kermarrec, "Lightweight Probabilistic Broadcast," in *Proc. DSN '01*. IEEE Computer Society, 2001, pp. 443–452.

[16] S. Floyd, V. Jacobson, C.-G. Liu, S. McCanne, and L. Zhang, "A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing," *IEEE/ACM Trans. Netw.*, vol. 5, no. 6, pp. 784–803, 1997.

[17] I. T. Foster, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations." in *Proc CCGRID'01*. IEEE Computer Society, 2001, pp. 6–7.

[18] I. T. Foster, J. Gieraltowski, S. Gose, N. Maltsev, E. N. May, A. Rodriguez, D. Sulakhe, A. Vaniachine, J. Shank, S. Youssef, D. Adams, R. Baker, W. Deng, J. Smith, D. Yu, I. Legrand, S. Singh, C. Steenberg, Y. Xia, M. A. Afaq, E. Berman, J. Annis, L. A. T. Bauerdick, M. Ernst, I. Fisk, L. Giacchetti, G. E. Graham, A. Heavey, J. Kaiser, N. Kuropatkin, R. Pordes, V. Sekhri, J. Weigand, Y. Wu, K. Baker, L. Sorrillo, J. Huth, M. Allen, L. Grundhoefer, J. Hicks, F. Luehring, S. Peck, R. Quick, S. Simms, G. Fekete, J. vandenBerg, K. Cho, K. Kwon, D. Son, H. Park, S. Canon, K. R. Jackson, D. E. Konerding, J. Lee, D. Olson, I. Sakrejda, B. Tierney, M. Green, R. Miller, J. Letts, T. Martin, D. Bury, C. Dumitrescu, D. Engh, R. Gardner, M. Mambelli, Y. Smirnov, J.-S. Vöckler, M. Wilde, Y. Zhao, X. Zhao, P. Avery, R. Cavanaugh, B. Kim, C. Prescott, J. L. Rodriguez, A. Zahn, S. McKee, C. T. Jordan, J. E. Prewett, T. L. Thomas, H. Severini, B. Clifford, E. Deelman, L. Flon, C. Kesselman, G. Mehta, N. Olomu, K. Vahi, K. De, P. McGuigan, M. Sosebee, D. Bradley, P. Couvares, A. DeSmet, C. Kireyev, E. Paulson, A. Roy, S. Koranda, B. Moe, B. Brown, and P. Sheldon, "The Grid2003 Production Grid: Principles and Practice." in *Proc. HPDC'04*. IEEE Computer Society, 2004, pp. 236–245.

[19] A. J. Ganesh, A.-M. Kermarrec, and L. Massouli, "SCAMP: Peer-to-Peer Lightweight Membership Service for Large-Scale Group Communication," in *Proc. NGC '01*. London, UK: Springer-Verlag, 2001, pp. 44–55.

[20] I. Gupta, "On The Design of Distributed Protocols from Differential Equations." in *Proc. PODC'04*, S. Chaudhuri and S. Kutten, Eds. ACM, 2004, pp. 216–225.

[21] I. Gupta, K. P. Birman, P. Linga, A. J. Demers, and R. van Renesse, "Kelips: Building an Efficient and Stable P2P DHT through Increased Memory and Background Overhead." in *Proc. IPTPS'03*, ser. Lecture Notes in Computer Science, M. F. Kaashoek and I. Stoica, Eds., vol. 2735. Springer, 2003, pp. 160–169.

[22] I. Gupta, R. van Renesse, and K. P. Birman, "A Probabilistically Correct Leader Election Protocol for Large Groups," in *Proc. DISC'00*, ser. Lecture Notes in Computer Science, M. Herlihy, Ed., vol. 1914. Springer, 2000, pp. 89–103.

[23] X. Hei, C. Liang, J. Liang, Y. Liu, and K. W. Ross, "Insight into PPLive: A Measurement Study of a Large-Scale P2P IPTV System," in *Workshop on IPTV services over World Wide Web in conjunction with Proc. WWW'06*, May 2006.

[24] J. Jannotti, D. K. Gifford, K. L. Johnson, M. F. Kaashoek, and J. O. Jr., "Overcast: Reliable Multicasting with an Overlay Network." in *Proc. OSDI'00*, 2000, pp. 197–212.

[25] M. F. Kaashoek and I. Stoica, Eds., *Peer-to-Peer Systems II, Second International Workshop, IPTPS 2003, Berkeley, CA, USA, February 21-22,2003, Revised Papers*, ser. Lecture Notes in Computer Science, vol. 2735. Springer, 2003.

[26] R. Karp, C. Schindelhauer, S. Shenker, and B. Vocking, "Randomized Rumor Spreading," in *Proc. FOCS '00*. Washington, DC, USA: IEEE Computer Society, 2000, p. 565.

[27] D. Kostoulas, D. Psaltoulis, I. Gupta, K. Birman, and A. Demers, "Decentralized Schemes for Size Estimation in Large and Dynamic Groups," in *Proc. NCA '05*. IEEE Computer Society, 2005, pp. 41–48.

[28] R. Melamed and I. Keidar, "Araneola: A Scalable Reliable Multicast System for Dynamic Environments," in *Proc. NCA '04*. IEEE Computer Society, 2004, pp. 5–14.

[29] A. Oram, *Peer-to-Peer : Harnessing the Power of Disruptive Technologies*. Sebastopol, CA: O'Reilly & Associates, 2001.

[30] S. Paul, K. K. Sabnani, J. C.-H. Lin, and S. Bhattacharyya, "Reliable Multicast Transport Protocol (RMTP)," *IEEE JSAC*, vol. 15, no. 3, pp. 407–421, 1997. [Online]. Available: citeseer.ist.psu.edu/paul97reliable.html

[31] T. Pongthawornkamol and I. Gupta, "AVCast : New Approaches for Implementing Availability-Dependent Reliability for Multicast Receivers," in *Proc. SRDS'06*, October 2006, to appear.

[32] J. Radcliffe, "Book reviews: *The Mathematical Theory of Infectious Diseases and its Applications. 2nd Edition*, by Norman T. J. Bailey," vol. 26, no. 1, pp. 85–87, 1977.

[33] S. Ratnasamy, P. Francis, M. Handley, R. M. Karp, and S. Shenker, "A Scalable Content-Addressable Network." in *Proc. SIGCOMM'01*, 2001, pp. 161–172.

[34] S. Ratnasamy, M. Handley, R. M. Karp, and S. Shenker, "Application-Level Multicast Using Content-Addressable Networks." in *Proc NGC'01*, ser. Lecture Notes in Computer Science, J. Crowcroft and M. Hofmann, Eds., vol. 2233. Springer, 2001, pp. 14–29.

[35] S. C. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz, "Handling Churn in a DHT," in *Proc. USENIX Annual Technical Conference, General Track*. USENIX, 2004, pp. 127–140.

[36] R. Rivest, “The MD5 Message-Digest Algorithm ,” RFC 1321, Apr. 1992.

[37] L. Rodrigues, S. B. Handurukande, J. O. Pereira, R. Guerraoui, and A.-M. Kermarrec, “Adaptive Gossip-Based Broadcast,” in *Proc. DSN ’03*. IEEE Computer Society, 2003, pp. 47–56.

[38] M. Roussopoulos, M. Baker, D. S. H. Rosenthal, T. J. Giuli, P. Maniatis, and J. C. Mogul, “2 P2P or Not 2 P2P?” in *Proc. IPTPS’04*, ser. Lecture Notes in Computer Science, G. M. Voelker and S. Shenker, Eds., vol. 3279. Springer, 2004, pp. 33–43.

[39] A. I. T. Rowstron and P. Druschel, “Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems.” in *Proc. Middleware’01*, ser. Lecture Notes in Computer Science, R. Guerraoui, Ed., vol. 2218. Springer, 2001, pp. 329–350.

[40] A. I. T. Rowstron, A.-M. Kermarrec, M. Castro, and P. Druschel, “SCRIBE: The Design of a Large-Scale Event Notification Infrastructure.” in *Proc NGC’01*, ser. Lecture Notes in Computer Science, J. Crowcroft and M. Hofmann, Eds., vol. 2233. Springer, 2001, pp. 30–43.

[41] J. H. Saltzer, D. P. Reed, and D. D. Clark, “End-To-End Arguments in System Design,” *ACM Trans. Comput. Syst.*, vol. 2, no. 4, pp. 277–288, 1984.

[42] D. Sandler, A. Mislove, A. Post, and P. Druschel, “FeedTree: Sharing Web Micronews with Peer-to-Peer Event Notification.” in *IPTPS*, ser. Lecture Notes in Computer Science, M. Castro and R. van Renesse, Eds., vol. 3640. Springer, 2005, pp. 141–151.

[43] S. Saroiu, P. K. Gummadi, and S. D. Gribble, “A Measurement Study of Peer-to-Peer File Sharing Systems,” in *Proc. MMCN ’02*, San Jose, CA, USA, January 2002. [Online]. Available: citeseer.ist.psu.edu/article/saroiu02measurement.html

[44] A. Singh, T.-W. Ngan, P. Druschel, and D. S. Wallach, “Eclipse Attacks on Overlay Networks: Threats and Defenses,” in *Proc. InfoCom ’06*, Barcelona, April 2006.

[45] I. Stoica, R. Morris, D. R. Karger, M. F. Kaashoek, and H. Balakrishnan, “Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications.” in *Proc. SIGCOMM’01*, 2001, pp. 149–160.

[46] C. Tang and C. Ward, “GoCast: Gossip-Enhanced Overlay Multicast for Fast and Dependable Group Communication,” in *Proc. DSN ’05*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 140–149.

[47] S. Voulgaris, D. Gavidia, and M. van Steen, “Cyclon: Inexpensive Membership Management for Unstructured P2P Overlays.” *J. Network Syst. Manage.*, vol. 13, no. 2, 2005.

[48] S. Voulgaris, E. Rivière, A.-M. Kermarrec, and M. van Steen, “Sub-2-Sub: Self-Organizing Content-Based Publish Subscribe for Dynamic Large Scale Collaborative Networks,” in *Proc. IPTPS’06*, 2006.

[49] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph, “Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing,” Berkeley, CA, USA, Tech. Rep., 2001.

[50] S. Zhuang, B. Y. Zhao, A. D. Joseph, R. H. Katz, and J. Kubiatowicz, “Bayeux: an Architecture for Scalable and Fault-Tolerant Wide-Area Data Dissemination.” in *Proc. NOSSDAV’01*. ACM, 2001, pp. 11–20.