Towards Robust Overlay Networks: Enhancing Adaptivity Mechanisms with Byzantine-Resilience

AAron Walters Kevin Bauer Cristina Nita-Rotaru
Department of Computer Science and CERIAS, Purdue University
250 N. University St., West Lafayette, IN 47907 USA
{arwalter,ksbauer,crisn}@cs.purdue.edu

Abstract— **Adaptive measurement-based overlay networks offer increased performance and resilience to benign failures for end-to-end communication by using aggressive adaptivity mechanisms. These mechanisms dynamically optimize application-centric metrics such as latency, jitter, bandwidth, and loss rate. However, end-systems are more vulnerable than core routers, making overlay networks susceptible to malicious attacks coming from untrusted outsiders, and especially from trusted (but compromised) members of the overlay. Unlike outsider attacks, insider (or Byzantine) attacks can not be prevented by simply deploying cryptographic authentication mechanisms. In this work, we identify and classify insider attacks against adaptivity mechanisms in overlay networks and demonstrate several of them against the ESM/Narada multicast overlay system. The attacks target the overlay network construction, maintenance, and availability and allow malicious nodes to control significant traffic in the network, facilitating further attacks such as selective forwarding and traffic analysis. We believe this work is the first to classify insider attacks against adaptivity mechanisms in distributed systems and the first to propose techniques to enhance the adaptivity mechanisms with Byzantine-resilience. We demonstrate the effectiveness of the newly proposed techniques through real-life deployments and emulations conducted on the PlanetLab and DETER testbeds, respectively.**

Keywords: Overlay Networks, Security, Insider Attacks, Byzantine-Resilience, Adaptivity

Technical areas: (1) Security; (2) Peer-to-peer; (3) Operating systems and middleware

Contact author: Cristina Nita-Rotaru, crisn@cs.purdue.edu

I. Introduction

Numerous collaborative Internet applications, such as conferencing and video broadcasting, have benefited tremendously from multicast services. Multicast overlay networks were proposed as a viable application level multicast architecture to overcome the scarcity of native IP multicast deployments. Examples of multicast overlay networks include ESM/Narada [1], Nice [2], ALMI [3], and Overcast [4]. In these types of networks, buffering and relaying functionality is moving from core routers toward end-systems. Efficient dissemination structures, such as trees and meshes, allow for reduced overhead as well as increased throughput and reliability. Furthermore, many overlay networks utilize adaptivity mechanisms to increase performance and provide fault tolerance for end-to-end communication. These mechanisms seek to dynamically optimize application-centric metrics such as latency, jitter, bandwidth, and loss rate. Such metrics are typically collected by overlay nodes through passive observation of their performance from the source (primary metrics) and through periodic probing of peer nodes about their performance from the source (secondary metrics). The resulting performance of the overlay network depends on the accurate interpretation of performance observations, as well as the correctness of the responses received from probed nodes.

While pushing functionality to end-systems allows overlay networks to achieve better scalability, it also makes them more vulnerable. In such systems, trust is often pushed to the fringes of the Internet, where end-nodes are more likely to be compromised than core routers [5]. As a result, end-system overlay networks are more vulnerable to malicious outsider attacks, as well as to insider attacks coming from (potentially colluding) attackers that infiltrate the overlay or compromise member nodes. In particular, attacks that target the overlay construction and maintenance can be extremely dangerous since they can allow an attacker to control a significant part of the traffic. This can be used as a mechanism to further facilitate other attacks such as selective data forwarding, cheating, traffic analysis, and attacks against availability (i.e., partitioning). Some of the resulting attacks, such as selective forwarding, may ultimately be noticed by the victim and a posteriori detection mechanisms can be deployed. However, other attacks, such as traffic analysis, do not have immediately observable results. It is therefore critical to address the primary attacks that allow the attacker to obtain advantageous positions in the overlay structure.

This work constitutes the first effort to identify and analyze a set of advanced threats against adaptive overlay networks. Unlike previous work [6], [7], our work considers the effects of insider adversaries, also referred to as Byzantine attackers. Current adaptivity mechanisms lack Byzantine-resilience and assume that the information reported by probed nodes is always correct. Furthermore, such mechanisms fail to take into account the effects of Byzantine attackers on their surrounding environment. Attackers in close proximity to a given node may influence the network in order to manipulate the metrics collected and the subsequent decisions made by that node.

In this paper, we provide an introduction to Byzantine attacks against standard adaptivity mechanisms in overlay networks, an initial analysis of how such attacks can be mitigated and prevented throughout the life-cycle of the overlay, and an in-depth solution to a critical aspect of the problem: preventing poor adaptation decisions in networks influenced by attackers. Our solution lies in performing spatial and temporal outlier analysis on primary (measured) and secondary (probed) metrics to allow an honest node to make better use of available information before making an adaptation decision. Furthermore, we demonstrate the effects of the identified attacks on an advanced adaptive overlay network operating over real Internet infrastructure. Finally, we experimentally show the usefulness of our outlier detection technique for preventing bad decisions in the face of Byzantine attackers. We summarize our key contributions:

- We provide a characterization of the types of mechanisms currently used to achieve adaptivity in overlay networks and identify attacks against these mechanisms. We refer to these attacks, which target overlay construction, maintenance, and stability, as *attraction*, *repulsion*, and *disruption*.
- We demonstrate the effectiveness of the above identified attacks against the well-known adaptive multicast system, ESM [1], and its Narada multicast protocol. Our experiments, which were conducted using both real-life deployments and emulations, demonstrate that, although ESM employs an advanced set of adaptivity mechanisms, it is unable to mitigate the attacks posed by a malicious adversary.
- We provide an analysis of the solution space for mitigating Byzantine attacks that exploit adaptivity: preventing unnecessary or unnatural adaptations, increasing stability by incorporating metrics that reflect stability into the decision process, detecting malicious behavior that results in observable degradation of service, and reacting to the detected malicious nodes.
- We focus on providing a solution for what we believe is the most critical and challenging problem: preventing bad adaptations. We propose techniques to reduce incorrect and unnecessary adaptations by using spatial and temporal correlations to perform context-sensitive outlier analysis. A key component of our solution is based on the observation that several estimated metrics are dependent variables and the overlay and multicast tree logical networks share overlapping physical links.
- We demonstrate the effectiveness of our defense mechanisms in the context of the ESM system through experiments conducted on the PlanetLab [8] and DETER [9] testbeds.

Roadmap: The rest of the paper is organized as follows. We provide a survey of adaptivity mechanisms employed by overlay networks and classify attacks against them in Section II. We demonstrate several attacks against the adaptivity mechanisms employed by ESM in Section III and propose defense mechanisms in Section IV. We overview related work in Section V. We present our conclusions in Section VI.

II. Attacks Exploiting Adaptivity Mechanisms

The benefits of employing adaptivity to address intermittent failures and degraded performance associated with dynamic network conditions have been recognized since the earliest days of the ARPANET [10]. Adaptivity mechanisms based on measurements of network characteristics have been used more recently in the design of overlay networks [1], [2], [4], wireless networks [11] and sensor networks [12]. However, these mechanisms can not overcome attacks performed by insider adversaries. As the benefits of adaptive protocols are evident, what is needed is to enhance these mechanisms with resilience to Byzantine attacks. As a first step towards this goal we first overview the main techniques that are used to augment adaptivity and improve resiliency in overlay networks. We then provide a characterization of insider attacks that exploit adaptivity mechanisms and affect the overlay's construction, maintenance, and availability.

A. Adaptivity Mechanisms

An adaptivity mechanism allows a network protocol or system to adjust to network environmental conditions with the goal of improving its performance and availability. The performance can be defined by application-centric metrics such as latency, jitter, bandwidth, and loss rate. The adaptation process consists of collecting information about the network conditions and making a decision to change, or *adapt*, based on some decision criteria. We now explore two areas, data quality and decision quality, in which protocol designers have typically developed mechanisms to improve adaptivity.

1) Data Quality: A critical factor for the effectiveness of an adaptivity mechanism is the quality of the data observation and estimation, as well as the ability of the metrics used to accurately reflect the state of the network environment. An additional factor can be scaling down the data during processing in order to decrease the associated overhead. Examples of factors that influence data quality are data freshness, variability and the presence of noise. Mechanisms related to these issues are data sampling, data smoothing, metric construction, and data summarization, and aggregation.

Data sampling. Frequent sampling or probing, also referred as *data sampling*, is a method used to prevent staleness of information and subsequent oscillations. It is desirable to have a high frequency of sampling to decrease mispredictions, as it was shown in the design of RON [13]. However, frequent sampling can introduce a significant overhead in the network.

Data smoothing. Measured variables often exhibit a large amount of variance or errors, especially in discrete measurements of continuous valued functions. Data smoothing is a technique used to reduce and eliminate the noise and variability in the samples of physical state variables. The method is also often used to reduce the effects of erratic changes in the measured values and results in effective filtering. For example, in [14] a smoothed Round Trip Time (RTT) is used as the metric for the cost of the overlay link to prevent wild oscillations in measurements.

Metric construction. Another technique commonly used to address the instabilities in measured data is metric construction. New metrics are constructed from existing metrics to improve the estimation process. These methods can be used to elucidate relationships between metrics useful for estimation that may not have been discernible when considering the metrics in isolation. An excellent example of the importance of metric construction can be found in [15], where a new metric is constructed by combining the latency as perceived by a neighbor node with the RTT to the neighbor. The newly constructed metric is then coupled with the bandwidth measurement. The authors demonstrate a substantial improvement over the approaches that considered both latency and bandwidth in isolation.

Summarization and aggregation. Summarization and aggregation are two techniques that are employed to provide high scalability and improve stability. These techniques are used by BGP [16], allowing it to scale by limiting the amount of information that is considered by the large number of routers. The obtained stability is at the expense of being less aggressive in the path exploration and maintenance, which sometimes inhibits BGP's ability to recover from faults [17].

2) Decision Quality: The response taken by an adaptivity mechanism has an associated cost that must be weighed against the degree of benefit that could occur as the result of the adaptation. Under some network conditions such decisions could lead to instabilities [18], [19], [20], such as oscillatory behavior commonly referred to as *flapping*, where nodes rapidly switch between seemingly equal alternatives. New techniques were deployed to mitigate these phenomena and provide a tradeoff between responsiveness to changes and instabilities. Examples of such techniques are damping and hysteresis. Below we provide more details about techniques related to the decision process.

Utility discretization. Utility measures are thresholded relative measurements used to quantify the potential usefulness of making a particular change. They are usually tied to the optimization strategy employed by the estimation algorithm used when taking a response. Because often times the numeric metrics that drive these utility functions have large variances and frequent updates, discretization is used to reduce the values of continued metrics by dividing the range of metric values into intervals. Utility discretization is used when the change results in substantial improvements over a threshold. For example, in [20] a routing change is only made from the current path when the improvement in bandwidth is above some threshold. Other examples include RON, where routes are changed to obtain at least a 50% improvement in throughput [17], and GoCast [21], which avoids "futile minor adaptations" by requiring new neighbors to have a 50% improvement in RTT.

Randomization. A common technique used to address deterministic and symmetric characteristics of distributed systems and network protocols is randomization. The technique is used to offer another dimension of variability to reduce the predictable behavior of network protocols. For example, randomization has been used to reduce the likelihood of a "thundering herd" effect where many nodes attempt to switch to the same link [22]. Another example is using randomization for path selection [20] to reduce the loss rates.

Damping. Frequent changes due to ephemeral failures can cause significant instability in systems. Damping was proposed as a stabilizing technique to reduce unwanted or excessive responses to network conditions and limit the propagation of unstable information. Damping creates a temporal diminution of metric qualities characteristic for generating oscillatory behavior. The benefits of the technique were demonstrated in the context of BGP. For example, damping has been used within BGP to suppress route changes caused by link flapping by distinguishing consistently unstable routes from routes that experience ephemeral failures [19]. Another example can be found in [20], where it is demonstrated that increasing the amount of time between route change decisions can improve the accuracy of measurements and inhibit the frequency of change at the expense of reactivity.

Hysteresis. Hysteresis is a technique commonly used in protocols to add a slowing effect to changes. This is achieved by adding a history dependence to the system where previous readings can influence the estimation effects of subsequent readings. Many protocols deploy different types of mechanisms to introduce hysteresis to the system. For example, RON demonstrated that smoothing was not enough to avoid flapping and thus it used smoothing hysteresis to avoid flapping between measurably equal routes [17]. Hysteresis was introduced by giving a bonus to the "last good" route. Hysteresis is coupled with the aforementioned randomization in Tapestry to further reduce the likelihood of a "thundering herd" effect [22] .

3) Summary: A brief analysis of the above techniques indicates that the data quality phase depends on the accurate interpretation of performance observations, as well as the correctness of the responses received from probed nodes. None of the mechanisms described above take into account the correctness of the responses received from probed nodes or the effect of Byzantine attackers on their surrounding environment.

B. Attacks Classification

In this section we present three classes of attacks that an adversary may seek to carry out on an adaptive overlay network. We first describe our adversary model and then present the attacks, which we refer to as *attraction*, *repulsion*, and *disruption*.

Attack model. We consider the model of an insider adversary. An insider adversary has access to the same data as any legitimate user, including cryptographic keys stored at a node. This access can be the result of the adversary bypassing the authentication mechanisms or compromising an overlay node through other means. In this case, system participants cannot be completely trusted although they are authenticated, and authentication mechanisms are no longer enough to protect the system. An insider adversary can have an arbitrary (or

Byzantine) behavior: it can send misleading information when probed, send inconsistent information to different nodes in the overlay, refuse to participate in the process of forwarding data traffic by selectively or entirely dropping data, or store the data traffic it has forwarded through it to perform traffic analysis.

Below we assume that data authentication and integrity mechanisms are deployed and we focus only on attacks directed at the adaptivity mechanisms.

The main asset in any communication network is its data. Thus, one of the main goals of an insider adversary is to control as much of the data disseminated in the overlay as possible. This can be achieved if the adversary, having infiltrated the overlay, manages to manipulate the path selection or the multicast structure maintenance to its advantage. Based on their effect on the control of path selection, we classify these attacks as *attraction attacks*, *repulsion attacks*, and *disruption attacks*. Any of these attacks can be conducted by an adversary by affecting the observed and collected metrics as follows:

- the malicious node lies about the observation space; In this case, the adversary manipulates the secondary sources of information used by the adaptivity mechanism.
- the malicious node imposes an artificial influence toward the observation space; In this case, the adversary manipulates the primary sources of information measured by a node.

Attraction attacks. Attraction attacks are a form of "bait-and-switch," where observed data is manipulated by a malicious node in order to draw attention. In such attacks, the malicious nodes are always presenting the network conditions to be better than they are, with the goal of gaining control over significant traffic. The attack can also target one particular node, in which case the attacker will persuade the victim to attach to a malicious parent in the dissemination structure. For example, if the dissemination structure is a tree, the goals of the attacker can be to attract many nodes to itself as children or to obtain a higher position in the tree. The final goal of the attack can be manipulating data, performing traffic analysis, man-in-the-middle attacks, causing disruption for specific nodes by isolating them, or selectively dropping packets for a particular destination.

A basic way to perform the attack is for a node to falsify the answers to probe requests to deceptively create the perception of a route with higher utility from the perspective of the victim node. The attacker can exploit characteristics, such as low frequency in the data sampling, to prolong the effects of his malicious action on the data smoothing mechanism. As a result, the utility function will create an incorrect adaption since the utility gain does not reflect reality. For example, if the utility function is based on the bandwidth from the source, a malicious node can attract other nodes in the tree by lying about its bandwidth from the source every time it is probed. The utility function will incorrectly choose to adapt and choose the malicious node since it appears that the change will guarantee a better bandwidth from the source.

Repulsion attacks. Repulsion attacks seek to reduce the attractiveness of other nodes or misrepresent the insufficiency of their own abilities. This is accomplished by means of lying and defamation in responses to active probes or by manipulating the physical or logical infrastructure in a way that creates the perception of routes with lower utility. As in the case of attraction attacks, repulsion attacks can target one particular node. The ultimate purpose of such attacks is to offer the malicious node opportunities for free-loading, traffic pattern manipulation, augmenting attraction attacks, or to just cause disruption.

One way a malicious node can conduct the attack is by lying about its performance. An example of such an attack is when a malicious node lies about route costs (i.e., hop count) in order to convince other nodes that it has a bad connection and thus should not be selected as a parent. The malicious node will then obtain a reduced burden. A particular attack that falls into this category was analyzed through simulation in [23]. The authors showed that selfish nodes (i.e., nodes that want to obtain advantage over other nodes but do not have destructive goals as malicious nodes) can selfishly improve their performance by manipulating distance measurements.

An attacker may instead choose to manipulate the existing environment, rather than lie about it, by exerting an influence of aversion toward the partially observable link state estimation. This constitutes an attack against primary sources of information – metrics that are directly observed by the node. One example of the attack was demonstrated in [13], where a flooding attack was conducted to demonstrate how quickly the overlay could respond. However, in this case, the attack was executed by an external attacker. An example of an internal repulsion attack was presented in [23]. In this case, a selfish node delayed its probe responses to affect the RTT measured by the node performing the probing. In both cases, the attackers manipulate the observation space of the victim to make things seem worse than they actually are.

Disruption attacks. Disruption attacks target the availability of the network by using the adaptivity mechanisms to turn the system against itself. An attacker can create significant disruption in the overlay by injecting or influencing the observation space metric data to generate self-destructive responses as a result of unnecessary adaptations. The ultimate goal of such attacks is to affect the infrastructure that supports the overlay with the intent to prevent or degrade service. These attacks can be classified as a form of denial of service (DOS) and can result in jitter, flapping, or partitioning the overlay.

An example from this class of attacks is the attack against TCP [6], which is an attempt to deny bandwidth to TCP flows by manipulating the observation space to create the perception of network congestion. The magnitude of the disruption is generally based on obtaining a steady-state [24] and a temporal measure of how long it takes to reach this state. The work in [7] generalizes the work presented in [6], as a form of low-rate reduction of quality (ROQ) attack that focuses on attacking adaptive control loops that drive resource allocation and affect the perceived service of a system.

Summary: We classified and described several attacks that exploit the data quality aspect of adaptivity mechanisms. Next,

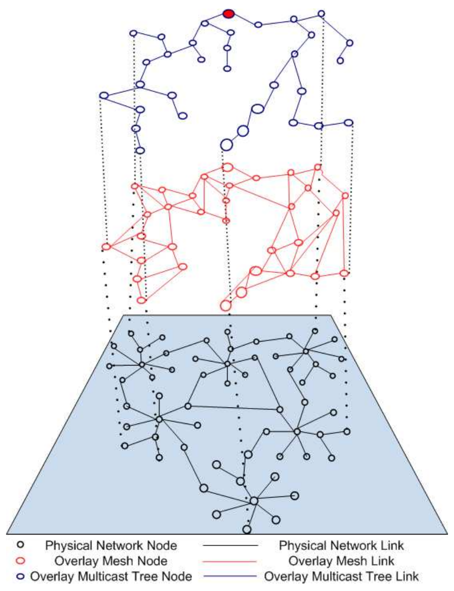


Fig. 1. Illustration of the physical network and the two logical networks: the overlay network and the multicast tree.

we examine the practicality of these attacks in a real-world network deployment. In addition, we study the degree to which mechanisms used within the adaptation decision are able to detect and recover from the attacks we identified.

III. An Example: Attacks against ESM and Narada

In this section we demonstrate through experimental results how the attacks identified above can be used against the ESM overlay system and the Narada multicast protocol. We first provide an overview of ESM, Narada, and the adaptivity mechanisms they use. We selected ESM to demonstrate the attacks because of its maturity, extensive deployment, and particularly because of the advanced set of techniques it uses for augmenting its adaptivity. We demonstrate that, even though ESM employs almost all of the mechanisms discussed in Section II-A, it is unable to mitigate the attacks posed by a malicious insider adversary.

A. Overview of ESM and Narada

ESM [1] is an overlay multicast system that shifts the multicast infrastructure to the end systems and forms a peer-to-peer overlay network. The system is largely used for broadcasting live events including academic conferences such as SIGCOMM and INFOCOM. ESM uses an application level multicast protocol, Narada, that builds an overlay tree for distributing multicast content, as seen in Figure 1.

A key component of Narada is the use of adaptivity mechanisms to dynamically change the multicast tree to improve application performance or maintain it when network conditions change. More specifically, this adaptivity serves to improve suboptimal overlay meshes that can occur as a result of random initial neighbor selection, aggressive partition repair, multicast group membership changes, and the transient nature of underlying physical network conditions.

Narada employs several adaptivity techniques from the set we identified in Section II-A. Data sampling and data smoothing are used to address variations in the metrics considered: available bandwidth, latency, RTT and loss rate. Both passive observation and probing are used in collecting the data used to make the adaptation decision. Narada also employs a number of combined metrics, damping, randomization, hysteresis and several different utility functions to address instabilities in the observed data. Below we provide more details about how these techniques are supported in ESM. The values of the parameters we discuss below were selected empirically by the ESM creators and were demonstrated to provide good performance for several deployments [25].

A dedicated subsystem in ESM, the *performance agent*, is responsible for collecting the metrics used in the adaptation scheme. This agent runs periodically (every 7 seconds) and sends probe requests to a random subset of neighboring nodes. Each probe response from a neighbor includes the bandwidth it perceives from the source, the latency, the saturation level, and the path. The probing also enables each node to determine the RTT to a neighbor node, which is used to augment the other metrics. In addition, every node also maintains its own view of the network conditions by passively observing the data traffic it is receiving from the source. Both the passively observed metrics and the metrics measured via probing are smoothed using an exponential smoothing function to obtain a long-term average.

Currently, ESM supports three utility functions: one based on bandwidth, one based on latency, and one based on a combination of bandwidth and latency. The bandwidth utility function constructs a metric by combining the data offered by the neighbor in the probe response with the measurements made passively. If the bandwidth currently being received by the node is equal to the source rate or the relative improvement is less than 10% of the current bandwidth, the utility value is set to 0. If a node's bandwidth is greater than the current bandwidth being received and has a greater discretized utility function than the current parent, then the utility value is augmented. The latency utility function is constructed by merging the link latency with the route latency. If the new latency is less than the current latency then the utility is set to the relative improvement over the current metric. On the other hand, if the link metric to the current parent is less than 5, then the utility is also augmented.

The decision whether to change the tree or not is made locally by each node based on the utility gain (computed as described above) and a damping factor, used to induce stability. First, the damping factor will determine if switching the parent will indeed happen. If a node has recently changed parents with high frequency, a change is less likely to occur. In addition, a randomization technique is also used to avoid the case where several nodes try to change to the same parent. Each node maintains a switch probability indicating that the parent change will indeed happen. A random number is selected before making the switch decision. If the random number is greater than the switch probability, then no parent change takes

TABLE I

EFFECT OF ONE MALICIOUS (LYING) NODE ON AN ESM DEPLOYMENT OF 30 NODES ON PLANETLAB OVER A 90 MINUTES RUN.

Experiment	Chosen as Parent	Parent Changes	Unique Children	Unique on Path	Ave(sec)
Node was Lying	72	369	10	12	794.42
Node was not Lying	15	216	8	9	72.69

place. Otherwise, the change to the newly selected parent will take place and the current parent is placed on a pending drop list, which is flushed after a period of time. Once the change happens, the switch probabilities are reduced if there have been many switches recently and the switch probability is above a threshold.

The process of selecting the parent to which a given node will switch works as follows. Each node maintains links to its parent in the tree and its neighbors in the overlay. The agent running on each ESM node periodically processes the cached probe responses to build a selection table of possible candidates for becoming its new parent, referred to as the “short list.” Nodes which are currently saturated, descendants of this node, and those that have recently failed a bandwidth test are not considered. If there is no utility gain, no node is selected as a potential candidate and the process will be repeated next cycle. If several nodes are possible candidates, then the first node on the list is selected as the new parent. The selection process also uses hysteresis to generate a negative bias against nodes that have performed poorly as experience through primary observations. As a result, those peers who have been chosen as a parent and performed poorly are less likely to be chosen again.

B. Attacks Against Narada

In this section we concretely demonstrate some of the attacks identified in Section II-B in the context of the ESM system through experiments performed in the DETER and PlanetLab testbeds. We first provide a brief overview of the testbeds and experiment setup, then we provide details about the attacks.

1) Testbed and Experiment Setup: We conducted our experiments on the PlanetLab [8] and DETER [9] testbeds. PlanetLab is the most well-known live Internet testbed. It consists of machines hosted by research institutions and allows for the deployment of overlay networks and distributed services. Currently, over 275 active research projects are using PlanetLab as their testing platform. We use PlanetLab in our experiments because it provides the opportunity to study ESM under real-world conditions.

In addition, for experiments that could be disruptive to PlanetLab, we used the DETER [9] testbed. DETER is a shared testbed infrastructure that is specifically designed for cyber-security research. Unlike PlanetLab, DETER is an emulation testbed that allows us to emulate real networks in terms of latency and network topology. In addition, DETER provides a stable controlled environment and repeatable experiments. This allowed us to isolate the issues we were investigating and overcome some of the uncontrollable variability that can be found on PlanetLab.

There are several parameters that characterize ESM deployments and our experiments. The most important are the number of nodes in the overlay, the degree saturation for the nodes, and the constant bit rate of the multicast source. We use deployments of 30 and 50 nodes and experiment durations of 30 and 90 minutes. Nodes usually join after the experiment begins and leave before it ends, with an average participation of 25 minutes and 85 minutes per node. We use a saturation degree of 4-6 nodes as in similar ESM deployments [25]. In all the experiments below, we use a constant bit rate of 480 Kbps, which is sufficient to transmit video at two different qualities and one audio channel; this value was used in previous ESM deployments [25].

2) Attraction Attacks: All the experiments demonstrating attraction attacks were performed on PlanetLab.

Compromised nodes may use their insider position to lie about their bandwidth, latency, and saturation with the goal of attracting as many nodes as possible as children in the multicast tree. To demonstrate the effect that one malicious node, who exploits the adaptive nature of ESM, has on the multicast tree construction, maintenance, and stability, we ran the following experiment. One randomly selected node lies every probing cycle about bandwidth, latency, and saturation. This experiment was performed using 30 nodes, a degree of 6, and a duration of 90 minutes. We summarize our findings in Table I. When it is not lying, the malicious node is selected only 15 times as a parent by other nodes. When it is lying, the malicious node is selected 72 times, almost 5 times more often. When the node is lying, the overlay becomes more unstable, as can be seen in the large number of total parent changes. This increased instability can be attributed to the fact that the new child will eventually realize the bait-and-switch and change again. In addition, Table I shows that the lying node manages to get selected on paths such that the traffic of 12 other nodes (out of 30) goes through the malicious node.

We next investigate the degree to which a higher percentage of malicious nodes can affect the network. As in the case of one malicious node, the experiment was performed using an ESM deployment of 30 nodes with a saturation degree of 4 nodes and a duration of 90 minutes. The results of the experiment are summarized in Figure 2. The graph depicts the percentage of nodes that have at least a malicious node on their path to the source at some point during the experiment. In addition, it also shows how many of these nodes have chosen a malicious node as a parent. Finally, it shows how many of

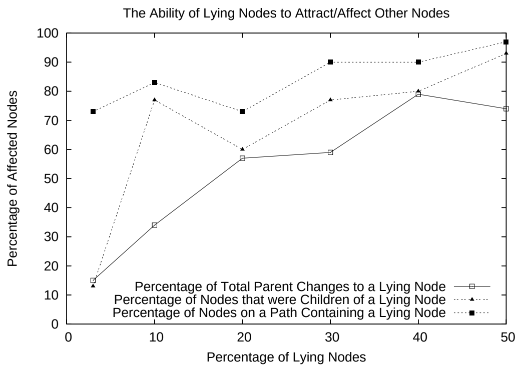


Fig. 2. This graph demonstrates the effect of attraction attacks on the correct nodes as a function of the percentage of malicious nodes. The experiment was conducted on PlanetLab with an ESM overlay of 30 nodes, for a duration of 90 minutes.

the decisions to change the parent were decisions that resulted in selecting a malicious node. Note that the malicious nodes were randomly selected and stronger attacks may exists if the nodes collude and perform a coordinated attack. Stronger attacks may also be possible by choosing nodes that have very good network connections.

As shown in Figure 2, a network in which 20% of nodes are malicious will result in those nodes controlling a significant amount of the traffic to other (non-malicious) nodes. For example, 20% of malicious nodes succeed in convincing more than 50% of the nodes in the network to select a malicious node as a parent. This potentially allows the adversary to monitor all traffic for the nodes it tricked into selecting it as a parent. In the case where 30% of the nodes are malicious, 90% of the nodes in the network have at least a malicious node on their path to the source. In this case, the malicious node can potentially affect any node that is positioned lower in the tree.

Having malicious parents can result in a severe degradation of service if the malicious parent decides to selectively drop data. One interesting aspect is to examine if the stability techniques and the decision function are able to detect the bad adaptations. In Figure 3, we demonstrate the potential impact of malicious nodes that use their positions on the tree to drop data traffic. The graphs plot the bandwidth averaged over all receivers as a function of time. The experiments are performed using an ESM deployment of 50 nodes with a saturation degree of 4 nodes. The duration of each experiment is 30 minutes, which corresponds to the duration of a conference presentation. At a predetermined time during each experiment (about 12 minutes since they joined the overlay), malicious nodes begin to drop 100% of the data traffic that they receive through the data dissemination tree. We vary the percentage of malicious nodes among 10%, 30%, and 50% of the total receivers to demonstrate the performance degradation that results from introducing malicious faults into the system. In each experiment, the malicious nodes are chosen at random from the receiver set and there is no colluding communication

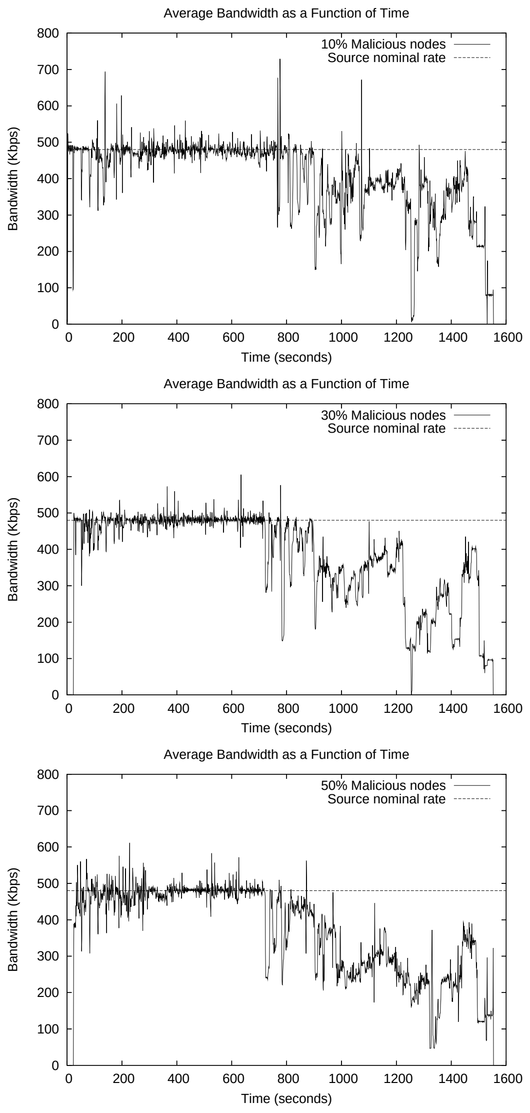


Fig. 3. These graphs demonstrate the effect on average bandwidth in ESM as a function of the number of malicious nodes. The experiment was conducted on PlanetLab with an overlay of 50 nodes.

between the malicious nodes.

In the case when 10% of the nodes behave maliciously about 10 minutes into the experiment, the average bandwidth depicted in the graph is shown to start decreasing. For the remainder of this experiment, the average bandwidth remains below the source bit rate of 480 Kbps. A similar effect is observed in the cases where 30% or 50% of the nodes are malicious. As expected, increasing the percentage of malicious nodes has a greater effect upon the system's performance. However, it can be noted that the effect of 10% nodes is already significant and increasing the number of malicious nodes to 30% and 50% does not change the effect on the average bandwidth dramatically. We believe this is because the tree structure is very vulnerable since it does not have any redundant paths and 10% of malicious nodes are enough to obtain advantageous positions in the tree. Note that in our

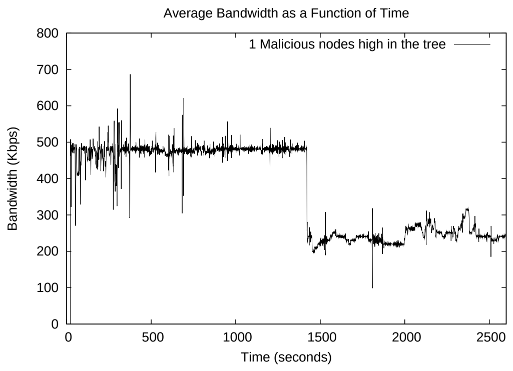
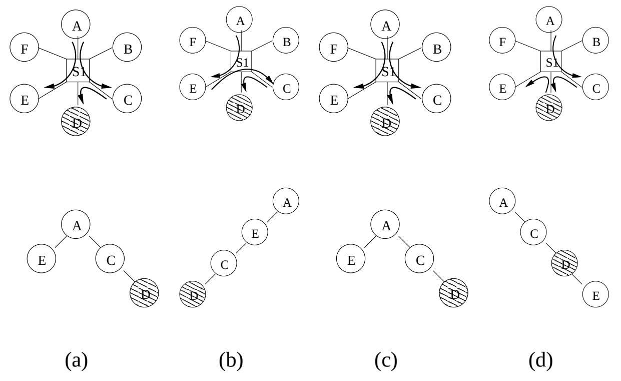
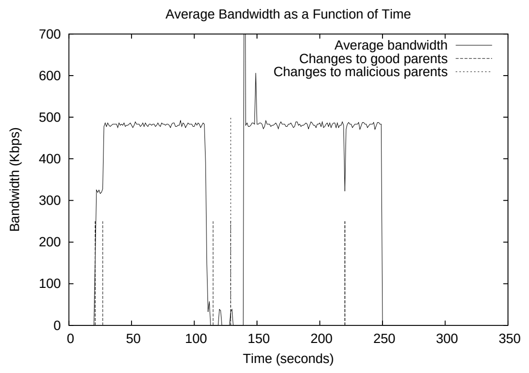


Fig. 4. This graph demonstrates the effect on average bandwidth in ESM of one malicious node that manages to maintain a high position in the tree while dropping 100% of the traffic, starting at 1400 seconds. The experiment was conducted on PlanetLab with an overlay of 50 nodes.





(e) The effects of the attack on the average bandwidth

Fig. 5. An example demonstrating a repulsion attack against the ESM multicast overlay system in a controlled experiment on DETER. (a) shows the overlay and the multicast tree before attack, (b), (c) and (d) show topology changes in the multicast tree as a result of the attack, at times 115 seconds, 128 seconds and 129 seconds from the beginning of the experiments, respectively. The result is that node E is manipulated by the attacker to attach to malicious node D, although this makes E to be three hops away from the source, instead of one at the beginning of the attack.

experiment the nodes lied just in the beginning to obtain the advantageous position in the tree, then they stop lying. The communication overhead associated with the attack is basically zero. As it can be seen in Figure 3 the adaptivity mechanisms react slowly and they do not manage to completely eliminate the malicious nodes from the tree structure by the end of the broadcast.

The closer a malicious node is to the source, the more nodes there are that use the malicious node in their path to the source. Thus, positioning malicious nodes connected directly or near the source makes this attack more devastating. In Figure 4, we demonstrate the effect of only one malicious node that lies during each probe cycle and manages to maintain its position in the tree near the source. The experiment was conducted using 50 nodes, a saturation degree of 4, and a duration of 90 minutes. As it can be seen, the effect is devastating as soon as the node starts dropping traffic. In spite of this, the adaptivity mechanisms are not able to react for more than 15 minutes.

3) Repulsion Attacks: The experiments demonstrating repulsion attacks were performed in the stable, controlled environment provided by the DETER testbed.

We demonstrate a repulsion attack performed by exerting an artificial influence of aversion toward the partially observable link state estimation in order to manipulate the routing tree topology. This attack example was also motivated by the fact that, while performing attraction attacks, we noticed certain nodes that were directly attached to the source, generally powerful positions within the tree, that could not be enticed by the lying node. Thus, we wanted to analyze how difficult it would be to actually displace these nodes by providing an external influence on their observation spaces.

In Figure 5, we show a simple topology to emphasize the susceptibility of the Narada protocol to repulsion attacks. We use a star topology composed of six nodes, all of which are connected with 100 Mbps links to the switch, S1. There is also no background traffic so each node has the potential to receive full bandwidth. The Narada protocol is configured to use a saturation degree of 2 and the utility function used takes into account both bandwidth and latency. In our example, node A is the source, nodes C, D, and E are end-systems in the overlay, nodes B and F are outsiders who collude with D, a malicious node that has infiltrated the overlay. During the attack, nodes B and F generate traffic to augment the attack of malicious node D, which lies about its bandwidth, latency, and saturation. Similar results of the attack will be obtained if nodes B and F are trusted members of the overlay attempting to improve their position in the tree or influence the path the data takes from the source to themselves or others.

As shown in the graph in Figure 5(e), the overlay converges to a stable structure, shown in Figure 5 (a), after about 30 seconds, at which point the mean bandwidth is approximately the same as the source rate (480 Kbps). Topology changes before this point were due to nodes attaching to the overlay tree, as represented by the impulses in Figure 5(e). The attack begins at 115 seconds when nodes B and F begin flooding 30 seconds worth of traffic at the source, node A. After several seconds of traffic, the attack is able to generate the first disturbance in the tree when node C detaches from the source and chooses node E as its new parent 5 (b). This occurs despite the fact that C now has an extra hop to the source. Then, 14 seconds later, C switches back to its previous position, but the overlay has yet to stabilize (Figure 5 (c)). Next, under a second later, node E detaches from the source and, instead of choosing node C, chooses the malicious node, D, as its parent (Figure 5 (d)). Note that node E was previously directly connected to the source but is now connected three hops away. The changes after 200 seconds are due to nodes leaving the experiment.

One important aspect of the experiment is the amount of traffic generated by the attackers. Two nodes create the attack by filling the 100 Mbps link with a 30 second burst of traffic and in some experiments it only required a 5 second burst. Note that in real Internet deployments, the cost of the attack will be substantially less since links will typically have a lower bandwidth.

A variant of the attack is to target the active probes on which the victim node relies. In this case, the victim's peers will be made to look unappealing for changes, thereby increasing the chances of the malicious node to move upward in the tree.

4) Disruption Attacks: In our experiments we have also been able to perform a number of disruption attacks against the Narada protocol. In the interest of space we have decided not to include those experiments. They were also excluded since it should be intuitive for the reader to see that the disruption caused in both the attraction and repulsion attack could be performed in a periodic manner to create a longer term disruption attack. These attacks are similar with the attacks presented in detail in [7], [6].

IV. Defending Against Attacks in Adaptive Overlay Networks

In this section we identify several components that will provide a comprehensive solution for mitigating Byzantine attacks that exploit adaptivity in overlay networks. These components are: reducing unnecessary or unnatural adaptations, increasing stability by incorporating metrics that reflect stability into the decision process, detecting malicious behavior that resulted in observable degradation of service, and reacting to the detected malicious nodes. In addition, we provide an in-depth description and experimental results for what we believe is the most critical and challenging problem: preventing bad adaptations. As the attacks we are concerned with are performed by compromised nodes controlled by adversaries, cryptographic mechanisms deployed to provide authentication and encryption will not be able to mitigate the attacks by themselves. The solution space components we describe below are complementary to cryptographic techniques and assume that authentication and integrity data protection are provided.

A. Solution Space

The primary cause of the attacks we identified is the ability of the attacker to influence the adaptation process. Thus, the most important component of a solution is preventing, or at least reducing, these unnecessary or bad adaptations. In fact, a perfect prevention will render the other components useless. Since perfect prevention is not possible, especially in the context of Byzantine faults, we discuss three other complementary components.

1) Reducing Unnecessary or Bad Adaptations: The adaptation process relies on two sources of information one measured by each node, the other obtained by probing a set of peer nodes. By blindly accepting the information reported by the – potentially malicious – probed nodes, correct nodes make bad adaptations. One way to prevent this from happening is to filter out the metrics reported by malicious nodes that may influence the adaptation decision. Our defense mechanism evaluates temporal and spatial correlations among data in the system to detect outliers and reduce the ability of malicious nodes to influence the adaptive responses, while still improving the adaptation decisions made locally. Although our solution is developed in the context of overlay networks we believe that it can be used to address the more general problem found in many network protocols associated with "blind acceptance" of routing metrics [26].

2) Increase Stability: Reducing the number of unnecessary adaptations has the potential to increase the stability. However, we believe that explicit efforts must be taken to reduce the overhead and the number of necessary changes by integrating stability in the function that drives the adaptation. Reducing the number of adaptations is obviously important since with every adaptation there is always the potential of making a bad adaptation. Several metrics can be included to reflect stability such as the time a node was connected to his current parent, the frequency of changes, or even the degree of variance in metrics. By making adaptations that consider stability as part of their optimization function, the nodes perceived as unstable will be pushed to the fringes of the tree as no other node will select them as a parent.

3) Detection and Recovery: Enhancing the adaptation decision with Byzantine robustness is not sufficient since no method will have perfect accuracy. The two mechanisms described above may still result in bad adaptations. Thus, a solution must also be designed that allows a node to quickly detect and recover when such an adaptation occurs. In this case, if the attack has observable effects such as severe degradation of the delivery service, we propose two recovery mechanisms that allow a node to determine that it is connected to a malicious parent and then to find a new parent. The first mechanism uses the low-bandwidth direct unicast link that every node in the overlay shares with the source to provide selective feedback information to the source about the received data. The second mechanism uses the low-bandwidth channel to provide information from the source to a node about its path to the source. Since the path is sent by the source (and can be protected cryptographically), the information can be trusted. The path information provides another means for a node to detect inconsistencies in the metrics reported by its parent and neighbors.

4) Response: One important aspect of any defense mechanism relates to reacting to malicious nodes. We believe a solution must also take action and neutralize the threat of the malicious nodes. Without taking such actions the convergence of the protocol, as well as the overall overhead, will increase as the malicious node continues to interfere with the system. To this end, we have explored building a shared black list of malicious nodes, but we do not eliminate them from the overlay. Instead, we force them to the edges of the tree such that they will not be able to affect other nodes. In addition, this mechanism can be enhanced with a reward procedure that allows bounding the amount of data that a system can lose due to adversarial behavior similar to [27].

B. Reducing Bad Adapatations Utilizing Local Spatial and Temporal Correlation for Outlier Detection

We now present an in-depth description and experimental results for the most critical component of a comprehensive solution: reducing the potential for poor adaptation decisions in overlays influenced by attackers. Our solution consists of determining which nodes are advertising inconsistent metrics by performing an outlier analysis on the information received from probed nodes and used in the decision process. An outlier is a data point that is significantly different from the rest of the data in the observation space based on some measure of distance. The nodes detected as outliers will then be discarded from the potential parent set so they will not be able to influence the multicast tree structure. The detection is performed locally by each node using spatial and temporal correlations. The *spatial outlier detection* compares the reported metrics received from each node in the set of probed nodes. The *temporal outlier detection* examines the consistency in the metrics received from an individual probed node over time. Considered metrics include probed latency, probed bandwidth and RTT.

In order to prevent suspicion from other nodes, a malicious node must insure that any lie it tells is: (1) consistent with what the other peers are reporting during a probe cycle about current conditions (external with respect to the rest of the world), (2) consistent with the bandwidth, latency, and influence yielded towards the RTT (internal with respect to other metrics within a set of dependent variables), and (3) consistent with what it said in the past. The spatial outlier detection targets the first and second aspects of consistency, while the temporal outlier detection targets the second and third aspects.

The intuition behind our solution is that an attacker will have difficulty lying consistently because it does not have perfect knowledge of the observation space and does not have a guaranteed feedback loop to coordinate with other attackers. For example, a set of colluding malicious nodes will have difficulty lying in a manner consistent with information provided by other peers probed during the same probe cycle. This is because malicious nodes cannot accurately predict the random subset of nodes that will be queried during the probe cycle and only have a finite amount of time, the probe period, to coordinate. In addition, the intrinsic dependency existent in several measured variables requires attackers to make sure that the "fake" metrics vary in a consistent manner. This dependency results from a fundamental characteristic of end-system multicast systems – that the distribution tree will overlap itself on the routing infrastructure, often represented as a measure called link stress [23]. Additionally, the process is made more difficult by the fact that attackers can only make the RTT worse, because it is a measured attribute, and yet, at the same time, the RTT must remain consistent with both the bandwidth and latency.

A key component of our approach is using the Mahalanobis [28] distance as the mechanism to detect outliers. The Mahalanobis distance has several advantages that make it appropriate for our problem:

- It has been shown to be better than other distance functions for detecting outliers with multiple attributes [29]. In our case, we can use several attributes in the detection process since each node reports latency, RTT, and bandwidth.
- It takes into account the variance and covariance of the attributes that are measured by scaling each variable based on its standard deviation and covariance. This means that the attributes with high variance receive less weight than components with low variance.
- It takes into consideration the correlation between attributes and how the measured attributes change in relation to each other. This makes it appropriate for our environment where there is a dependency between the attributes reported by each node.

1) Spatial Outlier Detection: Currently, the utility function at each node relies on the observation tuple consisting of bandwidth, latency, and RTT, recorded every probing cycle. Our spatial outlier detection uses the same observation space as the utility function. Thus, it does not add any communication overhead. Spatial outlier detection is performed during each probing period as follows. The observation tuples are used to compute the centroid of the data set. We then compare how far the observation tuple for each node is away from the centroid. The comparison is done by computing the Mahalanobis distance between each node and the centroid. The Mahalanobis distance takes into account the variability of the variables and the correlation between variables. It is computed as follows:

$$d(\vec{x},\vec{y}) = \sqrt{((\vec{x}-\vec{y})^T C^{-1} (\vec{x}-\vec{y}))}$$

$\vec{x}$ and $\vec{y}$ are the feature vectors which in this case include bandwidth, latency, and RTT. $\vec{x}$ is the value from the probe response and $\vec{y}$ is the average value that was calculated. C^{-1} is the inverse covariance matrix developed from the observation tuples.

Two important special cases must be considered. The first case is when there are not enough observation tuple responses received during a probe cycle. In this case we compare the observation tuples received against the most recent centroid, if available. The other special case is when there is no variance between the observation tuples that are received. (We actually encountered this situation when running experiments on a high speed local area network.) In this case, we can not compute the

Mahalanobis distance since the determinant of the covariance matrix becomes zero. To address the problem we randomly choose a parent from that probe set of observation tuples and compare it to the most recent centroid, if available. If no centroid is available we postpone the decision to the next probe cycle.

2) Temporal Outlier Detection: The use of temporal outlier detection is motivated by the fact that a node's view of its peers is manifested through periodic samplings of metrics that can be correlated over time. We use incremental learning to develop models for the nodes currently in a node's peer group during the course of a multicast session. Incremental learning allows our models to improve over time as more data is collected and old data is decayed [28]. In this context, we are using temporal correlation to maintain a sense of history within the system which allows us to compare the metrics received in the current cycle with the information we have received in the past from those nodes in our current peer list.

The technique we use is based on the "simplified Mahalanobis distance" presented in [28]:

$$d(x, \vec{y}) = \sum_{i=0}^{n-1}(|x_i - \vec{y_i}|/(\vec{\sigma_i} + \alpha))$$

In this equation n is 3, related to the three metrics we are currently using (bandwidth,latency,RTT), $\vec{\sigma_i}$ is the standard deviation, and α is the smoothing factor. In order to reduce the overhead of maintaining the entire set of observations we also make a simplifying assumption that the metrics are statistically independent. We trade-off accuracy in the distance function related to the covariance of the metrics for the amount of data we must maintain. Thus, we do not need to maintain the entire history of sampled values, which continues to grow over time.

Each node will additionally maintain the mean, standard deviation, and sample count associated with the observation tuple within the routing table entry for each of the peers. These values, which are stored in the routing table, will represent the temporal centroid associated with the respective peer. This centroid is incrementally updated with observations received during each probe cycle, as in [28], using the technique Knuth described in [30]. At the end of the probe cycle the latest observation tuple associated with each peer is compared with the centroid using the Mahalanobis distance. If the distance is greater than the threshold, then this node is considered a temporal outlier.

3) Utility Driven Spatio-Temporal Fusion: We now show how the spatial and temporal outlier detection techniques are used during the adaptation process, using a technique similar to a codebook [31]. The decision process checks if there are a large number of nodes that are substantially far from their historical centroid (temporal outliers). If there are, then an adaptation does not occur during this probe cycle (artificial influence of aversion repulsion attack). If not, then the decision process continues. Next, the peer nodes are ranked according to their spatial outlier distance from the centroid. Peer nodes are then traversed moving from those nodes closest to the centroid to those nodes farthest from the centroid. The node that is closest to the centroid, is neither a spatial or horizontal outlier, and has passed the utility function (made it to the short list) is chosen as the new parent. At this point, the new parent request is sent. If no peer is found which meets these criteria, then no adaptation is performed during this probe cycle.

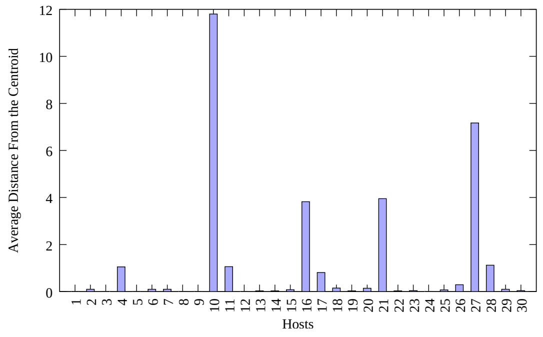


Fig. 6. Nodes identified as outliers on an ESM deployment of 30 nodes on PlanetLab over a 90 minutes run using spatial outlier detection.

C. Experimental Results

In this section we demonstrate the effectiveness of each outlier detection method proposed.

1) Testbed and Experiment Setup: We conducted our experiments on the PlanetLab [8] and DETER [9] testbeds. The experiments are similar to those presented in the Section III-B, but this time our outlier detection mechanisms have been added. There are several parameters that characterize our experiments. We use ESM deployments of 30 and 50 nodes as specified. All experiments were run for 90 minutes. The node join-leave pattern is a uniform join; all the nodes joined in the beginning and stayed for an average of 80 to 85 minutes. We use a degree saturation of 4 nodes. The source generates a constant bit rate of 480 Kbps. The probe cycle was set to 7 seconds.

2) Spatial Outlier Detection: We first investigated the effectiveness of the Mahalanobis distance in identifying outliers. The goal of the following experiment is to see if a malicious node is perceived as an outlier by all nodes in the system, when using spatial outlier detection. One random node was chosen to be malicious on an overlay of 30 nodes deployed on the PlanetLab testbed.

At the end of each probe cycle we calculated the centroid from all the probe responses received during the probe cycle and the distance of each probe response from the centroid. Next, we calculated the average distance from the centroid for each probe cycle for a single node. Finally, we calculated the average distance across all the end-system nodes involved in the experiment. This number is generated based on a total of 539,739 probe responses that were received during 19,465 probe cycles. Figure 6 presents the average distance each node was from the centroid across all the probe cycles and averaged across all end systems. Several nodes are detected as outliers, including the malicious node 4. The graph indicates that the malicious node was seen as anomalous on average across all the nodes in the system, despite the fact that their centroids

TABLE II

AVERAGED BANDWIDTH, RTT, AND LATENCY AND AVERAGE DISTANCE FROM THE AVERAGE CENTROID FOR A DEPLOYMENT OF 30 NODES ON PLANETLAB OVER A 90 MINUTES RUN.

Node	RTT	Bandwidth	Latency	Distance
4	109.83	480.00	0.00	1.05
10	2678.88	89.43	612.64	11.80
11	185.74	480.00	0.00	1.06
16	479.46	445.62	218.30	3.82
17	192.25	469.41	2.00	0.81
21	522.58	449.61	195.81	3.95
27	484.06	160.88	167.00	7.17
28	342.11	469.39	144.10	1.12
	265.36	**454.65**	**64.64**	

TABLE III

NUMBER OF UNIQUE HOSTS THAT EACH OF THE OUTLIER NODE ID APPEARED IN THE SHORT LIST BECAUSE THEY PASSED THE UTILITY FUNCTION.

Node ID	4	10	11	16	17	21	27	28
# Short Lists	9	2	18	2	3	2	1	9

were independently developed. As a result, it demonstrates the utility of Mahalanobis distance for distinguishing outliers.

We now analyze the results in detail and show how, by combining the utility function with the spatial outlier information, the system can avoid making decisions influenced by malicious adversaries. Table II presents the average values for the probed metrics of RTT, bandwidth, and latency for all noticeable outliers. The bottom row shows the average centroid of all nodes for each of these metrics. The right-most column shows the resulting distance between each outlier node and the centroid. In addition, Table III presents the number of times that outlier nodes in Figure 6 were considered as possible candidates for a parent, based only on the utility decision function.

As shown in Figure 6, there are 8 significant outliers. One of them, node 11, is the trusted source and can therefore be discarded from the outlier detection. Information provided in Table II shows that nodes 10, 16, 21, and 27 are far away from the centroid due to their poor performance. Thus, they will not be selected in the short list as possible parents because they will not pass the utility function. This is supported by Table III, which shows that nodes 10, 16, 21, and 27 were considered as potential parents for at most two nodes, which were most likely nodes in the same respective LAN as the outlier. While less dramatic, this is also the case with node 28, which had an average RTT greater than twice the mean. Based on this data, it is intuitive to see that it is possible to develop a threshold to distinguish the aforementioned nodes, as well as 4 and 17, as outliers. Empirically, we have found that an effective threshold typically lies between 1 and 2, although this could be set based on the security requirements of each node. Because they were flagged as outliers, these nodes would not have been chosen as parents. This avoids selecting the malicious node, 4, as a parent.

TABLE IV

THE EFFECTIVENESS OF SPATIAL OUTLIER DETECTION AT IMPROVING PARENT SELECTION ON A DEPLOYMENT OF 50 NODES ON PLANETLAB OVER 90 MINUTE RUNS.

	Malicious Selected	Total	Percentage
No lying	8	427	1.61%
Lying	649	1122	57.84%
Spatial	84	519	16.18 %
Spatial/Temp	22	282	7.80 %

Note that the experiment also shows that a node could be identified as an outlier because its performance is much worse than the performance of the other peers in the probe set, much better than other nodes in the probe set, or simply inconsistent. Regardless of the cause, by not choosing outliers the system achieves increased stability.

To demonstrate the effectiveness of spatial correlation and the Mahalanobis distance function at improving the parent selection process and the stability of the system, we repeated the same experiment as above for a deployment of 50 nodes and recorded the number of parent changes that took place for the duration of the experiment. The outcome of these experiments is shown in Table IV. The numbers in the table are summed across the 50 nodes in the experiment. The results indicate that using our outlier detection scheme has dramatically reduced the likelihood of choosing the malicious node as a new parent. Our method also dramatically improved the stability of the network, as measured by a decrease in parent changes, in spite of the presence of the malicious node. In fact, the number of adaptations is comparable to the number of adaptations that would occur with no malicious nodes present in the network.

As previously stated, the method does not completely eliminate bad adaptations. However, we believe this indicates the need to augment better decision techniques with an ability to detect and recover from the inevitable bad adaptations that will still occur. Our solution is therefore only one major piece of the final solution.

3) Temporal Outlier Detection: To demonstrate the effectiveness of temporal correlation and the “simplified Mahalanobis distance” function at detecting outliers, we conducted the following experiment on the PlanetLab testbed. An overlay of 50 nodes was deployed, and a random node was chosen to act maliciously by sending false reports about its metrics. Figure 7 presents the distance function as measured by a non-malicious node in the experiment. During our experiments the threshold for the simplified distance function was set to 3, which intuitively means that each metric can differ at least one standard deviation from its mean value. The graph presents

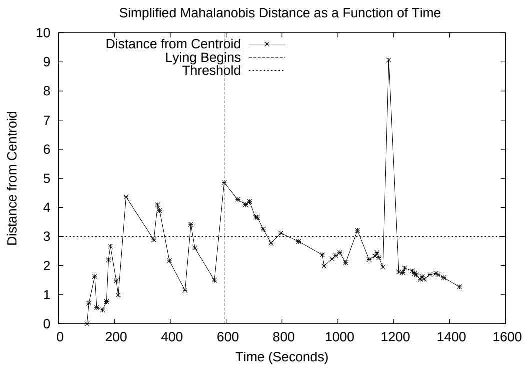


Fig. 7. The effectiveness of detecting temporal outliers on a 50 node ESM deployment on PlanetLab

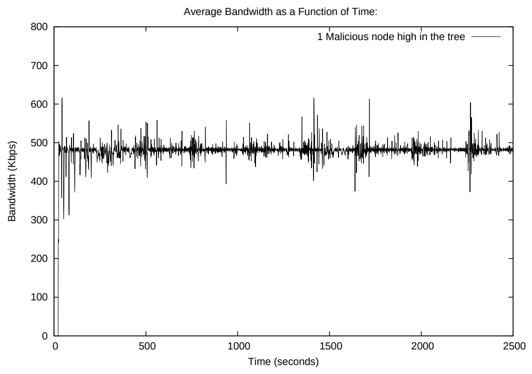


Fig. 9. This graph demonstrates the ability of the utility driven spatio-temporal fusion mechanism to mitigate the effects seen in Figure 4 where one malicious node manages to maintain a high position in the tree while dropping 100% of the traffic. As in Figure 4, the attack begins at 1400 seconds. The experiment was conducted on PlanetLab with an overlay of 50 nodes.

the first 24 of the 90 minutes of the experiment, focusing on details around the point when the node begins to lie. In the first 9 minutes we see a number of ephemeral fluctuations, which is typical as the correlation begins to improve its model over time. The non-malicious node first detects that the malicious node has started lying at 593 seconds, the initial distance being measured at 4.85 from the centroid. Then 7 out of the next 8 measured distances are above the threshold value as the malicious node continues to lie. This lasts for about 200 seconds before the observation tuple begins to be seen as normal as the centroid has adapted to these falsified values. After this point we see that the model begins to converge again. This effect on convergence demonstrates also why a complete solution must include the ability to respond and neutralize malicious nodes. It may also be possible for a node to try and create a high variance in the metrics it reports in order to manipulate the distance function. This type of activity motivates the need for including metrics that reflect stability in the decision process, since a node with extreme variations in metrics is undesirable from a stability perspective.

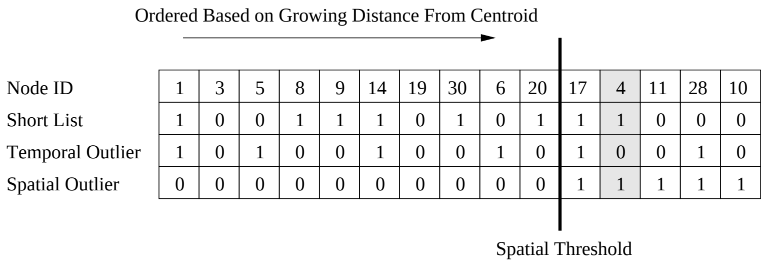
4) Utility Driven Spatio-Temporal Fusion: An example of how the decision process works can be seen in Figure 8. The data used is based off the information provided in Figure 6. Remember that node 4 was the lying node. The subset of nodes that responded to the probe request are ordered in increasing distance from the centroid. Now in order to choose if a new parent will be selected and who that parent will be the performance agent traverses the list beginning at those nodes closest to the centroid. The first node encountered that is on the short list (the performance agent previously selected as potential parent) and is not a temporal or spatial outlier is selected as the new parent. In Figure 8, node 1 is rejected for being a temporal outlier, node 3 is rejected for not being a suitable parent, node 5 is rejected for not being a suitable parent and a temporal outlier, and finally node 8 is selected as the new parent. The performance agent ceases traversing the list once it reaches the spatial threshold.

The effectiveness of this method can be also seen in the last row in Table IV. The row presents the results of combining the temporal and spatial correlation in the utility driven spatio-temporal fusion. In these experiments the threshold for spatial outlier detection was set at a conservative 1.5 and the threshold for temporal outlier detection was set to 3. From the results we can see that the combined technique has dramatically reduced the number of times the malicious node was chosen as a parent. With our outlier detection mechanisms enabled only 7.80% of the changes made during the experiment were to a malicious parent. In comparison, 57.84% changes to malicious parents occurred when the Narada protocol was run without our mechanisms. In addition, the combination of spatial and temporal outlier detection reduced the total number of changes that were made during the 90 minute experiment. Our method resulted in 66.04% of the total number of changes made during the experiment with no malicious nodes. Despite this reduction in the number of changes, there was no noticeable degradation on average bandwidth, indicating that our method inhibited potentially unnecessary adaptations.

The ability of the utility driven spatio-temporal fusion to mitigate threats can be seen in Figure 9 which depicts the same experiment as in Figure 4, but with the utility driven spatio-temporal fusion enabled. Remember that in Figure 4, one duplicitous node was able to have a significant impact on the average performance of the overlay multicast tree for an extended period of time by ascertaining a powerful postion in the tree and dropping 100% of the traffic. As it can be seen in Figure 9, our attack mitigation mechanism is able to prevent the malicious node from obtaining and maintaining the powerful position in the tree and thus the node is unable to inflict the previously described damage on the system as a whole.

D. Overhead

We present the overhead of our defense techniques by analyzing bandwidth, memory, and CPU utilization. Our methods do not introduce any extra bandwidth utilization since they use information that is already being exchanged between nodes. In fact, as an artifact of reducing the number of adaptations


Ordered Based on Growing Distance From Centroid
Spatial Threshold

Node ID	1	3	5	8	9	14	19	30	6	20	17	4	11	28	10
Short List	1	0	0	1	1	1	0	1	0	1	1	1	0	0	0
Temporal Outlier	1	0	1	0	0	1	0	0	1	0	1	0	0	1	0
Spatial Outlier	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1

Fig. 8. This is an example of how the utility driven spatio-temporal fusion is performed during a probe cycle based on data from Figure 6. The 15 nodes represent a possible probe set. In this example Node 4 was the lying node and node 8 was chosen as the new parent

the amount of control traffic being sent is also reduced. The memory utilization for spatial correlation only lasts for the span of a probe cycle and requires maintaining the observation tuple associated with each of the nodes in the probe set. This requires storing three additional values in the route table for the peer set maintained by each node. In order to perform the temporal correlation we modify the route table entries to store nine additional values: mean, standard deviation, and count for each of the three metrics.

Additional CPU utilization occurs only when the performance agent has selected a short list of possible parents. If the utility function does not find any suitable parents then no additional computations are performed during that probe cycle. The computational complexity is bound by the number of nodes in the probe set which is constant. The computation of the temporal outliers is a constant time calculation performed for each of the nodes in the probe set. The calculation of the spatial correlation is also computed in constant time.

V. Related Work

Our work focuses on adaptivity attacks from trusted compromised nodes (insiders) in the context of overlay networks and our solution uses concepts borrowed from anomaly detection. Below we review related work in several areas related to our work.

Attacks exploiting adaptivity. Previous research has shown the vulnerability of the TCP adaptivity mechanisms, i.e. the congestion control mechanism, to attacks from malicious outsiders [6]. The authors showed that by manipulating the end-system's perception of network congestion, the adaptivity mechanism could be used to perform a low-rate DOS attack with severe effects on TCP throughput.

In [7], the authors generalize the attack against TCP [6] as a form of low-rate ROQ attack targeting adaptive control loops that drive resource allocation and affect perceived service of a system (bandwidth, jitter, etc). The authors model the problem analytically by using control theoretic models. They cast the adaptivity as an optimization process where multiple control loops adaptively converge to a stable operating point. The analysis focuses on attacks that create noisy feedback for the controllers where the mechanisms being used by the attacker are short bursts of traffic (square wave pattern). Simulations and experiments demonstrate the validity of models for these attacks on active queue management (AQM) techniques and TCP's AIMD rules for congestion control. The work points out the interesting observation that the more aggressively or greedily a protocol attempts to optimize the more susceptible it becomes to these attacks.

The fundamental difference between the work in [7] and our work lies in the adversarial model: We consider Byzantine adversaries, which can cause stronger attacks without using a significant computational effort. The attacks identified in [7], are more general and the transients created by the attack are similar to those experienced in normal conditions. As a result, it makes it difficult for the resource to realize it is under attack since it would have to monitor a large range of times scales. In our case, the nature of the attacks and of the application and deployment environment allows us to go one step further than [7] and propose a solution. We have demonstrated that detection is possible by using context sensitive observation spaces and correlated information associated with the same information that drives the adaptation.

Anomaly detection. Anomaly detection has been previously leveraged to address insider threats in distributed protocols. For example, the benefits of using statistical anomaly detection to detect insider attacks against link-state routing protocols has been demonstrated in [32]. The proposed method uses intrusion detection systems which do not require changes to the protocol but passively monitor the network looking for perturbations in the observation space. In our work, we use context sensitive anomaly detection that is incorporated into the protocol so it has the semantic understanding of the data. The major advantage of our approach is that the observation space of the detection mechanism and adaptation mechanisms are tightly coupled allowing application centric semantically rich detection. This tight coupling is necessary in autonomic systems and also reduces it susceptibility to classic obfuscation attacks on intrusion detection [33].

Recently the benefits of the Mahalanobis distance for statistical anomaly detection have been demonstrated in the context of network intrusion detection [28], [34]. In [34] the authors present a comparative study of detection schemes based on data mining techniques for network based intrusion detection. In [28] the authors discuss an unsupervised payload based network anomaly detector based on the Mahalanobis distance and used to detect attacks like worms.

In our work we focus on reducing the likelihood of making unnecessary or unnatural adaptations as opposed to letting them happen and then trying to detect them. Recently, similar work has been done in the context of inter-domain routing messages for BGP. In [35] the authors use these techniques to reduce the likelihood of a router accepting invalid routes. Anomaly detection is used to detect inconsistencies in the topology information and geographical location data. A solution proposed for the multiple origin AS conflicts in BGP also makes use of similar techniques [26].

Use of spatial and temporal correlations. Spatial and temporal correlations were previously used in the context of network security. A notable work in this aspect is [31] where authors use temporal and spatial correlations to trace back attacks and detect attack scenarios, using a large amount of information available from intrusion detection systems, firewalls, and different software logs. Unlike the approach in [31], which was more general, our work focuses on overlay networks and does not look for correlations, but exploits the fact that they exist to detect inconsistent metrics and find suspicious nodes.

Selfish behavior in overlay networks. To the best of our knowledge the problem of malicious insider attacks was not studied in the context of overlay networks. The problem of selfish adversarial behavior in the context of overlay was studied in [23]. The authors showed through simulation that selfish nodes (i.e. nodes that want to obtain advantage over other nodes, but do not have destructive goals as malicious nodes) can selfishly improve their performance by manipulating distance measurements and cheating as independent end-systems. Our work is different in the fact that considers a malicious attacker and presents results in the context of a real system in real deployments over the Internet.

VI. Conclusions

In this work we provided a characterization of the mechanisms currently used to achieve adaptivity in overlay networks and identified insider attacks against these mechanisms. We believe that the attacks are relevant also in other contexts, as adaptivity mechanisms are used also in the design of sensor and wireless networks. The attacks are successful because adaptivity mechanisms are often not designed to handle degenerate inputs.

We demonstrated the effectiveness of the newly identified attacks against a well-known adaptive multicast overlay network, ESM [1]. Our experiments conducted in real-life deployments and emulations, demonstrate that although ESM employs an advanced set of adaptivity mechanisms it is unable to mitigate the attacks posed by a malicious adversary.

We provided an initial analysis of how such attacks can be mitigated and prevented throughout the life-cycle of the overlay, and an in-depth solution to a critical aspect of the problem: preventing poor adaptation decisions in networks influenced by attackers. Our solution lies in performing spatial and temporal outlier analysis on primary (measured) and secondary (probed) metrics to allow an honest node to make better use of available information before making an adaptation decision.

We demonstrated the benefits of using our outlier detection mechanisms to improve the adaptation process and the overall stability of the system in the context of ESM, through experiments conducted in real deployments. Our techniques must be combined with a detection and response mechanism to eliminate the malicious nodes. In future work we would like to address this aspect, which will also allow us to experiment with a larger number of malicious nodes under a diverse attack pattern. In addition, we would like to investigate in depth how to decrease even further the number of unnecessary changes by integrating metrics of stability in the function that drives the adaptation.

VII. Acknowledgments

We would especially like to thank Nick Petroni for his invaluable comments and insightful discussions that improved significantly the presentation of this work. We would also like to thank to Sanjay Rao and Ruben Torres for providing insights into the architecture of ESM and Narada, and to David Zage for his help with the logical and physical networks overlay picture.

This work is supported by National Science Foundation CyberTrust Award No. 0430276. The views expressed in this research are not endorsed by the National Science Foundation.

References

[1] Y. hua Chu, S. G. Rao, and H. Zhang, "A case for end system multicast (keynote address)," in *SIGMETRICS '00*, 2000.
[2] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable application layer multicast," in *ACM Sigcomm*, August 2002.
[3] D. Pendarakis, S. Shi, D. Verma, and M. Waldvogel, "ALMI: An application level multicast infrastructure," in *3^{rd} USENIX Symposium on Internet Technologies and Systems (USITS)*, 2001.
[4] J. Jannotti, D. K. Gifford, K. L. Johnson, M. F. Kaashoek, and J. W. O'Toole, Jr., "Overcast: Reliable multicasting with an overlay network," in *USENIX OSDI 2000*, 2000.
[5] "2005 e-crime watch survey - survey results." http://www.cert.org/archive/pdf/ecrimesurvey05.pdf.
[6] A. Kuzmanovic and E. W. Knightly, "Low-rate TCP-targeted DOS attacks: the shrew vs. the mice and elephants," in *SIGCOMM '03*, 2003.
[7] M. Guirguis, A. Bestavros, and I. Matta, "Exploiting the transients of adaptation for roq attacks on internet resources," in *The 12^{th} IEEE International Conference on Network Protocols (ICNP'04)*, 2004.
[8] "Planetlab." http://www.planet-lab.org/.
[9] "Deter." http://www.isi.edu/deter/.
[10] D. D. Clark, "The design philosophy of the darpa internet protocols," pp. 54–62, 1992.
[11] P. Gupta and P. R. Kumar, "A system and traffic dependent adaptive routing algorithm for ad hoc networks," in *The 36^{th} IEEE Conference on Decision and Control*, December 1997.
[12] R. B. Santashil PalChaudhuri, Rajnish Kumar and D. B. Johnson, "Design of adaptive overlays for multi-scale communication in sensor networks," in *the International Conference on Distributed Computing in Sensor Systems (DCOSS 2005)*, June 2005.
[13] D. ANDERSEN, "Resilient overlay networks," 2001.
[14] D. Bauer, S. Rooney, P. Scotton, S. Buchegger, and I. Iliadis, "The performance of measurement-based overlay networks.," in *QofIS*, pp. 115–124, 2002.
[15] Y. hua Chu, S. G. Rao, S. Seshan, and H. Zhang, "Enabling conferencing applications on the internet using an overlay multicast architecture," in *ACM SIGCOMM 2001*, (San Diago, CA), ACM, Aug. 2001.
[16] Y. Rekhter and T. Li, "A Border Gateway Protocol 4 (BGP 4)." Internet Engineering Task Force: RFC 1771, March 1995.

[17] D. G. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris, “Resilient Overlay Networks,” in *18th ACM SOSP*, October 2001.

[18] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian, “Delayed internet routing convergence,” *IEEE/ACM Trans. Netw.*, vol. 9, no. 3, pp. 293–306, 2001.

[19] Z. M. Mao, R. Govindan, G. Varghese, and R. H. Katz, “Route flap damping exacerbates internet routing convergence.,” in *SIGCOMM*, pp. 221–233, 2002.

[20] M. Seshadri and R. H. Katz, “Dynamics of Simultaneous Overlay Network Routing,” Tech. Rep. UCB//CSD-03-1291, University of California, Berkeley, November 2003.

[21] C. Tang and C. Ward, “Gocast: Gossip-enhanced overlay multicast for fast and dependable group communication,” in *DSN ’05: Proceedings of the 2005 International Conference on Dependable Systems and Networks (DSN’05)*, (Washington, DC, USA), pp. 140–149, IEEE Computer Society, 2005.

[22] B. Y. Zhao, L. Huang, J. D. Kubiatowicz, and A. D. Joseph, “Exploiting routing redundancy using a wide-area overlay,” Tech. Rep. CSD-02-1215, U. C. Berkeley, Nov 2002.

[23] L. Mathy, N. Blundell, V. Roca, and A. El-Sayed, “Impact of simple cheating in application-level multicast.,” in *INFOCOM*, 2004.

[24] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian, “Delayed internet routing convergence,” *IEEE/ACM Trans. Netw.*, vol. 9, no. 3, pp. 293–306, 2001.

[25] Y. hua Chu, A. Ganjam, T. E. Ng, S. G. Rao, K. Sripanidkulchai, J. Zhan, and H. Zhang, “Early experience with an internet broadcast system based on overlay multicast,” in *USENIX Annual Technical Conference, General Track*, pp. 155–170, 2004.

[26] X. Zhao, D. Pei, L. Wang, D. Massey, A. Mankin, S. F. Wu, and L. Zhang, “Detection of invalid routing announcement in the internet,” in *DSN ’02: Proceedings of the 2002 International Conference on Dependable Systems and Networks*, (Washington, DC, USA), pp. 59–68, IEEE Computer Society, 2002.

[27] B. Awerbuch, D. Holmer, C. Nita-Rotaru, and H. Rubens, “An on-demand secure routing protocol resilient to byzantine failures,” in *Proceedings of ACM Workshop of Wireless Security (WiSe)*, September 2002.

[28] K. Wang and S. J. Stolfo, “Anomalous Payload-based Network Intrusion Detection,” in *Proceedings of the Recent Advances in Intrusion Detection (RAID) Conference*, September 2004.

[29] C. Lu, D. Chen, and Y. Kou, “Multivariate spatial outlier detection,” *International Journal on Artificial Intelligence Tools, World Scientific*, vol. 13, pp. 801–812, December 2004.

[30] D. E. Knuth, *The Art of Computer Programming, 2nd Ed. (Addison-Wesley Series in Computer Science and Information*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1978.

[31] G. Jiang and G. Cybenko, “Temporal and spatial distributed event correlation for network security,” in *In American Control Conference (ACC)*, 2004.

[32] D. Qu, B. M. Vetter, F. Wang, R. Narayan, S. F. Wu, Y. F. Jou, F. Gong, and C. Sargor, “Statistical anomaly detection for link-state routing protocols,” in *ICNP ’98: Proceedings of the Sixth International Conference on Network Protocols*, (Washington, DC, USA), p. 62, IEEE Computer Society, 1998.

[33] T. Ptacek and T. Newsham, “Insertion, evasion, and denial of service: Eluding network intrusion detection,” tech. rep., Secure Networks, Inc., 1998.

[34] A. Lazarevic, L. Ertoz, V. Kumar, A. Ozgur, and J. Srivastava, “A comparative study of anomaly detection schemes in network intrusion detection.,” in *Proceedings of the Third SIAM International Conference on Data Mining*, 2003.

[35] C. Krügel, D. Mutz, W. Robertson, and F. Valeur, “Topology-based detection of anomalous bgp messages.,” in *RAID* (G. Vigna, E. Jonsson, and C. Krügel, eds.), vol. 2820 of *Lecture Notes in Computer Science*, pp. 17–35, Springer, 2003.