On the Resilience of Multicasting Strategies in a Failure-Propagating Environment

Meng-Jang Lin, Aleta Ricciardi and Keith Marzullo

TR TR-PDS-1998-002 February 1998

Parallel & Distributed Systems group

Department of Electrical & Computer Engineering

University of Texas at Austin

Austin, Texas 78712

On the Resilience of Multicasting 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

February 6, 1998

1 Introduction

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

It seems plausible to mask the effects intruders in the same way that one masks arbitrary failures. Masking the effects of failures requires replication, and several protocols have been designed to use replication to mask the effects of intruders [1, 2]. 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 intruded processes when reaching agreement [3].

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.

We call such behavior an *infection* and a process that is being seized by an intruder and that can propagate the attack on the intruder's behalf an *infected* process. An infection spreads when more processes become faulty and infected in such a fashion. This implies that the techniques used to mask arbitrarily faulty processes cannot be applied to infected processes because a sufficient degree of replication cannot be guaranteed.

An ideal approach is to make it impossible for infected processes to infect other processes, but this is not always possible. Security attacks can be hard to

*This work is 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.

prevent, especially when building on top of commercial systems that have not been designed with security against attack a high concern. A second approach is to slow down the rate of infection sufficiently to allow tools like anti-virus programs and intrusion detection systems to examine enough of a process's behavior in order to diagnose the infection and allow for its removal.

We consider the second problem in this paper. More specifically, 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 efficacy of the infection method changes, and as different multicast strategies are used. The answers to these questions can be used to decide how often a "disinfection" mechanism must be run so that the assumptions made by the failure-masking protocols mentioned above can be sustained.

These questions, however, have a broader applicability. For example, in a mobile agent system, one can model an infected process as a compromised landing pad and a mobile agent capable of compromising a landing pad 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 wishes to collect information in a web-crawl-like manner in such a system, a compromised landing pad can corrupt information carried by agents that pass through that pad. Hence, a multicast strategy could be used to limit the damage by compromised landing pads. The work in this paper can help one decide what multicast strategy to use, and how often landing pads must be checked for being compromised.

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 The 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. 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.

We denote with p_{init} the probability that a process is infected in the initial state, where p_{init} is small enough to ignore the probability of there being more than one initially infected process. Each message sent by an infected process is infective with probability p_{infect}. We make the 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 parametrized by* $< n, p_{init}, p_{infect} >$, *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 determined by 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 the message from its predecessor, process $(i-1) \bmod n$. Protocols that fit this structure include the Totem group-based protocols [2] 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. Protocols that use this structure include the Isis [4] 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. Protocols that use this structure include the Psync [5] 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. Figure 2 illustrates the actual messages that we consider for each pattern. We assume that $\ell \geq 0$, each process initiates exactly one multicast for multicasts with sequence numbers in the range $\Big[n \times \ell, n \times (\ell+1) - 1\Big]$.

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*

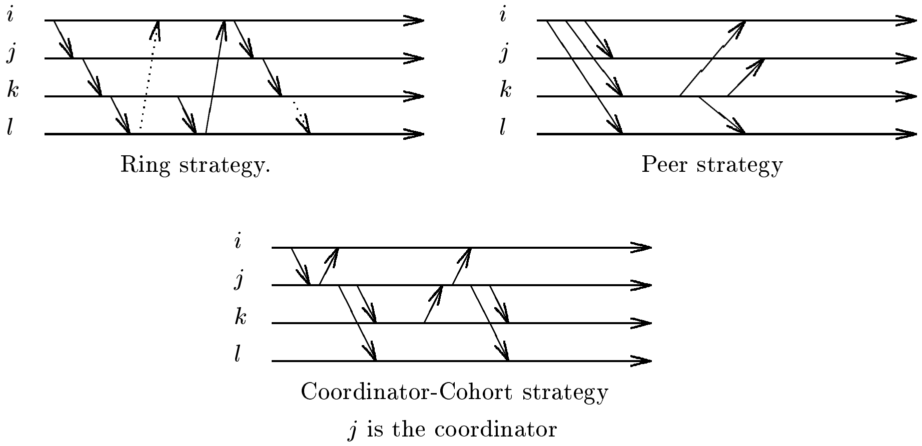

Figure 1: Three communication patterns examined in this paper. In each pattern shown, processes i and k initiate multicasts.

and are displayed with two types of graphs, which we call *infection 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 any 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.* $A(\sigma, f, m) \;=\; (1 - p_{init}) + P(\sigma, f, m) \times p_{init}$.

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.

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.

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

Figure 2: Notation and defintions.

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

We now describe how we estimate $P(\sigma,f,m)$ through simulation. Figure 3.1 summarizes all definitions and notation. 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 $r \in \rho$ and $m \le 300$, 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 $N(\sigma,f,m)$ to be the number of runs $r \in \rho$ in which $F(r,m) \le f$.

$$N(\sigma,f,m) = \Big\lvert\{r \in \rho \mid F(r,m) \le f\}\Big\rvert \ .$$

Finally we estimate

$$P(\sigma,f,m) \ = \ \frac{N(\sigma,f,m)}{R} \ .$$

From these definitions, the first question above is answered by computing

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

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

$$N(\sigma,f,m) \ge \frac{(a + p_{init} - 1)R}{p_{init}} \ . \quad (2)$$

Since $N(\sigma,f,m)$ is nonnegative, when $a \le 1 - p_{init}$ the above inequality holds for all f and m. When a is this low, the percentage of runs in which there is no initial infection is large enough to ensure that for all f and m, $A(\sigma,f,m) \ge a$.

3.2 Displaying Availability and Infection Rates

We display the results of the simulation in two different formats.

3.2.1 Infection Graphs

The first format is called an *infection graph*, which plots the average number of infected processes for each value of m. That is, the infection graphs plots $\sum_{r\in\rho} F(r,m)/R$ versus m. We also indicate on infection graphs the variability of the values $\{r \in \rho : F(r,m)\}$ for each m. Since these values are not distributed normally, we define this variability in terms of a parameter c that captures the middle c per-cent chunk of the values of $\{r \in \rho : F(r,m)\}$.

For example, consider a simulation of 10 runs in which we are interested in the number of processes infected by the end of the second multicast. Suppose the values obtained are

bin 1 $F(r_1,2) = F(r_3,2) = F(r_4,2) = F(r_7,2) = 1$,

bin 2 $F(r_2,2) = F(r_6,2) = 2$

bin 3 $F(r_5,2) = F(r_8,2) = F(r_9,2) = 3$,

bin 4 $F(r_{10},2) = 4$.

Then, if we choose $c = 0.8$, we keep the middle 80% of values, and disregard one of the values in the bin 1 and the value in bin 4.

The terms $H_c(m)$ and $L_c(m)$ describe the endpoints of this chunk. Specifically, if the values of $F(r,m)$ are sorted by increasing value, then $H_c(m)$ is the value in position $R-(R(1-c)/2)$ in the sequence and $L_c(m)$ is the value in position $1+(R(1-c)/2)$. In the example, $H_{0.8}(2) = 3$ and $L_{0.8}(2) = 1$. The variability for m is the range from $L_c(m)$ to $H_c(m)$.

Infection graphs can be used to answer the second question posed in Section 3 above. By definition, $\frac{(1+c)}{2}$ percent of the runs have no more than $H_c(m)$ infected processes at multicast m; in our example, 90%, of the runs have no more than 3 infected processes after the second multicast. Said another way,

$$\frac{N(\sigma, H_c(m), m)}{R} = \frac{1+c}{2} \quad \text{and} \quad A(\sigma, H_c(m), m) = 1 - p_{init} + \left(\frac{1+c}{2}\right) \times p_{init} .$$

Solving the above for c, we get

$$c = \frac{2 \times \Big(A(\sigma, H_c(m), m) + p_{init} - 1\Big)}{p_{init}} - 1.$$

Having chosen a desired availability a, one sets $c = \Big(2 \times (a + p_{init} - 1)/p_{init}\Big) - 1$ and generates an infection graph using this value of c. The largest value of m for which $A(\sigma, f, m) \geq a$ is the largest value of m for which $H_c(m)$ is less than f.

An example infection graph, simulating the Peer group strategy, is given in Figure 3. The y-axis is the number of infected processes, and the x-axis is the number of completed multicasts. In this example $n = 7, c = 0.8, p_{init} = 0.001$, and $p_{infect} = 0.1$. The integer-valued terms $L_c(m)$ and $H_c(m)$ are represented by the step functions, which bound the continuous function of the average number of infected processes. In all infection graphs, the leftmost step function will be $H_c(m)$ and the rightmost $L_c(m)$.

From this infection graph, $L_{0.8}(30) = 2$ which means that in 90% of runs, at least 2 processes will be infected after 30 multicasts; the average over all runs

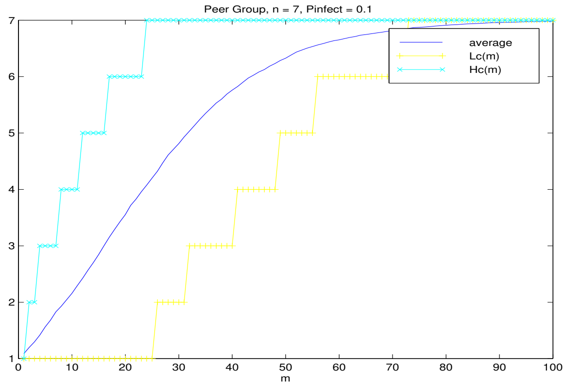

Figure 3: A typical infection graph. $\sigma =< 7,\ 0.001,\ 0.1 >, c = 0.8$. Peer strategy

after 30 multicasts is 5 infected processes. The value $H_{0.8}(m)$ attains the value of 4 at $m = 10$, which means that this strategy can be run for nine multicasts until 10% of runs show more than 3 processes infected. Hence, if our goal is to ensure that $A(\sigma, 3, m) \geq .9999$, then disinfection must occur before the tenth multicast.

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 $N(\sigma, f, m) = R - \sum_{i=1}^{m} \mathcal{I}(\sigma, f+1, m)$, 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 4 shows an f-graph simulating the Peer group strategy in which $n = 7, f = 2$ and $p_{infect} = 0.1$. The x-axis is the number of completed multicasts and the y-axis is the percentage of runs in which the second process becomes infected. The histogram plots $\mathcal{I}(\sigma, f, m)/R$ versus m, while the lower curve plots one minus the integral of the histogram, which is $N(\sigma, f-1, m)/R$. From this f-graph, the second process becomes infected for $m = 8$ in 31 runs, for $m = 15$ in 21 runs, and for $m = 30$ in only 6 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.

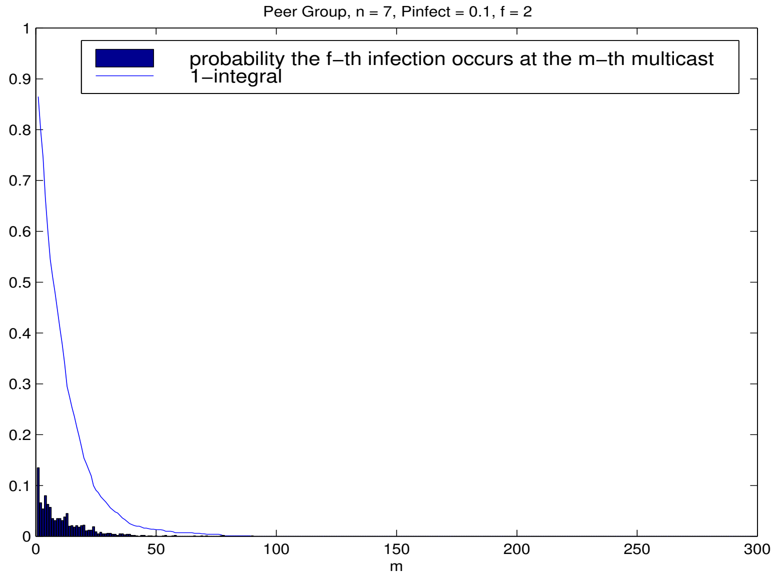

Figure 4: A typical f-graph. $\sigma =< 7, 0.001, 0.1 >, f = 2$. Peer strategy.

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 all processes to become infected when $p_{infect} = 0.1$, which is the value we normally used in the simulations. Exactly one process is infected in the initial state of each run.

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 what we expected to see. 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. The rate of decrease 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.

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

- With the Coordinator-Cohort strategy, the coordinator will send $nn'(n' - n)$ additional messages in the second system, and each cohort will send $n' - n$ *less* 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.

- 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}.

4.3 Comparing Group Strategies

In this section we compare the three strategies' resilience to spreading infection. The analyses of the previous section play a role here, as the Ring strategy far exceeds Peer (the favorite for small n) for larger systems. We examine a special case of Coordinator-cohort, called Coordinator-safe, that can out-perform Peer for small n, though it looses this edge for large n.

In all cases we use infection graphs to compare strategies' upper bound of variability, $H_c^{\mathrm{ring}}(m)$, $H_c^{\mathrm{peer}}(m)$, and $H_c^{\mathrm{cc}}(m)$. We have $c = 0.8$, so the desired availability is $a = (1 - p_{init}) + (\frac{(1+c)}{2}) \times p_{init} = 0.999 + 0.0009 = 0.9999$. The analyses hinge on an observation made in Section 3.2.1:

> ... the largest value of m for which $A(\sigma, f, m) \geq a$ is the largest value of m for which $H_c(m)$ is less than f.

4.3.1 The Resilience of the Peer Strategy

Figure 5 compares the Peer and Ring strategies for $\sigma =< 7,\ 0.001,\ 0.1 >$. Examining the graph for the case $f = 4$, we see that Peer reaches this plateau with multicast $m = 8$ (that is, $H_c^{\mathrm{peer}}(8) = 3$ and $H_c^{\mathrm{peer}}(9) = 4 = f$); Ring, on the other hand, reaches the same plateau with multicast $m = 4$ (that is, $H_c^{\mathrm{ring}}(4) = 3$, while $H_c^{\mathrm{ring}}(5) = 4 = f$). Similarly for $f = 6$, Peer reaches the plateau at multicast $m = 17$ ($H_c^{\mathrm{peer}}(17) = 5, H_c^{\mathrm{peer}}(18) = 6$), while Ring reaches it at $m = 10$ ($H_c^{\mathrm{ring}}(10) = 5, H_c^{\mathrm{ring}}(11) = 6$). All processes become infected, $f = 7$ in Peer at multicast $m = 24$, but in Ring at $m = 14$.

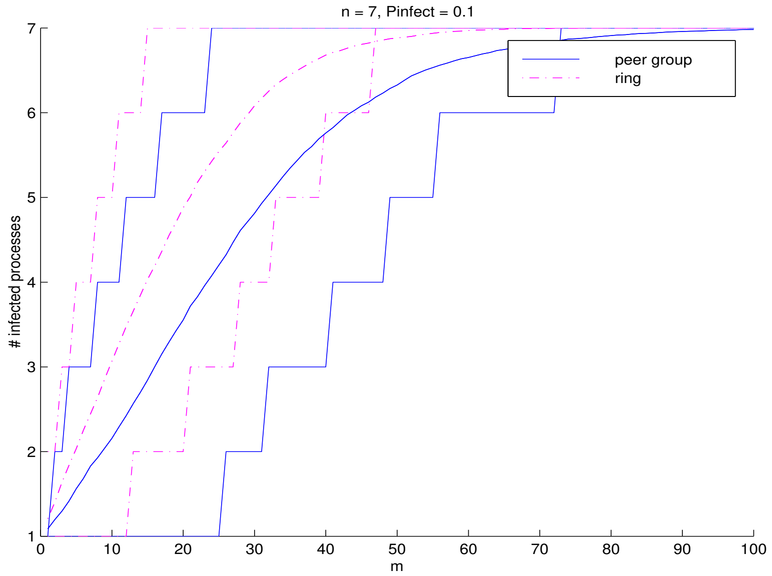

Figure 5: Peer vs. Ring. $\sigma =< 7,\ 0.001,\ 0.1 >, c = 0.8$.

Figure 6 shows the comparison for Peer and Coordinator-cohort. Peer reaches the plateaux for $f = 4, f = 6$, and $f = 7$ at multicasts $m = 8, m = 17$, and $m = 24$ respectively, Coordinator-cohort reaches them at $m = 7, m = 13$, and $m = 20$.

As the next section shows, Coordinator-cohort can be improved relative to Peer.

4.3.2 Effect of Securing Coordinator

In Coordinator-cohort, the infection rate greatly depends on whether the coordinator or a cohort is the initially infected process. Before the coordinator becomes infected, the infection cannot spread at all. In the general version of Coordinator-cohort strategy, the coordinator is as likely as any other process to be the one that is initially infected. If it is not initially infected, the coordinator will not become infected, on average, until the initially infected process has issued $1/p_{infect}$ multicasts. Therefore, one expects that availability would increase greatly by securing the coordinator. The f-graphs depicted in Figures 7-9 bear this out.

In Figure 7, the histogram value for $m = 1$ is unusually large precisely because it represents the case in which the coordinator is initially infected. That is, when the coordinator is initially infected, it is rather likely to infect another process with the first multicast. In Figure 8, the coordinator has been 'secured', in that in all runs of the simulation, the coordinator is not initially infected (we refer to this as the *Coordinator-safe* strategy). In this graph, the large value for $m = 1$ is absent.

Figure 10 shows that the Coordinator-safe strategy is more resilient than Peer. For $f = 4, f = 6$, and $f = 7$, Coordinator-safe reaches its plateaux at multicasts $m = 13, m = 21$, and $m = 29$. Figure 4.3.2 summarizes these

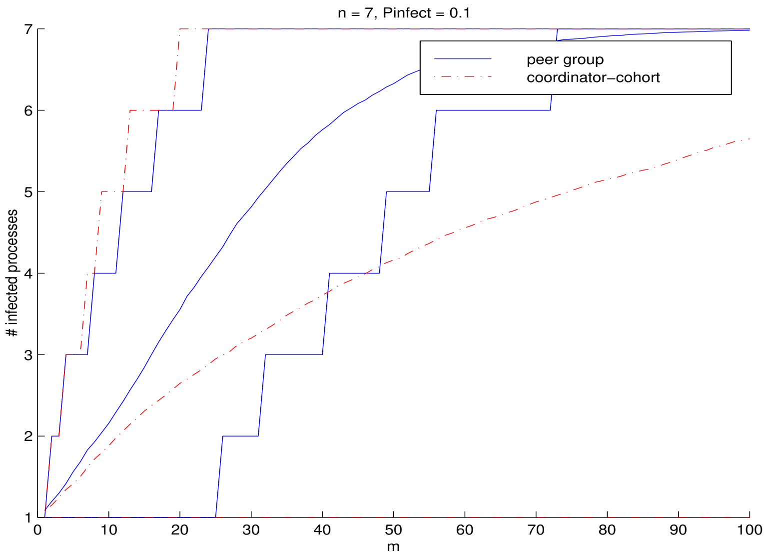

Figure 6: Peer vs. Coordinator-cohort. $\sigma =< 7, 0.001, 0.1 >, c = 0.8$.

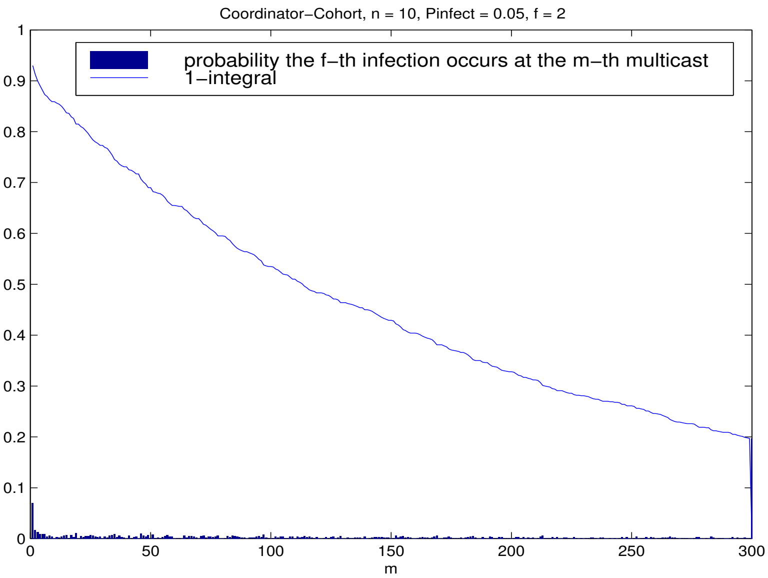

Figure 7: Coordinator-cohort f-graph. $\sigma =< 10,\ 0.001,\ 0.05 >, f = 2$.

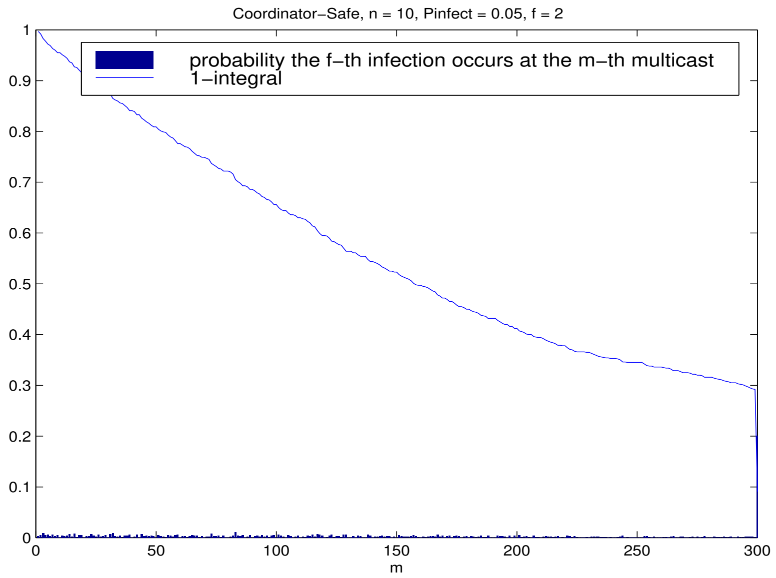

Figure 8: Coordinator-safe f-graph. $\sigma =< 10,\ 0.001,\ 0.05 >, f = 2$.

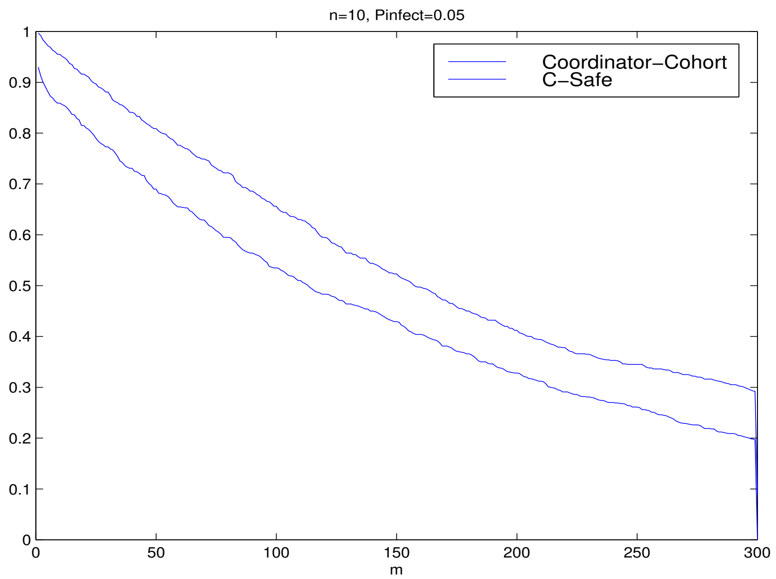

Figure 9: Overlay of Figures 7 and 8 showing the effect of securing the coordinator.

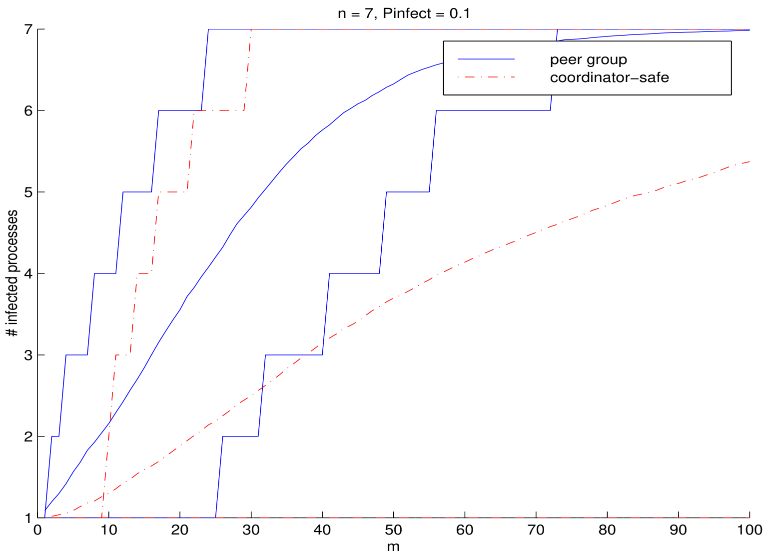

Figure 10: Peer vs. Coordinator-safe. $\sigma =< 7,\ 0.001,\ 0.1 > c = 0.8$

f	Ring	Coordinator-Cohort	Peer	Coordinator-safe
4	4	7	8	13
6	10	13	17	21
7	17	20	24	29

Figure 11: For $\sigma =< 7,\ 0.001,\ 0.1 >$, and $c = 0.8$, for each strategy the value max $1 \leq m \leq 300 : H_c(m) < f$.

comparisons.

Not surprisingly, it seems wise to invest in the security of a static coordinator; this would be less expensive than securing all entities in the system, in addition to being easier to monitor and maintain.

4.3.3 Larger System Sizes

Finally, we compare the three strategies for their scalability with system size. It turns out that as the number of processes in a system grows, the Ring strategy far exceeds Peer and Coordinator-safe, and that Coordinator-safe degrades quickly.

Figure 12 shows that $H_c^{\text{peer}}(m)$ is consistently and significantly larger than $H_c^{\text{ring}}(m)$. For $f = 7, f = 15$, and $f = 24$, Peer reaches plateaux at multicasts $m = 12, m = 22$, and $m = 33$. Ring however reaches the same plateaux at multicasts $m = 15, m = 38$, and $m = 72$.

Finally, Figure 13 shows that Coordinator-safe is briefly very resilient; however, once the coordinator becomes infected, the resilience of this strategy degrades very quickly.

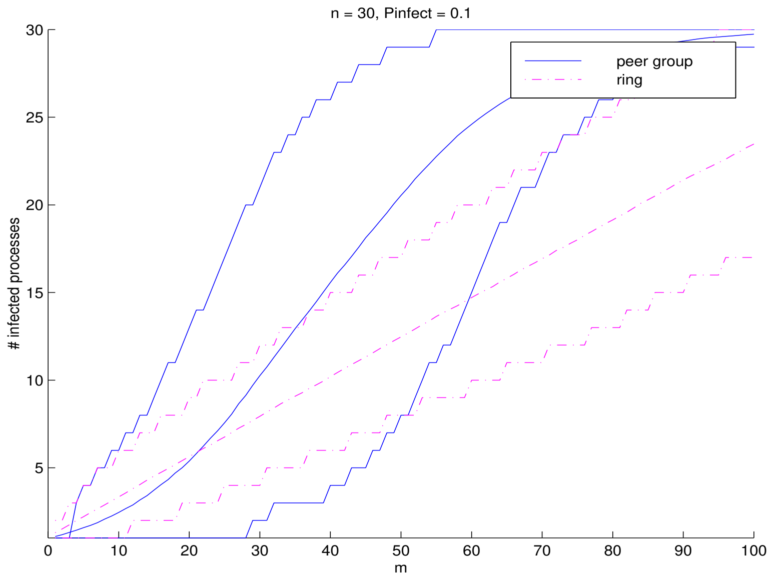

Figure 12: Peer vs. Ring. $\sigma =< 30,\ 0.001,\ 0.1 >$ $c = 0.8$

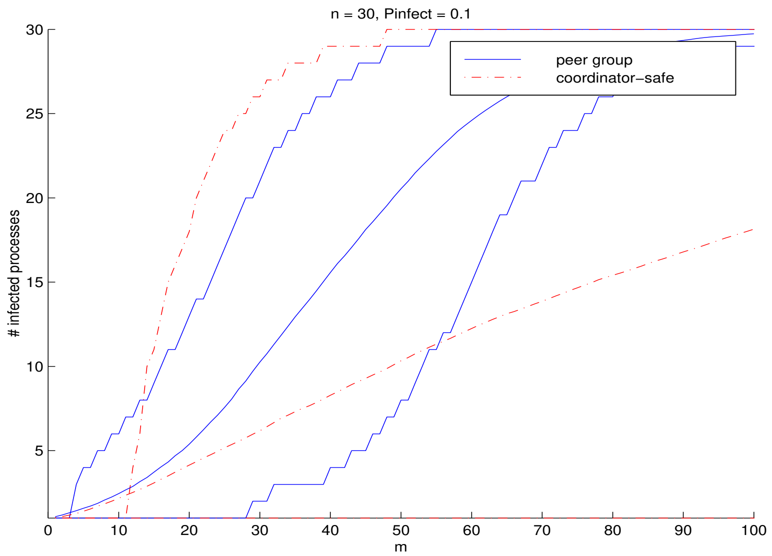

Figure 13: Peer vs. Coordinator-Safe. $\sigma =< 30,\ 0.001,\ 0.1 >$ $c = 0.8$

f	Ring	Peer	Coordinator-safe
7	15	12	0
15	38	22	17
24	72	33	24

Figure 14: For $\sigma =< 30,\ 0.001,\ 0.1 >$, and $c = 0.8$, for each strategy the value $\max\ 1 \leq m \leq 300 : H_c(m) < f$.

5 Conclusion

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 many cases, not realistic. For example, the Morris Internet Worm [7] had a fixed set of tricks that it used to try to replicate itself. In this case, a more realistic 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 and other more realistic 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, [6]), 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] 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.

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

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

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

[5] L. L. Peterson and N. Buchholz and R. D. Schlichting. "Preserving and Using Context Information in Interprocess Communication," in *ACM Transactions on Computer Systems,* 7(3):217–246, 1989.

[6] O. Babaoglu. On the reliability of consensus-based fault-tolerant distributed computing systems. ACM Transactions on Computer Systems 5(4):394–416, November 1987.

[7] D. Seeley. A Tour of the Worm. USENIX Winter'89 Conference. pp. 287–304.