On the Resilience of Broadcasting Strategies in a Failure-Propagating Environment [*]

Meng-Jang Lin Aleta M. Ricciardi [†]
Department of Electrical and Computer Engineering
The University of Texas at Austin

Keith Marzullo
Department of Computer Science and Engineering
University of California at San Diego

Abstract

A process that is under the control of an intruder may masquerade as a legitimate process and, like an arbitrarily faulty process, may not follow the specification that other processes expect it to. Given this similarity, it seems plausible to mask the effects of such compromised processes in the same way that one masks arbitrary failures. One must, however, be able to bound the number of such compromised processes. We examine this problem in the context of multicast protocols. We cast the problem into terms of *availability*, which is the probability that no more than a certain number processes are infected. We consider the two questions "what is the availability of the system after having run for some period of time?" and "how long can a system run until the availability is unacceptably low?" We examine how the answers to these questions change as the number of processes grows, as the probability of a message being infective changes, and as different multicast strategies are used.

[*]Supported by DOD-ARPA under contract number F30602-96-1-0313.
[†]Also supported by the Texas Higher Education Coordinating Board, under contract ARP-003658-260.

1 Introduction

There are similarities between problems associated with processes that are under the control of an intruder and problems associated with processes that are arbitrarily faulty. A process that is under the control of an intruder may masquerade as a legitimate process and, like an arbitrarily faulty process, may not follow the specification that other processes expect it to.

Given this similarity, it seems plausible to mask the effects of such compromised processes in the same way that one masks arbitrary failures. Masking the effects of failures requires replication, and several protocols have in fact been designed to use replication to mask the effects of such processes [10, 11]. The bounds for masking arbitrary failures hold for these protocols, such as the need for either digital signatures or $(3f+1)$–fold replication in order to mask f compromised processes when reaching agreement [6].

However, an intruder may wreak more damage than what is captured by the arbitrary failure model. For example, an intruder may launch a malicious attack towards other processes on the system. It can create other seemingly benign processes by exploiting *transitive trust* that is assumed with the use of, for example, a *.rhosts* file, or it can co-opt otherwise correct processes through mechanisms like trap doors and race condition attacks. This implies that the techniques used to mask arbitrarily faulty processes may not be applicable, because too many processes may become compromised thereby violating the replication assumption. Accepting that the natural occurrence of Byzantine faults is small, and that one likely source for such faults is due to malicious attacks, then the self-propagating nature of these attacks should also be considered.

In this paper, we examine how different multicast strategies effect the efficacy of such attacks. We model these attacks as a simple form of infection. We assume that intruders can infect processes with a given probability by sending it a message. We consider only the messages in multicast strategies that carry the user's data, since these are the messages over which an application process has the most control.

We measure this effect in terms of *availability*, which is the probability that no more than a certain number processes are infected. We consider the two questions "what is the availability of the system after having run for some period of time?" and "how long can a system run until the availability is unacceptably low?" We examine how the answers to these questions change as the number of processes grows, as the probability of a message being infective changes, and as different multicast strategies are used.

The results in this paper affect the design of middleware for group-based communication systems [4]. An attack at this level of a system would be very hard to detect. Using the results of this paper, one could use a standard protocol for masking arbitrary failures. Given a desired availability, one can determine how long this protocol can run before the processes must be "cleaned", either by restarting processes from known clean images or by running a diagnostic program. This periodic cleaning ensures that the initial assumption of no more than f processes being arbitrarily faulty is maintained with an acceptable level of likelihood.

Our results are applicable to more than group-based communications. For example, in a mobile agent system, one can model a compromised landing pad as an infected process, and a mobile agent capable of compromising a landing pad as a message from an infected process. In this context, availability is the probability that no more than a given number of landing pads are compromised. If one uses mobile agents to collect information in a web-crawl-like manner, a compromised landing pad can corrupt the information that agents carry while passing through that pad. The results of this paper can be used to choose a dissemination mechanism for the agents and decide how often the landing pads must be "cleaned".

¿From a modeling point of view, our work resembles epidemiological modeling of algorithms [2]. Our model is essentially that of a simple epidemic with a zero latency period. If cleaning were also modeled, then one would have a general epidemic. There has been work on modeling the spread of computer viruses as a general epidemic (*e.g.*, [7, 8]). However, our work differs from existing epidemiological approaches in several aspects. First, we are interested in how the infection process affects the availability of the system rather than the expected number of infected processes. This is important because we wish to apply the results to multicast protocols. If one builds a protocol that can mask f arbitrarily faulty processes, then one would like to know how likely this assumption holds. Knowing the average number of infected processes does not address this question. Second, our transmission of infection is more restricted than general mixing of populations or a mixing restricted to undirected graphs. Third, we separate infection from death because death (for us, cleaning by restarting processes from trusted object files) is not a stochastic process. Instead, cleaning is periodically initiated based on the rate of infection and the desired availability, which ensures that any initial assumptions regarding the maximum number of (arbitrarily) faulty processes is maintained with an acceptable likelihood. At a pragmatic level, cleaning is expensive, and so should only be done as infrequently as possible rather than at random times chosen from some distribution.

The observations in our paper are consistent with some earlier work in epidemiological algorithms. For example, [8] observes that when connectivity is low, a higher transmission rate is required for an epidemic to become widespread. We observe a similar effect. Our work could also be used to extend prior research. For example, in [5], different epidemiological techniques are applied to propagate database updates. By understanding the effect of communication patterns on the speed of propagation, one might be able to design faster epidemic-based information diffusing mechanisms.

The rest of the paper proceeds as follows: in Section 2 we describe our system model and the multicast strategies that we consider. In Section 3 we give a definition for availability and describe how we simulate the execution of a system in order to determine how availability changes with different system parameters and different multicast strategies. Section 4 interprets the results of our simulations. We conclude and comment on future work in Section 5.

2 System Model

We consider an asynchronous distributed system composed of *processes* that communicate with one another by passing messages over a network of point-to-point channels. The communication is reliable, in that every message sent on a channel will be received by the destination process. In such a system, an infected process can infect another process only through messages. In particular, an infected process may send an *infective* message to an uninfected process, which will then become infected as soon as it receives this message (*i.e.* latency period = 0). We assume that there is no mechanism available to uninfected processes to determine when another process is infected nor when a particular message is infective.

Messages are not corrupted or otherwise tampered with once in transit. Because we are interested in comparing dissemination strategies according to how quickly they spread infection, we model only the ability of an infected process to infect another process, and do not consider spontaneous infection or infection via some external entity.

Each process has a probability of being infected in the initial state. Unless stated otherwise, we assume that this is the same for all processes, and we denote it with the simple variable p_{init}; the model can be easily tuned by customizing this value for specific platforms, security domains, and so forth. We further assume that p_{init} is small enough to ignore the probability that more than one process is initially infected; thus, p_{init} is not the percentage of n processes that are initially infected. Each message sent by an infected process is infective with probability p_{infect}. Again, we make a simplifying assumption that p_{infect} is constant and identical for all intended destinations. Allowing p_{infect} to differ for various destinations, yet remain constant, would model heterogeneity within the system. Allowing p_{infect} to further vary, as in a probability distribution for each intended destination, models more complex infective behavior.

Definition 1 *An* infective system σ *is parameterized by* $< n, p_{\text{init}}, p_{\text{infect}} >$, *where* n *is the number of processes,* p_{init} *is the probability that a process is infected in its initial state, and* p_{infect} *is the probability that a message sent by an infected process is infective.*

Since infection spreads through communication, the rate of infection depends on the pattern of communication. The pattern of communication that a multicast protocol uses is chosen based on several factors, including message efficiency, time efficiency, and the underlying network topology. We consider three patterns of communication, each of which also maps to a common network topology:

1. *Ring*: The processes are totally ordered, $\{0, 1, \cdots, n-1\}$. Process i starts a multicast by sending the message to its successor, process $(i+1) \bmod n$. The multicast terminates when i receives its own message from its predecessor, process $(i-1) \bmod n$; since only one point-to-point message may be in transit at any time, n sequential hops are needed to complete one multicast with the Ring strategy. Protocols that fit this structure include the Totem group-based protocols [10] and any lower level protocols designed to run on ring networks.

2. *Coordinator-cohort*: A process starts a multicast by sending its message to a special process called the *coordinator*. The coordinator forwards the message to all of the other processes; it takes two sequential hops to complete one multicast with the Coordinator-cohort strategy, the length of the second determined by the longest delay between the coordinator and any cohort. Protocols that use this structure include the Isis [3] group-communication protocols. One could also say that any network gateway acts as a coordinator for incoming traffic to that network.

3. *Peer*: A process starts a multicast by sending its message to all other processes; since all $n-1$ point-to-point messages may be in transit simultaneously, the time to complete a multicast is the longest delay between the initiator and any other process. Protocols that use this structure include the Psync [9] group-communication protocols.

The flow of information in any particular protocol can be very complex, depending on the exact properties of the multicast. We assume that the only messages that could be potentially infective, however, are the messages that contain the data to be delivered to the application. Other messages, such as those acknowledging the receipt of a message, seem less likely to offer the security holes that could be exploited to infect another process. We assume that for all integral

$\ell \geq 0$, each process initiates exactly one multicast for multicasts with sequence numbers in the range $\Big[n \times \ell + 1, n \times (\ell + 1)\Big]$.

3 Infection and Availability

In this section we define the terms by which we measure and evaluate our simulation results. These results are defined in terms of what we call *availability* and are displayed with two types of graphs, which we call *availability graphs* and *f-graphs.*

Definition 2 *Given an infective system σ, $P(\sigma, f, m)$ is the probability that, after m multicasts, no more than f processes in σ are infected, given that one process is initially infected.*

We estimate $P(\sigma, f, m)$ through simulation as described below. The availability of σ is derived from $P(\sigma, f, m)$ and removes the effect of having assumed that a process is initially infected.

Definition 3 *Given an infective system σ, the availability $A(\sigma, f, m)$ is the probability that, after m multicasts, no more than f processes in σ are infected.*

For example, $A(\sigma, 2, 100) = 0.99$ means that in σ, in 99% of the runs, no more than 2 processes will be infected after 100 multicasts. Thus, a protocol that masks failures can assume that no more than 2 processes are infected.

¿From conditional probability, we get $A(\sigma, f, m) = (1 - p_{init}) + P(\sigma, f, m) \times p_{init}$. Note that $P(\sigma, 0, m)$ is by definition 0, and so $A(\sigma, 0, m)$ is simply $1 - p_{init}$. This is obviously correct given that we assume $(p_{init})^2$ is zero and that infections can not spontaneously occur.

We consider two natural questions about infective systems:

1. Given p_{infect}, p_{init}, some maximum number of infected processes f and some number of multicasts m, what is the availability of the system? This is a question that one might ask to determine whether a given strategy is acceptable.

2. Given a desired availability, a, and given p_{infect}, p_{init}, and f, how many multicasts can occur before the availability falls below a? The answer to this question determines how often some anomalous behavior detection utility or disinfecting program must be run to enforce the assumption that no more than f processes are infected.

3.1 Estimating $P(\sigma, f, m)$ Through Simulation

A *simulation* is a set of runs ρ in which initially exactly one process is infected and in which M multicasts are sent and delivered. Throughout this work, $M = 300$ and $R = |\rho| = 1000$. For each run $r \in \rho$, we define $F(r, m)$ as the number of infected processes once all processes have received multicast m. Since we assume one process is initially infected, $F(r, 0) = 1$.

Define $N(\sigma, f, m) = \Big|\{r \in \rho \mid F(r, m) \leq f\}\Big|$, which is the number of runs $r \in \rho$ in which $F(r, m) \leq f$. Finally we estimate $P(\sigma, f, m) = N(\sigma, f, m)/R$. From these definitions, the first question above is answered by computing

$$A(\sigma, f, m) = 1 - p_{init} + P(\sigma, f, m) \times p_{init} = 1 - p_{init} + \frac{N(\sigma, f, m) \times p_{init}}{R} \; . \qquad (1)$$

Since processes do not become uninfected, $F(r,m)$ is monotonically increasing in m, and so $N(\sigma,f,m)$ is monotonically decreasing in m. Thus, the second question above is answered by finding the largest m such that $N(\sigma,f,m) \geq (a+p_{init}-1)R/p_{init}$. Since $N(\sigma,f,m)$ is nonnegative, this inequality trivially holds for all f and m when $a \leq 1-p_{init}$. Indeed, if the availability a is that low, then the percentage of runs in which there is no initially infected process is large enough to ensure that for all f and m, $A(\sigma,f,m) \geq a$.

Notation	Definition	Comments
ρ	set of runs of M multicasts	$M = 300$ in simulations
R	$\lvert\rho\rvert$	$R = 1000$ in simulations
$F(r,m)$	# infected after m multicasts	$F(r,0) = 1$
$N(\sigma,f,m)$	$\Big\lvert\{r \in \rho \mid F(r,m) \leq f\}\Big\rvert$	
$P(\sigma,f,m)$	$N(\sigma,f,m)/R$	Estimated
$A(\sigma,f,m)$	$(1-p_{init}) + P(\sigma,f,m) \times p_{init}$	Availability
$H(\sigma,a,m)$	$\min f : A(\sigma,f,m) \geq a$	for availability graphs
$\mathcal{I}(\sigma,f,m)$	$\Big\lvert\{r \in \rho \mid F(r,m-1) < f, F(r,m) \geq f\}\Big\rvert$	for f-graphs

Summary of notation and definitions.

3.2 Displaying Availability and Infection Rates

We display simulation results in two different formats.

3.2.1 Availability Graphs

Let $H(\sigma,a,m)$ be the smallest f that ensures an availability of at least a given σ and m,

$$H(\sigma,a,m) = \min f : A(\sigma,f,m) \geq a.$$

An *availability graph* plots the curves $H(\sigma,a,m)$ versus m for different values of a. One can think of $H(\sigma,a,m)$ as being an "iso-availability" curve for a given value of a. $H(\sigma,a,m)$ is a monotonically increasing step function in m; to keep the same availability, the number of possibly infected processes cannot diminish. By definition, for (unacceptably low) values of $a \leq 1-p_{init}$, $H(\sigma,a,m)$ is the horizontal line $f = 0$.

If a multicast strategy spreads infection quickly, the plot of $H(\sigma,a,m)$ will be nearly vertical; if it spreads infection slowly, each step of $H(\sigma,a,m)$ will endure for many multicasts. Consequently, the most resilient strategies will show intervals lasting many multicasts between $H(\sigma,a,m) = f$ and $H(\sigma,a,m) = f+1$.

Figure 1 is a sample availability graph, plotting $H^{peer}(\sigma,\ 0.99995,m), H^{ring}(\sigma,\ 0.99995,m)$, and $H^{cc}(\sigma,\ 0.99995,m)$ for $\sigma =<\ 7,\ 0.0001,\ 0.04\ >$. It shows that Ring and Coordinator-cohort are nearly equally poor under these circumstances, while Peer is best, especially so for $H(\sigma,\ 0.999995,m) \geq 4$, which Peer does not attain until multicast $m = 14$, while Ring and Coordinator-cohort attain at $m = 10$. No strategy can maintain an availability of 0.999995 for 50 multicasts without the entire system becoming infected.

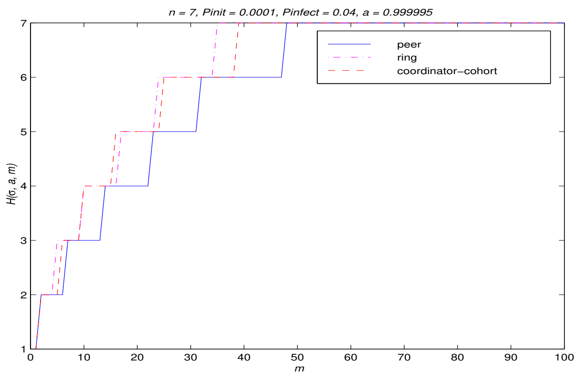

Figure 1: A typical availability graph. For $\sigma =< 7,\ 0.0001,\ 0.04 > anda = 0.9999995$, Peer is more resilient than Coordinator-cohort, which is more resilient than Ring.

3.2.2 f-Graphs

The second format of graph is called an *f-graph*. An f-graph plots, for a given value of f, the percentage of runs in the simulation in which the f^{th} infection occurs with multicast m. More precisely, define $\mathcal{I}(\sigma, f, m)$ to be the number of runs $r \in \rho$ for which $F(r, m) \geq f$ and $F(r, m-1) < f$. An f-graph plots $\mathcal{I}(\sigma, f, m)/R$ versus m. We also plot on an f-graph the values $N(\sigma, f-1, m)/R$ versus m. Note that $N(\sigma, f, m) = R - \sum_{i=1}^{m} \mathcal{I}(\sigma, f+1, i)$, and so the graph of $N(\sigma, f-1, m)/R$ is simply one minus the integral of the graph $\mathcal{I}(\sigma, f, m)/R$.

Figure 2 shows an f-graph for thee Coordinator-cohort strategy in which $n = 7, f = 2$ and $p_{infect} = 0.15$. The x-axis is the number of completed multicasts, the left y-axis is the percentage of runs in which the second process becomes infected with that multicast, and the right y-axis plots $N(\sigma, f-1, m)/R$. From this f-graph, the second process becomes infected in the first multicast, $m = 1$, in 12.5% of ($R = 1000$) simulation runs, in the third multicast in 4.3% of simulation runs, and in the 100th multicast in 0.3% of simulation runs.

Since f-graphs show $N(\sigma, f-1, m)/R$, they can be used to answer both of the questions posed above. f-graphs can also be used to understand the sensitivity of availability to the number of multicasts. As the number of multicasts grows, the availability for a fixed f must monotonically decrease. The rate of the decrease of $A(\sigma, f, m)$ is the same as the rate of decrease of $N(\sigma, f, m)$. Thus, one can examine the $(f+1)$-graph to determine how $A(\sigma, f, m)$ decreases with m. For example, if $N(\sigma, f, m)$ decreases slowly then one can increase m without a rapid decrease in availability.

4 Simulation Results

We ran several sets of simulations to determine the effects on availability of varying n, of varying p_{infect}, and of varying the multicast strategy. Each simulation generated 1,000 runs, each run lasting 300 multicasts. This is sufficiently long for most processes to become infected when $p_{infect} = 0.04$, which is the value we normally used in the simulations. Exactly one process is infected in the initial state of each run. For completeness, we include illustrative graphs in the Appendix; all graphs can be found at `http://www.nile.utexas.edu/~mj/disc-graphs/`.

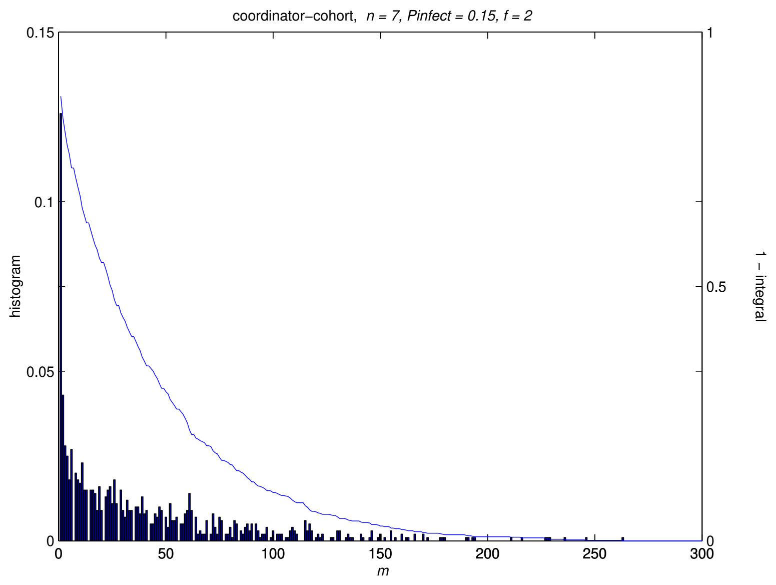

Figure 2: A typical f-graph showing, for Coordinator-cohort, when the second ($i.e.$, $f = 2$) process becomes infected. $\sigma =< 7,\ 0.0001,\ 0.15 >$.

4.1 Effect of Varying p_{infect}

Our simulations show that within each group strategy, and for the same values of f and n, scaling p_{infect} by a factor of $1/k$ is equivalent to scaling m by k. This is expected. If p_{infect} is reduced by k, then an infected process must send k times more messages to generate the same expected number of infective messages.

4.2 Effect of Varying n

Our simulations show that as the number of processes n increases, the availability $A(\sigma, f, m)$ decreases. This is significant because it is at odds with the traditional reasoning behind fault tolerance based on masking. A system of n processes can mask $n/3$ malicious failures or a minority of crash failures, and so as n increases more failures can be masked. Our results show that larger systems, though able to tolerate a larger absolute number of failures, may actually incur that fraction of failures very rapidly. Said another way, resilience to infection does not scale.

The rate at which availability decreases depends on the multicast strategy. Consider two systems, using the same multicast schema and having the same value of p_{infect}. The first system has n processes, and the second has $n' > n$ processes. Consider a run of each system, where each run contains nn' multicasts. In this and subsequent discussions, unless otherwise stated $p_{init} = 0.0001,\ p_{infect} = 0.04$ and the desired availability $a = 0.999995$.

- In the Peer strategy, each process in the first system multicasts n' times and each process in the second system multicasts n' times. When multicasting, a process in the first system sends $n - 1$ messages while a process in the second system sends $n' - 1$ messages. Thus, a process in the second system sends $n' - n$ more messages than its equivalent in the first system. The initially infected process therefore sends more messages, and the infection will spread faster.

For example, simulations show that at multicast $m = 25$, to maintain $a = 0.999995$, one must allow three of $n = 4$ processes to be infected, five of seven, five of ten, and six of 30 processes to be infected.

- In the Coordinator-cohort strategy, the coordinator sends $nn'(n' - n)$ additional messages in the second system, and each cohort sends $n' - n$ *fewer* messages in the second system. If the coordinator is the initially infected process, then the expected number of infective messages it generates will be much larger in the second system. If a cohort is the initially infected process, then the expected time for the coordinator to become infected is longer in the second system, but once the coordinator becomes infected it spreads much faster. When the initially infected process is chosen uniformly from all processes, then it is less likely that the coordinator is the initially infected process in the second system. The coordinator must be either the first or second infected process, however, and so the infection still spreads rapidly even when the coordinator is not initially infected.

  For example, at multicast $m = 25$, one must allow all of $n = 4$ processes to be infected, but six of seven, seven of ten, and 17 of 30 processes to be infected.

- With the Ring strategy, each process sends nn' messages in both systems. Hence, the infection rate is relatively insensitive to n. There are runs in the second system in which more processes are infected for small values of m just because there are more processes that can be infected, but these runs are very rare for small values of p_{infect}.

  Again, at multicast $m = 25$, one must allow all of $n = 4$ processes to be infected, but six of seven, six of ten, and only six of 30 processes to be infected.

$n =$	4	7	10	30	remarks
CC	20	39	53	93	
Peer	26	48	68	> 100	$H^{peer}(\sigma, a, 100) = 27$
Ring	11	35	57	$\gg 100$	$H^{ring}(\sigma, a, 100) = 15$

Ring would seem to fare best as system size increases, and Coordinator-cohort worst. The table on the left is extracted from Figure 3. It indicates the multicast at which the last of n processes becomes infected while ensuring that $a = 0.999995$. For masking-based strategies, the table immediately below, also taken from Figure 3,

$f/n =$	2/4	4/7	6/10	16/30
CC	2	10	15	23
Peer	2	15	22	56
Ring	2	12	17	> 100

indicates the multicast at which $\lfloor \frac{n}{2} \rfloor + 1$ processes become infected while maintaining $a = 0.999995$. While all strategies fare nearly equally for small system sizes, Ring shows a marked advantage for $n = 30$.

This comes at a cost: the time to complete a single multicast increases linearly with n as well.

4.3 Effect of Securing the Coordinator

In Coordinator-cohort, the infection rate greatly depends on whether the coordinator or a cohort is the initially infected process. For example, in Figure 2 the values for $1 \leq m \leq 6$ are unusually large precisely because they reflect runs in which the coordinator is initially infected. If it is not initially infected, it will not become infected, on average, until the initially infected process has initiated $1/p_{infect}$ multicasts. In all such cases, the coordinator becomes the second infected process, and this accounts for the tail of the distribution in Figure 2. Clearly, one expects availability to increase by securing the coordinator. Figure 4 bears this out; it shows that the Coordinator-safe strategy is more resilient than even Peer. Not surprisingly, it seems wise to invest in the security of a static coordinator; this is be less expensive than securing all entities in the system, in addition to being easier to monitor and maintain.

Unfortunately, these gains are short-lived and do not scale with system size (Figure 5); while the $H^{c-safe} = 1$ step endures longer than H^{peer} and H^{ring}, by multicast $m = 18$ Coordinator-safe has become, and thereafter remains, far worse than either.

4.4 Effect of Varying Desired Availability

For all strategies, lessening the desired availability means that more multicasts can be completed before the number of infected processes must increase. However, some strategies are more sensitive to these changes than others, to the extent that strategies' relative resilience changes. While a tight availability of $a = 0.999995$ favors Peer over Coordinator-cohort, (Figure 1), if $a \leq 0.999975$ is acceptable, then the reverse is true; Coordinator-safe is still the absolute best. Coordinator-cohort and Coordinator-safe show large improvements across the range of decreased availability; Ring, at the other extreme shows the least improvement throughout the range.

$a =$	0.999995	0.999970	improvement
CC	39	95	144%
Peer	48	87	81%
Ring	35	59	69%
C-safe	57	119	109%

For $\sigma =< 7,\ 0.0001,\ 0.04 >$, the accompanying table indicates the multicast at which $H(\sigma, a, m) = 7$ for $a = 0.999995$ and $a = 0.999970$.

The edge experienced by Coordinator-based strategies for low availabilities is attributable to the increased importance of any cohort, as opposed to the coordinator, being the initially infected process. The infected cohort sends only one message in each group of n multicasts whereas in Peer, the initially infected process sends $n - 1$ messages (in each group of n multicasts), and in Ring it sends n. With lower availabilities, this discrepancy becomes visible.

The improvement in the Coordinator-based strategies prompts a further inquiry into the effect of system size, this time coupled with decreased desired availability. Coordinator-cohort shows a tremendous improvement with respect to its own performance at tight availability (compare Figure 3 with Figure 6). For as long as 67 multicasts, one only has to allow 1 of 30 processes to be infected to maintain $a = 0.9999975$. After that, unfortunately, nearly one more infected process per multicast has to be allowed to maintain the same availability.

5 Conclusions and Future Work

In this paper, we examined a model of how the effects of intrusion spreads through a system. Our model is based on a simple notion of spreading: a compromised process can compromise another process by sending it a message, but it is only successful in doing so (that is, in generating an infective message) with a given probability p_{infect}. Each attempt to generate an infective message is independent.

This simple model of infection is, in some cases, may not be appropriate. For example, the Morris Internet Worm [12] had a fixed set of tricks that it used to try to replicate itself. In this case, a more appropriate model would have p_{infect} drop rapidly with time. For another example, many sites run the same operating system on many machines. If the infected process is able to exploit a trap door in one process, then it should be able to do the same for a large number of processes. In this case, if one message a process sends during a given multicast is infective, then most of the messages it sends during that multicast are infective. We are interested in exploring these models of infection.

There have been several recent protocols that treat compromised processes as being arbitrarily faulty as a basis to mask their effects. The results in this paper are a step towards understanding when such protocols can, in fact, be used. However, we have abstracted the message flow of multicast protocols to a degree that the results are not immediately applicable. To be applicable, one would need to choose specific protocols and to refine the infection model. Such research would generalize the work done on reliability for reliable multicast protocols (for example, [1]), and is an obvious next step for us.

Our model is also applicable to mobile agent systems, in which an infective message is equivalent to a malicious mobile agent. The results of this paper suggest that if one wishes to allow for the collection of information in a large network, then the mobile agents should be forced to visit the landing pads in a fixed sequence. This raises the question of how landing pads could cooperate to enforce an ordering on the sequence of landing pads an agent visits. This also begs the question of the effect of compromised landing pads. Assuming that an compromised landing pad can change the information that an agent carries, the best strategy a non-malicious agent should adopt is to send independent copies of itself to all landing pads. It would be interesting to see how these two goals, protection against malicious agents and compromised landing pads, could be balanced.

References

[1] Ö. Babaoğlu, "On the reliability of consensus-based fault-tolerant distributed computing systems", in *ACM Transactions on Computer Systems,* vol. 5, pp. 394–416, 1987.

[2] N. T. Bailey, *The Mathematical Theory of Infectious Diseases and Its Applications,* second edition, Oxford University Press, 1975.

[3] K. P. Birman and R. van Renesse, *Reliable Distributed Computing with the Isis Toolkit*, IEEE Computer Society Press, 1994.

[4] K. P. Birman and R. van Renesse, "Software for Reliable Networks," *Scientific American,* May 1996.

[5] A. Demers et al., "Epidemic Algorithms for Replicated Database Maintenance,"

[6] L. Lamport, R. Shostak and M. Pease, "The Byzantine Generals Problem," in *ACM Transactions on Programming Languages and Systems,* vol. 4, pp. 382–401, 1982.

[7] J. O. Kephart and S. R. White, "Measuring and Modeling Computer Virus Prevalence," in *Proc. of the 1993 Computer Society Symposium on Research in Security and Privacy,* pp. 2–15, 1991.

[8] J. O. Kephart and S. R. White, "Directed-Graph Epidemiological Models of Computer Viruses," in *Proc. of the 1991 Computer Society Symposium on Research in Security and Privacy,* pp. 343–359, 1991.

[9] L. L. Peterson, N. Buchholz and R. D. Schlichting, "Preserving and Using Context Information in Interprocess Communication," in *ACM Transactions on Computer Systems,* vol. 7, pp. 217–246, 1989.

[10] L. E. Moser et al., "Totem: A Fault-Tolerant Multicast Group Communication System," in *Communications of the ACM,* April 1996.

[11] M. K. Reiter, "Secure Agreement Protocols: Reliable and Atomic Group Multicast in Rampart," in *Proc. of the 2nd ACM Conference on Computer and Communication Security,* pp. 68–80, November 1994.

[12] D. Seeley, "A Tour of the Worm," in *USENIX Conference Proceedings*, pp. 287–304, Winter 1989.

Appendix of Graphs

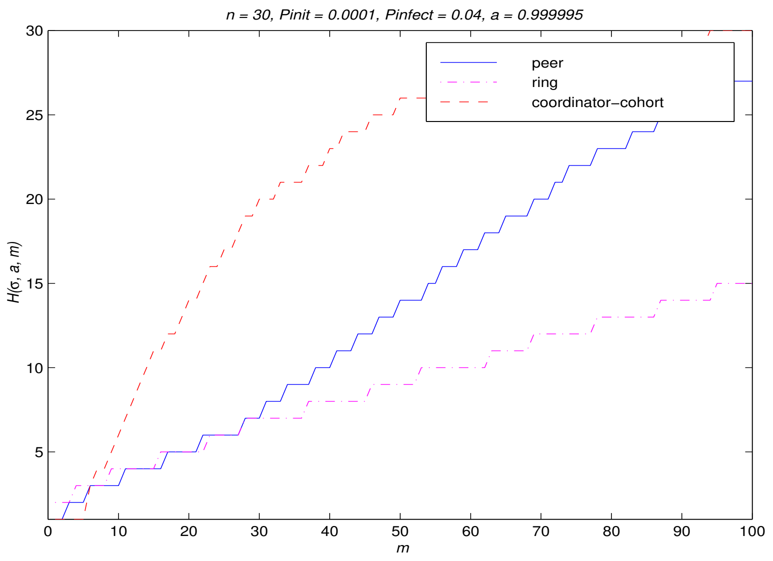

Figure 3: Effect on availability of increasing system size. $\sigma =< 7,\ 0.0001,\ 0.04 > a = 0.999995$.

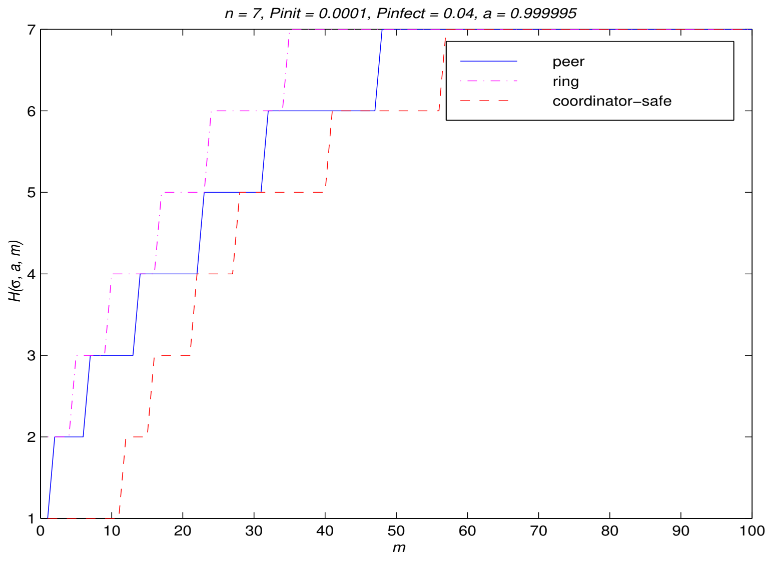

Figure 4: Effect on availability of ensuring the coordinator is not initially infected. $\sigma =< 7,\ 0.0001,\ 0.04 > a = 0.999995$.

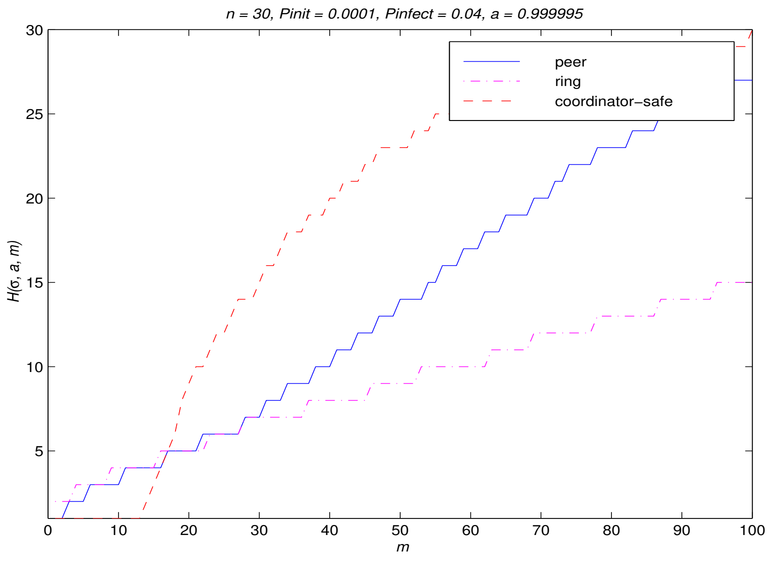

Figure 5: The Coordinator-safe strategy for larger system sizes. $\sigma =< 30,\ 0.0001,\ 0.04 > a = 0.999995$.

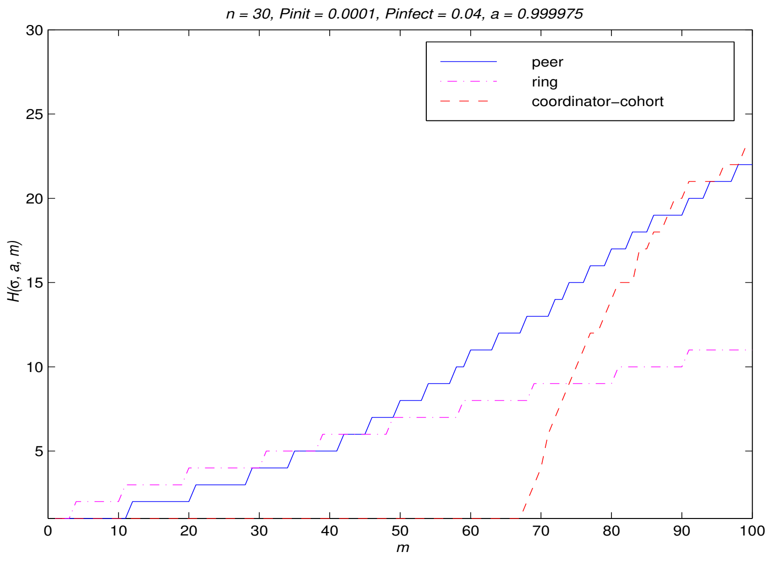

Figure 6: Lower availability and larger systems. $\sigma =< 30,\ 0.0001,\ 0.04 > a = 0.999975$.