Δ-Reliability*

Patrick Th. Eugster Rachid Guerraoui Petr Kouznetsov
Swiss Federal Institute of Technology
Lausanne, Switzerland, CH 1015

Abstract

This paper presents a new, non-binary measure of the reliability of broadcast algorithms, called Δ-Reliability. *This measure quantifies the reliability of practical broadcast algorithms that, on the one hand, were devised with some form of reliability in mind, but, on the other hand, are not considered reliable according to the "traditional" notion of broadcast reliability [8].*

Our specification of Δ*-Reliability suggests a further step towards bridging the gap between theory and practice in the reliability of broadcast algorithms. We illustrate the use of* Δ*-Reliability through two case studies, namely* Bimodal Multicast *and* IP Multicast.

Contact author: Petr Kouznetsov (petr.kouznetsov@epfl.ch)

Student paper: Patrick and Petr are full-time students

Category: Regular paper

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 [8]:

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

Integrity. For any message m, every correct process delivers m at most once, and only if m was previously broadcast by sender(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 [3], contradicting the stringent scalability properties claimed by today's peer-to-peer applications.

Best-effort broadcast. More pragmatic approaches to broadcast focus on performance in very large-scale settings, and sacrifice strong reliability guarantees (in the sense of [8]) to scalability. Examples include the Internet *Multicast Usenet* (MUSE) protocol [10], the *XPress Transfer Protocol* (XTP) [15] or a broad range of so-called *network-level* protocols building on *IP Multicast* [4].[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

*This work is partially supported by Agilent Laboratories, Lombard & Odier, and the Swiss National Science Foundation (project number 510-207).

[1] E.g., *Reliable Multicast Transport Protocol* (RMTP) [11], *Reliable Multicast Protocol* (RMP) [14], *Log-Based Receiver-Reliable Multicast* (LBRM) [9], *Scalable Reliable Multicast* (SRM) [7].

are simply not intended to satisfy the traditional properties of Reliable Broadcast [8].

Probabilistic broadcast. Birman et al [2] 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* [5] *Bimodal Multicast* algorithm [2]), 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 [2] the authors concentrate on giving a behavioral analysis of the Bimodal Multicast algorithm.

Reliability measure. The motivation of this work is the observation that, beyond the approach of [2], there is a lack for a precise but also practical measure to estimate the reliability of inexpensive and scalable best-effort algorithms. Intuitively, those are less reliable than algorithms that comply with the strong properties of [8] 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.

The aim of this work is to introduce a *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.

[2]The "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.

Contributions. This paper introduces a new *non-binary probabilistically* flavored specification of the reliability of broadcast algorithms called Δ-*Reliability*. Through this measure 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 [2], 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 another important class of algorithms often used in practice, namely the network-level protocols, we discuss IP Multicast [4].

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.

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 our 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 reliability: 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. Appendix A discusses more in detail our definition of Δ-Reliability, in particular with respect to alternatives we have considered. Appendix B recalls details of the Bimodal Multicast algorithm.

2 Δ-Reliability: 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 [8]) system Π of processes $\{p_1, .., p_n\}$. 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. To simplify presentation, we do not consider byzantine failures, and we assume that crashed processes do not recover.

However, 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 Σ, to refer to the set of relevant system properties and algorithm parameters.

2.2 Δ-Reliability

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

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

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

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

Properties Δ-*Validity* and Δ-*Integrity* here are the same as *Validity* and *Integrity* in traditional Reliable Broadcast [8], except that we only require them to be satisfied with a given probability. *Agreement*, as defined in [8], is transformed here into Δ-*Agreement* which is less restrictive in terms of the number of processes that need to deliver the message.[3]

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 [8].[4] 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.

[3] In Appendix A we discuss alternative approaches to defining the reliability of a broadcast algorithm in a probabilistic context.

[4] Note that Δ-Reliability is stronger than ensuring with a probability of ψ the traditional properties of Reliable Broadcast [8] for a fraction ρ of the system: in the former case, *Integrity* and *Validity* are ensured with a probability ψ for the entire system, not only for a fraction ρ.

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

For instance, to satisfy the properties of Δ-Reliability 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-Reliability does not aim at giving a binary interpretation for the reliability of a broadcast algorithm as in [8]. 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 $\Delta_1 ... \Delta_s$ 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 Δ_{B_1}=(0.9, 0.9) and Δ_{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} is only 0.7. To compare two algorithms in a more general manner, we define a *reliability distribution function* $\psi(\rho)$:

$$\psi : [0, 1] \mapsto [0, 1] \tag{1}$$

Note however, that by "a fraction ρ of the system of size n" we mean $\lfloor \rho n \rfloor$. Accordingly, $\psi(\rho)$ is not represented by a continuous function, but manifests steps.

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* function.

Since the reliability distribution function of an algorithm is strongly coupled with the considered environment Σ, we will also write $\psi(\rho,\Sigma)$, in particular when comparing broadcast algorithms. Indeed, a comparison of algorithms in different environments is not very meaningful.

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 Δ-Reliability sense, in the environment Σ, an algorithm B_1 is *more reliable in* $\nabla = [\rho_1, \rho_2]$ than an algorithm B_2 iff

$$\begin{array}{c} \forall \rho \in \nabla :\ \psi_{B_1}(\rho, \Sigma) \geq \psi_{B_2}(\rho, \Sigma) \wedge \\ \exists \rho_0 \in \nabla :\ \psi_{B_1}(\rho_0, \Sigma) > \psi_{B_2}(\rho_0, \Sigma)^5 \end{array} \tag{2}$$

Similarly, in the environment Σ, 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

$$\forall \rho \in \nabla,\ \rho \neq 0 :\ \psi_{B_1}(\rho, \Sigma) > \psi_{B_2}(\rho, \Sigma)^6 \tag{3}$$

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

Note that two algorithms B_1 and B_2 might have different sets of parameters in their respective environments Σ_{B_1} and Σ_{B_2}. The environment for the comparison can in a simplified sense

[5] This second condition is necessary to avoid that two equally performing algorithms are "each more reliable than the other".

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

be viewed as a *compound* environment; a union of the two environments ($\Sigma = \Sigma_{B_1} \cup \Sigma_{B_2}$). In Section 5 we will illustrate these comparison criteria.

2.6 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.

Dreamcast. One can easily see that an algorithm implementing traditional Reliable Broadcast [8] is Δ-Reliable with $\Delta = (1, 1)$. Since ψ_{RB} is a monotonically decreasing function, this sample univocally defines ψ_{RB}: $\forall \rho \in [0, 1]\ \psi_{RB}(\rho) = 1$. 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) also conforms to the specification of probabilistic reliability with $\forall \rho \in]0, 1]\ \psi_{UB}(\rho) = 0$ (and, as stated previously, $\psi_{UB}(0) = 1$).

In short, 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, namely Bimodal Multicast and IP Multicast respectively.

3 Bimodal Multicast

This section focuses on the *Bimodal Multicast* algorithm [2]. 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 [10]. The first is an unreliable, hierarchical multicast (IP Multicast [4] 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* [5] 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 [2], which differs from the original protocol in ways that simplify the discussion without changing the analytical results. The algorithm is presented in Appendix B, 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 epidemiological theory [1, 2].

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 (see Appendix B), 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 [2], 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 number of process crashes does not exceed $f < n$. That is, the probability of a process crash during the protocol execution is bounded by $\tau = f/n$. Furthermore, 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 [2]. Following [2], we describe the state of the system in round t using the following random variables:

s_t - the number of infectious processes.

r_t - the number of susceptible processes.

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

$$\begin{aligned} &s_0 = 1,\ r_0 = n-1 \\ &s_{t+1} + r_{t+1} = r_t \\ &r_T + \textstyle\sum_{t=0}^{T} s_t = n \end{aligned} \tag{4}$$

3.3 Analysis

We define $p = \beta(1-\varepsilon)(1-\tau)$ as the probability that a given gossip message m is successfully received by a given process p_i, that is: (a) a gossiping (infectious) process chooses p_i as destination, (b) message m is not lost in transmission, and (c), process p_i is not crashed. Respectively, $q = 1 - p$ is the probability that a certain process did *not* receive a given gossip message from a particular infectious process.

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

$$\begin{aligned} p_{ijkl} =& P(s_{t+1} = k, r_{t+1} = l | s_t = i, r_t = j) \\ =& \begin{cases} \binom{j}{k}\ (1-q^i)^k q^{il} & k+l = j \\ 0 & k+l \neq j \end{cases} \end{aligned} \tag{5}$$

The distribution of r_{t+1} and s_{t+1} can be defined as follows:

$$\begin{aligned} &P(s_{t+1} = k, r_{t+1} = l) \\ &= \sum_i \sum_j P(s_t = i, r_t = j) p_{ijkl} \end{aligned} \tag{6}$$

Using (4),(5) and (6), we can build a distribution of s_T and r_T. We are interested in the probability that, for some $\rho \in [0,1]$, not less than $\lfloor \rho n \rfloor$ processes are infected up to round T:

$$\begin{aligned} &\Phi_{BM}(\rho, \Sigma_{BM}) \\ &= P(r_T < \lceil (1-\rho)n \rceil) \\ &= \sum_{0 \le i \le \lfloor \rho n \rfloor} \sum_{j < \lceil (1-\rho)n \rceil} P(s_T = i, r_T = j) \end{aligned} \tag{7}$$

where $\Sigma_{BM} = (\varepsilon, \tau, n, \beta, T)$ 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 [2].

Proposition 1 *For any environment* $\Sigma_{BM} = (\varepsilon, \tau, n, \beta, T)$ *and any* $\rho \in [0,1]$ *the reliability distribution function* $\psi_{BM}(\rho, \Sigma_{BM})$ *of Bimodal Multicast [2] satisfies the condition:*

$$\psi_{BM}(\rho, \Sigma_{BM}) \geq \Phi_{BM}(\rho, \Sigma_{BM}) \tag{8}$$

Proof: To prove the result it is sufficient to show that, for any $\rho \in [0,1]$ and any Σ_{BM}, Bimodal Multicast is Δ-Reliable with $\Delta = (\psi_{BM}(\rho, \Sigma_{BM}), \rho)$, such that $\psi_{BM}(\rho, \Sigma_{BM})$ satisfies (8).

The proof of Δ-*Validity* and Δ-*Integrity* follows directly from the algorithm description and the absence of byzantine failures (see Appendix B): the sender of a broadcast message delivers the message immediately and a process that receives the broadcast message delivers it only once. Thus, Δ-*Validity* and Δ-*Integrity* are satisfied with probability $1 \geq \Phi_{BM}(\rho, \Sigma_{BM})$, for any $\rho \in [0,1]$.

The proof of Δ-*Agreement* follows from the analysis above. Since $\Phi_{BM}(\rho, \Sigma_{BM})$ gives a lower bound on the probability of successfully infecting at least a fraction ρ of processes, the effective probability given by a real execution of Bimodal Multicast in Σ_{BM} is higher, thus $\forall \rho \in [0,1]$

$$\psi_{BM}(\rho, \Sigma_{BM}) \geq \Phi_{BM}(\rho, \Sigma_{BM}) \qquad \square$$

Remark. Note that an approximation consists in computing the *expected value* for the infected fraction of the system. Let $E[r_t]$ be the expected number of susceptible processes at time t. We can roughly estimate $E[r_t]$ using the following recursive relationship:

$$\begin{aligned} E[r_0] &= n - 1, \\ E[r_1] &= (n-1)q, \\ E[r_{t+1}] &= q^{(E[r_{t-1}] - E[r_t])} E[r_t] \end{aligned} \tag{9}$$

This gives us a simple way to estimate the fraction of infected processes after T rounds $E_{BM}[\rho] \approx (n - E[r_T])/n$, which is relevant for a large-scale system (the variance is comparatively small for a large n [13]).

4 IP Multicast

In this section, we illustrate Δ-Reliability through a second, in the traditional sense [8] inherently unreliable algorithm, namely IP Multicast [4].

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) [6] protocol.

Spanning tree. In conformance with what is usually supposed for the analysis of such protocols (e.g., [12]), we suppose a k-ary spanning tree of depth d. In other terms, we consider a regular spanning tree with a single broadcasting process (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 $n = k^d$,[7] but

[7] To 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.

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 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 as ε. In addition, we define as γ the probability of a router failure.

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 ($s_0 = 1$). Due to message losses, only $s_1 \leq k$ will receive m. In any round $1 < t < d$, the s_{t-1} "infectious" routers of level $t-1$ forward m to their ks_{t-1} child nodes (maximum of k^t). At round $T = d$, the routers composing level $d-1$ finally send m to the processes constituting the leaves of the tree. The probability p_t of a successful reception of m by an entity at level t is therefrom given by:

$$p_t = \begin{cases} (1-\gamma)(1-\varepsilon) & 1 \leq t < d \\ (1-\tau)(1-\varepsilon) & t = d \end{cases} \quad (10)$$

The probability of having a given number s_t of "infected" entities at a given level t 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 processes in Π which have received m. For that end, we require the probability of having j infected entities at level t based on the number i of infected entities at the previous level:

$$p_{ijt} = \binom{ik}{j} p_t^j (1-p_t)^{(ik-j)} \quad (11)$$

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

$$P(s_t = j) = \sum_{0 \leq i \leq k^{t-1}} P(s_{t-1} = i) p_{ijt} \quad (12)$$

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

$$\begin{aligned} \Phi_{IPM}(\rho, \Sigma_{IPM}) =& P(s_d \geq \lfloor \rho n \rfloor) \\ =& \sum_{\lfloor \rho n \rfloor \leq j \leq n} P(s_d = j), \end{aligned} \quad (13)$$

where the environment Σ_{IPM} is defined in this case as the set of parameters $\Sigma_{IPM} = (\varepsilon, \tau, \gamma, n, k)$.

4.4 Δ-Reliability of IP Multicast

Based on 13 we are now able to formally characterize the Δ-Reliability of IP Multicast.

Proposition 2 *For any environment* $\Sigma_{IPM} = (\varepsilon, \tau, \gamma, n, k)$ *and* $\forall \rho \in [0,1]$ *IP Multicast is* Δ*-Reliable with* $\Delta = (\Phi_{IPM}(\rho, \Sigma_{IPM}), \rho)$.

Proof: The proof of Δ-*Integrity* follows from the semantics of IP and the absence of byzantine failures, and Δ-*Validity* is assured with prevalent operating systems. Thus, Δ-*Validity* and Δ-*Integrity* are satisfied with probability 1.

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

Remark. Note that the *expected value* for the fraction ρ of processes which will receive m, $E_{IPM}[\rho]$, is given by:

$$E_{IPM}[\rho] = \prod_{1 \leq t \leq d} p_t = (1-\varepsilon)^d (1-\gamma)^{d-1} (1-\tau) \quad (14)$$

Furthermore, the probability that *all* n processes will receive a given broadcast message m, $P(s_d =$

$n) = \psi(1)$ can be easily expressed in this model through:

$$P(s_d = n) = \prod_{1 \leq t \leq d} p_t^{k^t} \quad (15)$$

5 Comparing Bimodal Multicast and IP Multicast

This section illustrates the use of Δ-Reliability in comparing broadcast algorithms. Based on the analytical results presented in Sections 3 and 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.

Reliability distribution functions. Figure 1 presents a lower bound on the probability of successful execution $\Phi_{BM}(\rho, \Sigma_R)$ of Bimodal Multicast and the reliability distribution function $\psi_{IPM}(\rho, \Sigma_R)$ of IP Multicast in some "realistic" compound environment $\Sigma_R = (\varepsilon = 0.01, \tau = 0.05, \gamma = 0.001, n = 256, k = 4, \beta = 0.02, T = 6)$. Indeed, relevant to the intuition, $\forall \rho \in [0.5, 1]$: $\psi_{BM}(\rho, \Sigma_R) \geq \Phi_{BM}(\rho, \Sigma_R) > \psi_{IPM}(\rho, \Sigma_R)$. One can conclude that Bimodal Multicast is *strictly more reliable in* $\nabla = [0.5, 1]$ in the environment Σ_R. However, in a "better" environment Σ_{R+} (with much smaller values for ε, τ and γ), IP Multicast may guarantee the same level of reliability as Bimodal Multicast. At the extremum, in a perfect environment Σ_P with $\varepsilon_P = \tau_P = \gamma_P = 0$, where we have neither message losses nor node failures, $\psi_{IPM}(\rho, \Sigma_P) = 1$, $\forall \rho \in [0, 1]$. 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. So $\psi_{BM}(\rho, \Sigma_P)$ 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,[8] and thus on the result of the comparison.

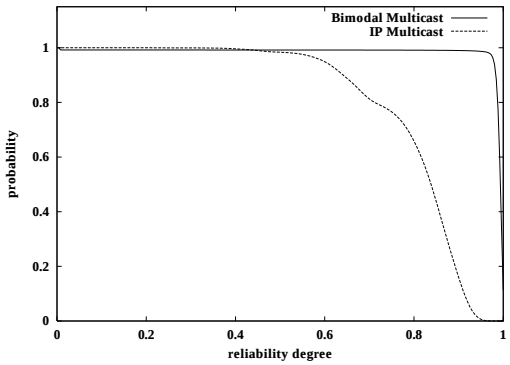

Figure 1: $\Phi_{BM}(\rho, \Sigma_R)$ and the reliability distribution function $\psi_{IPM}(\rho, \Sigma_R)$.

Expected reliability degrees. Figure 2 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 while, for IP Multicast, $E_{IPM}[\rho]$ is exponentially decreasing. This confirms the advantage of Bimodal Multicast over IP Multicast in terms of Δ-Reliability. This result might seem surprising, since Bimodal Multicast uses a first quick dissemination phase based on a tree-based algorithm, possibly IP Multicast itself. As we already mentioned in Section 3, this first phase does not impact the analysis of the propagation of knowledge.

Acknowledgements

We are very grateful to Ken Birman and Robert van Renesse for affording us an insight

[8] It is worth noting that IP Multicast is "more efficient": to obtain the same reliability degree it requests a smaller number of message sends and consumes less time.

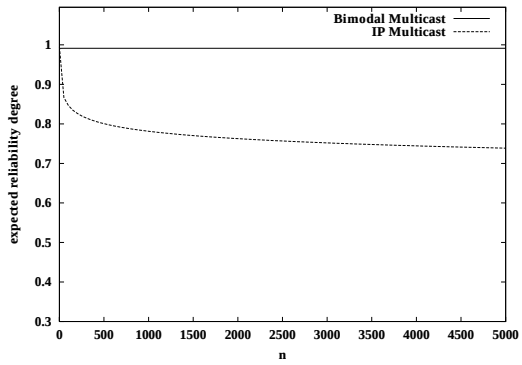

Figure 2: Expected reliability degrees $E_{BM}[\rho]$ and $E_{IPM}[\rho]$ for a given system size n (otherwise the environment is identical to Σ_R).

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*. Charles Griffen and Company, London, England, United Kingdom, 2nd edition, 1975.

[2] K. Birman, M. Hayden, O. Ozkasap, Z. Xiao, M. Budiu, and Y. Minsky. Bimodal multicast. In *ACM Transactions on Computer Systems*, volume 17(2), pages 41–88, 1999.

[3] D. Cheriton and D. Skeen. Understanding the limitations of causally and totally ordered communication. *ACM SIGOPS Oper. Syst. Rev.*, 28(1), January 1994.

[4] S. Deering. *Multicast Routing in a Datagram Internetwork*. PhD thesis, Stanford University, 1991.

[5] 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 Annual ACM Symposium on Principles of Distributed Computing (PODC'87)*, pages 1–12, Aug. 1987.

[6] 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.

[7] 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*, Nov. 1996.

[8] V. Hadzilacos and S. Toueg. A modular approach to fault-tolerant broadcast and related problems. Technical report, Cornell University, Computer Science, May 1994.

[9] 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.

[10] K. Lidl, J. Osborne, and S. Ghemawat. Drinking from the firehose: Multicast usenet news. In *USENIX Winter, Jan.*, pages 33–45, Berkley, CA, 1994. USENIX Assoc.

[11] 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.

[12] 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.

[13] B. Pittel. On spreading of a rumor. *SIAM Journal of Applied Mathematics*, 47:213–223, 1987.

[14] 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 Verlag, 1995.

[15] XTP Forum. Xpress transfer protocol specification. In *XTP Rev. 4.0*, pages 95–120, 1995.

Appendix A Reliability Distribution Function

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 all properties. An alternative specification considering different probabilities ψ_V, ψ_I 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.

Alternatively, one could also assume that Δ-*Validity* and Δ-*Integrity* are *always* fulfilled. In other terms, these first two properties would be satisfied with $\psi_{V,I}(1) = 1$, while Δ-*Agreement* would have a separate reliability distribution function $\psi_A(\rho)$. Presupposing that both Δ-*Integrity* and Δ-*Validity* are de facto always fulfilled however bears the danger of ruling out useful algorithms.

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

$$F(x) = P(X \leq x),\ \forall x \in\ ]-\infty, \infty[\qquad (16)$$

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

$$\psi(\rho_0) = P(\rho \geq \rho_0),\ \forall \rho_0 \leq 1 \qquad (17)$$

Together with the assumption that $\forall \rho > 1\ \psi(\rho) = 0$, 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

$$F(\rho_0) = 1 - \psi(\rho_0) + P(\rho = \rho_0), \forall \rho_0 \in\ ]-\infty, \infty[\qquad (18)$$

where the last term disappears when considering $\psi(\rho)$ as a continuous function.

Lower bounds. As illustrated through our examples in Sections 3 and 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.

Appendix B Abstract Version of Bimodal Multicast

This section presents a simplified version of the gossip-based phase of the Bimodal Multicast algorithm [2] used in this paper.

```
{* 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| = nβ
    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)
```