A New Model for Availability in the Face of Self-Propagating Attacks *

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

1 Overview

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 yet, 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 [7, 8]. 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 arbitrary failures is vanishingly small, and that the likely explanation for such failures is due to malicious attacks, then the self-propagating nature of these attacks should also be considered.

We have been examining 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 for us 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.

1.1 Implications for Secure Middleware

Our results impact the design of middleware for group-based communication systems [3]. An attack at this level of a system would be very hard to detect. Using the results of our work, 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 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. Our results can be used to choose a dissemination mechanism for the agents and decide how often the landing pads must be cleaned.

1.2 Relation to Other Work

Our definition of the metric *availability* differs from that commonly used in the security community, but both are instances of how availability is generally defined. Consider a system and a specification. The system's availability with respect to the specification is the probability that the system satisfies the specification. In security, the specification of interest is usually "the service responds in a timely manner

*Supported by DOD-ARPA under contract number F30602-96-1-0313. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force or the U.S. Government.

1998 NSPW 9/98 Charlottsville, VA, USA

to all requests", and so the availability measures the probability of the service not suffering from a denial of service. In this paper, the specification we are concerned with is "no more than f processes are infected". The two can be related. If one considers infection due to multicast protocols, then the processes using the multicast may together be implementing a service that masks f failures. If more than f processes are infected and therefore faulty, then the masking technique will be unable to stop the faulty processes from providing an arbitrarily faulty service.

From a modeling point of view, our work resembles the application of epidemiology [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.*, [5]). However, our work is different from existing epidemiological approaches in several aspects. For example, our transmission of infection is more restricted than general mixing of populations or a mixing restricted to undirected graphs. A second difference is that we measure availability rather than the expected percentage of infected processes as a function of time. 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.

We differ in a third way as well: rather than using a general epidemic model, we use a simple epidemic model and treat disinfection as a separate and unmodelled activity. Inherent in our notion of availability is the assumption that the system will not misbehave if no more than f processes are infected. And, disinfection is most likely an expensive operation: either diagnostics must be run to locate infected processes or all the processes need to be reloaded from trusted images. Hence, disinfection should be done as infrequently as possible as long as not too many processes become infected.

Some of the observations we draw in this paper are consistent with earlier work in epidemiological algorithms. For example, [5] observe 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 [4], 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.

2 Results to be Discussed

2.1 Model and Simulation Set-up

We consider an asynchronous distributed system composed of n processes that communicate with one another by passing messages over a network of point-to-point channels. 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).

Each process has a probability of being infected in the initial state. We assume that this is the same for all processes, and we denote it with the simple variable p_{init}. We assume that p_{init} is small enough to ignore the probability that more than one process is initially infected. Each message sent by an infected process is infective with probability p_{infect}. Both parameters can be enriched to tune the model to specific platforms, security domains, and so forth.

Definition 1 *An infective system σ is parameterized by $< n, p_{init}, p_{infect} >$.*

Since infection spreads through communication, the rate of infection depends on the pattern of communication, which is determined by considerations including message efficiency, time efficiency, and the underlying network topology. We have considered four patterns of communication:

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 $(i-1) \bmod n$.

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.

Peer: A process starts a multicast by sending its message to all other processes.

Tree: The processes are partially ordered into a single, unidirected, balanced binary tree. Being undirected, each process can be considered to be a root of this tree (of course, the tree will not be balanced for all of these roots). A process starts a multicast by considering itself to be the root of this tree and sending the message to its children in the tree. The multicast continues by having a process, upon receiving a message from its parent in this tree, send the message to its children.

Definition 2 *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. $A(\sigma, f, m) = (1 - p_{init}) + P(\sigma, f, m) \times p_{init}$.*

We estimate availability through simulation, and then consider the two questions posed earlier:

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 can 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? This 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.

We have run simulations to determine the effects on availability of varying p_{infect}, of varying n, 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.

2.2 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 not surprising. 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.

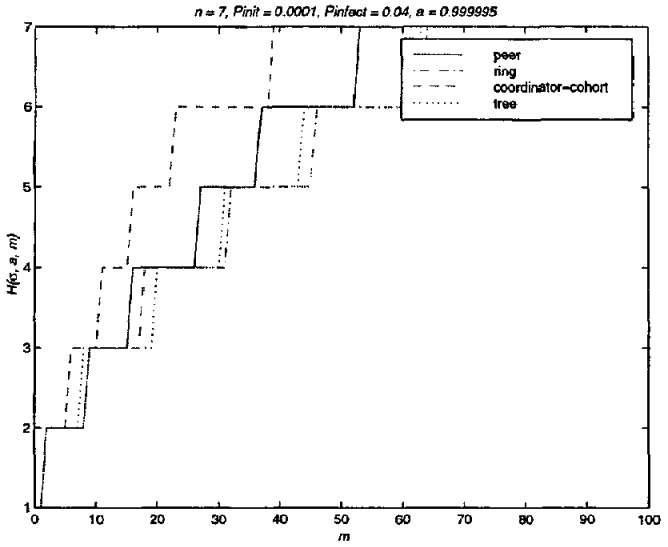

Figure 1: $\sigma =< 7, 0.0001, 0.04 >, a = 0.999995$.

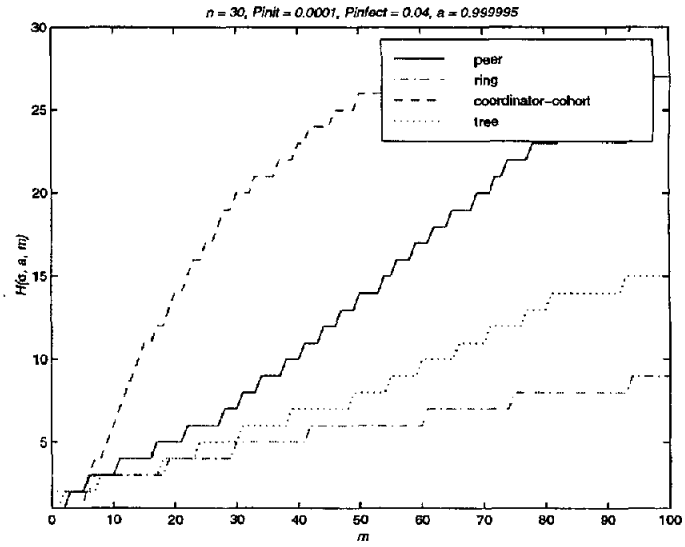

Figure 2: $\sigma =< 30, 0.0001, 0.04 >, a = 0.999995$.

2.3 Effect of Varying Protocol

Let $H(\sigma, a, m)$ be the smallest f that ensures an availability of at least a given σ and m. We plot the curves $H(\sigma, a, m)$ versus m for different protocols, given σ and a.

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 plots $H^{peer}(\sigma, 0.99995, m)$, $H^{ring}(\sigma, 0.99995, m)$, and $H^{cc}(\sigma, 0.99995, m)$ for $\sigma =< 7, 0.0001, 0.04 >$. It shows that Ring is the most resilient, followed by Tree, followed by Peer. Coordinator-cohort is the least resilient strategy. No strategy can maintain an availability of 0.999995 for 65 multicasts without the entire system becoming infected.

The explanation for Ring's resilience is intimately related to its drawback. Each multicast of the Ring strategy is composed of $n-1$ sequential point-to-point messages; before the next hop can be undertaken, all previous hops must be completed. Thus, the time to complete a Ring multicast is long, roughly the sum of the times to complete each of the $n-1$ hops. Relatedly, each process can only infect one specific target, its successor, so before the fourth process can become infected, the third must be, and before it, the second; infection, like message hops, is sequential and each process's target is restricted with Ring.

Tree is similar to Ring. The only difference is that each process can infect up to 2 processes (*i.e.*, its children) simultaneously. With Peer, an infected initiator may, with each multicast, infect any other process. However, the time to complete a multicast is roughly the maximum transmission delay between the initiator and the other processes.

Coordinator-cohort can also infect $n-1$ processes per multicast, once the coordinator becomes infected. The reason why it is less resilient than Peer is because in Peer if the initiator of a multicast is not infected, then the messages it sends are not infective, while in Coordinator-cohort an infected coordinator can send out infective messages in every single multicast.

2.4 Effect of Varying n

Our simulations show that as the number of processes n increases, the availability $A(\sigma, f, m)$ decreases. This finding 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; thus, as n increases more failures can be masked. Our results show that larger systems, though able to tolerate a larger absolute number of failures, will actually incur that fraction of failures at a faster rate than will small systems.

The trend of strategies' relative resilience remains the same, Ring the best and Coordinator-cohort the worst. However, the advantage of, say, the Ring strategy is more obvious when $n = 30$, since it is still the case that each process can only infect one specific target. Compared, for example, to Peer where more process can become infected with each multicast.

3 Conclusions

This simple model of infection, in some cases, may not be appropriate. For example, the Morris Internet Worm [9] 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. Or, many sites run the same operating system on many machines, and should infected process 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.

Our results are a step towards understanding when Byzantine masking protocols can 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. Our results 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 a 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, "Software for Reliable Networks," in *Scientific American,* May 1996.

[4] A. Demers et al., "Epidemic Algorithms for Replicated Database Maintenance," in *Proceedings of the ACM 6th Annual Symposium on Principles of Distributed Computing,* pp. 8–32, 1987.

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

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

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

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