ContagAlert: Using Contagion Theory for Adaptive, Distributed Alert Propagation

Michael Treaster, William Conner, Indranil Gupta, Klara Nahrstedt
Department of Computer Science
University of Illinois
Urbana-Champaign, IL 61820
Email: {treaster, wconner, indy, klara}@cs.uiuc.edu

***Abstract*— The widespread uses of large-scale distributed systems, e.g., Grid networks and distributed storage systems, raise the possibilities of large-scale attacks on such systems. Although current technology for detecting worms and viruses in the Internet can be applied, few existing systems support *fast propagation* of alerts *during the attack itself*. This paper proposes and studies a new system towards this problem. The system, called "ContagAlert", uses contagion spreading behavior, somewhat like the spread of fads in society, in order to spread alerts. ContagAlert is able to propagate an alert while the attack is in progress, while at the same time suppressing disruptive signals generated by adversaries or false positives. The core contagion protocols in the system are completely localized, involving simple threshold checks at each node, but resulting in desired emergent threshold behavior at the network scale. Signals with too few sources fail to spread, and signals exceeding the threshold propagate across the entire network. Contagion protocols can be analyzed using bootstrap percolation models. We also present experimental results from contagion protocols running in a wide variety of topologies. Finally, we present experiments based on two real-life applications: Internet worm attacks and DoS attacks on p2p systems.**

I. Introduction

In recent years, there has been a substantial increase in the deployment of large-scale distributed systems such as Grid networks [12], federated testbeds such as PlanetLab [5], and storage networks such as HP's Federated Array of Bricks (FAB) [15] and IBM's Collective Intelligent Bricks (CIB) [20]. However, this increase in the availability of such valuable computing resources has led to a rise in the occurrence of large-scale attacks on these systems. These attacks may come in the form of worms like Code Red [30] or Sapphire [10], or as explicit hacking, such as the recent Cisco attacks [6] or intrusions of TeraGrid systems [27].

Today's technology handles such attacks in a mostly human-oriented manner which is both slow and error-prone. Intrusions are detected by human observation of symptoms. Once the diagnosis is made, intrusion data is manually spread among system administrators, who repair the damage and patch the vulnerability by hand.

Even when single-host and cooperative automatic intrusion detection systems [11], [14], [39], [41] are deployed, the distribution of alert data is typically regarded as secondary to detection capabilities. This allows adversaries to interfere with the proper functioning of the system by exploiting vulnerabilities in the alert distribution mechanism, and presents weaknesses in the filtering of false positives.

This paper proposes new techniques for automatic alert propagation. Unlike prior techniques, however, we focus not only on quickly propagating alerts, but also on suppressing the false positives and messages of malicious intent even when these disruptive messages cannot be explicitly identified. This recognition of messages as legitimate or disruptive is performed implicitly by leveraging ideas from contagion theory. Since disruptive messages are not explicitly identified by are prevented from interfering with the system, the protocol is *tolerant* of such messages.

Contagion theory studies the spread of influences, such as social fads, power grid failures, political movements, or biological epidemics, across interconnected populations. These populations are modeled as a graph, describing the relationships (edges) between individuals (vertices) in the population. The influence originates in some number of seed individuals in the graph, and spreads across the graph using a threshold-based activation rule [40]. Individuals which have adopted an influence are said to be *infected*[1]. A *contagion* has occurred if all nodes eventually become infected.

Both in this model and in the real world, a few carefully placed seeds can result in a cascade effect as the idea infects the rest of the population [16]. These cascades are more likely to occur as the number of seeds increases. Contagion theory shows that the probability of such cascades quickly moves from very low to very high as the number of seeds passes a threshold. Our protocol exploits this cascade behavior by adjusting the infection threshold on nodes in order to control the network threshold at which cascades occur.

Standard contagion theory employs a fractional threshold model, where a node becomes infected after a certain fraction of its neighbors are infected. We use a variation on this model in which a node becomes infected after some fixed number of its neighbors are infected. This modification simplifies the implementation of the protocol by reducing the number of messages that must be sent and the bookkeeping that is required, while also enabling the protocol behavior to be analyzed by comparing it with bootstrap percolation [4].

[1]Later on, we use "alerted" instead of "infected" to refer to a node that has adopted an alerted state. We reserve the term "infected" to refer to a node under the influence of a worm or other attack.

II. Related Work

The goal of our work is to provide a protocol that prevents malicious messages and false positives from interfering with the well-behaved nodes in the network. Most systems accomplish this by attempting to keep malicious nodes out of a system entirely using a trust mechanism. There are two basic approaches to instituting trust relationships in a network.

The first approach introduces a central certificate authority that validates each node as legitimate and trustworthy. Since the legitimacy of each node is established through an out-of-band operation, all nodes can be assumed to be trustworthy. In the event that a node changes its behavior from benign to malicious, message signatures allow harmful signals to be traced back to their source such that the sender can be held accountable for its actions [19], [34].

Central certificate authorities, like any centralized technology, have difficulties with scalability. As the user population expands, it becomes more difficult to verify the credibility of each individual user and credibility of the authority decreases. There exist several projects which implementing distributed certificate authorities which avoids this issue [9], [43].

The second approach is a distributed system which constructs a "web of trust" among peers. Although there are a variety of variations on this approach, they all share the same basic premise. Each node trusts certain other nodes in the system based on a system-specific trust metric. Regardless of the basis of the trust, these immediate relationships are treated as somewhat transitive as a function of the distance to the stranger peer and the trust levels of the intervening nodes. When a signal is received, a node can weight the message based on the level of trust and act accordingly [1], [21], [28], [37].

In both the centralized and decentralized approaches, trust-based systems have problems if a malicious agent is able to attain a trusted status. This could occur through several means. For example, one user could feign good behavior until a trusted status is built up, at which point he attempts to subvert the normal behavior of the network, or an adversary could compromise the system of a legitimate user, thus hijacking the user's trusted status. The fundamental problem is that trust is a complex commodity to manage. Once given, it is difficult to revoke, and once attained, it is easily misused.

The ContagAlert protocol avoids these issues by doing away with the idea of trust entirely, and instead relying on the premise that legitimate messages will be generated by a larger fraction of the network than an adversary would be able to subvert to his own uses. However, this does limit the class of applications for which the ContagAlert protocol is suited to those that have a high redundancy in the generation of signals. On the other hand, the ContagAlert protocol is effective at suppressing false positives that are accidentally generated by well-meaning peers in the system.

The work of Nojiri, et al [33] bears a similarity to the ContagAlert protocol. As in the ContagAlert protocol, each node has some number of neighboring peers, and if the number of alerted neighbors exceeds a certain threshold, the node becomes alerted itself and notifies its neighbors of the new alert. The threshold setting is fixed, and the number of neighbors is treated as an input parameter that can be modified rather than as a fixed aspect of the network.

The ContagAlert protocol has some important differences, however. The threshold setting is adaptively set to adjust to a desired tolerance for false signals, and the number of neighbors is treated as a parameter over which we have no control. This allows the ContagAlert protocol to operate in a wider variety of network conditions, and allows the threshold separating legitimate messages and disruptive messages to be specified.

III. Network Topologies

We explored the behavior of the ContagAlert protocol on a variety of network topologies. Some of the topologies resemble those used in common peer-to-peer overlays, while others are more abstract in nature. Six network topologies were studied: random, circle, semi-circle, affinity groups, exponential, and power law.

Figure 13 depicts an example of each topology on a 16-node network. In these diagrams, the black segments on the gray edge lines indicate the node that "possesses" the directed edge. A large number of black segments on a node indicates that the node knows of many neighbors.

A *random* network has fully randomized connectivity. When a network with n nodes and average degree k is initialized, $n \times k$ directed edges with random end points are created. The exact degree of connectivity varies according to a binomial distribution.

The *exponential* topology closely resembles any of several peer-to-peer overlay networks such as Pastry [36], Chord [38], or Tapestry [42], that use O(log n) storage at each node and O(log n) messages to reach an arbitrary destination. In a network of size k with nodes of degree d, a node n shares data with all nodes $n + k^{i/d} < k : i \in \mathbf{Z}^*$. For example, in a network of size 16 with nodes of degree 4, node 0 would have neighbors 1, 2, 4, and 8.

In a network with a *circle* topology and nodes of degree k, each node communicates with its $k/2$ immediate neighbors. For example, in a circular network with nodes of degree 4, node 8 would have links to 6, 7, 9, and 10.

The *semi-circle* topology is a hybrid of the circle topology and the random topology. When constructing this topology, it is initially identical to the circle topology, with each node fully connected to its $k/2$ neighbors to either side. However, for each edge there is a probability p that the neighbor is chosen at random from all nodes instead of being deterministically set to one of the k neighbors. This generates a structure in which there is a high degree of local clustering, but the random edges reduce the average path length to nodes that would be distant in a normal circle topology. In our simulations, p was set to $.25$.

The *affinity group* network topology is designed to approximate the connectivity used by the Kelips [17] peer-to-peer overlay. In this design, a network of size n is subdivided into

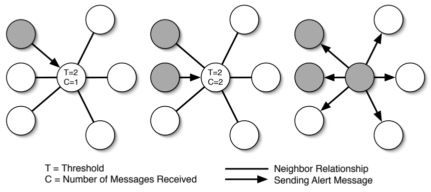

Fig. 1. Threshold-based Activation

groups of size $\sqrt{n}$. Nodes in each affinity group are fully connected within their group. Additionally, each node has k directed links to a random node in each other affinity group. In our model, k was set equal to 1.

The *power law* topology uses the commonly occurring power law distribution [29] to describe how many nodes a given node knows about. In such a network, a relatively small number of nodes maintain a relatively high fraction of all the edges in the graph, but there is a heavy tail of nodes with a small number of links. More specifically, the probability that a given node will have degree x is proportional to x^{-k}, for some constant k. The links are directed, so if node A has a link to node B, B may not have a link to A. When constructing power law networks, we use $.55 <= k <= .65$, based on the size of the network being constructed. The value of k is chosen to keep the average degree of nodes similar to the average node degree for other topologies of the same size.

Whenever any network was generated for any simulation presented here, it was tested to determine if there were any disconnected components. If there were, the network was discarded and a new one was generated. This guarantees that all graphs that were studied had only a single component. This more closely represents the real world, where it makes little sense to consider how two disconnected peer-to-peer networks might spread information between each other.

IV. Protocol Design

The purpose of the ContagAlert protocol is to distribute legitimate messages across a peer-to-peer network, while suppressing malicious messages and false positives. We refer to such messages as *disruptive*.

The protocol is intended to be used in applications where legitimate signals have a high degree of redundancy. There must exist a threshold, stated as a fraction of the number of peers in the system, which separates the legitimate signals from disruptive signals based on the number of sources of the signal. In other words, disruptive nodes cannot compose an arbitrarily large fraction of the network.

We make two additional assumptions about the network environment. First, all network connections are reliable. There are no dropped messages, unexpected message delays, or failed network links. Secondly, neighbor relationships need not be reflexive. It is allowable, but not required, for node B to be in the neighbor list of node A, but for node A to be unknown to B.

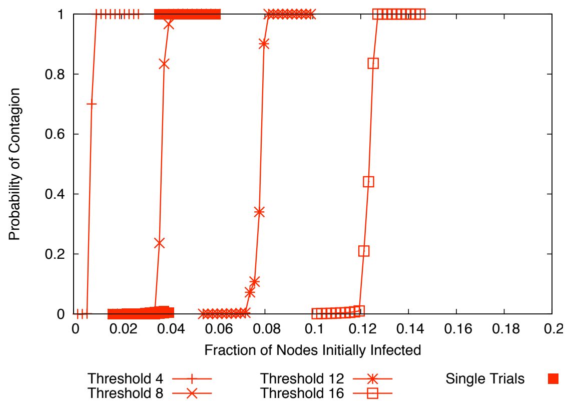

Fig. 2. Comparison of Contagion Threshold for Different Node Activation Thresholds – Random Topology, 4096 Nodes with Average Degree 64

A. Basic Protocol

The basic approach of the ContagAlert protocol is very simple. Peers in the cooperative network each have a small list of neighbors whose network locations are known. At various times, certain peers will observe an attempted intrusion or attack and will generate an alert which is sent to its neighbors to provide them with advance warning. The neighbors each independently decide whether to act on the alert, taking defensive measures and forwarding the alert to their respective neighbors. This decision procedure is the core of the ContagAlert protocol.

Each peer in the system has its own, independent threshold to determine whether or not it will forward a message to all of its neighbors. This threshold is expressed as a constant number of messages. For example, if a node has a threshold of 3, it will forward a message to all of its neighbors if it first receives a copy of the message from at least three different nodes. This behavior is illustrated in Figure 1. We refer to this piece of the protocol as the *basic* ContagAlert protocol.

This type of decision procedure results in very clear threshold behavior emerging from the network as a whole, as shown in Figure 2. This figure shows the probability of a contagion occurring given the fraction of nodes seeding the alert. A 4096-node network with a random topology and an average of 64 directed edges per node was used for these trials. The X-axis represents the number of instances of a simulated alert that were used to seed the propagation. The Y-axis represents the fraction of the network that accepted the alert as legitimate and forwarded the data to neighboring nodes. Thirty trials were run for each seed value. Four of the trend lines in the graph show the average value of all thirty trials for each seed point. The fifth point set (filled squares) in the graph shows the results of each individual trial in the data for the threshold of 8.

The frequency of contagion is 0% at the left of the graph and 100% at the right. In the middle is a region where, for certain numbers of seeds, a contagion may or may not occur. The single-trial data show that there is never a “half-contagion” where a cascade spreads to a large number of non-seed nodes but does not complete the contagion. This observation allows

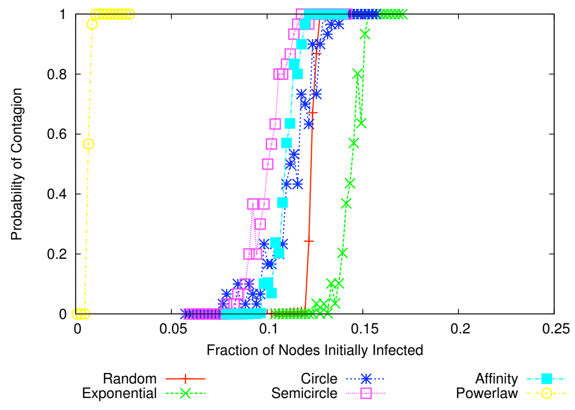

Fig. 3. Comparison of Contagion Threshold for Different Network Topologies – 4096 Nodes with Average Degree 64, Activation Threshold 16

the average line (X symbols in the figure) to be interpreted as the probability that a contagion will occur, given a certain number of seeds. We call the steeply sloped region of the line the *threshold region*.

It is a crucial to observation: given just a single trial, it is impossible to assess where the trial lies with regard to the threshold region because the result indicates only whether or not a contagion occurred.

This type of threshold behavior occurs regardless of the network size, topology, and average node degree. However, these characteristics will cause the network's threshold region to shift to the left or right, or to stretch wider or narrower. As an example, Figure 3 shows the effects of network topology on the network's threshold behavior. All other factors are held constant. It can be seen that using the Semicircle topology results in a left shift, while the Exponential topology results in a right shift. The Random topology produces a narrower threshold region, showing that the network's behavior transitions from no-contagion to contagion very quickly as the number of seeds is increased.

In order to shift the threshold region to a specified point, the activation thresholds on individual nodes must be adjusted to account for changes in these traits. Since these traits are difficult to ascertain for a decentralized, peer-to-peer system, and since they can change over time, the protocol parameters cannot be specified ahead of time or computed offline. Instead, the protocol must *adaptively* set the threshold setting for each node in the system at runtime.

B. Adaptive Protocol

The goal of the ContagAlert protocol as a whole is to generate a network-level threshold behavior, with the threshold region centered on the fraction of nodes that must be seeded for a contagion to be possible. We call this number the *target tolerance*, because it describes the approximate number of messages that can be tolerated by the protocol before the message is propagated throughout the rest of the network. Signals with too few seeds are assumed to be disruptive and will not spread, while signals with enough seeds are assumed to be legitimate and will result in contagion.

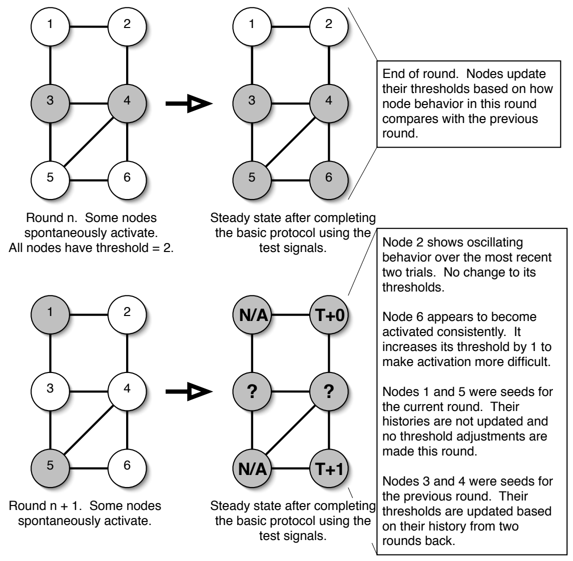
1
2
3
4
5
6
Round n. Some nodes spontaneously activate. All nodes have threshold = 2.
Steady state after completing the basic protocol using the test signals.
End of round. Nodes update their thresholds based on how node behavior in this round compares with the previous round.
Round n + 1. Some nodes spontaneously activate.
Steady state after completing the basic protocol using the test signals.
N/A
T+0
?
?
N/A
T+1
Node 2 shows oscillating behavior over the most recent two trials. No change to its thresholds.
Node 6 appears to become activated consistently. It increases its threshold by 1 to make activation more difficult.
Nodes 1 and 5 were seeds for the current round. Their histories are not updated and no threshold adjustments are made this round.
Nodes 3 and 4 were seeds for the previous round. Their thresholds are updated based on their history from two rounds back.

Fig. 4. Adaptive Procedure

The principal difficulty in constructing such a system is the adjustment of the contagion threshold to the target tolerance. Even with an omniscient, centralized controller it is not clear how this would be accomplished. It is difficult to assess how the many network parameters will affect the threshold region, and it is not possible to estimate the current position of the threshold region with a single evaluation of the network.

The difficulty is compounded when each node in the system is responsible for adjusting its own threshold based on its limited, local view. Additionally, a node has control only over itself. If it should decide that another node should adjust its threshold in a certain way, it has no way to force that node to listen. Finally, although a node n possesses a list of neighboring nodes, these are not necessarily the nodes that are sending alerts to n. A node has no direct means of knowing how many other nodes it should expect to hear from or what local conditions exist for those nodes.

The ContagAlert protocol addresses these difficulties with a second piece of the protocol. This piece runs in parallel to the basic piece of the protocol, and adjusts the threshold of nodes based on their number of incoming edges in the connectivity graph, despite the fact that it is unknown exactly how many such edges there are.

To accomplish this adaptivity, the network periodically executes "practice" runs of the basic ContagAlert protocol to allow nodes to experiment with different threshold settings. Nodes in the network periodically generate synthetic test messages and attempt to spread these messages as though they were real. Messages are generated at the target tolerance level of the protocol. Each node adjusts its threshold such that the contagion result appears one-half of the time, resulting in a shift of the threshold point to the target tolerance fraction at the network-wide level. This behavior is illustrated in Figure 4, and is additional detail below.

More specifically, each node periodically generates a test message with probability p, where p is the target tolerance fraction for the network, used to differentiate between legitimate and malicious nodes. This inserts approximately $p \times n$ signals into the system. Ideally, the sloped region of the network threshold graph should be centered over this point. With this number of seeds, sometimes a contagion will occur and sometimes it will not. Node thresholds are configured perfectly and the contagion threshold is positioned perfectly when, on sequential test rounds, the network alternates between contagion and no-contagion. Attaining this alternating behavior is the goal of the adaptive part of the ContagAlert protocol.

After deciding whether to generate a message, each node adheres to the basic ContagAlert protocol, forwarding the practice messages (or not) according to their own threshold T. This continues for a period of time, until no messages are received for a timeout period and it can be assumed that the contagion would have completed if a contagion were going to occur. Each node decides on its own when to time out and end the practice round for itself.

After the time period expires, each node assesses whether or not a contagion has occurred in the network. If it decides a contagion has occurred, it assumes (perhaps incorrectly) that the network as a whole is somewhere to the right of the threshold region of the graph. Otherwise it assumes the network is left of the threshold region. The node then adjusts its personal threshold to inhibit or facilitate future contagions based on its assumption history.

After each round, each node adjusts its threshold in an attempt to more closely achieve its desired alternating behavior. The node examines its behavior history for the previous two rounds to attempt to determine if it is presently alternating, if its threshold is too high, or if its threshold is too low. This two-round time window allows the node to classify itself into one of three categories:

1) Both of the last two rounds resulted in alerts. This suggests that the node's threshold is set too low and that the node is becoming alerted too easily. In this case, the node responds by increasing its threshold by 1.
2) Both of the last two rounds resulted in no alert. This suggests that the node's threshold is set too high and that the node is having too much difficulty becoming alerted. In this case, the threshold is reduced by 1.
3) The last two rounds have produced differing alert results. This indicates that the node has attained the desired alternating behavior. No changes are necessary.

Each node adjusts its threshold as dictated by its self-assessed categorization. Over many test rounds, its threshold approaches the point where, when considered with all other nodes in the network, the network's tolerance threshold for malicious nodes is shifted to the specified level. Signals with a number of sources below the threshold will be suppressed, while signals exceeding the threshold will be distributed throughout the network.

If a node acts as a spontaneously activated seed for a practice round, it does not update its history or adjust its threshold at the end of the round. The spontaneous activation probability has no bearing on the accuracy of the current threshold value, and making adjustments based on the spontaneous activation prevents the node from converging to the appropriate value.

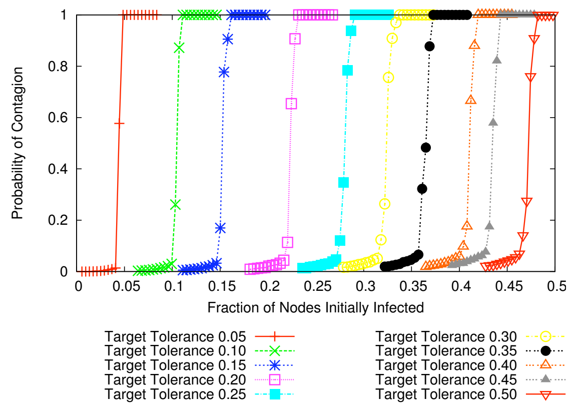

Fig. 5. Threshold Behavior of the ContagAlert Protocol Adapted to Different Target Tolerances – Random Topology, 4096 Nodes of Average Degree 64

C. *Churn*

Churn is an ever-present condition in any distributed system. The ContagAlert protocol has churn resistance built-in through the adaptive aspect of the protocol. Provided that the rate of churn is sufficiently low, the adaptation will adjust the threshold on each node to accommodate changes in membership in the network or changes in connectivity between nodes.

During periods of dramatic network change and the following period during which the adaptive protocol re-stabilizes, the performance of the protocol is unpredictable. After the protocol has time to re-adjust the node thresholds, behavior returns to normal. Therefore, if the rate of churn is too high, the adaptation will not have time to converge before more changes occur in the network structure. Higher churn rates can be handled by increasing the frequency of the adaptation periods if necessary.

Churn reduces the precision of the network level threshold adjustment to a degree proportional to the severity of the churn. More severe churn will result in more severe, longer duration disruption to the ability of the protocol to adhere to the target tolerance. If churn is minor but continuous, or if churn is severe but widely spaced in time, the ContagAlert protocol will attain more accurate performance.

V. Protocol Simulation Results

We implemented a custom simulator to examine the adaptivity performance of the protocol. The simulator generates a network graph of a specified size, topology class, and degree. The adaptive protocol is executed for a large number of rounds with a specified target tolerance to allow the node thresholds to stabilize.

Once the adaptive phase is completed, the simulator runs a large number of trials to assess the behavior of the network when subjected to "real" signals with various numbers of starting seeds. The simulation for each seed value was repeated

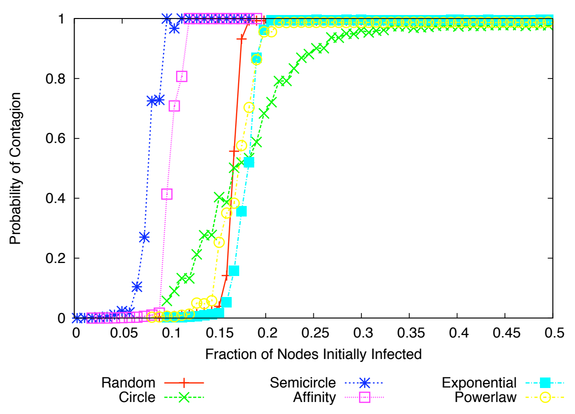

Fig. 6. Adaptive Performance of ContagAlert Protocol on Various Network Topologies – 1024 Nodes of Average Degree 32, Target Tolerance 0.15

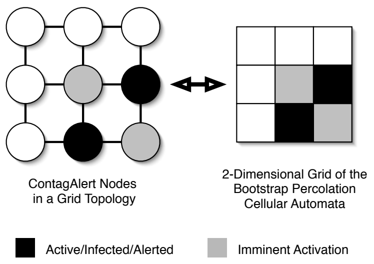
ContagAlert Nodes in a Grid Topology
2-Dimensional Grid of the Bootstrap Percolation Cellular Automata
Active/Infected/Alerted
Imminent Activation

Fig. 7. Mapping of ContagAlert onto Bootstrap Percolation

30 times in order to assess the probability that a contagion would occur for any given number of seeds. The large number of different seed values, combined with the large number of repetitions for each seed value, allows the probability curve of the network's behavior to be closely estimated.

The simulation proceeds in rounds. On each round, each node counts the number of alerts it has received from any neighbors. If this number exceeds its threshold, the node becomes alerted and broadcasts the alert to all of its neighbors. This round procedure repeats until the network state achieves a stable, unchanging state.

Figure 5 shows the behavior of simulated networks after the adaptive period has completed. Each trend line represents a different target tolerance. This data indicates that the adaptation and various target tolerance settings definitely have an effect on the network behavior. Although the adaptation is not exactly perfect, each trend line is appears in the correct region of the plot. The ordering of the lines is correct, indicating that setting a higher target tolerance for the protocol will result in a higher contagion threshold for the network.

Figure 6 shows the effects of network topology on the adaptive performance of the ContagAlert protocol. Note that four of the topologies have a 50% probability of contagion when almost exactly 0.15 of the nodes are seeds. This is precisely the behavior desired from the protocol. In the Affinity and Semicircle topologies, the adaptation was less effective.

VI. Protocol Analysis

The ContagAlert protocol views a peer-to-peer network as being akin to a cellular automaton on an irregular topology. In particular, the basic form of the protocol bears at close resemblance to *bootstrap percolation*, a family of deterministic cellular automata using a class of activation rules on an n-dimensional lattice.

Although ContagAlert is inspired by and draws ideas from contagion theory, due to the modification of the decision procedure on a node to use a constant threshold instead of a fractional one, it is more straightforward to analyze the system from the angle of *bootstrap percolation*. Bootstrap percolation describes a specific deterministic cellular automaton rule on a two-dimensional lattice of cells. Each cell is activated with probability p at time $T = 0$. On each subsequent time step, inactive cells become active if at least two of their neighbors are active. Cells never deactivate. More general, but less common models of bootstrap percolation allow for d dimensional lattices, and cells activate when l neighbors are active.

The study of bootstrap percolation focuses on determining the probability that the entire lattice will become activated, given the fraction of lattice nodes that are active at the beginning of the process. It has been shown that this probability shows a sharp threshold behavior. The probability shifts from nearly 0 to nearly 1 over a very small increase in the fraction of seed nodes. The ContagAlert protocol leverages this threshold effect to suppress malicious signals and false positives, while allowing legitimate signals to spread normally.

The basic aspect of the ContagAlert protocol maps closely onto this definition of the bootstrap percolation scenario, as illustrated in Figure 7. Each peer participating in the protocol corresponds to a cell in bootstrap percolation. The neighbors of the peer correspond to the neighboring cells in the lattice. The threshold number of alert messages that a peer must receive before becoming alerted corresponds to l, the threshold number of activated neighbors required to cause a bootstrap percolation cell to become activated. The fraction of peers acting as the source of an alert or other signal corresponds to the activation probability p. The only difference between the two systems is in the topology of the neighbor relationships. While the ContagAlert protocol can function on any network topology, bootstrap percolation is limited to only d-dimensional lattices.

In both bootstrap percolation and the ContagAlert protocol, it is easy to see that if p is too small, few or no nodes will be initially activated. Any spreading of the activation state will stall very quickly, and most of the lattice will remain inactive. Inversely, if p is high, most of the nodes will be initially activated, and the remaining inactive nodes will activate very quickly as time progresses.

The probability of the active state completely filling the network (a contagion) can be considered as a function of p. Although one might expect this probability to begin at 0 when p is small and gradually increase to 1 as p nears 1, this is not actually the case. In fact, this probability remains close to 0 up to some threshold fraction p, at which point it rapidly

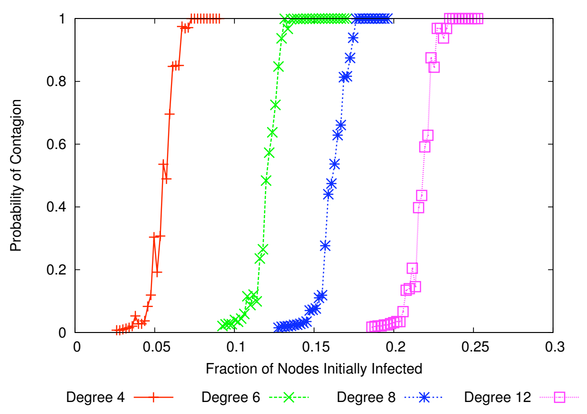

Fig. 8. Threshold Behavior of the Lattice Topology – 4096 Nodes, Activation Threshold = Node Degree / 2

increases to 1 as p increases just a small amount. When p exceeds this threshold point, the lattice is almost guaranteed to achieve 100% activation.

It is exactly this threshold behavior in bootstrap percolation systems that the ContagAlert protocol exploits. The nature of this threshold point is the common topic of research regarding bootstrap percolation. If it is assumed that the ContagAlert protocol is running on a d-dimensional lattice topology, the results of bootstrap percolation research can be directly applied to this work.

A. Lattice Network Topology (Bootstrap Percolation)

In bootstrap percolation, it has been proven that for an $L \times L$ lattice, as L becomes large, the probability that the entire lattice becomes alerted approaches 1 if the fraction of seed nodes is greater than $(\pi^2/18)/\ln(L)$, and it approaches 0 if the fraction of seed nodes is less than this value [18]. Therefore, for a 4096 node network with a 2-dimensional lattice topology and an activation threshold of 2 on every node, we expect the contagion threshold to occur at $(\pi^2/18)/\ln(\sqrt(4096)) = 0.1318$, or 540 seed nodes. Although the degree 4 trend line in Figure 8 shows that this is not actually the case, this can be attributed to the fact that the convergence to this threshold is extremely slow [2].

A similar result has been proven for n-dimensional bootstrap percolation for $n > 2$ dimensions with arbitrary activation threshold l [3]. In this case, however, the result is more complex and less precise.

For $2 < l \leq d$, there exist two constants, $\alpha_-(d,l)$ and $\alpha_+(d,l)$, such that

$$0 < \alpha_-(d,l) \leq \alpha_+(d,l) < \infty$$

independent of p, such that if

$$L_\pm(d,l,p) := exp^{\circ(l-1)}(\alpha_\pm p^{-\frac{1}{d-l+1}})$$

then

$$P(contagion) \to 1 \text{ if } (p,L) \to (0,\infty) \text{ with } L \geq L_+(d,l,p)$$
$$P(contagion) \to 1 \text{ if } (p,L) \to (0,\infty) \text{ with } L \leq L_-(d,l,p)$$

This result is proven by induction, showing how scenarios with arbitrarily large d and l can be reduced down to the 2-dimensional, threshold 2 case, one step at a time. However, they are unable exactly to specify the threshold. Instead, they are able to prove that the threshold exists, and they are able to reduce the uncertainty to the values α_+ and α_-, which are functions of only d and l.

The results presented above correspond only to n-dimensional lattices. Obviously, very few peer-to-peer overlays will have this type of topology. As topologies become more irregular, these thresholds will vary in a manner that may be impossible to quantify. The theoretical results above are largely based on the predictable way in which an activation state will propagate across the lattice. This predictability is not shared by networks with any kind of random component, and it can be hard to see even in topologies with regular structures.

However, we can qualitatively consider the behavior we would expect to observe from these more common topologies. We approach the problem by considering the clustering coefficient of each structure. If nodes A and B are neighbors, and if nodes B and C are neighbors, the clustering coefficient of the topology is the probability that node A and C are neighbors. A high clustering coefficient suggests that large cascades are easier to trigger, because if any one node in a cluster becomes activated it is likely that the others in the cluster will follow. The lattice topology has a clustering coefficient of 0, since there exist no triangular loops of edges in this type of topology.

Note that the following analyses apply only when all nodes use the same activation threshold. When activation thresholds are allowed to vary from one another, as in the ContagAlert protocol when the adaptive procedure is applied, even this type of qualitative analysis is intractable.

B. Circle Topology

The circle topology has a completely regular structure, the highest possible clustering coefficient without being fully connected, and clear activation behavior. Due to the high clustering coefficient, once the threshold of any single node is exceeded and the node becomes alerted, the activation will cascade to eventually encompass the entire ring. As a result, the probability of the entire network becoming active is equal to the probability of the first node becoming active, which is equal to the probability of the number of seed nodes appearing within range of a single inactive node exceeding the threshold of that node.

C. Random, Semicircle, and Affinity Topologies

Although the clustering coefficient of these topologies is not as high as the circle topology, it is higher than the lattice, exponential, or power law topologies. According to the clustering coefficient hypothesis, this suggests that the threshold region of these topologies should lie somewhere right of the circle, but left of the exponential and power law topologies. However, the power law topology is a special case, as described below.

D. Exponential Topology

Although the exponential topology is very regular, its directed edges only look in one direction around the ring. This results in a very low clustering coefficient, and therefore makes it difficult to generate cascading activations. As a result, we expect the threshold point for this topology to occur at a higher number of seeds than for other topologies.

E. Power Law Topology

On a network possessing a power law topology, an activation state propagates extremely easily when using the same fixed threshold level for all nodes. Due to their large number of neighbors, the few highly connected nodes are likely to reach the requisite number of active neighbors in order to trip their thresholds. The large number of poorly connected nodes are likely to have one or more of these highly connected, now active nodes as neighbors. As a result, the few highly connected nodes are able to rapidly activate most of the poorly connected nodes, and those that are left out quickly activate once the majority of the poorly connected nodes are activated. This easy propagation when using a fixed threshold for all nodes forces the power law network to employ the adaptive protocol in practice in order to ensure that the highly connected nodes do not activate too easily. This behavior can be seen in Figure 3, where the trend line for the power law topology appears on the far left of the plot.

VII. Application Data

We studied the behavior of the protocol in two applications to assess its effectiveness outside of an entirely abstract environment. The first application is an extension of the earlier simulation, and it models the spread of an Internet worm, such as Code Red, while the ContagAlert protocol disseminates warnings of the attack. The second application, called DoS-Alert, is built on top of the Pastry peer-to-peer object location service [36]. This application uses the ContagAlert protocol to inform peers of the identities of denial-of-service attackers so that requests from those malicious nodes can be suppressed.

A. Defending Against an Internet Worm Attack

Internet worms represent a significant threat to online resources in the present Internet environment. Not only do they harm individual systems, but they also overload the Internet infrastructure as they spread at an exponentially increasing rate. If a few nodes are able to detect the worm early on, they can spread alerts to other nodes about the imminent attack in order to protect those nodes, slow the spread of the worm, and reduce the impact on the Internet infrastructure. Due to the wide-ranging attack pattern of Internet worms, the number of worm observations should easily exceed the contagion threshold for the ContagAlert protocol.

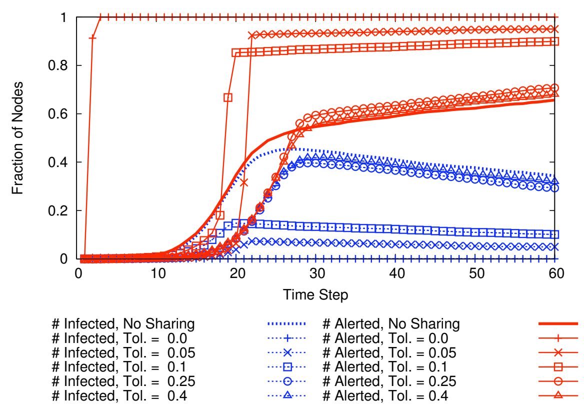

Fig. 9. Comparison of Alerted Nodes to Infected Nodes over Time – 4096 Nodes, Random Topology, Average Degree 64

We modified our existing simulator by replacing the fixed number of seeds with an increasing number of seeds modeling the spreading behavior of a worm.

The spreading behavior of the worm is based on the model presented by Kephart in [25]. This model is an example of a Susceptible-Infected-Removed infection model [26], [32]. By default, nodes are considered Susceptible to infection. On each time step, each infected node selects another node in the network at random to infect. If the target node is Susceptible, it becomes Infected with probability p, and it becomes Removed with probability $1-p$. If the node is already infected, there is no effect. If the target node is already Removed, it is immune to the infection. On each time step, infected nodes have a probability c of becoming Removed as administrators repair the infected systems and patch the vulnerability.

In our variant on this model, nodes become alerted to the attack when they are Removed by any means – either by successfully resisting infection, or by being repaired by an administrator. Alerted nodes spread warning information to other nodes using the ContagAlert protocol in an attempt to inhibit the further spread of the worm. Susceptible nodes which become alerted by ContagAlert become Removed. Infected nodes do not participate in the protocol.

This model allows the infected population to at most double on each step, if no infected or immune nodes are targeted and if all susceptible nodes become infected. When the simulation begins, the rate of spreading will increase for a certain period as more nodes become infected. However, at a certain point most of the network will be infected, reducing the number of susceptible nodes, and the rate of spreading will begin to slow.

The speed of the spread of an alert across the network is primarily dependent on the target tolerance of the network. If the tolerance is set higher, more seeds are required to tip the network past the contagion threshold and the propagation of warnings will be delayed. This allows the worm to attain a stronger foothold in the network before defensive measures are raised. If the target tolerance is set lower, the opposite behavior will occur, but it may open the possibility for disruptive messages to spread.

The simulator generated a network of a specified topology, size, and average degree. As with the earlier simulations, the network was adapted to a specific target tolerance with interference before testing its performance. After the adaptive period was completed, a simulated worm was released on the network. The initial infection was seeded at a single, randomly selected node. After the initial infection, we used an infection probability of $p = .5$ and a cure rate of $c = .01$. A trace of the network state was collected, recording how many nodes were in an infected state and how many nodes were in an alerted state at each time step.

Figure 9 depicts the number of infected nodes (dashed lines) and the number of alerted nodes (dotted lines) in the network at each time step (X axis) for a typical simulation run. Both the infected count and the immune count start at 0 at time $T = 0$. As time progresses in the simulation, both the infected and immune counts rise due to the probabilistic nature of the infections. As the immune count passes the target tolerance of the network, the ContagAlert protocol goes into action and the alert rate of nodes increases dramatically. However, if the target tolerance is too high, the worm is able to attain a solid foothold and the ContagAlert protocol cannot catch up. Eventually, all nodes become alerted as administrators repair the infected nodes.

The solid lines in the figure indicate the infected and immune counts in networks where worm alerts are not shared using a protocol like ContagAlert. Although nodes can still become alerted when an infection attempt fails or when repaired by an administrator, this information is not used to protect other nodes.

The figure shows that a low target tolerance of between .05 and .15 results in a dramatic reduction in the number of nodes that become infected by the worm. If the target tolerance is much higher, however, it is of limited effectiveness when trying to compete against a rapidly spreading signal like a worm. It is important to note, however, that the worm simulated in this scenario represents an extremely high-speed attack, and ContagAlert will be even more effective against slower attacks.

For some target tolerances, the sharp rise in infections occurs later than for other target tolerances. This is a side effect of the probabilistic infection model. If the earliest infection attempts by the worm fail, it can significantly delay when the worm attains its maximum infection rate.

B. Defending a P2P Network Against a DoS Attack

The ContagAlert protocol is also suitable for suppressing requests from malicious nodes during a denial-of-service (DoS) attack on a peer-to-peer (P2P) network. In such an attack, malicious nodes will attempt to overwhelm victim nodes by exhausting their resources with useless requests. Most research on DoS attacks has been at the network layer on how to prevent networks from being flooded with useless network-layer packets, rather than preventing P2P applications from being flooded with useless application-layer requests [22], [23], [31], [35]. However, Daswani et al. identified availability in P2P systems as one of the major open problems in P2P systems research [8].

We implemented a P2P application, called DoSAlert, as a testbed for the ContagAlert protocol. The goal of DoSAlert is for benign nodes to collectively suppress requests from an attacker once they have identified that attacker. Once a benign node becomes aware of an attacker, that node should inform its neighbors of the attacker's identity. However, DoSAlert must also avoid the situation where attackers collude to cause benign nodes to mistakenly suppress legitimate requests due to false accusations.

The DoSAlert application presented in this section is a distributed approach to minimizing the number of malicious requests processed during a DoS attack on a P2P system. Each peer monitors for attack patterns in the traffic passing through its node. If such an attack is detected, the node begins suppressing messages and propagates an alarm identifying the attacker to other nodes in the system.

If a node detects that a target destination is being overwhelmed by a particular source, then that node will start suppressing messages from that source and will propagate an alarm signal to other nodes identifying that source as an attacker. Also, if a node receives enough distinct alarm signals identifying a particular source as an attacker, then that node will begin suppressing messages from that attacker and will propagate the alarm to all of its neighboring nodes.

It is observed in [24] that attackers have constant per-client request rates during a DoS attack while legitimate clients have per-client request rates that tend to get lower during a flash event. This difference between legitimate and malicious clients allows us to distinguish between high levels of traffic due to legitimate events (e.g., flash crowds) and malicious events (e.g., DoS attacks). This observed legitimate client behavior during a flash event is possibly due to increased server processing time and transmission delay caused by network congestion. Legitimate clients will be responsive to congestion and slow down their requests, while malicious clients will continue to inject requests at a regular rate because their goal is to overwhelm the target.

DoSAlert leverages this observation to distinguish between legitimate requests and malicious attacks. Benign nodes enforce exponentially increasing inter-arrival request times between each source-destination pair whose messages pass through that node.

We evaluated the effectiveness of the ContagAlert protocol in this environment with a trace-based simulation. Using the FreePastry API, we implemented DoSAlert on top of the Pastry P2P routing substrate [13], [36]. The application underlying the DoS detection in the simulation was a peer-to-peer cooperative Web caching scheme where each peer stores and requests Web objects.

Each benign node makes requests based on a randomly chosen trace for a single machine from the BU Web Client Traces [7]. For each request appearing in the trace, the request object identifier used in the simulation is obtained by computing a hash of the URL appearing in the request.

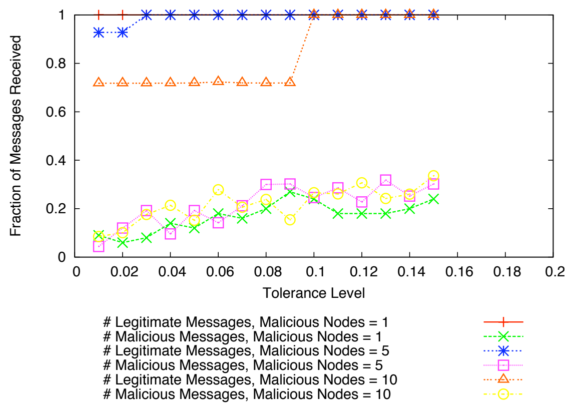

Fig. 10. Fraction of Messages Received in Pastry Application

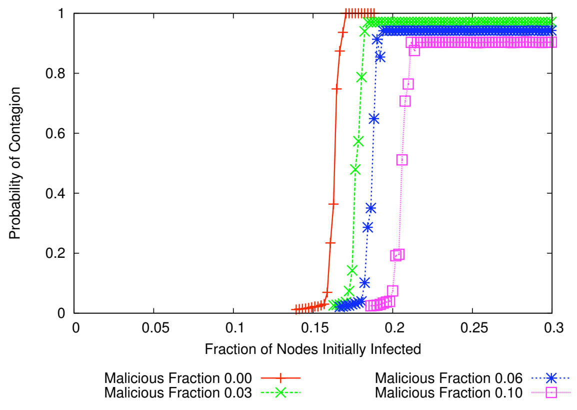

Fig. 11. Effects of Malicious Interference on ContagAlert Performance – Driving the Contagion Threshold Higher

Malicious nodes randomly choose twenty different targets and bombard them with useless requests at a constant rate. Malicious nodes also generate and inject false alarm signals identifying benign nodes as attackers into the system. More specifically, the malicious nodes collude to select a set of benign nodes and inject false alarm signals identifying each node in this set as malicious. The number of benign nodes in this set is equal to the number of malicious nodes in that particular simulation run.

In the simulation, each attacker sent 10 requests that are spaced 200 milliseconds apart to each one of its twenty targets. The minimum initial wait time enforced at benign nodes that monitor traffic between source-destination pairs is 1000 milliseconds. Every time a request arrives before its wait time, a violation occurs and the wait time is doubled. If three violations occur for a source-destination pair, then the source is considered an attacker and the detecting node sends an alarm identifying the attacker.

There are two important metrics to consider when evaluating the ContagAlert protocol in this scenario: the number of false positive alerts, and the number of false negative alerts. False positives can be measured as the effect of the target tolerance level on the percentage of legitimate messages that are delivered. False negatives can be measured as the effect of the target tolerance level on the percentage of malicious messages that are delivered.

Ideally, the delivery rate of legitimate messages should be high, while the delivery rate of malicious messages should be low. Legitimate messages might be blocked if malicious nodes are successfully able to spread misinformation about the intention of nodes that are actually benign. Malicious messages might be allowed through if benign nodes fail to heed legitimate alert messages generated by other nodes. It should be noted that a small number of malicious messages are guaranteed to be successfully delivered, since several messages must be observed before a message source is identified as malicious.

Figure 10 shows that lower tolerance levels often lead to malicious requests being suppressed more aggressively. This desirable behavior is a result of the lower threshold required for benign nodes to become aware of attackers and start suppressing attacker requests. However, these lower thresholds also make it easier for attackers to inject false alarm signals into the system that identify benign nodes as attackers. This is an undesirable behavior and causes some legitimate messages to be suppressed in the above table. However, in the above results, when the percentage of malicious nodes is less than or equal to the tolerance level, no legitimate messages are dropped (i.e., false alarms from attackers are completely ineffective) and over 60% of the attackers' requests are suppressed.

C. Malicious Interference

Although the ContagAlert protocol is effective under normal operating conditions, it is important to consider the impact on performance of malicious nodes that refuse to adhere to the rules of the protocol. These malicious nodes might attempt to disrupt normal operation in order to facilitate the easier propagation of maliciously generated alerts, or they might attempt to suppress the spread of legitimate alerts. To accomplish these goals, nodes can interfere with the basic aspect or the adaptive aspect of the protocol.

One example of a possible attack is as follows: Malicious nodes attempt to drive up the activation thresholds of benign nodes during the adaptive rounds. A malicious node can accomplish this by generating a practice alert on every adaptive round with a 100% probability, instead of using probability equal to the target tolerance of the network. The higher than expected fraction of seed signals will result in the network adapting to a higher contagion threshold than it intends. Then, in the basic aspect of the protocol, the malicious nodes refuse to relay real alert signals. This results in a lower propagation rate of the real alerts, driving the effective contagion threshold of the network even higher. The end result is that real attacks will be allowed to proceed for a longer period of time before the network becomes alerted to its presence, increasing the amount of damage that it will be able to inflict.

The opposite attack could also be executed. Malicious nodes drive down the activation thresholds of benign nodes during adaptive rounds by refusing to send any practice alert messages. After the network stabilizes at an abnormally low

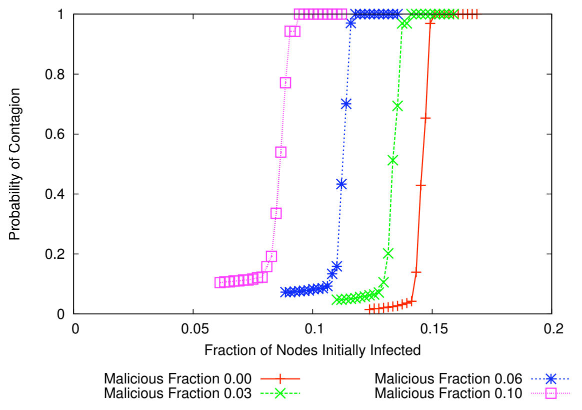

Fig. 12. Effects of Malicious Interference on ContagAlert Performance – Driving the Contagion Threshold Lower

contagion threshold, malicious nodes collude to generate false alerts in an attempt to cause benign nodes to take misguided defensive measures. If the adaptive phase of the attack is sufficiently effective, the small population of malicious nodes might be effective at causing the false alerts to propagate.

This attack could be countered if benign nodes record the sources from which they receive practice messages. This serves as an indication of which nodes are participating fully in both aspects of the protocol. Then, during the propagation of real alerts, alert messages are only heeded if they are received from one of these fully participating nodes. With this defense in place, malicious nodes cannot abstain from the practice rounds entirely. In the test simulation modeling this attack, malicious nodes participated in each practice round with a probability of .1

Figures 11 and 12 present data showing the behavior of the ContagAlert protocol when it is subjected to these two attacks. The target tolerance in these simulations was set to 0.15 on a network of 4096 nodes with a random topology. In both cases, the adaptive aspect of the protocol was able to closely match this target when no malicious nodes were present. As the fraction of malicious nodes increases, the contagion threshold diverges further from the target, as is expected given the nature of the attacks. However, even when 10% of the nodes are actively working to interfere with the protocol, the contagion threshold of the network is deflected by by no more than .07 from the target tolerance for the first type of attack, or by .05 for the second type of attack. Although this shift is noticeable, it is relatively small given the amount of resources an adversary would need to organize in order to launch these types of attack.

A variety of other methods might also be used to subvert the functioning of the protocol. A comprehensive study of these attacks, there effects on the ContagAlert protocol, and possible defenses that could be used to block them is beyond the scope of this paper. However, the following list enumerates several possible approaches that might be used to generate interference.

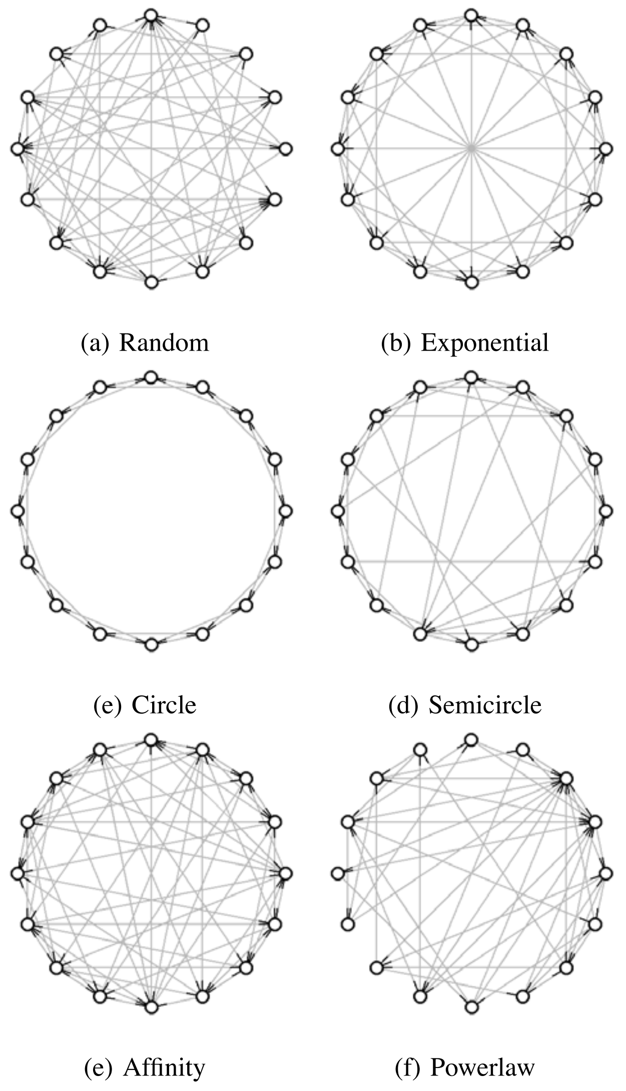

Fig. 13. Example Network Topologies – 16 Nodes with Average Degree 4

- Malicious nodes do not follow the basic aspect of the protocol. Alerts may be blocked, always forwarded, or forwarded at random with some probability.
- Malicious nodes do not adhere to the adaptive aspect of the protocol. Practice messages may be blocked, always forwarded, or forwarded at random with some probability.
- Malicious nodes do not obey to the network topology. Messages are broadcast to a wider population of nodes than they should normally be able to reach.
- Malicious nodes choose their position in the network topology to maximize their influence over the network or to reduce the influence of benign nodes.

VIII. Conclusions and Future Work

We introduced the ContagAlert protocol, a new protocol oriented towards disseminating messages across a peer-to-peer network. The protocol leverages the threshold behavior studied in contagion theory to suppress messages of malicious intent while causing legitimate messages to spread. The protocol is able to adaptively set the contagion threshold of the network to specify a target tolerance level for malicious messages. This is accomplished by adjusting the activation threshold of individual nodes using an adaptive distributed algorithm.

We showed that the protocol disseminates information sufficiently quickly that it is useful when attempting to thwart rapidly spreading Internet worms. Additionally, we demonstrated that the protocol is able to distinguish between legitimate and false alerts in a P2P DoS detection application.

We believe that communicating alerts in Grid computing environments may be an ideal application for the ContagAlert protocol. Grid resources are typically high-value, and are therefore tempting targets for adversaries and in need of protection. Additionally, the low churn rate of Grid environments would be easily accommodated by the adaptive aspect of the protocol, allowing ContagAlert to operate at the target target tolerance level without interference. Finally, the reputable nature of Grid resources would reduce the probability that nodes in the network would attempt to subvert the proper functioning of the protocol in ways that will significantly undermine its performance.

Acknowledgments

This work was supported in part by NSF ITR grant CMS-0427089 and in part by NSF CAREER grant CNS-0448246.

References

[1] A. Abdul-Rahman and S. Hailes. A distributed trust model. In *Proceedings of the 1997 workshop on New security paradigms*, pages 48–60. ACM Press, 1997.

[2] J. Adler, D. Stauffer, and A. Aharony. Comparison of bootstrap percolation models. *Journal of Physics A*, 22:L297–L301, 1989.

[3] R. Cerf and F. Manzo. The threshold regime of finite volume bootstrap percolation. In *Mathematical Physics Preprint Archive*, July 2001.

[4] J. Chalupa, G. R. Reich, and P. L. Leath. Bootstrap percolation on a Bethe lattice. *J. Phys. C*, 12(1):L31–L35, 1979.

[5] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman. PlanetLab: An Overlay Testbed for Broad-Coverage Services. *ACM SIGCOMM Computer Communication Review*, 33(3), July 2003.

[6] Breach that hit cisco wider than thought. *CNN/Money*, May 2005.

[7] C. Cunha, A. Bestavros, and M. Crovella. Characteristics of www client-based traces. Technical Report TR-95- 010, Boston University Department of Computer Science, 1995.

[8] N. Daswani, H. Garcia-Molina, , and B. Yang. Open problems in data-sharing peer-to-peer systems. In *9th International Conference on Database Theory*, 2003.

[9] D. Dhillon, T. S. Randhawa, M. Wang, and L. Lamont. Implementing a fully distributed certificate authority in an OSLR MANET. In *Wireless Communications and Networking Conference (WCNC)*, 2004.

[10] D. M. *et al.* The spread of the Sapphire/Slammer worm. http://www.caida.org/outreach/papers/2003/sapphire/, 2003.

[11] J. S. B. *et al.* An architecture for intrusion detection using autonomous agents. In *ACSAC*, pages 13–24, 1998.

[12] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the Grid: Enabling scalable virtual organizations. *International Journal of High Performance Computing Applications*, 2001.

[13] FreePastry. http://www.cs.rice.edu/CS/Systems/Pastry/FreePastry/.

[14] D. Frincke, D. Tobin, J. McConnell, J. Marconi, and D. Polla. A framework for cooperative intrusion detection. In *21st National Information Systems Security Conference*, pages 361–373, October 1998.

[15] S. Frolund, A. Merchant, Y. Saito, S. Spence, and A. Veitch. FAB: Enterprise storage systems on a shoestring. In *9th Workshop on Hot Topics in Operating Systems (HOTOS)*, June 2003.

[16] M. Gladwell. *The Tipping Point: How Little Things Can Make a Big Difference*. Little, Brown, and Company, 1999.

[17] I. Gupta, K. Birman, P. Linga, A. Demers, and R. van Renesse. Kelips: Building an efficient and stable P2P DHT through increased memory and background overhead. In *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03)*, 2003.

[18] A. E. Holroyd. Sharp metastability threshold for two-dimensional bootstrap percolation. *Probability Theory and Related Fields*, 125(2):195–224, 2003.

[19] R. Hunt. PKI and digital certification infrastructure. In *Networks*, 2001.

[20] Collective intelligent bricks. `http://www.almaden.ibm.com/StorageSystems/autonomic_storage/CIB/index.s%html`, August 2003.

[21] The International PGP Homepage. www.pgpi.org.

[22] J. Ioannidis and S. Bellovin. Implementing pushback: Router-based defense against DDoS attacks. In *Network and Distributed Systems Security Symposium*, 2002.

[23] C. Jin, H. Wang, and K. Shin. Hop-count filtering: An effective defense against spoofed DDoS traffic. In *10th ACM Conference on Computer and Communications Security*, 2003.

[24] J. Jung, B. Krishnamurthy, and M. Rabinovich. Flash crowds and denial-of-service attacks: Characterization and implications for CDNs web sites. In *11th International World Wide Web Conference*, 2002.

[25] J. O. Kephart and S. R. White. Directed-graph epidemiological models of computer viruses. In *IEEE Computer Society Symposium on Research in Security and Privacy*, 1991.

[26] W. O. Kermack and A. G. McKendrick. A contribution to the mathematical theory of epidemics. In *Proceedings of the Royal Society of London*, 1927.

[27] B. Krebs. Hackers strike advanced computing networks. *Washington Post*, April 2004.

[28] A. D. R. Marc Waldman and L. F. Cranor. Publius: A robust, tamper-evident, censorship-resistant, web publishing system. In *Proc. 9th USENIX Security Symposium*, pages 59–72, August 2000.

[29] M. Mitzenmacher. A brief history of generative models for power law and lognormal distributions. *Internet Mathematics*, 1(2):226–251, 2001.

[30] D. Moore, C. Shannon, and J. Brown. Code-red: A case study on the spread and victims of an Internet worm. In *Proceedings of the Internet Measurement Workshop (IMW)*, 2002.

[31] D. Moore, G. Voelker, and S. Savage. Inferring Internet denial-of-service activity. In *10th USENIX Security Symposium*, 2001.

[32] J. D. Murray. *Mathematical Biology*. Springer, third edition, 2002.

[33] D. Nojiri, J. Rowe, and K. Levitt. Cooperative response strategies for large scale attack mitigation. In *DARPA Information Survivability Conference and Exposition*, pages 293–302, 2003.

[34] J. S. Park and R. Sandhu. Binding identities and attributes using digitally signed certificates. In *Computer Security Applications (ACSAC)*, 2000.

[35] K. Park and H. Lee. On the effectiveness of route-based packet filtering for distributed DoS attack prevention in power-law internets. In *ACM SIGCOMM*, 2001.

[36] A. Rowstron and P. Druschel. Pastry: scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Middleware*, 2001.

[37] B. Sniffen. Trust economies in the Free Haven project, 2000.

[38] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for Internet applications. In *Proceedings of the 2001 ACM SIGCOMM Conference*, pages 149–160, 2001.

[39] V. Vlachos, S. Androutsellis-Theotokis, and D. Spinellis. Security applications of peer-to-peer networks. *Comput. Networks*, 45(2):195–205, 2004.

[40] D. J. Watts. A simple model of global cascades on random networks. *Proceedings of the National Academy of Sciences (PNAS)*, 99(9), 2002.

[41] G. White, E. Fisch, and U. Pooch. Cooperating security managers: A peer-based intrusion detection system. *IEEE Network*, 10(1), 1994.

[42] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. D. Kubiatowicz. Tapestry: A global-scale overlay for rapid service deployment. *IEEE Journal on Selected Areas in Communications*, 2003. Special Issue on Service Overlay Networks.

[43] L. Zhou, F. B. Schneider, and R. V. Renesse. Coca: A secure distributed online certification authority. *ACM Trans. Comput. Syst.*, 20(4), 2002.