



PEPITO
IST-2001-33234
PEer-to-Peer Implementation and TheOry

Deliverable no: D2.4

First progress report on specification of probabilistic reliable broadcast

REPORT VERSION: first

REPORT PREPARATION DATE: 2003.06.30

CLASSIFICATION: Public

DELIVERABLE NO: D2.4 DUE DATE: Month 18 DELIVERY DATE: Month 18

PROJECT START DATE: 2002.01.01 PROJECT DURATION: 36 months

RESPONSIBLE PARTNER: Swiss Institute of Technology in Lausanne (EPFL)

PARTICIPATING PARTNERS: EPFL

PROJECT COORDINATOR: Swedish Institute of Computer Science AB

PROJECT PARTNERS: SICS, EPFL Lausanne, INRIA Paris, KTH Stockholm, UCL Louvain, University of Cambridge UK



**Project funded by the European Community under the
'Information Society Technologies' Programme (1998–
2002)**

Project Number: IST-2001-33234

Project Acronym: PEPITO

Title: PEer-to-Peer Implementation and TheOry

Deliverable No: D2.4

First progress report on specification of probabilistic reliable
broadcast

Due date: project month 18

Delivery date: 2003-06-30

Responsible Partner: EPFL

25th June 2003

Prepared by Rachid Guerraoui, with input from Patrick Th. Eugster and Petr Kouznetsov

Contents

1	Introduction	3
2	Δ-Reliable Broadcast: Specification	5
2.1	System and Environment	5
2.2	Δ -Reliable Broadcast	5
2.3	Interpretation of ρ and ψ	6
2.4	Reliability Distribution Function	6
2.5	Comparing Broadcast Algorithms	7
2.6	Atomicity	7
2.7	Reliable Broadcast: From Perfect to Useless	7
3	Bimodal Multicast	8
3.1	Protocol Overview	8
3.2	Model	8
3.3	Analysis	10
3.4	Δ -Reliability of Bimodal Multicast	11
3.5	Approximation of $\Phi_{BM}(\rho, \mathcal{E}_{BM})$	12
3.6	Expected reliability degree of Bimodal Multicast	13
4	IP Multicast	13
4.1	Protocol Overview	13
4.2	Model	14
4.3	Analysis	14
4.4	Δ -Reliability of IP Multicast	15
4.5	Expected reliability degree of IP Multicast	16
5	Comparing Bimodal Multicast and IP Multicast	16
6	Related Issues	18

1 Introduction

Reliable broadcast. The growing interest in peer-to-peer computing has underlined the importance of reliable broadcast algorithms. Traditionally, the reliability of broadcast algorithms has been defined by three properties [9]:

Integrity. For any message m , every correct process delivers m at most once, and only if m was previously broadcast by sender(m).

Validity. If a correct process p broadcasts a message m , then p eventually delivers m .

Agreement. If a correct process delivers a message m , then every correct process eventually delivers m .

To obtain these strong properties in a system with process and link failures, one employs costly, traditionally acknowledgement-based algorithms. These can be effective in a local environment, but may give unstable or unpredictable performance under stress, and hence tolerate limited scalability [4], contradicting the stringent scalability properties claimed by today's peer-to-peer applications.

More pragmatic approaches to broadcast focus on performance in very large-scale settings, and sacrifice strong reliability guarantees (in the sense of [9]) to performance. Examples include the Internet *Multicast Usenet* (MUSE) protocol [11], the *XPress Transfer Protocol* (XTP) [17] or a broad range of so-called *network-level* protocols building on *IP Multicast* [5].¹ The reliability of such protocols is typically expressed in *best-effort* terminology: if a participant discovers a failure, the "most reasonable" effort is made to overcome it, but there is no guarantee that such an attempt will be successful. In short, best-effort reliable algorithms are simply not intended to satisfy the traditional properties of Reliable Broadcast [9].

Birman et al [3] proposed a new look at broadcast reliability. They informally characterized a *useful* reliable broadcast algorithm through a set of properties (illustrated by their *gossip-based* [6] *Bimodal Multicast* algorithm [3]), including the following:

Atomicity. *The protocol provides a bimodal delivery guarantee, under which there is a high probability that each broadcast will reach almost all processes, a low probability that each broadcast will reach just a very small set of processes, and a vanishingly small probability that it will reach some intermediate number of processes. That is, the traditional atomic "all or nothing" guarantee becomes "almost all or almost none".*

This property is very appealing from a practical viewpoint, but still rather informal,² and in [3] the authors concentrate on giving a behavioral analysis of the Bimodal Multicast algorithm.

An interesting approach to analyze the reliability of broadcast algorithms can be found also in [12], where reliability is the probability that either all or no correct processes deliver a broadcast message if the sender is correct. However, the formal proof of the correctness of an algorithm with respect to a specification is not presented.

¹E.g., *Reliable Multicast Transport Protocol* (RMTP) [13], *Reliable Multicast Protocol* (RMP) [16], *Log-Based Receiver-Reliable Multicast* (LBRM) [10], *Scalable Reliable Multicast* (SRM) [8].

²The "almost all or almost none" is in fact "almost always almost all or almost none"; the use of the term "almost" is indeed intuitive, but gives a rather informal nature to this property.

Reliability measure. The motivation of this work is the observation that there is a lack of a precise but also practical measure to estimate the reliability of inexpensive best-effort algorithms. Intuitively, those are less reliable than algorithms that comply with the strong properties of [9] but more reliable for instance than a simple *multisend*. But what is the actual meaning of “more reliable” and “less reliable”? Addressing this question is not trivial, yet fundamental, since these algorithms are precisely those used in practice.

Contributions. The aim of this work is to introduce a precise *measure* to quantify the intuitively understandable notion of reliability used in practice. In other terms, we do not aim at introducing an original broadcast algorithm which is more reliable than others, but at defining what the very statement “more reliable” may mean.

To this end, we introduce a new *non-binary probabilistically flavored* specification of the reliability of broadcast algorithms called Δ -Reliable Broadcast. Through the measure of Δ -Reliability, we contribute to bridging the gap between theory and practice in broadcast reliability. In short, our specification leads to describing the reliability *distribution* of a broadcast algorithm, that is, a *probability* distribution for the reliability *degree* of an algorithm. The use of probabilities enables the capture, to a certain degree, of the nondeterminism inherent to large-scale systems.

We illustrate our specification through two well-known examples. The first one, Bimodal Multicast [3], is a representative of the rapidly proliferating family of gossip-based algorithms which have received much attention lately, precisely because they are “pretty reliable”. As a representative of the class of best-effort algorithms often used in practice, namely the network-level protocols, we discuss IP Multicast [5].

We also demonstrate the use of Δ -Reliability in comparing broadcast algorithms by contrasting Bimodal Multicast and IP Multicast, confirming the intuition that, in most practical environments, Bimodal Multicast is “more reliable” than IP Multicast, especially as the system grows in size.

The practical use of our Δ -Reliability measure is furthermore illustrated through the scalability analysis of Bimodal Multicast which illuminates very attractive scalability properties of the algorithm.

Limitations. There is no universal way to analyze and quantify the reliability of a broadcast algorithm: when talking about “reliability” we actually mean “reliability in a certain environment”. To quantify the reliability of such an algorithm in a probabilistic sense, we need the precise knowledge of system parameters and an accurate model of the behavior of the algorithm based on former ones. Such parameters are not always available, and models usually represent approximations. This outlines the main limitation of our notion of Δ -Reliable Broadcast: not every system model (and algorithm) matches it well. Moreover, even with the most precise model, calculations might require approximations. The weak consolation is that there does not seem to be any alternative perfect notion of a broadcast reliability which covers all possible system models.

Roadmap. Section 2 introduces Δ -Reliability. Section 3 discusses the Δ -Reliability of Bimodal Multicast. Section 4 similarly applies our specification of Δ -Reliability to IP Multicast. Section 5 illustrates the use of Δ -Reliability in comparing broadcasting algorithms through Bimodal Multicast and IP Multicast. Section 6 discusses alternative reliability measures and related issues.

2 Δ -Reliable Broadcast: Specification

This section presents our approach to measuring, in a probabilistic sense, the reliability of a broadcast algorithm.

2.1 System and Environment

We consider an asynchronous (in the sense of [9]) system Π of processes $\{p_1, \dots, p_n\}$. Processes are connected through fair lossy channels of infinite capacity. Let m be any message, uniquely identified and equipped, in particular, with a parameter $sender(m)$. Processes communicate by message passing defined by the primitives $send(m)$ and $receive(m)$. Broadcast is defined by the primitives $broadcast(m)$ and $deliver(m)$. Processes are subject to *crash* failures. A *correct* process is one that never crashes.³ To simplify presentation, we do not consider Byzantine failures, and we assume that crashed processes do not recover.

The analysis of a broadcast algorithm usually depends on more properties of the underlying system than only its size and composition, as well as on parameters of the algorithm itself. Henceforth, we will use the term *environment*, denoted \mathcal{E} , to refer to the set of relevant system properties and algorithm parameters. Environment \mathcal{E} represents a point in an *environment space* \mathbb{E} , a set of all possible combinations of parameters: $\mathcal{E} \in \mathbb{E}$.

Let B_1 and B_2 be two broadcast algorithms that have different sets of parameters in their respective environments \mathcal{E}_1 and \mathcal{E}_2 . To compare the algorithms we introduce a *compound* environment - a union of the two environments, $\mathcal{E} = \mathcal{E}_1 \cup \mathcal{E}_2$. Note that the composition makes sense only if the related parameters in \mathcal{E}_1 and \mathcal{E}_2 do not contradict. For example, if the system models for B_1 and B_2 comprise the probabilities of an end-to-end message loss, respectively, $\varepsilon_1 \in \mathcal{E}_1$ and $\varepsilon_2 \in \mathcal{E}_2$, then $\varepsilon_1 = \varepsilon_2$. Otherwise, the comparison does not seem meaningful. In Section 5 we will illustrate this through the concrete examples.

2.2 Δ -Reliable Broadcast

Let Δ be any pair of real numbers (ψ, ρ) ($\psi, \rho \in [0, 1]$). We say that a broadcast protocol *complies with the specification of Δ -Reliable Broadcast* (or a broadcast protocol *is Δ -Reliable*) iff the following properties are *simultaneously satisfied with probability ψ* :

Integrity. For any message m , every correct process delivers m at most once, and only if m was previously broadcast by $sender(m)$.

Validity. If a correct process p broadcasts a message m then p eventually delivers m .

Δ -Agreement. If a correct process delivers a message m , then eventually at least a fraction ρ of correct processes deliver m .

Properties *Validity* and *Integrity* here are the same as in traditional Reliable Broadcast [9].

Integrity is a *safety* property: it says that some “bad” thing never happens, in particular, that every delivered message is earlier broadcast and there are no duplicate deliveries. Prevailing broadcast algorithms *always* guarantee the first part of the property: no “bogus” messages are delivered. However, to guarantee the absence of duplicate deliveries in an asynchronous system one should track infinitely large message history which is not feasible in realistic settings. That is why we define Integrity in the probabilistic context.

³In practice, we mean that a correct process does not crash *in a given run* of an algorithm.

Although Validity can be viewed as a necessary property, it is not always fulfilled for some broadcast algorithms. For example, some acknowledgement-based broadcast algorithms do not deliver a broadcast message at the source until the message has become stable (received by most of the correct processes in the system), and there is a probability that a stable message is never delivered at the source (Validity is violated). That is why Validity is probabilistically guaranteed in the specification above.

Agreement, as defined in [9], is transformed here into Δ -*Agreement* which is less restrictive in terms of the number of processes that need to deliver the message and also has a probabilistic flavor.

2.3 Interpretation of ρ and ψ

$\Delta = (\psi, \rho)$ represents a basic “reliability measure” of a broadcast algorithm. The values of ψ and ρ are intrinsically coupled: ψ can roughly be pictured as the probability with which at least a fraction ρ of processes behave according to the properties of Reliable Broadcast [9]. More precisely, a sample $\Delta = (\psi, \rho)$ is characterized by:

Reliability probability ψ : ψ is the probability that a protocol run is completed “successfully”. That is, once a message m is broadcast and delivered by a correct process, “enough” correct processes eventually deliver m .

Reliability degree ρ : ρ defines the fraction of correct processes which eventually deliver m .

For instance, to satisfy the properties of Δ -Reliable Broadcast with $\Delta = (\psi = 0.95, \rho = 0.9)$, once a message m is broadcast, an algorithm should, with probability 0.95, deliver m to 90% of correct processes in the system. In other terms, in a run of the system with 10 correct processes, one can expect 95% of all messages which are broadcast to be delivered by at least 9 processes (not necessarily the same processes for every message).

2.4 Reliability Distribution Function

Δ -Reliable Broadcast does not aim at giving a binary interpretation for the reliability of a broadcast algorithm as in [9]. Instead, it defines a *measure* of reliability, such that *any* broadcast protocol can be proven to be Δ -Reliable with *some* set of parameters $\Delta = (\psi, \rho)$.

In a practical system, with a given required reliability degree ρ , several broadcast algorithms can easily be compared along the ψ they offer for the given ρ . To give an informal measure of the general performance in terms of reliability of a broadcast algorithm, several samples $\Delta_1 \dots \Delta_s$ are usually sufficient. A precise expression of the reliability of such an algorithm requires however the consideration of the probabilities for all possible $\rho \in [0, 1]$, especially when comparing two algorithms in general. Indeed, consider two algorithms B_1 and B_2 and a set $\Delta_{B_1} = (0.9, 0.9)$ and $\Delta_{B_2} = (0.85, 0.9)$. Algorithm B_1 seems to perform better for $\rho_{B_1} = \rho_{B_2} = 0.9$. However, this information is not sufficient to promote algorithm B_1 as “more reliable” than algorithm B_2 , since for $\rho'_{B_1} = \rho'_{B_2} = 0.95$, algorithm B_2 might offer a ψ'_{B_2} of 0.8, while in the case of algorithm B_1 , ψ'_{B_1} might be only 0.7.

To compare two algorithms in a more general manner, we define a *reliability distribution function* $\psi_B(\rho, \mathcal{E})$ of a broadcast algorithm B :

$$\psi_B : [0, 1] \times \mathbb{E} \mapsto [0, 1], \quad (1)$$

such that for any $\rho \in [0, 1]$ and $\mathcal{E} \in \mathbb{E}$, B is Δ -Reliable with $\Delta = (\psi_B(\rho, \mathcal{E}), \rho)$.

As a direct consequence of the definition of Δ -Agreement — a sample in which a fraction ρ_0 of processes deliver every message is also a sample in which *at least* any fraction $\rho \in [0, \rho_0]$ of the processes deliver every message — $\psi(\rho)$ is a *monotonically decreasing* (with respect to ρ) function.

Note however, that by the size of “a fraction ρ of n processes” we mean $\lceil \rho n \rceil$. Accordingly, $\psi(\rho)$ is not represented by a continuous function, but manifests steps.

2.5 Comparing Broadcast Algorithms

Consider a *reliability range* $\nabla = [\rho_1, \rho_2]$, $\rho_1 \leq \rho_2 \in [0, 1]$, that is, a range of values for the reliability degree ρ which is of interest in the context of a comparison.

In the sense of Δ -Reliable Broadcast, in the environment \mathcal{E} , an algorithm B_1 is *more reliable in* $\nabla = [\rho_1, \rho_2]$ than an algorithm B_2 iff

$$\begin{aligned} \forall \rho \in \nabla : \psi_{B_1}(\rho, \mathcal{E}) &\geq \psi_{B_2}(\rho, \mathcal{E}) \wedge \\ \exists \rho_0 \in \nabla : \psi_{B_1}(\rho_0, \mathcal{E}) &> \psi_{B_2}(\rho_0, \mathcal{E})^4 \end{aligned} \quad (2)$$

Similarly, in the environment \mathcal{E} , an algorithm B_1 is said to be *strictly more reliable in* $\nabla = [\rho_1, \rho_2]$ ($\rho_2 > 0$) than an algorithm B_2 iff

$$\forall \rho \in \nabla, \rho \neq 0 : \psi_{B_1}(\rho, \mathcal{E}) > \psi_{B_2}(\rho, \mathcal{E}) \quad (3)$$

We exclude here $\rho = 0$ since for any algorithm B : $\psi_B(0) = 1$.

Finally, in the environment \mathcal{E} , an algorithm B_1 is *more reliable* than an algorithm B_2 iff, in \mathcal{E} , B_1 is *more reliable* than B_2 in $\nabla = [0, 1]$. Analogously, in the environment \mathcal{E} , an algorithm B_1 is *strictly more reliable* than an algorithm B_2 iff, in \mathcal{E} , B_1 is *strictly more reliable* than B_2 in $\nabla = [0, 1]$.

2.6 Atomicity

The reliability distribution function can be used to define the probability that a certain number of processes deliver the message as a result of an algorithm run. More precisely, the probability that the fraction ρ of correct processes that delivered a broadcast message (in a given environment \mathcal{E}) is larger than ρ_1 but smaller than ρ_2 ($0 \leq \rho_1 < \rho_2 \leq 1$) can be defined as:

$$\begin{aligned} P(\rho_1 \leq \rho < \rho_2) &= \\ &= \psi(\rho_1, \mathcal{E}) - \psi(\rho_2, \mathcal{E}). \end{aligned} \quad (4)$$

Thus, the following *Atomicity predicate* (see more examples in [3]) defines a *failed broadcast* to be one that reaches more than a fraction σ of correct processes, but less than a fraction $1 - \sigma$ of correct processes in a system ($\sigma < 1/2$).

$$\begin{aligned} P(\sigma \leq \rho < 1 - \sigma) &= \\ &= \psi(\sigma, \mathcal{E}) - \psi(1 - \sigma, \mathcal{E}). \end{aligned} \quad (5)$$

Section 6 discusses some alternative *non-binary* specifications of broadcast algorithms.

2.7 Reliable Broadcast: From Perfect to Useless

A reliability distribution function ψ in the sense of (1) can be found for any algorithm. We demonstrate this through the following extreme cases.

⁴This second condition is necessary to avoid that two equally performing algorithms are “each more reliable than the other”.

Dreamcast. One can easily see that an algorithm implementing traditional Reliable Broadcast [9] in a given environment \mathcal{E} is Δ -Reliable with $\Delta = (1, 1)$. Since ψ_{RB} is a monotonically decreasing function, this sample univocally defines ψ_{RB} : $\forall \rho \in [0, 1] \psi_{RB}(\rho, \mathcal{E}) = 1$. One may call such an algorithm *perfectly* reliable. As we mentioned earlier in the introduction, its practical implementation in a network with unreliable processes and channels is expensive and not scalable.

Spellcast. A bogus algorithm which does nothing (*useless* broadcast) conforms to the specification of Δ -Reliable Broadcast in such a way that $\forall \mathcal{E} \in \mathbb{E}$ with at least one correct process and $\forall \rho \in]0, 1]$: $\psi_{UB}(\rho, \mathcal{E}) = 0$ (as stated previously, $\psi_{UB}(0, \mathcal{E}) = 1$).

Thus, the reliability level of any broadcast algorithm can be found somewhere between these two extreme cases. The following two sections illustrate this through two well-known and more meaningful examples, *Bimodal Multicast* and *IP Multicast*, respectively.

3 Bimodal Multicast

This section focuses on the *Bimodal Multicast* algorithm [3]. While providing a lower reliability in terms of Δ -Reliability than a perfectly reliable protocol, it is in most cases more scalable and efficient. We first recall the algorithm, and then discuss its Δ -Reliability.

3.1 Protocol Overview

The algorithm uses the idea of gossip-based protocols that dates back to the original USENET news protocol developed in early 1980's (*Network News Transport Protocol* — NNTP). In this protocol, a communication graph is superimposed on a set of processes, and neighbors gossip to diffuse news postings in a reliable manner over the links. If process p_i receives a news posting and then establishes communication with process p_j , p_i would offer p_j a copy of that news message, and p_j solicits the copy if it has not already seen the message.

Bimodal Multicast is composed of two subprotocols structured roughly as in the Internet MUSE protocol [11]. The first is an unreliable, hierarchical multicast (IP Multicast [5] can be used where available) that makes best-effort attempt to efficiently deliver each message to its destination. The second is a two-phase *anti-entropy* [6] protocol that operates in a series of asynchronous rounds. During each round, the first phase detects message losses; the second phase corrects such losses and executes only if needed.

In the present work, we are concerned only with *the first phase of the anti-entropy protocol*, namely the gossip-based knowledge propagation. For the analysis below, we use a simplified version of Bimodal Multicast, which differs from the original protocol in ways that simplify the discussion without changing the analytical results. The abstract version of the algorithm [3] is presented in Figure 1, where the parameter β is the so-called *fanout*, such that $n\beta$ is the size of the fraction of the system which is chosen as a destination set for the current gossip, and the parameter T is the number of *receive(m)* events in the longest causal chain for the message m . That is, a message m is consequently forwarded at most T times.

3.2 Model

Gossip protocols such as Bimodal Multicast can be analyzed with a stochastic approach as used in the epidemiological theory [1, 3].

```
{* Auxiliary function. *}
deliver_and_gossip(m, round)
  {* Do nothing if already received.*}
  if received_already then return

  {* Mark the message as received and deliver it.*}
  received_already:=true
  pbDeliver(m)

  {* if last round, don't gossip.*}
  if round=T then return

  let S be a randomly chosen subset of the system,
    such that  $|S| = n\beta$ 
  for each p in S send to p gossip(m,round+1)

{* Initial settings. *}
received_already:=false
initialize(T)

{* Initiate a Bimodal Multicast. *}
On a pbcast(m):
  deliver_and_gossip(m,0)

{* Handle message receipt. *}
On receive gossip(m,round)
  deliver_and_gossip(m,round)
```

Figure 1: Abstract version of Bimodal Multicast [3]

Breakdown in synchronous rounds. The stochastic analysis below is based on the assumption that the execution of a broadcast algorithm can be broken up into a sequence of synchronous *rounds*, such that, during each round t , only processes which have gossips with round number t are gossiping, and every round happens strictly after all the transmission of the previous round are completed. Of course, in a real execution, each process autonomously proceeds in its own rounds which are completely unsynchronized with respect to other processes. Indeed, a recently infected process starts gossiping immediately without waiting for the previous gossip to complete. But as outlined in [3], the actual execution performs better, and the obtained lower bound does give useful results.

Assumptions and definitions. For the following analysis, we assume that failures are *stochastically independent*. In particular, the probability of a message loss does not exceed a predefined $\varepsilon > 0$, and the probability of a process crash during the protocol execution is bounded by $\tau > 0$. For simplicity, we assume that all incorrect processes are initially crashed. This implies that dependent link failures like a network partition are outside of our failure model. Furthermore, at any moment and for any message m :

An **infected process** is one that already received m .

An **infectious process** is an infected one which is gossiping m in the current round.

A **susceptible process** is one that is not infected yet by m .

We consider a system of n participants using Bimodal Multicast [3]. Following [3], we describe the state of the system in round t using the following random variables:

X_t - the number of susceptible processes.

Y_t - the number of infectious processes.

We assume that, initially, only one process is infected (the process which broadcasts). To summarize the constraints on the state of the system:

$$\begin{aligned} X_0 &= n - 1, Y_0 = 1 \\ X_{t+1} + Y_{t+1} &= X_t \\ X_T + \sum_{t=0}^T Y_t &= n \end{aligned} \tag{6}$$

Note that at any round t , the number of infected processes is $n - X_t$.

3.3 Analysis

Let $F = f$ be the number of incorrect processes in a given run. We define $\beta(1 - \varepsilon)(n - f)/n$ as the probability that a given gossip message m sent by an infectious process is successfully received by a given process p_i , that is: (a) the gossiping (infectious) process chooses p_i as destination, (b) message m is not lost in transmission, and (c), process p_i is correct. Respectively, $q_f = 1 - \beta(1 - \varepsilon)(n - f)/n$ is the probability that a certain process did *not* receive a given gossip message from a particular infectious process. If j processes are gossiping in a given round, susceptible process p_i is *not infected* in this round with probability q_f^j .

The corresponding stochastic process can be expressed in the form of a *homogeneous Markov chain* with a transition matrix defined by:

$$\begin{aligned}
p_{ijklf} &= \\
&= P(X_{t+1} = k, Y_{t+1} = l | X_t = i, Y_t = j, F = f) \\
&= \begin{cases} \binom{i}{l} (1 - q_f^j)^l q_f^{jk} & k + l = i \\ 0 & k + l \neq i \end{cases} \quad (7)
\end{aligned}$$

The distribution of X_{t+1} and Y_{t+1} can be defined as follows:

$$\begin{aligned}
&P(X_{t+1} = k, Y_{t+1} = l | F = f) \\
&= \sum_i \sum_j P(X_t = i, Y_t = j | F = f) p_{ijklf} \quad (8)
\end{aligned}$$

Using (6),(7) and (8), we can build a distribution of X_T and Y_T . We are interested in the probability that, for some $\rho \in [0, 1]$, not less than a fraction ρ of correct processes are infected up to round T :

$$\begin{aligned}
\Phi_{BM}(\rho, \mathcal{E}_{BM}) &= \\
&= \sum_f P(F = f) P(X_T \leq n - \lceil \rho(n - f) \rceil | F = f) \\
&= \sum_f \binom{n}{f} (1 - \tau)^{n-f} \tau^f \\
&\quad \sum_{i \leq n - \lceil \rho(n - f) \rceil} \sum_j P(X_T = i, Y_T = j | F = f), \quad (9)
\end{aligned}$$

where $\mathcal{E}_{BM} = (\varepsilon, \tau, n, \beta, T)$ is the set of system and algorithm parameters defining the current environment.

3.4 Δ -Reliability of Bimodal Multicast

Based on this, we formally characterize the Δ -Reliability of Bimodal Multicast [3].

Proposition 1 *For any environment $\mathcal{E}_{BM} = (\varepsilon, \tau, n, \beta, T)$ and any $\rho \in [0, 1]$ Bimodal Multicast [3] is Δ -Reliable with $\Delta = (\Phi_{BM}(\rho, \mathcal{E}_{BM}), \rho)$.*

Proof: *Validity* and *Integrity* follow directly from the algorithm description (Figure 1) and the absence of Byzantine failures: the sender of a broadcast message delivers the message immediately and a process that receives the broadcast message delivers it only once. Thus, *Validity* and *Integrity* are always satisfied.

The proof of *Δ -Agreement* follows from the analysis above. Since $\Phi_{BM}(\rho, \mathcal{E}_{BM})$ gives the probability of successfully infecting at least a fraction ρ of correct processes, given that initially one process is infected, Δ -Reliability with $\Delta = (\Phi_{BM}(\rho, \mathcal{E}_{BM}), \rho)$ is guaranteed. \square

3.5 Approximation of $\Phi_{BM}(\rho, \mathcal{E}_{BM})$

Here we present a way to approximate the function $\Phi_{BM}(\rho, \mathcal{E}_{BM})$. We describe the state of the system using the stochastic process $X(t)$ - the proportion of susceptible processes in round t .

Neglecting the fluctuation of $X(t)$ around its conditional expectation $x(t)$, we have the following *deterministic* approximation of the stochastic process:

$$x(t+1) = x(t)q^{n(x(t-1)-x(t))}, \quad (10)$$

with the following initial conditions:

$$x(0) = \frac{n-1}{n}, \quad (11)$$

$$x(1) = x(0)q.$$

Here $q = 1 - \beta(1 - \varepsilon)(1 - \tau)$. The approach is robust for large n , when the deviation of $X(t)$ is comparatively small [15]. From (10) we can derive the following relationship:

$$x(t+1)q^{nx(t)} = x(t)q^{nx(t-1)} = \frac{n-1}{n}q^n. \quad (12)$$

Denote $q = 1 - \mu/n$, where $\mu = \beta n(1 - \varepsilon)(1 - \tau)$. For large n , $(1 - \mu/n)^n \approx e^{-\mu}$. Thus, we have the following recursive relationship:

$$x(0) = \frac{n-1}{n}, \quad (13)$$

$$x(t+1) = e^{\mu(x(t)-1)},$$

which is approximated by the following Cauchy's problem:

$$x(0) = \frac{n-1}{n}, \quad (14)$$

$$\dot{x} = e^{\mu(x-1)} - x, \quad (15)$$

Note that, according to (14, 15), $x(t)$ is a monotonically decreasing function.

The question is: what is the lower-bound asymptote of the susceptible fraction of the system $x(t)$ and how does it depend on n ?

One can easily see that equation (15) does not depend on t and n , that is if $x = \varphi(t)$ is a solution of (15), then, for any t_0 , $x = \varphi(t - t_0)$ is also a solution of (15). The system size n only impacts the initial condition (14). Thus the lower-bound asymptote of $x(t)$ does not depend on n : (14) defines the *time* necessary to approach it.

The lower-bound asymptote x_l can be roughly estimated for $x \ll 1$ through the following consideration:

$$e^{\mu(x-1)} \approx e^{-\mu} + e^{-\mu}\mu x \Rightarrow \quad (16)$$

$$x_l \approx \frac{e^{-\mu}}{1 - \mu e^{-\mu}}.$$

Assume that the maximal number of rounds T is sufficient to approach *closely* the upper bound of the infected fraction $1 - x_l$ (that is $T = O(\log n)$ [1, 15]). Hence, we can approximate the probability

that a given process is infected as a result of the run as $1 - x_l$. The reliability distribution function is approximated as:

$$\begin{aligned} \Phi_{BM}(\rho, \mathcal{E}_{BM}) \\ \approx \sum_{\lceil \rho n \rceil \leq i \leq n} \binom{n}{i} (1 - x_l)^i x_l^{n-i} \end{aligned} \quad (17)$$

Note that we are approximating $\Phi_{BM}(\rho, \mathcal{E}_{BM})$ by the probability that at least fraction ρ of *all* processes is infected. This is valid when $\tau \ll 1$. In general, (17) defines a *lower bound* on $\Phi_{BM}(\rho, \mathcal{E}_{BM})$: the probability of infecting at least fraction ρ of *correct* processes can be only larger.

3.6 Expected reliability degree of Bimodal Multicast

The presented analysis allows to state the following result:

Proposition 2 *For any environment $\mathcal{E}_{BM} = (\varepsilon, \tau, n, \beta)$ in which $\tau \ll 1$, the expected reliability degree $E_{BM}[\rho](n)$ as a function of system size n is such that:*

$$E_{BM}[\rho](n) \xrightarrow{n \rightarrow +\infty} 1 - \frac{e^{-\mu}}{1 - \mu e^{-\mu}}, \quad (18)$$

where $\mu = \beta n(1 - \varepsilon)(1 - \tau)$.

The proof follows directly from the approximations presented above.

Note that if we choose fanout $\beta = \frac{k}{n}$ (such that the number of partners a process gossip to each round, $k = \beta n$ is constant), then the right-hand side of (18) is constant with respect to the system size. In other words, the expected reliability degree of Bimodal Multicast is stable with respect to the scale of the system. This a very valuable property for self-organizing systems, since for some *fixed* set of parameters of the algorithm, its reliability degree does not degrade with the increase of the system size. As we will see in the following section, IP Multicast is not scalable in this sense: its reliability degree $E_{IPM}[\rho](n)$ is decreasing exponentially with the increase of n .

4 IP Multicast

In this section, we illustrate the notion of Δ -Reliable Broadcast through a second, in the traditional sense [9] inherently unreliable algorithm, namely IP Multicast [5].

4.1 Protocol Overview

IP Multicast is a so-called *network-level* broadcast algorithm. As its name reveals, it is directly based on IP, and is used to broadcast datagrams. The transmission of such datagrams is not reliable, and basic IP Multicast does not consider message loss detection and reparation, making it inherently unreliable. In the context of IP Multicast, many different protocols have been described and deployed, for instance in the *MBone*, the Internet's IP Multicast backbone.

4.2 Model

While certain protocols are targeted at dense distribution of processes and thus rely on flooding techniques, we focus here on a sparse distribution of processes. We presuppose a spanning tree, as for instance the ones that are encountered with the *Protocol-Independent Multicast — Sparse Mode* (PIM-SM) [7] protocol.

Spanning tree. In conformance with what is usually supposed for the analysis of such protocols (e.g., [14]), we suppose a k -ary spanning tree of depth d . In other terms, we consider a regular spanning tree with a single broadcasting process⁵ (the broadcaster of a given message) located at the root, k^d receiving processes constituting the leaves of the tree, and every non-leaf node of the tree representing a router with k outgoing links. The system size is thus given by $n = k^d$,⁶ but we will consider n and k as parameters of the environment, and, since we are interested in large systems, we use $d = \log_k n$. Note that a spanning tree obtained in a real use case can always be captured by a possibly bigger spanning tree with a number of leaves of order n conforming to the above description.

Failures. In conformance with the analysis of Bimodal Multicast presented in the previous section, we consider as τ the probability that a given process fails, and the probability of a message loss in a link between two nodes in the spanning tree as ε_l . In addition, we define as γ the probability of a router failures. We assume that all incorrect entities are initially crashed and the link failures are stochastically independent.

4.3 Analysis

Similarly to the analysis presented in the previous section, we propose a breakdown in successive rounds. These rounds however correspond to the levels in the spanning tree, that is, at round 1, the router of a broadcasting process forwards a given message m to the k routers representing its child nodes ($Y_0 = 1$). Due to failures, only $Y_1 \leq k$ will receive m . In any round $1 < t < d$, the Y_{t-1} “infectious” routers of level $t - 1$ forward m to their kY_{t-1} child nodes (maximum of k^t). The probability p of a successful reception of m by an entity at level $t < d$ is therefrom given by $p = (1 - \gamma)(1 - \varepsilon_l)$. At round $t = d$, the routers composing level $d - 1$ finally send m to the processes constituting the leaves of the tree. We assume that $F = f$ processes are correct in a given run.

The probability of having a given number Y_t of “infected” entities at a given level $t > 0$ can be computed recursively based on the probabilities of any number of infected entities at level $t - 1$. Finally, the probability of obtaining a given number of infected processes at the leaves of the spanning tree enables the computation of the fraction ρ of the correct processes in Π which have received m . For that end, we require the probability of having j infected entities at level $0 < t < d$ based on the number i of infected entities at the previous level:

$$p_{ij} = \binom{ik}{j} p^j (1 - p)^{(ik-j)} \quad (19)$$

⁵For simplicity, we assume that the broadcasting process is correct.

⁶To be absolutely precise, we would have to consider $n = k^d + 1$ processes, since the broadcasting process is itself receiving. At an increased system size n , this does not significantly impact the result.

Thus, the probability of having j infected entities at round $0 < t < d$ is given recursively by:

$$P(Y_t = j) = \sum_{0 \leq i \leq k^{t-1}} P(Y_{t-1} = i) p_{ij} \quad (20)$$

Let $F = f$ be the number of incorrect processes in a given run. The probability p_d of successful transmission of message m from an infected router at level $d - 1$ to a process at level d is given by $p_f = (1 - \varepsilon_l)(n - f)/n$ and the probability of having j infected processes at level $t = d$ based on the number i of infected entities at the previous level:

$$p_{ijf} = \binom{ik}{j} p_f^j (1 - p_f)^{(ik-j)} \quad (21)$$

Thus, the probability of having j infected processes at round d is given recursively by:

$$P(Y_d = j | F = f) = \sum_{0 \leq i \leq k^{d-1}} P(Y_{d-1} = i) p_{ijf} \quad (22)$$

As a direct consequence, the probability of having infected at least fraction ρ of correct processes in a k -ary spanning tree of depth d is given by:

$$\begin{aligned} & \Phi_{IPM}(\rho, \mathcal{E}_{IPM}) \\ &= \sum_f P(F = f) P(Y_d \geq \lceil \rho(n - f) \rceil | F = f) \\ &= \sum_f \binom{n}{f} (1 - \tau)^{n-f} \tau^f \\ & \quad \sum_{\lceil \rho(n-f) \rceil \leq i \leq n-f} P(Y_d = i | F = f), \end{aligned} \quad (23)$$

where M is the number of correct processes in a given run and \mathcal{E}_{IPM} is the environment defined as the set of parameters $\mathcal{E}_{IPM} = (\varepsilon_l, \tau, \gamma, n, k)$.

4.4 Δ -Reliability of IP Multicast

Based on (23), we are now able to formally characterize the Δ -Reliability of IP Multicast.

Proposition 3 *For any environment $\mathcal{E}_{IPM} = (\varepsilon_l, \tau, \gamma, n, k)$ and $\forall \rho \in [0, 1]$ IP Multicast is Δ -Reliable with $\Delta = (\Phi_{IPM}(\rho, \mathcal{E}_{IPM}), \rho)$.*

Proof: The proof of *Integrity* follows from the semantics of IP and the absence of Byzantine failures, and *Validity* is assured with prevalent operating systems. Thus, *Validity* and *Integrity* are always satisfied in this model.

The proof of Δ -*Agreement* follows from the analysis above. $\Phi_{IPM}(\rho, \mathcal{E}_{IPM})$ is equal to the probability of successfully infecting at least a fraction ρ of processes. Thus Δ -Reliability with $\Delta = (\Phi_{IPM}(\rho, \mathcal{E}_{IPM}), \rho)$ is guaranteed. \square

4.5 Expected reliability degree of IP Multicast

Although the reliability distribution function $\Phi_{IPM}(\rho, \mathcal{E}_{IPM})$ is computed recursively, the *expected value* for the fraction ρ of correct processes which receive m , $E[\rho]$, is given by:

$$E[\rho] = (1 - \varepsilon_l)p^{\log_k n - 1}. \quad (24)$$

Furthermore, the probability that *all* n processes are correct and receive a given broadcast message m , $P(Y_d = n) = \psi(1)$, can be easily expressed in this model through:

$$P(Y_d = n) = p^{\frac{n-k}{k-1}}(1 - \varepsilon_l)^n(1 - \tau)^n \quad (25)$$

5 Comparing Bimodal Multicast and IP Multicast

This section illustrates the use of Δ -Reliable Broadcast in comparing broadcast algorithms. Based on the analytical results presented in Section 3 and Section 4, we present here estimations of the reliability distribution functions (and also the expected values of the reliability degrees for both algorithms), which enable the comparison of Bimodal Multicast and IP Multicast in the context of Δ -Reliability. As we will show, our analysis confirms the intuition that, in many cases, Bimodal Multicast is “more reliable” than IP Multicast and that Bimodal Multicast, unlike IP Multicast, manifests no considerable reliability degradation as the system grows in size.

Environment. We assume that the system topology allows each process to maintain dynamically a spanning tree with d layers and and arity k , whose leaves represent the other processes and non-leaf members represent the routers: $n = k^d$. The probability of a message loss on the way from one process to another used in the analysis of Bimodal Multicast (Section 3) is thus bounded by $\varepsilon = 1 - (1 - \varepsilon_l)^d(1 - \gamma)^{d-1}$, where ε_l is the probability of a message loss in a link between two corresponding nodes in the spanning tree and γ is the probability of a router failure. We consider Bimodal Multicast and IP Multicast in the following *compound* environment $\mathcal{E}_R = (\mathcal{E}_{BM} \cup \mathcal{E}_{IPM} = (\varepsilon_l = 0.05, \tau = 0.01, \gamma = 0.001, n = 256, k = 4, \beta = 0.02, T = 6)$ (see the definition of a compound environment in section 2). Non-common parameters of the environments \mathcal{E}_{BM} and \mathcal{E}_{IPM} , β and T , are chosen in order to approach closely the upper-bound reliability degree $1 - x_l$ defined by (16).

Reliability distribution functions. Figure 2 presents the reliability distribution functions $\Phi_{BM}(\rho, \mathcal{E}_R)$ of Bimodal Multicast and $\Phi_{IPM}(\rho, \mathcal{E}_R)$ of IP Multicast in the “realistic” compound environment \mathcal{E}_R . Indeed, relevant to the intuition, $\forall \rho \in [0.55, 1] : \Phi_{BM}(\rho, \mathcal{E}_R) > \Phi_{IPM}(\rho, \mathcal{E}_R)$, that is Bimodal Multicast is *strictly more reliable* in $\nabla = [0.55, 1]$ in the environment \mathcal{E}_R . However, in a “better” environment \mathcal{E}_{R+} (with much smaller values for ε_l , τ and γ), IP Multicast may guarantee the same level of reliability as Bimodal Multicast. At the extremum, in a perfect environment \mathcal{E}_P with $\varepsilon_l = \tau = \gamma = 0$, where we have neither message losses nor node failures, $\psi_{IPM}(\rho, \mathcal{E}_P) = 1, \forall \rho \in [0, 1]$. Bimodal Multicast on the other hand, due to its randomized nature, even in the perfect system, admits the case when all the gossips of any given round are sent to already infected members and some part of the system will never get the broadcast message. Thus, $\forall \rho \in]0, 1[$, $\psi_{BM}(\rho, \mathcal{E}_P)$ is strictly less than 1 (but can be made arbitrarily close to 1). This conveys the strong impact of the choice of the environment, in which two algorithms are to be compared, on the respective reliability distributions, and thus on the result of the comparison.

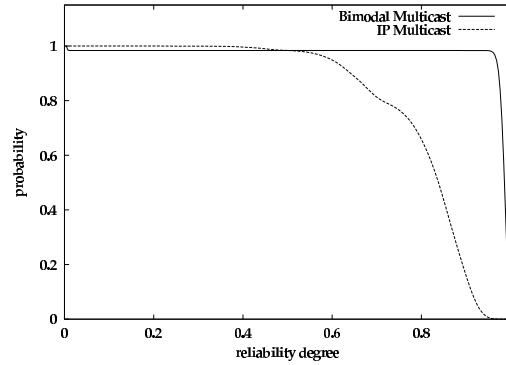


Figure 2: Reliability distribution functions $\Phi_{BM}(\rho, \mathcal{E}_R)$ and $\Phi_{IPM}(\rho, \mathcal{E}_R)$.

Scalability measure. Our non-binary approach to quantify the reliability of a broadcast algorithm can be used to state out a new measure of the *scalability* of the algorithm. Basically, we can consider an algorithm to be scalable if its expected reliability degree as function of the system size $E[\rho](n)$ is constant (or decreasing slowly).

Such an approach is innovative in the sense that it uses a new, more flexible (compared to [9]) notion of reliability and covers the cases when broadcast performance degrades with the increase of the system size.

Remark. The presented scalability criterion obviously reflects just one dimension of scalability. It is worth noting that IP Multicast is “more scalable” in terms of message complexity and time: to obtain the same reliability degree it requests a smaller number of messages and consumes less time. Note that traditional Broadcast[9] is scalable in this context: its reliability degree is $E_{RB}[\rho](n) = 1$, although it is not scalable in terms of message complexity and time.

Expected reliability degrees. Figure 3 presents the expected values of the reliability degrees for Bimodal Multicast and IP Multicast ($E_{BM}[\rho]$, resp. $E_{IPM}[\rho]$) given a system size n . As expected, the system size does not have a noticeable impact on the reliability of Bimodal Multicast (see Proposition 2) while, for IP Multicast, $E_{IPM}[\rho]$ is significantly decreasing. This confirms the advantage of Bimodal Multicast over IP Multicast *in terms of Δ -Reliability* (in the “realistic” environment \mathcal{E}_R).

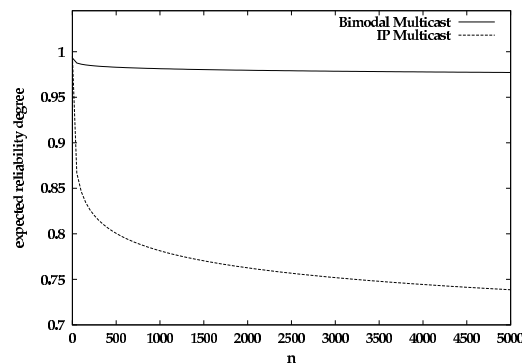


Figure 3: Expected reliability degrees $E_{BM}[\rho]$ and $E_{IPM}[\rho]$ for a given system size n (otherwise the environment is identical to \mathcal{E}_R).

6 Related Issues

In this section, we give more details on the nature of our reliability distribution function $\psi(\rho)$, and we discuss some alternative measures that we have been exploring.

Separating properties. Δ -Reliability defines the same probability for Integrity, Validity and Δ -Agreement properties to be satisfied. An alternative specification, considering different probabilities ψ_I , ψ_V and ψ_A of satisfying each property respectively, would lead to an underspecified system: many possible and not quantified intersections between the domains in which each respective property is verified would be introduced. In a practical context furthermore, it is sufficient to separate “good” and “bad” runs, without any further distinction.

Correct vs. all. For particular kinds of applications, such as quorum replication [2], a broadcast message must reach *clear majority* of *all* processes in the system (including failed processes) in a “good” run. Thus, an alternative definition of Δ -Agreement in Δ -Reliable Broadcast might require fraction ρ of *all* (not only correct) processes to deliver a broadcast message m , once one correct process delivers m . The specification does not directly apply to the notion of Reliable Broadcast [9]: Δ -Reliable Broadcast with $\Delta = (1, 1)$ guarantees now that *all* processes are *correct* and *infected*, which does not make much sense in a failure-prone model.

Cumulative distribution function. From a probabilistic point of view, our reliability distribution function $\psi(\rho)$ expresses a similar, but not equivalent measure than a *cumulative distribution function*. In fact, for any given random variable X , the cumulative distribution function of X is given by

$$F(x) = P(X \leq x), \forall x \in] - \infty, \infty[\quad (26)$$

In contrast, ψ expresses for a given random variable ρ :

$$\psi(\rho_0, \mathcal{E}) = P(\rho \geq \rho_0), \forall \rho_0 \leq 1, \mathcal{E} \in \mathbb{E} \quad (27)$$

Together with the assumption that $\forall \rho \notin [0, 1] \psi(\rho, \mathcal{E}) = 0$, we are able to express the relationship between $\psi(\rho)$ and the cumulative distribution function of a random variable ρ describing the fraction of processes which deliver a given message

$$F(\rho_0, \mathcal{E}) = 1 - \psi(\rho_0, \mathcal{E}) + P(\rho = \rho_0, \mathcal{E}), \forall \rho_0 \in] - \infty, \infty[\quad (28)$$

where the last term disappears when considering $\psi(\rho, \mathcal{E})$ as a continuous function with respect to ρ .

Lower bounds. As illustrated through our examples in Section 3 and Section 5, a lower bound on the probability of successful execution Φ_{B_1} for a given broadcast algorithm B_1 can help to estimate the reliability distribution function $\psi_{B_1}(\rho)$ for that algorithm, which can be useful when comparing B_1 with another algorithm B_2 , especially if $\psi_{B_2}(\rho)$ is known and is smaller than $\psi_{B_1}(\rho)$ in some ∇ . In practice, it is important to find a lower bound which reflects most truly the effective reliability distribution $\psi(\rho)$ of an algorithm. The lower bound presented in the case of Bimodal Multicast is reasonably close to the real probability distribution and provides useful information for comparisons with other broadcast algorithms, but for many algorithms, finding a precise reliability distribution function remains a difficult task.

Acknowledgements

We are very grateful to Ken Birman and Robert van Renesse for affording us an insight into the subtle approach of Bimodal Multicast. The fruitful discussions with them have strongly influenced this work.

References

- [1] N. Bailey. *The Mathematical Theory of Infectious Diseases and its Applications (second edition)*. Hafner Press, 1975.
- [2] K. Birman. *Building Secure and Reliable Network Applications*. Manning Publications Co., 1996.
- [3] K. Birman, M. Hayden, O. Ozkasap, Z. Xiao, M. Budiu, and Y. Minsky. Bimodal Multicast. *ACM Transactions on Computer Systems*, 17(2):41–88, May 1999.
- [4] D. R. Cheriton and D. Skeen. Understanding the limitations of causally and totally ordered communication. In *Proceedings of the 14th Symposium on Operating Systems Principles*, pages 44–57, Dec. 1993.
- [5] S. Deering. *Multicast Routing in a Datagram Internetwork*. PhD thesis, Stanford University, 1991.
- [6] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. Epidemic Algorithms for Replicated Database Maintenance. In *Proceedings of the 6th ACM Symposium on Principles of Distributed Computing (PODC '87)*, pages 1–12, Aug. 1987.
- [7] B. Fenner, M. Handley, H. Holbrook, and I. Kouvelas. Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification (Revised). *Internet Engineering Task Force (IETF)*, Nov. 2000.
- [8] S. Floyd, V. Jacobson, S. McCanne, C. G. Liu, and L. Zhang. A Reliable Multicast Framework for Light-Weight Sessions and Application Level Framing. *IEEE/ACM Transactions on Networking*, pages 784–803, Nov. 1996.
- [9] V. Hadzilacos and S. Toueg. A modular approach to fault-tolerant broadcast and related problems. Technical report, Cornell University, Computer Science, May 1994.
- [10] H. Holbrook, S. Singhal, and D. Cheriton. Log-Based Receiver-Reliable Multicast for Distributed Interactive Simulation. In *Proceedings of the 1995 ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM '95)*, pages 328–341, Aug. 1995.
- [11] K. Lidl, J. Osborne, and J. Malcolm. Drinking from the firehose: Multicast USENET news. In USENIX Association, editor, *Proceedings of the Winter 1994 USENIX Conference*, pages 33–45, Jan 1994.
- [12] M.-J. Lin, K. Marzullo, and S. Masini. Gossip versus Deterministically Constrained Flooding on Small Networks. In *Proceedings of the 14th International Conference on Distributed Computing (DISC 2000)*, pages 253–267, Oct. 2000.

- [13] S. Paul, K. Sabnani, J. Lin, and S. Bhattacharyya. Reliable Multicast Transport Protocol (RMTP). *IEEE Journal on Selected Areas in Communications*, 15(3):407–421, Apr. 1997.
- [14] G. Phillips, S. Shenker, and H. Tangmunarunkit. Scaling of multicast trees: Comments on the Chuang-Sirbu scaling law. In *Proceedings of the 1999 ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM'99)*, pages 41–52, Aug. 1999.
- [15] B. Pittel. On Spreading of a Rumor. *SIAM Journal of Applied Mathematics*, 47:213–223, 1987.
- [16] B. Whetten, T. Montgomery, and S. Kaplan. A High Performance Totally Ordered Multicast Protocol. In *Theory and Practice in Distributed Systems*, number 938 in LNCS, pages 33–54. Springer, 1995.
- [17] XTP Forum. Xpress transfer protocol specification. In *XTP Rev. 4.0*, pages 95–120, 1995.