A PLATFORM FOR CREATING EFFICIENT, ROBUST, AND RESILIENT PEER-TO-PEER SYSTEMS

A Dissertation

Submitted to the Faculty

of

Purdue University

by

David J. Zage

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

May 2010

Purdue University

West Lafayette, Indiana

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Zage, David J. Ph.D., Purdue University, May 2010. A Platform for Creating Efficient, Robust, and Resilient Peer-to-Peer Systems. Major Professor: Cristina Nita-Rotaru.

The rapid growth of communication environments such as the Internet has spurred the development of a wide range of systems and applications based on peer-to-peer ideologies. As these applications continue to evolve, there is an increasing effort towards improving their overall performance. This effort has led to the incorporation of measurement-based adaptivity mechanisms and network awareness into peer-to-peer applications, which can greatly increase peer-to-peer performance and dependability. Unfortunately, these mechanisms are often vulnerable to attack, making the entire solution less suitable for real-world deployment. In this dissertation, we study how to create robust systems components for adaptivity, network awareness, and responding to identified threats. These components can form the basis for creating efficient, high-performance, and resilient peer-to-peer systems.

1 INTRODUCTION

The rapid growth of communication networks such as the Internet and ad hoc wireless mesh networks has spurred the development of numerous collaborative peer-to-peer (P2P) systems. These systems employ distributed resources to achieve some end goal in a distributed manner. The nodes in the system act as equals, simultaneously functioning as both clients and servers. Each node provides part of its own resources (storage, network bandwidth, *etc.*) in order to facilitate the goal of the overall system such as providing video content or creating shared collaborative spaces. As more nodes arrive in the system, greater demands are made of the system, but further total capacity is also added by the sharing of individual node resources. This allows the systems to efficiently scale to hundreds or even thousands of users and support a wide variety of usage patterns. This scalability, flexibility, and power drives the growth of peer-to-peer systems and the desire to identify further applications which can effectively use this computational model.

In order to understand the current popularity of peer-to-peer systems, one just has to examine the applications which they make possible. The following applications, while just a subset of the myriad of possible peer-to-peer applications, have been identified as some of the most useful and promising:

- ***File Download and Distribution*** Currently, one of the most common uses of peer-to-peer applications is sharing files between network participants. Each participant has the ability to download files and is expected to provide files for download, distributing the cost and system load over multiple users. Besides load balancing, the applications provide robustness against nodes leaving the network and node failure as multiple copies of the data are often stored across the system. Example systems include BitTorrent [1], Emule [2], Gnutella [3],

Direct Connect [4], and OpenNap [5]. Each of these systems claims user membership in the millions, with thousands of active users per day.

- ***Voice over IP*** Systems such as Skype [6] utilize a peer-to-peer model in order to efficiently scale to millions of nodes (customers) without the need for fixed infrastructure.

- ***Video Broadcasting and Real Time Television*** These applications provide the ability to efficiently multicast video on demand to a large number of users. The use of peer-to-peer technology allows the systems to scale by distributing processing and bandwidth loads. Moreover, some of these systems are adaptable to network conditions, providing increased throughput and connectivity over a traditional server-client model. There has been a considerable amount of research and commercial interest in video multicasting, resulting in systems such as End System Multicast (ESM) [7], Scribe [8], SplitStream [9], Nice [10], Overcast [11], Bullet [12], Chainsaw [13] and CoolStreaming/DONet [14]. Along with on-demand video, there recently has been an explosion of interest in real-time peer-to-peer streaming of audio and video, resulting in systems such as SOPCast [15], PPLive [16], PPStream [17], TVAnts [18], QQLive [19], Feidian [20], Mysee [21], Pdbox [22], PPMate [23], UUSee [24], VGO [25], CTV [26], StreamerOne [27], TVUnetworks [28], Joost [29], Veoh [30], and Zattoo [31].

- ***Distributed Computation and Data Processing*** Projects such as Folding@home [32] utilize the unused resources of each node to study computationally and storage intensive applications such as protein folding, DNA sequencing, and star analysis. Computing power previously realized only in high cost supercomputing centers can now be harnessed for a fraction of the cost using individuals' personal computers.

As the number of networked devices is forecasted to reach 15 billion by 2010 [33], peer-to-peer systems will only continue to grow in importance.

1.1 Motivation

As peer-to-peer systems continue to evolve, there is an increasing focus on creating high-performance systems. This desire for performance has led to the incorporation of *adaptivity mechanisms* and *network awareness* into current systems. Adaptivity mechanisms allow the systems to satisfy changes in user requirements or fulfill performance guarantees under changing conditions by dynamically changing the system structure. This fluidity allows the application to improve suboptimal system structure resulting from random initial neighbor selection, aggressive partition repair, group membership changes, and transient conditions in the underlying physical network. Additionally, an increasing number of peer-to-peer applications are designed to be network aware and not blindly choose neighbor nodes at random. In order to avoid excessive network measurements or the overhead of reactive routing protocols, applications depend on accurate and efficient services that allow hosts on the Internet to determine network properties such as the latency between nodes. The use of virtual coordinate systems to enable peer-to-peer applications to accurately and efficiently select the best nodes for a desired task is one concrete example of such a service. Both adaptivity and network awareness have been shown to significantly improve peer-to-peer system performance.

In contrast to great deal of research that has resulted in the rapid creation of high-performance peer-to-peer solutions, little work has focused on creating *robust* and *resilient* peer-to-peer systems. While the inclusion of adaptivity and network awareness can greatly increase the performance and dependability of peer-to-peer applications, care must be taken not to introduce new attack vectors in which the attacker is able to target these system components. Unfortunately, in many of the proposed systems, these mechanisms are vulnerable to attack, making the system unsuitable for real-world deployment. For example, one single malicious node attacking the adaptation mechanism of the system can significantly impact the performance of the system.

Additionally, many of the current peer-to-peer systems and components were designed without security principles in mind [34]. This has left gaps in current system design, leaving the systems vulnerable to attack. This is especially troublesome as many of the peer-to-peer systems are designed to operate in open environments and have open memberships. Traditional security mechanisms, such as digital signatures and access control, are either not applicable or ineffective since the attackers have the necessary credentials to be a part of the system and cannot be mitigated using these mechanisms. It is therefore critical that integrating high-performance components into peer-to-peer applications leads to improved performance while remaining resilient to malicious activity.

1.2 Thesis Focus and Contributions

This thesis constitutes an effort to reconcile the performance and security of peer-to-peer systems through the development of robust system components. We identify and characterize three critical components we envision will form the basis for creating efficient, high-performance, resilient peer-to-peer systems: robust adaptation mechanisms, robust network awareness, and responsiveness to identified threats. For each component, we identify its associated security risks, provide techniques to make each component robust to malicious activity, and demonstrate how to effectively integrate it into current systems. Along with real-world experimentation, we provide analyses of the components in order to reason about their security and integration into peer-to-peer applications.

We summarize our key contributions:

- We demonstrate the vulnerability of adaptive system components and propose multiple solutions designed to prevent adversarially-induced mis-adaptation. We demonstrate the effectiveness of our solutions in the context of the unstructured multicast system, ESM [35], by incorporating robust adaptation tech-

niques into the system. In addition to real-world experimentation, we provide a theoretical analysis of the robustness achieved by our proposed design.

- We create a viable solutions for robust network awareness through the creation of a robust virtual coordinate system. We classify and demonstrate the vulnerabilities of decentralized virtual coordinate systems and create a robust solution by using spatial and temporal correlations to perform context-sensitive outlier analysis. In keeping with the design goals of virtual coordinate systems, our solution is efficient, distributed, and contain no trusted network components.

- We provide a viable solution for responding to threats once they have been identified in the peer-to-peer system. Through the aggregation of local information on malicious behavior, we utilize a robust reputation system to build a global reputation for each system node. Using this global reputation, the peer-to-peer system can take appropriate reactive steps to the malicious activity. We demonstrate the effectiveness of utilizing reputation systems in the context of the unstructured multicast system, ESM [35].

- We elucidate the impact malicious attacks against lower-level system components have on higher-level applications, showing how the system is only as strong as its "weakest link". We demonstrate how the inclusion of components for robust network awareness into distributed information retrieval applications can optimize system performance in *both* non-attack and attack scenarios.

1.3 Thesis Roadmap

The rest of the thesis is organized as follows: We provide an overview of peer-to-peer systems and our adversarial model in Chapter 2. We demonstrate the vulnerability of adaptive mechanisms in peer-to-peer systems to malicious activity and provide solutions to mitigate this activity in Chapter 3. We identify virtual coordinate systems as one method to provide network awareness and demonstrate techniques to

increase their robustness to attack in Chapter 4. We provide a viable solution for responding to malicious activity through the integration of reputation systems into peer-to-peer applications in Chapter 5. We examine and experimentally demonstrate how peer-to-peer applications can benefit from our system components in Chapter 6. We review related work in Chapter 7. Finally, we present our conclusions in Chapter 8.

2 SYSTEM AND ATTACKER MODEL

In this chapter, we describe our basic system model and present the adversarial model under which we study the vulnerabilities of peer-to-peer systems and their components.

2.1 Peer-to-peer Systems

A peer-to-peer system constitutes a set of N nodes connected via a set of bi-directional virtual network links to form an *overlay network*. Each virtual link is composed of one or more physical links on the underlying network. Such networks offer properties such as functionality, performance, robustness, anonymity, or isolation that are not typically possible with the existing infrastructure.

Overlay networks can be divided into two basic categories: *unstructured* overlay networks and *structured* overlay networks. In unstructured overlay networks, there are no constraints in the selection of the neighbor set and no imposed constraints in the resulting overlay [36]. On the other hand, structured overlay networks create an overlay topology which offers pre-defined bounds and organizational invariants by constraining the set of nodes eligible to become neighbors of a given node [37, 38]. Unstructured overlay networks are often easier to maintain (as any network node can be chosen as a neighbor) and allow for greater flexibility and adaptability than structured overlays. However, the constraints imposed by the structured variant can guarantee average case performance and can make structured overlay networks easier to secure. In the end, the type of overlay network employed by a peer-to-peer system is highly dependent on the desired performance bounds and envisioned network stability.

In our work, we assume the overlay construction is completely self-organized, forming unstructured networks. No node has complete knowledge of the system

topology. While nodes can be connected through a variety of manners, we assume traditional wired links unless otherwise noted. In order to join the overlay network, a node n contacts a *membership server* and receives a small set of network nodes with which to communicate. This set of network nodes is known as a *neighbor*, *reference*, or *peer set* of n. Also, depending on the application, the overlay often includes a dedicated source or seed which creates the initial data or traffic being disseminated through the network. Each of the nodes has similar functionality and often plays the role of both a server and a client.

2.2 Attacker Model

In general, attackers fall into two categories: outsiders and insiders. An outside attacker does not possess the credentials (*e.g.*, secret keys, digital certificates, *etc.*) necessary for passing the imposed authentication process and is prevented from accessing system resources. Inside attackers are those entities who have legitimate access to the system and can participate according to the system specifications (*i.e.*, authenticated entities within the system). As peer-to-peer application design often pushes functionality to end-nodes to achieve better scalability and performance, it makes these systems vulnerable to attack as trust is pushed to the fringes of the network where end-nodes are more likely to be compromised than core routers [39]. The open nature of peer-to-peer applications leads us to assume all attackers are insider as it is straightforward for an outsider to obtain legitimate credentials.

Examining insider attackers, we can further classify them based on their motivations: rational attackers performing egoistic actions designed to maximize their own benefit [40, 41] versus Byzantine attackers performing arbitrary actions from which they may not directly benefit [42, 43]. A common goal in the construction of many peer-to-peer systems is that of incentivizing nodes to faithfully follow system specifications and not manipulate protocols at the expense of system performance [44]. To achieve this adherence to the design specifications, systems employee various mea-

sures, such as the tit-for-tat schemes [45] or creating balanced messaging patterns in which all nodes share the same burden [46]. While it often achievable to obtain faithful participation from rational nodes, it is difficult to improve or repair systems that are vulnerable to Byzantine attackers as they do not respond to incentives [47]. Additionally, even if rational nodes are present, the presence of Byzantine nodes will exasperate their effect and exert a greater detrimental effect on the performance and stability of the overall system [47, 48].

Due to these motivations, unless otherwise noted, we consider a constrained-collusion Byzantine adversary model similar to that proposed by Castro *et al.* [49], with a system size of N and a bounded percentage of malicious nodes f $(0 \leq f < 1)$. The malicious nodes behave arbitrarily and are only limited by the constraints of the cryptographic methods deployed [50]. The set of malicious nodes may collude. We assume a malicious adversary has access to all data at a node as any legitimate user would (insider access), including cryptographic keys stored at a node. This access can be the result of the adversary bypassing the authentication mechanisms or compromising a node through other means. As malicious nodes have insider access, nodes cannot be completely trusted although they are authenticated.

3 CREATING ROBUST ADAPTIVE COMPONENTS

With the tremendous growth of the Internet, a wide range of applications taking advantage of peer-to-peer technologies have emerged in recent years. In fact, recent studies indicate that over 60% of all Internet traffic is generated by peer-to-peer systems [51]. As these application are often deployed in rapidly changing environments, *adaptive components* have become critical building blocks, enabling them to satisfy system requirements under changing conditions.

To provide increased performance and fault tolerance to benign failures, many peer-to-peer applications are built utilizing adaptive overlay networks [14, 35, 52–54]. In general, the adaptation improves suboptimal performance conditions experienced in many peer-to-peer applications. Each node monitors its performance and that of a subset of the network known as a peer set. When the measured performance becomes inadequate, the node will change its upstream node(s) in the network. We refer to this process as *adaptation* and the mechanisms used to achieve it as *adaptation mechanisms*. There are no restrictions in the selection of the peer set nor in the adaptation process, resulting in an unstructured overlay network.

The proliferation of peer-to-peer applications using adaptive mechanisms raises questions about how to design and deploy these applications in a secure and robust manner [55]. In particular, attacks that exploit the adaptivity mechanisms employed by many overlay networks can be extremely dangerous because they target the overlay construction and maintenance while requiring no additional communication bandwidth on the attacker side. Such attacks can allow an adversary to control a significant part of the traffic and further facilitate other attacks such as selective data forwarding, cheating, traffic analysis, network partitioning, and disconnecting victim nodes. Not only are the attacks damaging from an end-user perspective, but they also have a large economic impact, potentially costing businesses millions of

dollars in lost revenue [56]. As the number of Internet businesses grow and compete for the same niche markets, such as streaming video using peer-to-peer overlays, the risk of subversive attack will only increase. In fact, reports of such attacks designed to disrupt the service provided by Internet businesses are starting to appear in the media [57].

In this chapter, we focus on insider attacks against adaptation mechanisms in unstructured overlay networks. We make the following contributions:

- We demonstrate the benefit for the attacker to subvert adaptation mechanisms over other methods of attack such as simple data dropping by showing the susceptibility of adaptation mechanisms to malicious attacks through real-life deployments on the PlanetLab [58] Internet testbed using the End System Multicast (ESM) [35] system.

- We propose to increase the resiliency of the system to attacks and mitigate the effect of malicious adversaries by aggregating and correlating data-plane and control-plane information to determine the reliability and utility of received information. This information is incorporated into the adaptation process of the overlay network, constraining the ability of the attacker to lie to honest nodes and increasing the robustness and stability of the network. Network topology has previously been used to detect data-dropping attacks [59]. In our approach, we use network topology to detect and mitigate a more general set of attacks against the adaptation process used to augment other attacks such as data-dropping.

- We propose techniques to reduce incorrect or 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 logical networks share overlapping physical links.

- We demonstrate the effectiveness of the identified attacks and the benefits of both of our defense mechanisms through deployments of ESM on the PlanetLab Internet testbed.

The rest of this chapter is organized as follows: We provide an overview of adaptive peer-to-peer multicast systems in Section 3.1 and attacks targeting adaptation in Section 3.2. We propose and demonstrate the effectiveness of aggregating and correlating data-plane and control-plane information at increasing the robustness of the system in Section 3.4. We demonstrate the utility of context-sensitive outlier analysis at preventing unnecessary adaptation in Section 3.5. Finally, Section 3.6 concludes this chapter.

3.1 Adaptive Peer-to-peer Multicast

Multicast functionality was initially envisioned to run on the IP layer [60]. However, limited deployment and other limitations have driven the need to provide an alternative architecture where multicast functionality is implemented by end systems and not by the infrastructure supporting routers. Peer-to-peer multicast is designed to efficiently disseminate information to a set of distributed nodes connected to an overlay network of logical links across an actual physical infrastructure. Recently, a great deal of research and interest has been generated in this area by the establishment of increasingly popular peer-to-peer streaming sites [15–31]. To deal with different demands placed on the multicast system by the varying environments and requirements, many different peer-to-peer multicast overlay networks have been created [7–14].

For our research, we consider an overlay network providing support for single-source broadcasting applications that are high-bandwidth (hundreds of kilobits to megabits per second) and real-time. The system consists of a set of nodes and a data source communicating via unicast links. All nodes receive data and contribute to the routing process by forwarding data. The overlay construction is self-organized and

distributed. No node has complete knowledge of the dissemination topology. Each node maintains a peer set, a routing table, and the upstream node forwarding the data, referred to as the node's parent. The peer set is a subset of nodes that are currently reachable in the overlay from which performance information is gathered. This set is bootstrapped at join time by contacting the source and is continually updated via a membership protocol. There are no constraints placed on the members of a node's peer set. The routing table represents a set of nodes that the node is responsible for routing data to, also referred to as *children*. The size of this set is limited by a system characteristic termed the *saturation degree*, representing the number of concurrent data streams the node is able to support before saturating the allocated bandwidth of the underlying physical network link.

3.1.1 End System Multicast(ESM)

ESM [7] is a multicast system used for broadcasting live events such as academic conferences, including major computer science conferences such as SIGCOMM and INFOCOM. We provide a high-level description below. For further details, the reader is referred to the work by Chu *et al.* [7].

As seen in Figure 3.1, ESM forms a peer-to-peer overlay tree over the physical infrastructure for distributing multicast content. One key idea behind the construction of the ESM overlays 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.

In order to make the adaptation decisions, each node maintains a set of performance variables for each member of its peer set collected through passive observation and probing of peers. These variables consist of bandwidth (throughput), latency

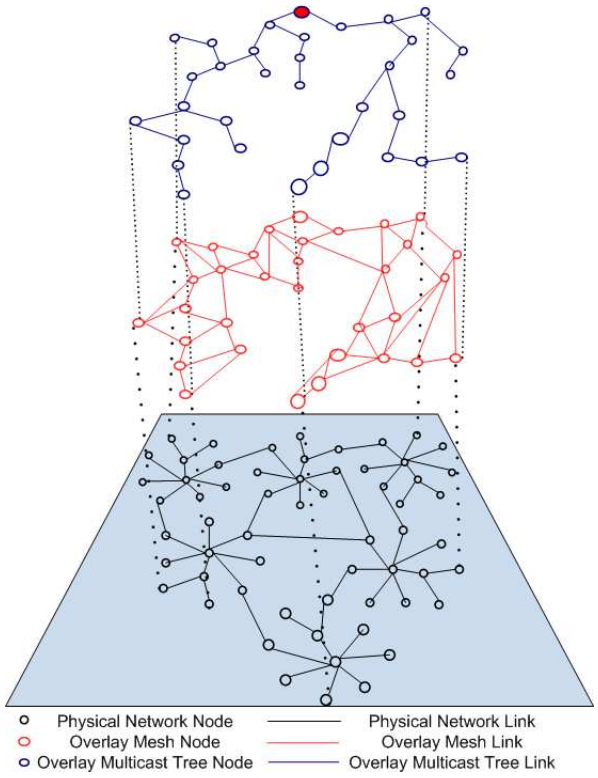


Figure 3.1. Illustration of the physical network and the two logical networks of the ESM overlay multicast system: the adaptive overly network and the multicast tree.

(one-way delay), and round-trip-time (RTT). As ESM uses TCP as the data transport protocol, loss rate is not considered. Once the data has been received at a node, the node must decide if it needs to change its parent in the overlay to improve the application performance. To improve the quality of the data and subsequent adaptation decision, an extensive set of mechanisms is employed, including but not limited to data sampling, data smoothing, decision randomization, and hysteresis.

Once the performance of a node has become inadequate, it uses the collected metrics to select a new parent from its peer set. The parent selection algorithm used in ESM is presented in Algorithm 1. In order to select a new parent, a node first computes a list of potential candidates from its neighbor set. Nodes which are currently saturated, descendants, or did not respond when recently probed are not considered. If there is no utility gain, no node is selected and the process will be repeated in the next cycle. If several nodes are candidates, then the first candidate is selected as the new parent. The selection process uses hysteresis to generate a negative bias against nodes that have performed poorly in the past.

We selected ESM as a representative peer-to-peer multicast system because of its maturity, extensive deployment, and the advanced set of adaptation techniques it employs.

3.2 Attacks Targeting Adaptation

Any adaptive network protocol based on measurements involves periodically observing and estimating the network conditions, followed by making an adaptation decision. For unstructured multicast overlays, the adaptation decision consists of a node selecting a new parent based only on weighing the associated costs versus benefits quantified through a utility function.

Previous work studied the quality of the data observation and estimation, as well as the ability of the metrics to accurately reflect the state of the network. Examples of factors that influence data quality include data freshness, variability and noise. Mech-

Algorithm 1: Parent Selection Algorithm Used in ESM

Input: Set of probe responses tuples (<bandwidth, latency, RTT>)

Output: New Parent Node

Create list of potential parent candidates, PCL, excluding nodes: 1) currently saturated, 2) descendant of this node, and 3) unresponsive

foreach *rnode in* PCL **do**
    `// Combined Metrics Component`
    if *(utilityGain(throughput, latency)* $\geq$ *currentUtility*1.1)* **then**
        keep *rnode* in PCL;
    else
        remove *rnode* from PCL;
    end
end
foreach *rnode in* PCL **do**
    `// Hysteresis Component`
    if *(rnode has performed well in the past)* **then**
        `// Dampening Component`
        if *(local node has not switched parents recently)* **then**
            `// Randomization Component`
            if *(node switch probability > rand())* **then**
                Select *rnode* as next parent;
                return *rnode*
            end
        end
    end
end
return no change

anisms proposed to address these issues are data sampling [61], data smoothing [62], metric construction [63], as well as data summarization and aggregation [64]. Previous work also studied instabilities [65–67], such as the oscillatory behavior referred to as flapping, occurring when nodes rapidly switch between seemingly equal alternatives. New techniques such as utility discretization [67, 68], randomization [67, 69], damping [66], and hysteresis [61, 69] were deployed to mitigate these phenomena and provide a tradeoff between responsiveness to change and instability.

None of mechanisms described above take into account adversarial environments. While adaptivity can greatly increase the performance and dependability of peer-to-peer applications, care must be taken not to introduce new attack vectors in which the attacker is able to target the adaptivity mechanism.

In an adversarial network, compromised overlay nodes can take advantage of the adaptation process to gain control over overlay traffic by lying about their observed performance metrics to manipulate the path selection of the overlay topology. We classify three types of attacks:

- ***Attraction*** attacks are a form of "bait-and-switch" attack in which a malicious node manipulates the observed data in order to persuades a victim to attach to a malicious parent in the tree.

- ***Repulsion*** attacks seek to reduce the attractiveness of other nodes or misrepresent their ability, with the ultimate goal of free-loading, traffic pattern manipulation, or augmenting attraction attacks.

- ***Disruption*** attacks target the availability of the network by using the adaptation process to turn the system against itself causing a state of constant change.

Each of the attacks cause undesired changes in the overlay network which can lead to further attacks such as manipulating data, performing traffic analysis, performing man-in-the-middle attacks, causing disruption for specific nodes by isolating them, or selectively dropping packets for a particular destination. For detailed information on each attack, we direct the reader to [70].

3.3 Preventing Adversarial-Induced Adaptation

When a node in a peer-to-peer system makes an adaptive decisions, there are two types of information used to reach this decision: performance metrics collected locally and performance metrics obtained from neighboring nodes. By blindly accepting information reported by potentially malicious nodes, a benign node may make incorrect decisions. In order to increase the chances of making "good" decisions, we propose two methods to improve the decision process and prevent incorrect adaptations:

- ***Correlation of data-plane and control-plane information*** We increase the resiliency of peer-to-peer systems to attacks by improving the decision process through the aggregation and correlation of data-plane and control-plane information. This information is incorporated into the adaptation process of the overlay network to determine the reliability of received information, constraining the ability of the attacker to lie to honest nodes and increasing the robustness and stability of the network.

- ***Outlier detection*** We prevent incorrect adaptations by detecting and filtering out outliers in the metrics reported by probed nodes. We detect inconsistencies in reported metrics by performing outlier analysis which evaluates the temporal and spatial correlations on the information received from probed nodes used in the adaptation decision process.

Each of the techniques can be used individually or in conjunction with one another. They are also widely applicable as many peer-to-peer systems exchange similar performance metrics.

3.4 Using Information to Add Robustness

The impetus behind the attacker's ability to subvert the overlay networks is the fact the attacker can influence the adaptation process by manipulating the performance metrics. This stems from the assumption that peer nodes are altruistic and

respond with correct metrics to queries from any node. We propose to increase the resiliency of the system by aggregating and correlating both data-plane and control-plane information, allowing each node to use the derived information to make better adaption decisions.

The control-plane information pertains to the metrics necessary to manage connectivity while the data-plane metrics are derived from overlay data and are often used to improve application performance. In general, the data-plane information reported by a node was measured by that node while control-plane information was received from external sources (such as parents or the source). One important facet of our solution is that it uses information *already* present in the system, avoiding extra link stress. It should also be noted that many adaptive protocols utilize similar metrics for adaptation, to which our solution can readily be applied.

Table 3.1
ESM probe response components

Metric	Data Origin
Bandwidth received from parent	Data-Plane
Latency from the parent	Data-Plane
Latency from the source	Data-Plane
Time connected to parent (stay time)	Data-Plane
Current parent in the overlay structure	Control-Plane
Number of children	Control-Plane
Path to the source of the data	Control-Plane

During the system lifetime, each node periodically performs a *probe cycle* in which it sends requests for system metrics to its peer set and subsequently receives responses for a fixed interval of time. The contents of each probe response along with the origin of the data are listed in Table 3.1. Once the time interval expires, a node uses this gathered information, along with locally measured performance, to determine if it

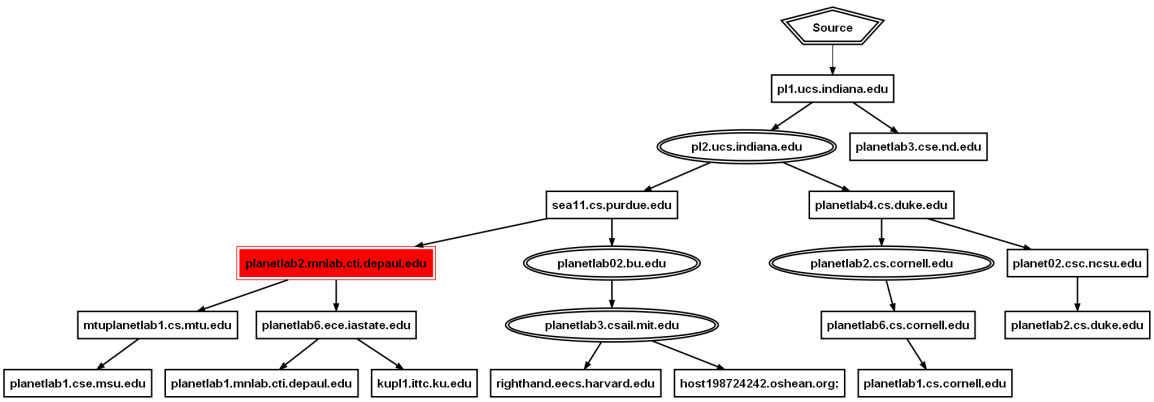


Figure 3.2. The aggregation of data-plane and control-plane information into a graph structure used to improve adaptation decision quality. The square nodes represent peers from which data has been received while the ellipses are placeholders used to complete the graph structure. The highlighted node represents a malicious node reporting it has no children.

should change parents. If a malicious insider responds with falsified information, it can bias the adaptation decision and cause incorrect choices.

As part of this solution, a node receives probe responses and aggregates this information into a new, in-memory graph structure called the *path graph*. Pictured in Figure 3.2, the root of the path graph is the data source and all other vertices contain information pertaining to a specific node. Each of the edges represent a connection in the overlay topology. Since only a subset of the entire overlay is contacted by each node, nodes for which information is not directly obtained are represented by named placeholders. For example, in Figure 3.2, the ellipses depict nodes whose existence were discovered by examining the path to the source used by other probed nodes. The location of the probing node in the path graph (if it close to the source, is a leaf, *etc.*) will vary depending on the node. The path graph is reset after each adaptation attempt.

The intuition behind our solution is that while each node makes adaptation decisions independently, many of the choices can be ameliorated by having data-plane and control-plane information associated with a partial view of the network topology. Even though unstructured overlays have no tight topological invariants such as those present in structured overlays, the system forms a stable topology which can be examined for inconsistencies. By correlating data from multiple sources into a path graph at each node, it becomes possible to check the consistency of the reported information and constrain the attackers ability to lie about system metrics.

Using this path graph, benign nodes can prevent many unnecessary adaptations by avoiding the use of falsified data in adaptation decisions. During a node n's adaptation process, when n is considering a new potential parent p, n will check the sanity of p's contextual information contained in the path graph. A node p will be avoided if the number of children graphed is inconsistent with what is reported by p or is greater than the system saturation degree, if the bandwidth reported by a current child is inconsistent with what is reported by the parent, the reported path is inconsistent with the graph, or if the time multiple children are connected to a parent is small. The more often these inconsistencies occur, the more often a node will be avoided as a parent. For example, if the highlighted node in Figure 3.2 reports having no children while the path graph has two children for the same node, the node is malicious and should not be consider for a parent change. Using the path graph, a benign node is able to augment its adaptation decision using Algorithm 2, allowing for more robust decisions.

Algorithm 2: Procedure to exclude malicious nodes from the adaptation process.

Input: Potential Adaptation Candidates List ($PACL$) and the Path Graph PG

Output: Updated $PACL$

```
1  foreach rnode in PACL do
       // Inconsistent number of children
2      if (rnode.numChildren != PG.rnode.numChildren) then
3          remove rnode from PACL;
       // Too many children
4      else if (PG.rnode.numChildren > SystemSaturationDegree) then
5          remove rnode from PACL;
       // Inconsistent bandwidth reported
6      else if (rnode.BW - PG.rnode.children.ActualBW > (SourceRate*.1)) then
7          remove rnode from PACL;
       // Inconsistent path to the source
8      else if (rnode.path != PG.rnode.path) then
9          remove rnode from PACL;
       // Too many small stay times
10     else if (PG.rnode.children.stayTimes < 100 sec) then
11         remove rnode from PACL;
12     else
13         keep rnode in PACL;
14     end
15 end
```

3.4.1 Discussion

It should be noted that the goal of our defense technique is not to create a "perfect" solution, but to provide improved robustness in a lightweight solution that can be readily incorporated into a variety of overlay networks. While the effect of the malicious nodes can be greatly diminished, there may still be some degradation in the system performance. This technique should be considered one useful tool in the tool-

box of system designers for enhancing the robustness of the system without adding extra overhead or excess complexity.

3.4.2 Experimental Evaluation

To study the effects of the attacks and our defense technique under real-world conditions, we conducted experiments on the widely-used PlanetLab [58, 71] Internet testbed. As we can see from Figure 3.3, Planetlab is a global research network that currently consist of over 1000 nodes distributed across 487 sites located on six continents. It provides a research platform for large-scale distributed experimentation of peer-to-peer systems over the Internet [72]. To mitigate the possible limitations of using a testbed, such as those discussed by Spring *et al.* [72], multiple experiments were conducted at different times of the day and different days of the week. Further, experimental nodes were selected randomly for different experiments to validate the statistical significance of results and nodes were chosen to span multiple operational and administrative domains. Each experiment was conducted multiple times and the results were averaged.

The baseline configuration for the following experiments consists of a 30 minute deployment of 100 nodes in which the nodes join after the experiment begins and leave before it ends, with an average participation time of 26 minutes. Each node is probed every seven seconds, each node probes 30 peers, the saturation degree of benign nodes is four, and the source streaming rate is 480Kbps. We use a bit rate of 480 Kbps as it is sufficient to transmit video at two different qualities of audio. All of the following experiments use these parameters unless otherwise noted.

The ESM System Under Attack

We study the effect different percentages of malicious nodes have on the overlay topology if they lie about their performance metrics to gain beneficial positions in the tree. As mentioned in Section 3.1.1, ESM uses several techniques designed to

Figure 3.3. Illustration of the physical location of a subset of the PlanetLab nodes used for experimental evaluations.

tolerate *benign* errors, it has no built-in mechanisms to defend against malicious nodes and is thus prone to instability and poor performance when under attack. For the experiments that incorporate our defense technique, we integrate it into the ESM decision process *prior* to the preexisting data cleansing techniques. As our defense and the existing techniques are orthogonal, they do not conflict with each other. Additionally, using our technique first allows for the removal of much of the malicious data during an adaptation decision, allowing the existing data cleansing techniques to take place in a environment similar to that of a benign system.

To determine the efficacy of subverting the adaptivity of the system, we examine the following attack scenarios in our experiments:

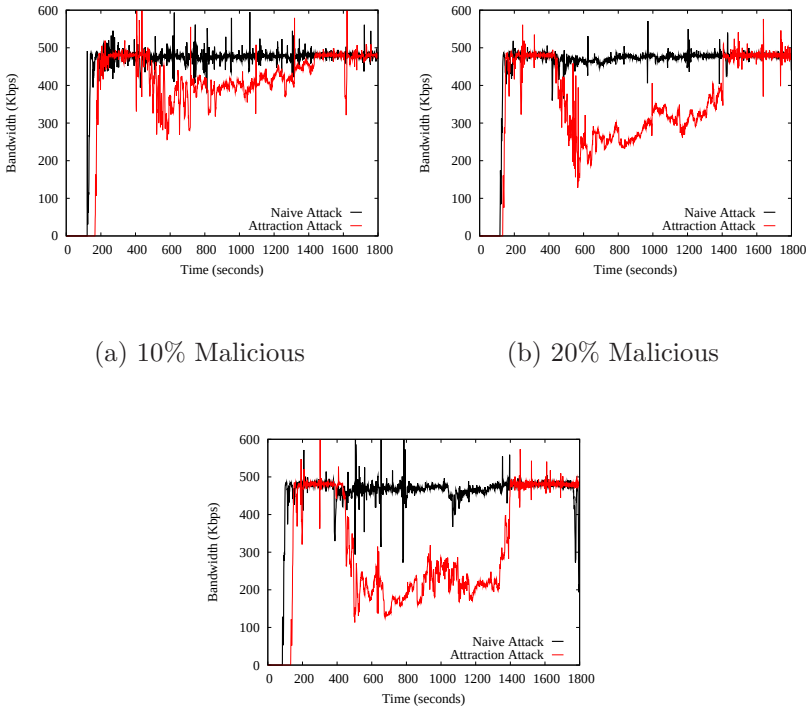
• **Naive Attack**: This is a naive attack method in which the malicious coalition of nodes simply obtain random positions in the overlay and, after an initial starting period, drop a percentage of the data.

• **Attraction Attack**: This is the attraction attack discussed in Section 3.2, in which the malicious nodes lie about their performance metrics. In this attack, the malicious nodes report having the best bandwidth (480Kbps), smallest latency (0ms), and no saturation. After an initial starting period which the malicious nodes use to optimize their positions in the overlay network, they drop 90% of the data as this percentage bypasses ESMs mechanisms for tolerating benign errors.

In both attack scenarios, other secondary actions besides dropping data could be performed. However, dropping data visibly demonstrates the severity of the attacks.

Impact of Percentage of Malicious Nodes. Figure 3.4 demonstrates the impact different percentages of malicious nodes can exert on the bandwidth of benign nodes for the two attack methods. In the experiments, malicious nodes start dropping 90% of the data traffic received through the data dissemination tree five minutes after joining the overlay and continue to drop data for the next fifteen minutes. The nodes only drop data for the specified interval to demonstrate the system bandwidth returns to the source rate and the degradation is caused solely by the attack. We vary the percentage of malicious nodes to 10%, 20%, and 30% of the overlay size to demonstrate the performance degradation that results when more nodes behave maliciously. As can be seen from the graphs of Figure 3.4, the naive attack has little effect on the bandwidth of the system. Even when 30% of the network is comprised of malicious nodes (Figure 3.4(c)), the average bandwidth only drops by 20Kbps to 460Kbps. This is not the case for the attraction attack, in which the average bandwidth of the system markedly decreases as more malicious nodes are introduced. For example, when 30% of the network is malicious (Figure 3.4(c)), the average bandwidth is reduced by over 50%, from 480Kbps to 220Kbps.

The difference between the two attacks is due to the fact that without the supplementary mechanism of attacking the adaptivity of the system, most of the malicious coalition is unable to obtain advantageous locations (positions with many children) in the overlay and are rendered ineffective. This insight elucidates not only the potential



(a) 10% Malicious

(b) 20% Malicious

(c) 30% Malicious

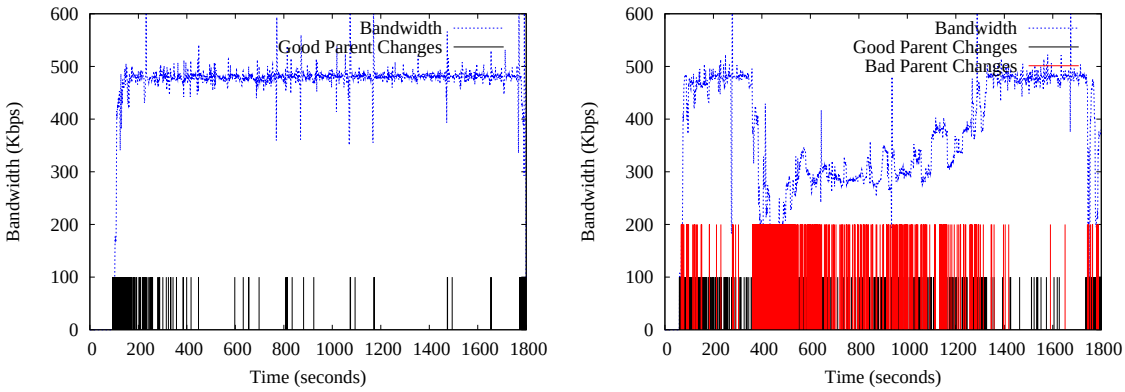
Figure 3.4. Average system bandwidth for a 100 node ESM overlay deployed on PlanetLab under different percentages of attackers. The graphs depict a naive data-dropping attack and more powerful attraction attack which drops data and lies about performance variables.

harm malicious nodes can cause by attacking the adaptivity of the system, but the need to makes these mechanisms more robust.

Table 3.2
System instability as represented by the number of adaptations of an ESM overlay of 100 nodes on PlanetLab

Experiment	Changes to Malicious Parents	Total Parent Changes
No lying	0	411
10% Naive	74	488
10% Attraction	392	1507
20% Naive	148	711
20% Attraction	674	1671
30% Naive	235	717
30% Attraction	931	1634

System Instability. Not only does the malicious activity decrease the system utility, it also leads to system instability and unnecessary adaption. Under benign conditions, when a node determines the advertised performance metrics do not match the actual performance, the node will attempt to connect to a new, better-performing parent. In the presence of adversaries, this change may actually result in performance loss, necessitating further adaptation and causing excess churn in the network. We can visually see the instability caused by the malicious nodes in Figure 3.5(b), where the number of adaptations caused by an attraction attack is four times higher than in Figure 3.5(a). Table 3.2 presents the average number of adaptations for different percentages of malicious nodes. As the percentage of malicious nodes increases, so does the number of adaptations. Also, we note that the naive attack causes far less instability in the system due to its inability to gain advantageous locations in the dissemination structure.



(a) 0% Malicious

(b) 20% Malicious

Figure 3.5. The average system bandwidth and number of parent changes for for a 100 node ESM overlay deployed on PlanetLab with 0% and 20% malicious nodes. The short, black impulses represent changes to benign parents while the taller, red impulses represent changes to malicious parents.

Attack Strength. To quantitatively compare experiments with different percentages of malicious nodes and possible defense techniques, we utilize an augmented form of the relative strength of attack measure τ [73].

The relative strength of a particular attack as is defined as:

$$\tau = \frac{B_{norm} - B_{adv}}{B_{norm} \times N_{adv}} \times 100 \tag{3.1}$$

where B_{norm} and B_{adv} represent the average throughput in the absence and presence of adversaries respectively, and N_{adv} is the number of adversaries. Intuitively, τ represents the amount of damage an attack created in the system normalized by the number of adversaries. The greater the performance degradation observed in the system (the difference between B_{norm} and B_{adv}), the higher the value of τ and the more damage an attack inflicts on the overlay.

Figure 3.6 depicts τ varying over the percentage of attackers for both attacks. As expected, the naive attacks resulted in very low τ values as they were largely ineffec-

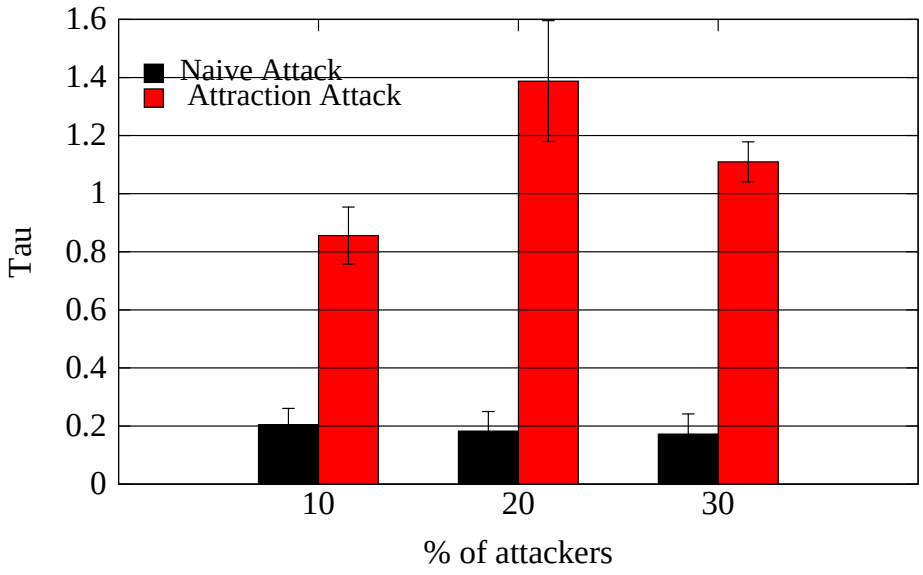


Figure 3.6. Attack strength for different percentages of malicious nodes for ESM deployments of 100 nodes on PlanetLab using the naive and attraction attacks.

tive. However, we can see the attack focusing on the adaptivity of the system, the attraction attack, has significant impact on the performance of the system for even a small percentage of malicious nodes (10%) and thus has a high τ value. Increasing the percentage of malicious nodes yields higher τ values, with the maximum effectiveness for the attacker occurring when 20% of the network was malicious. With percentages greater than 20%, the average system bandwidth continues to decrease, but relative measures like τ experience diminishing returns as each individual malicious node is less effective.

The Utility of Control-Plane and Data-Plane Information

To demonstrate the effectiveness of aggregating and correlating network topology and metrics to improve the adaptation decision quality and mitigate the effects of malicious activity, we use the following scenarios:

• **Smart Attack**: We use this name to denote a scenario in which a percentage of the nodes was malicious and performed a smarter version of the attraction attacks (as described in Section 3.4.2). Specifically, not only do the nodes report incorrect performance data, but they also try to attack the defense scheme itself by lying about the data collected in our path aggregation scheme (*e.g.*, reporting longer connectivity times than in actuality).

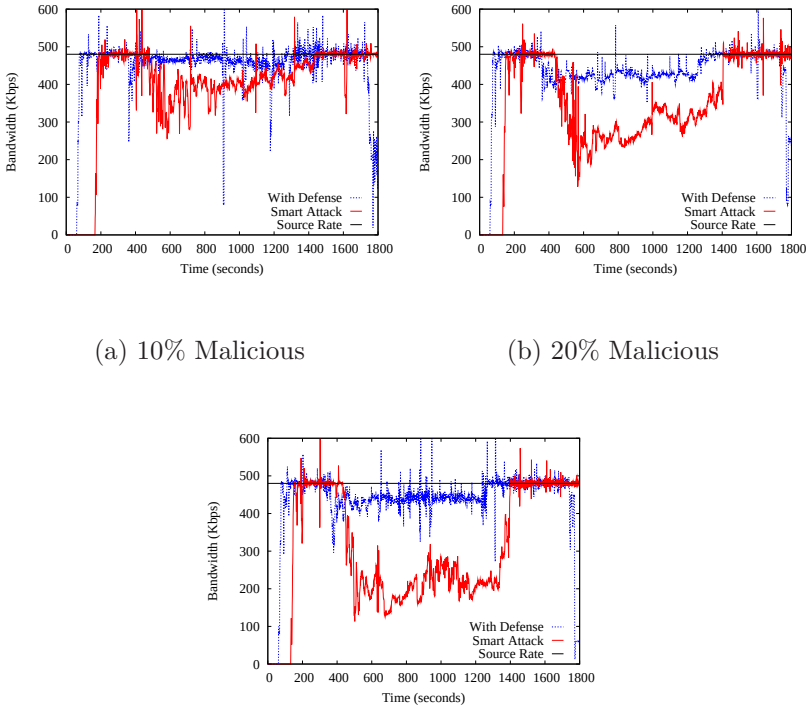
• **With Defense**: We use this name to denote a scenario in which an attacker performs the Smart Attack, but the defense technique presented in Algorithm 2 are also enabled.

As the naive attacks were ineffective, they are not included.

Table 3.3
Number of system adaptation and the percent improvement over an undefended system for an ESM overlay of 100 nodes on PlanetLab under different percentages of attackers

Experiment	Changes to Malicious Parents	Total Parent Changes
No lying	0	411
10% Using Paths	82 (79%)	644 (57%)
20% Using Paths	139 (79%)	658 (60%)
30% Using Paths	423 (55%)	926 (43%)

Attack Mitigation. Figure 3.7 shows both the severity of the attacks and the ability of our solution to mitigate much of their effect. As can be seen from each of the graphs, the bandwidth of the original system is greatly reduced when the malicious nodes are dropping data. However, through the utilization of data-plane and control-plane information to improve adaptation decisions, the systems maintains performance (average bandwidth) much closer to the optimal source rate of 480Kbps. For example, with the system using our defense technique, we can see from



(a) 10% Malicious

(b) 20% Malicious

(c) 30% Malicious

Figure 3.7. Average system bandwidth for 100 node ESM overlay deployed on PlanetLab under different percentages of attackers. The graphs show the degradation of the system performance and the utility of using our defense technique to mitigate the effects of the attack.

Figure 3.7(c) the average system bandwidth is 428Kbps, over 200Kbps higher than the undefended deployment. While there is still some degradation in the system performance, as previously discussed in Section 3.4.1, this is in accordance with our goal of creating a lightweight solution that is applicable to a broad range of systems.

Along with helping maintain system performance, our defense technique helps to increase the stability of the overlay by avoiding unnecessary adaptations. From Table 3.3, we can see our solution greatly improves the stability of the system under attack. On average, the total number of parent changes is halved and the changes to malicious nodes are reduced by a factor of 4 over deployments with no defense. This

reduction comes from the fact fewer malicious nodes are chosen as parents, which induces fewer instances of multiple adaptations.

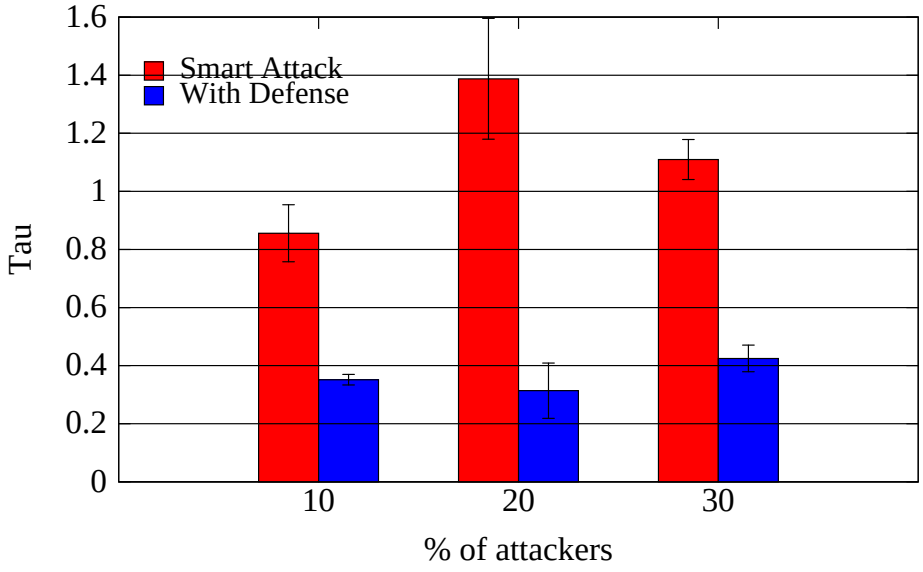


Figure 3.8. Attack strength for different percentages of malicious nodes for ESM deployments of 100 nodes on PlanetLab using our control-plane and data-plane information defense mechanism.

Figure 3.8 presents the effect of the Algorithm 2 on the relative strength of the attacks, τ, as defined in Equation 3.1. It confirms the intuition that constraining the ability of the attacker to subvert the adaptation of the system diminishes the strength of the attack for all percentages of malicious nodes. For the smaller percentages of malicious nodes, τ is reduced to levels near those of the naive attacks, demonstrating our solution greatly increases the robustness of the adaption mechanism and the system as a whole.

Potential False Positives. As each node performs its own probe cycle independently and the overlay topology may change as result, inconsistencies occasionally occur in nodes' path graphs which will cause them to avoid otherwise valid parents (a false positive). As the majority of the overlay is stable, nodes have multiple choices

for parents, and no node is permanently banned from overlay, the system is able to tolerate these occasional false positives without detrimental effect.

Impact on Nodes Stability. To determine the effects our solution has on individual nodes, we also analyzed the time nodes were connected to their parent, known as *stay time*. Stay time is a good indication of system stability. As can be seen from Figure 3.9, the stay times in a benign environment are biased towards long durations, with an average of approximately 400 seconds. As malicious nodes are introduced into the system, the number of short stay times dramatically increases. Irrespective of the percentage of malicious nodes, the resulting average stay for the overlay drops to 100 seconds. Using our solution, while we are unable able to match the performance of the benign case, we are able to increase the number of longer stay times seen in the system and achieve a better average stay time of 220 seconds.

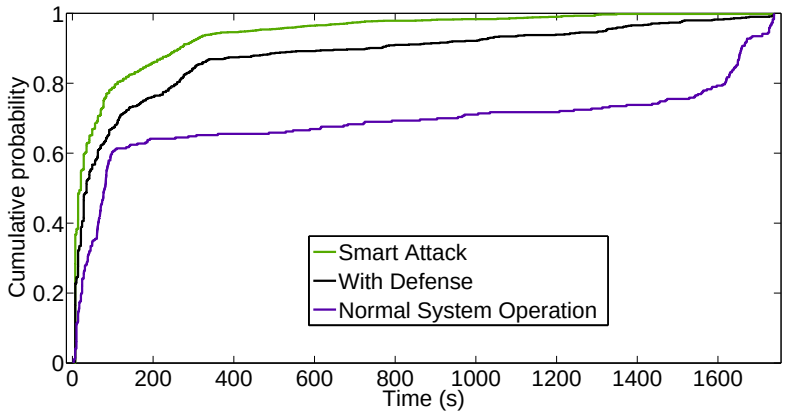


Figure 3.9. Distribution of the length of time a node is connected to its parents under the attraction attack and defense scenarios for an ESM deployment of 100 nodes containing 20% malicious nodes

Malicious Coalitions

All defense techniques and protocols resilient to insiders have limitations regarding the number of attackers they can tolerate. We now consider the constrained collusion

model presented in Section 2.2 in which the faulty nodes are part of the same coalition. We use the following scenario:

• **Malicious Coalition.** We use this name to denote the scenario in which a coalition of colluding attackers attempts to bypass the defense mechanism itself by strategically lying about system metrics to have members of the malicious coalition selected as parents high in the dissemination structure.

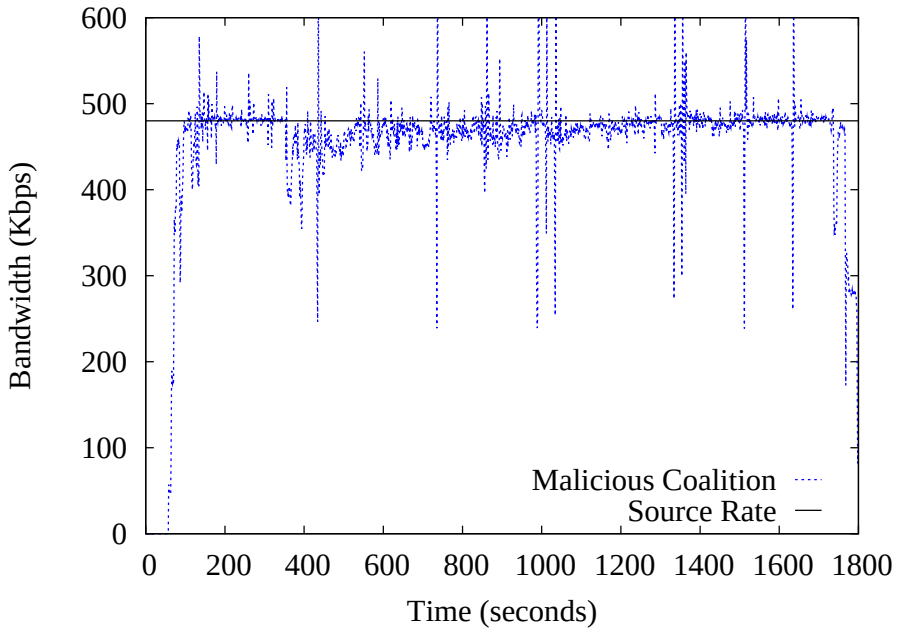


Figure 3.10. Resilience of our control-plane and data-plane information defense mechanism to malicious coalitions attempting to bypass the defense mechanism through coordinated lying while conducting an attraction attack on a representative ESM deployment of 100 nodes containing 20% malicious.

To subvert our defense mechanism, the attacker must control multiple nodes in the same section of the overlay, allowing for coordinated lying between parents and children. Each malicious parent attempts to have several malicious children so they can report good metrics (such as high throughput and low latency). This allows the parent node to attract benign nodes with less chance of inconsistencies appearing in the benign node's path graph. However, the greater the number of malicious nodes

that must be connected to each other, the fewer the number of benign children can be connected to a malicious node without creating inconsistencies in the path graph (*e.g.*, more children than the saturation degree), thereby lessening the effectiveness of the malicious nodes. Also, the malicious children are constrained to accurately reporting their path or they risk identifying themselves or their parent as being malicious. For example, from Figure 3.10 we can see that our solution is able to mitigate the attack, even when malicious collectives are actively trying to bypass the path graph detection. In fact, the relative attack strength τ is reduced by one-third of that seen in Figure 3.8 to 0.19. This implies that for maximum effectiveness in bypassing our solution, the malicious nodes should work alone or in small groups. However, from Section 3.4.2, we have seen our solution mitigates much of the effect of this strategy.

Overhead and System Performance

Our defense technique introduces minimal link stress since it uses information already being exchanged between nodes, with the exception of adding the stay time to the gossip messages. The memory utilization for the aggregation of information into the path graph lasts for the span of one probe cycle and requires maintaining the address, number of children, stay time, reported latency, reported bandwidth, and path from the node to the source for each of the probed neighbors. On average, each node receives 60 Bytes of information per node from 30 nodes per probe cycle, requiring 1.8KB of extra storage at any given time. There is negligible computational complexity associated with the aggregation of node information into the path graph.

3.5 Preventing Malicious Adaptation Using Outlier Detection

In this section, we examine our second solution to mitigating attacks aimed at the adaptation process of the system. We propose to detect inconsistent metrics by performing outlier analysis on the information received from probed nodes that is used in the decision process. An *outlier* is a data point that is significantly different

(greater than a threshold) from the rest of the data in the observation space based on a measure of distance.

As mentioned in Section 3.4, during the system lifetime, each node periodically performs probe cycles in which it receives information from remote its neighboring nodes. The outlier detection is performed locally by each node at the end of the probe cycle 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. Our outlier detection does not affect the link stress in the system, as it uses the metrics already reported by nodes: latency, bandwidth, and RTT. Both latency and RTT are utilized because they are highly correlated metrics collected in different manners (one is probed, while the other is measured). In order to avoid being suspected by benign nodes, a malicious node must insure that any lie it tells: (1) is consistent with what the other peers are reporting during a probe cycle about current network conditions, (2) ensures consistency between the different dependent metrics (bandwidth, latency, and RTT), and (3) is consistent with metrics it reported 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. Spatial and temporal data correlations have been previously shown effective in detecting network attack scenarios [74]. Unlike the general approach proposed by Jiang and Cybenko [74], our work does not look for correlations but exploits the fact that they exist to detect suspicious nodes.

The intuition behind our solution is that the dependency existent in the measured variables requires attackers to make sure the "fake" metrics vary in a consistent manner. Lying is made more difficult by the fact that, in most cases, 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. Our solution also forces an attacker to lie consistently with other peers. This is difficult to achieve as an attacker does not have perfect knowledge of the observation space,

must accurately predict the random subset of nodes that will be probed, and only has a finite amount of time (the probe period) to coordinate with other attackers.

Our approach uses the Mahalanobis [75] distance to detect outliers. We selected this distance function because it has been shown effective at detecting outliers with multiple attributes [76], scales each variable based on its standard deviation and covariance, and takes into account how the measured attributes change in relation to each other [73]. These features make it appropriate for our environment where there is a dependency between several of the attributes reported by each node.

3.5.1 Spatial Outlier Detection

The outlier detection is performed by a node as follows. Each probe cycle, the node first computes the centroid of the data over the three dimensional space formed by the observation tuples from all probed nodes. An *observation tuple* is represented by bandwidth, latency, and RTT. The node then computes the Mahalanobis distance between the observation tuple from each probed node and the centroid as follows [75]:

$$d(a, \bar{b}) = \sqrt{((a - \bar{b})^T C^{-1} (a - \bar{b}))} \tag{3.2}$$

where a and $\bar{b}$ are two feature vectors whose elements consist of an observation tuple. a is the value from the from the probe response and $\bar{b}$ is the averaged feature vector (the centroid) computed from all of the recently received observation tuples. C^{-1} is the inverse sample covariance matrix computed from the observation tuples.

If there are not enough tuples during a probe cycle, the tuples are compared with the most recent centroid. If there is no variance between the received observation tuples, the Mahalanobis distance cannot be computed since the determinant of the covariance matrix becomes zero. In this case, a node is randomly selected from that probe set of observation tuples and compared to the most recent centroid. If no centroid is available, the decision is postponed to the next probe cycle.

Spatial Threshold Selection

The threshold for our outlier detection can be mathematically derived as by Smith and Cheeseman [77] and Ribeiro [78], assuming a multivariate Gaussian distribution for the metrics vector. The contours of equal probability of this distribution create a 3-dimensional ellipsoid and the outlier threshold reflects the probability of a vector being within the ellipsoid whose semi-axes are determined by k. The probability that a random vector lies within the ellipsoid increases with the value of k. Thus, for a given value of k the probability that a probed tuple lies within the ellipsoid can be computed as:

$$P = -\frac{1}{\sqrt{2}\pi} + 2\left(\frac{1}{\sqrt{2\pi}}\int_0^k e^{\frac{y^2}{2}}dy\right) - \sqrt{\frac{2}{\pi}}ke^{\frac{-k^2}{2}} \tag{3.3}$$

We initially selected a k of 2.37, creating a threshold which half of the probes would successfully pass. Through testing in over 539,739 probe responses during 19,465 probe cycles, we found an ellipsoid determined by a threshold of k equal to 1.5 will contain approximately 80% of the nodes. Thus, we selected a threshold of 1.5 for our experiments. This variation from the mathematically derived value can be attributed to the fact that the used metrics do not form a perfect normalized distribution and have a smaller variance than assumed in Equation 4.4. A node may select smaller threshold distances for stronger security guarantees, with the drawback that it may find itself isolated due to aggressive filtering.

3.5.2 Temporal Outlier Detection

We use temporal correlations to detect inconsistencies in the performance metrics reported over time by a node. We develop models for the peers of a given node during the course of a multicast session by using incremental learning. Our technique is based on the "simplified Mahalanobis distance" presented by Wang and Stolfo [75]:

$$d(x, \bar{y}) = \sum_{i=0}^{n-1}(|x_i - \bar{y_i}|/(\bar{\sigma_i} + \alpha)) \tag{3.4}$$

where n is the number of metrics, three in our case (bandwidth, latency, and RTT), $\bar{\sigma}_i$ is the standard deviation, and α is a smoothing factor empirically set to .001 to help to avoid over-fitting and reduce false positives [75]. We trade-off accuracy of the distance function to minimize the amount of data a node must store by assuming that the metrics are statistically independent. Thus, a node does not need to maintain the whole history. Instead, a node maintains a temporal centroid for each peer consisting of the mean, standard deviation, and sample count computed from the observation tuples received over time. The centroid for each peer is incrementally updated with observations received during each probe cycle, similar to Wang and Stolfo [75], using the technique described by Knuth [79]. At the end of the probe cycle, the latest observation tuple for each peer is compared with the corresponding temporal centroid using the simplified Mahalanobis distance.

Temporal Threshold Selection

We used a threshold of 3.0 for our temporal outlier detection, to allow each of the three features to vary within one standard deviation from their temporally developed mean. The value was chosen based on the formula of the simplified Mahalanobis distance as presented by Wang and Stolfo [75].

3.5.3 Spatio-temporal Outlier Detection

We combine the two outlier detection mechanisms described above by using a codebook technique similar to Jiang and Cybenko [74]. The peer nodes are ranked according to their spatial outlier distance from the spatial centroid and traversed from the closest to the farthest node. The node that is the closest to the spatial centroid and it is not a spatial or temporal outlier is chosen as the new parent. If no peer is found meeting these criteria or if there are a large number of temporal outliers, no adaptation is performed during that probe cycle.

3.5.4 Potential Limitations of Outlier Detection

While outlier detection is an extraordinarily powerful tool, it has potential limitations that must be addressed. First, the system must start in a clean state free from attack, allowing the system to establish a correct centroid and correct running averages of historical data [80]. Next, without a proper response to the attacks identified by outlier detection, the attack data may eventually become incorporated in the corpus of data used to create the view of normal system functionality, allowing the attacks to go unnoticed and rendering the system ineffective [81]. One possible response method is the removal of nodes identified as malicious through the use of a reputation or consensus protocol, such as that proposed by Chapter 5. Finally, an adversary can attempt to bypass the outlier detection by reporting metrics that are marginally incorrect [81]. However, there are several techniques that can be employed in conjunction with our solution, such as using regularization to bias hypothesis testing, dynamically adapting thresholds of the system, introducing randomization into data collection and hypothesis calculations, and providing disinformation to potentially malicious nodes that create solutions robust to attackers specifically targeting the defense mechanisms [80].

3.5.5 Experimental Evaluation

To study the effects of using our defense solution under real-world conditions, we conducted experiments on the PlanetLab [58, 71] Internet testbed. We conducted multiple experiments at different times of the day and different days of the week. Further, experimental nodes were selected randomly for different experiments to validate the statistical significance of results and nodes were chosen to span multiple operational and administrative domains. Each experiment was conducted multiple times and the results were averaged.

The baseline configuration for the following experiments consist of 60 minute long ESM deployments of 100 nodes in which the nodes join after the experiment begins

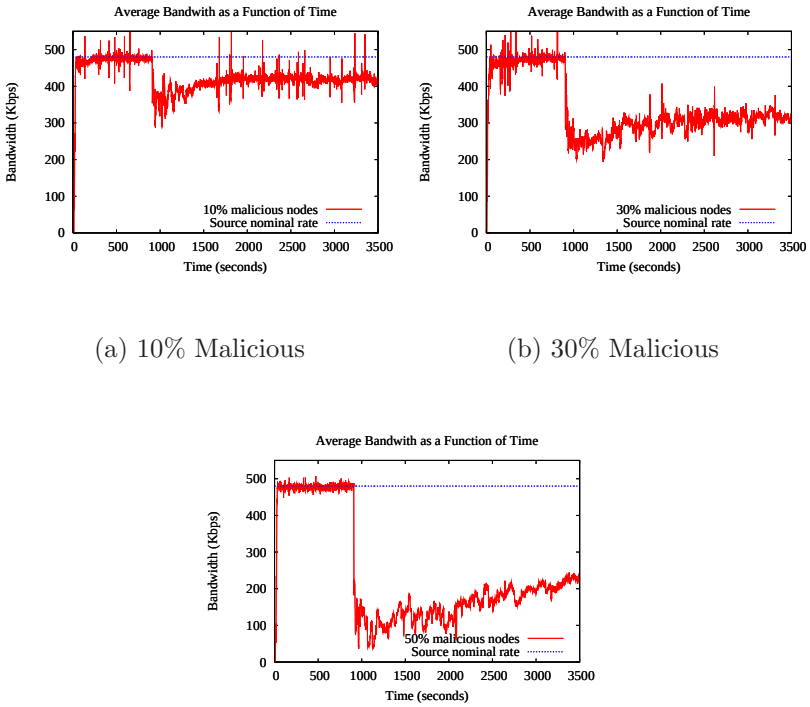
and leave before it ends, with an average participation time of 45 minutes. Each node is probed every seven seconds, each node probes 30 peers, the saturation degree of benign nodes is six, and the source streaming rate is 480Kbps. We use a bit rate of 480 Kbps as it is sufficient to transmit video at two different qualities of audio. The experimental time and saturation degree were increased over the previous section in order to validate the trends observed occur in longer duration experiments and under different system configurations. All of the following experiments use these parameters unless otherwise noted.

Effect of Malicious Nodes on Average Bandwidth

We studied the effect multiple malicious nodes can have on the overlay topology if they decide to selectively drop data. In Figure 3.11, we demonstrate the impact malicious nodes that use their position in the tree can exert on the bandwidth of benign nodes. The graphs plot the bandwidth averaged over all receivers as a function of time. Malicious nodes start dropping 90% of the data traffic received through the data dissemination tree fifteen minutes after they joined the overlay. We vary the percentage of malicious nodes to 10%, 30%, and 50% of the overlay size to demonstrate the performance degradation that results when more nodes behave maliciously.

Effectiveness of Outlier Detection

To demonstrate the effectiveness of our outlier detection at improving the parent selection process and the stability of the system, we considered one malicious attacker and recorded the number of parent changes that took place for the duration of the experiment considering two cases, one when only the spatial outlier is used, and one when the temporal-spatial outlier is enabled. The outcome of these experiments is shown in Table 3.4. The results indicate that using the spatial outlier detection scheme has dramatically reduced the likelihood of choosing a malicious parent since the number of times the malicious node was selected as a new parent is reduced from



(a) 10% Malicious

(b) 30% Malicious

(c) 50% Malicious

Figure 3.11. The average bandwidth over time for an ESM overlay of 100 nodes on PlanetLab for a duration of 60 minutes with different percentages of malicious nodes.

172 to 70. The addition of the temporal outlier detection further reduces this to only 35 times.

Our method also dramatically improved the stability of the overlay in spite of the presence of the malicious node, as measured by the decrease in total parent changes denoted in third column of Table 3.4. In fact, the number of adaptations is comparable to the number of adaptations that would occur when no malicious nodes are present in the overlay.

Table 3.4
The effectiveness of outlier detection at improving parent selection for an ESM overlay of 100 nodes on PlanetLab over 60 minutes

Experiment	Changes to Malicious Parents	Total Parent Changes
No lying	5	833
Lying	172	1032
Spatial	70	800
Spatial/Temp	35	604

Overhead and System Performance

Our outlier detection does not introduce any extra link stress since it uses information that is already being exchanged between nodes. 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 probed nodes, while the storage requirements consist of three additional values in the route table for the peer set maintained by each node. In the case of the temporal outlier detection, the memory usage consists of maintaining the temporal centroid. By incrementally updating the centroid, we do not need to maintain the entire history for each probed node. The temporal outlier detection also requires modifying the route table entries to store nine additional values: mean, standard deviation, and count for each of the three metrics.

3.6 Summary

In this chapter, we demonstrated the utility for an attacker in subverting adaptation mechanisms over other methods of attack and we proposed two separate techniques to mitigate the impact of these attacks. First, we proposed a lightweight solution which aggregates and correlates network topology and application metrics at each node, allowing the nodes to use the derived information to make better adaptation decisions and to constrain the actions of malicious nodes. Second, we proposed the use of context sensitive anomaly detection to detect spatial and temporal outliers of measured and probed metrics, allowing nodes to avoid malicious data and unnecessary adaptations. Our experiments conducted on the PlanetLab Internet testbed using the mature, unstructured overlay multicast system ESM showed that although ESM employs an advanced set of adaptation mechanisms, it was unable to mitigate the attacks posed by a malicious adversary. The experiments demonstrated that both of our techniques improved the adaptation process and the overall stability of the system by limiting the effect of malicious nodes.

4 CREATING ROBUST SOLUTIONS FOR NETWORK AWARENESS

As noted in Chapter 3, many peer-to-peer applications optimize their structure based on network topology. For example, the construction of multicast trees or the selection of a replica for file sharing applications can be greatly improved by taking advantage of network locality. One basic approach to obtain this network locality information is to probe all hosts in the network to determine attributes such as latency. The cost associated with active monitoring to estimate such attributes is non-negligible [35,82] and is further exacerbated by the presence of multiple applications performing this task on a common network infrastructure.

Virtual coordinate systems provide an efficient network service that allows hosts on the Internet to determine the latency to arbitrary hosts without actively monitoring all nodes in the network. The accuracy and stability of virtual coordinate systems relies on the assumption that peer nodes on which the coordinate computation relies are altruistic and correctly participate in the system. However, this assumption may be violated by compromised nodes acting maliciously to degrade the accuracy of the coordinate system. While this violation may be prevented for landmark-based virtual coordinate systems by securing the small set of infrastructure nodes, it is not easily mitigated for decentralized systems where any node can act as a reference node for other nodes in the system. As a result, decentralized virtual coordinate systems are vulnerable to insider attacks conducted by attackers that infiltrate such systems or compromise some of their nodes [83,84]. Since virtual coordinate systems are designed to be network services providing latency estimation for a wide variety of peer-to-peer applications, such as distributed hash tables (DHTs) [54] and routing [85], they are likely to be prime candidates for attack. It is critical that virtual coordinate systems are designed to be robust to attackers that influence the accuracy of the coordinates in order to maintain upper-level application performance.

In this chapter, we demonstrate the vulnerability of decentralized virtual coordinate systems to inside attackers. We propose techniques to make the coordinate assignment robust to malicious attackers. We use both spatial and temporal correlations to perform context-sensitive outlier analysis to reject malicious updates and prevent unnecessary and erroneous coordinate changes. Our solution does not induce extra communication in the system or use trusted components, complying with the virtual coordinate system design goals of flexibility and low communication overhead. We summarize our key contributions:

- We classify three types of attacks against virtual coordinate systems based on their impact on the coordinates: *coordinate inflation, coordinate deflation*, and *coordinate oscillation*. The attacks are conducted by insiders that have infiltrated the system or compromised some of the nodes. The low-rate nature of the attacks (*i.e.*, they do not require the attacker to generate a noticeable amount of traffic) makes them difficult to detect, while their epidemic nature makes them very dangerous, as a small number of attackers can significantly influence the accuracy of the entire system.

- We propose techniques to reduce incorrect coordinate mappings by using spatial and temporal correlations to perform context-sensitive outlier analysis. A key component of our solution is based on the observation that the behavior of attackers can be constrained by correlating dependent metrics.

- We demonstrate the impact of the attacks and the effectiveness of our defense mechanisms through p2psim [86] simulations, in the context of the well-studied Vivaldi virtual coordinate system [87] using three representative real-world topologies of hosts and corresponding RTTs: King [88], Meridian [89], and AMP [90]. We found through analytical and empirical studies that a spatial threshold of 1.5 and a temporal threshold of 4.0 provided a low system error under attack while maintaining an low false positive rate. Our experiments also

show that the method starts to degrade when a coalition of malicious nodes in the reference set of a node increases over 30% of the reference set size.

The rest of the chapter is organized as follows: We provide an overview of virtual coordinate systems in Section 4.1 and of the attacks against them in Section 4.2. We propose our solution in Section 4.3. We present experimental results demonstrating the impact of the attacks and the effectiveness of our solution in Section 4.4. Finally, Section 4.5 concludes this chapter.

4.1 Decentralized Virtual Coordinate Systems

Virtual coordinate systems [87,88,91–99] have been proposed as a low communication cost service to accurately predict latencies between arbitrary hosts in a network. These systems allow a node to map itself to a *virtual* coordinate based on a small number of actual network distance estimates to a subset of *reference nodes*. By comparing virtual coordinates, nodes can trivially estimate the latency between them.

Two main architectures for virtual coordinate systems have emerged: landmark-based and decentralized. Landmark-based systems rely on infrastructure components (such as a set of landmark servers) to predict distance between any two hosts. The set of landmarks can be pre-determined [91, 94, 98] or randomly selected [93, 99]. Decentralized virtual coordinate systems do not rely on explicitly designated infrastructure components, requiring any node in the system to act as a reference node. Examples of decentralized virtual coordinate systems include PIC [95], Vivaldi [87], and PCoord [97, 100].

The design goal of decentralized virtual coordinate systems is to efficiently create and maintain a stable set of virtual coordinates that accurately predict the latency between nodes without using fixed infrastructure nodes. Although each specific virtual coordinate system differs in some details, most of them follow a common design. The most important characteristics that define decentralized coordinate systems are

(1) the *reference or neighbor set*, (2) the *latency prediction mechanism*, and (3) the *error minimization technique*.

In a decentralized virtual coordinate system, each node calculates its coordinates based on the information obtained from a small set of nodes in the network, which we refer to as the *reference set*. There are several methods used to select the reference set. One of the most promising methods identifies a set of close and a set of distant network nodes and selects a random subset of each [87,95]. Nodes may have different reference sets. Different systems use different sizes of the reference set due to the frequency of actual network measurements, the number of nodes queried per measurement interval, and the error minimization technique utilized. For example, Vivaldi uses a reference set size of 64 nodes [84], PCoord uses 10 nodes [100], and PIC uses 32 nodes [95].

Once a reference set has been selected, a node determines its coordinate based on a predefined *latency prediction mechanism*, such as the Euclidean distance. Each system typically maintains coordinates in either low dimensional (usually 2 to 8 dimensions) Euclidean space [95], an augmented Euclidean space [87], or a non-Euclidean (*e.g.*, hyperbolic) space [101]. In general, it has been shown that none of the embedding spaces dominates the others in performance [102] and lower dimensionality Euclidean spaces are often sufficient [87]. A node determines its position and then successively refines it by periodically querying nodes in its reference set. Queried nodes respond with metrics that can include local error, perceived system error, local coordinates, and RTT.

Virtual coordinate systems provide accurate latency prediction, achieved through *error minimization techniques* of a chosen latency error function. Examples include:

- Generic multi-dimensional minimization designed to minimize a relative system error measure (such as logarithmic transformed error) using techniques such as the downhill simplex method [95].

- Minimizing coordinate error by simulating Newtonian mechanics. Each node in the system is simulated as a particle influenced by the field force induced

between nodes. Each pair of particles (nodes) either pulls or repulses each other, thereby reducing the total system error [101].

- Minimizing system error by simulating spring relaxation, where the state of the springs at rest is the optimal embedding. The system minimizes the squared system latency estimation error by iteratively finding the low-energy point of the spring-based system [87].

While each technique has benefits, systems based on multi-dimensional minimization are often slow to converge, sensitive to initial system conditions, and sensitive to high-error measurements. Simulation techniques such as spring relaxation are computationally inexpensive, less sensitive to high-error nodes, and more amenable to general decentralized system design.

In general, virtual coordinate systems achieve the overall goals of accuracy and stability while reducing traffic by as much as two orders of magnitude when compared with active monitoring to estimate RTT [95]. Systems such as Vivaldi [87], PCoord [97], and PIC [95] stabilize at an average system latency estimation error of ten milliseconds for large scale simulations and deployments.

4.1.1 Vivaldi Virtual Coordinate System

Vivaldi is a fully decentralized virtual coordinate system which assigns each host synthetic coordinates in a multi-dimensional Euclidean coordinate space. Conceptually, Vivaldi attaches a spring between each pair of neighbor nodes, where the length of the spring is the estimated RTT between the nodes. By minimizing the tension on these springs across the entire network, the protocol minimizes the error in the system.

When a node joins the system, it establishes its reference set and initializes its coordinates to the origin of the Euclidean coordinate space. As seen in Figure 4.1(a), each node i then periodically probes a reference set member j to obtain j's current coordinate and perceived error. As i receives the coordinate (x_j), the remote error

Algorithm 3: Vivaldi Coordinate Update

Input: Remote node observation tuple ($\langle x_j,\ e_j,\ RTT_{ij} \rangle$)

Output: Updated Node Coordinate

1 $w = e_i/(e_i + e_j)$;

2 $e_s = |\|x_i - x_j\| - RTT_{ij}|/RTT_{ij}$;

3 $\alpha = c_e \times w$;

4 $e_i = (\alpha \times e_s) + ((1 - \alpha) \times e_i)$;

5 $\delta = c_c \times w$;

6 $x_i = x_i + \delta \times (RTT_{ij} - \|x_i - x_j\|) \times u(x_i - x_j)$;

estimate (e_j), and measured RTT from j, i uses this information to update its coordinates using Algorithm 3. First, node i calculates the reliability of the observation based on the local and remote error (line 1) and relative error of the observation tuple (line 2). Next, node i updates its local error (line 4) and the movement dampening factor (line 5). Finally, as depicted in Figure 4.1(c), node i updates its coordinate in the last line of the algorithm. As the Vivaldi virtual coordinate system stabilizes, the average system error is on the order of a few percent. Once the coordinate system has stabilized, the latency between two nodes is trivially estimated by computing the Euclidean distance between their coordinates.

We selected Vivaldi as a representative example since it is a mature system, conceptually easy to understand and visualize, and has been shown to produce low error embeddings. For further details of the protocol, we refer the reader to the work by Dabek *et al.* [87].

4.2 Vulnerability of Virtual Coordinate Systems

The correct operation of virtual coordinate systems is dependent on the assumption that the reference set nodes are altruistic and respond with correct metrics to queries from any node computing its corresponding coordinates. An attacker con-

trolling reference set nodes has the ability to influence the coordinate maintenance process by manipulating the information, such as remote node error and coordinates, returned in response to a query. By blindly accepting this malicious information, a correct node computes incorrect coordinates. While we examine the attacks in the context of the Vivaldi virtual coordinate system, the attacks generalize to any virtual coordinate system which uses external information to maintain the correctness of the system.

A malicious node is able to indirectly take advantage of the error minimization techniques and chosen error function by manipulating the metrics it reports as a reference set node. In doing so, an attacker is able to make a victim node move away from its correct position by either pulling it closer or pushing it towards a location specified by the attacker. For example, as depicted in Figure 4.2(b), a malicious node can pull a victim node towards a designated location in quadrant slowromancapi@ and away from the victim's correct position. As we can see from Algorithm 3, if the malicious node reports a low error (used to calculate the movement dampening factor in Lines 1 and 5) in conjunction with an actual RTT less than the estimated RTT (the Euclidean distance between the victim node's coordinates and the reported location), a node will move towards the reported location. Similarly, if a malicious node reports a position close to the victim node that is in the opposite direction of the desired malicious location along with an artificially high RTT (caused by delaying probe responses) the node will be pushed towards the desired location (Figure 4.2(b)). The larger the induced delay, the farther the victim node will re-calculate its positions away from the reported position. We refer to such attacks that result in coordinate mappings farther from the correct location as *coordinate inflation*.

An attacker may cause a victim node to remain immobile by reporting positions that minimize the difference between the actual and estimated RTT. Examining line 6 of Algorithm 3, if the attacker is able to force $(RTT_{ij} - \|x_i - x_j\|) = 0$, the victim node will remain stationary. To achieve this, the malicious node can either report coordinates which cause the difference between the measured and estimated RTT to

be zero or influence the measured delay so that it matches the delay estimated by the coordinates. In addition, since the difference between the actual and estimated RTTs is used to update the victim's local error estimate, the estimate will be artificially low. We refer to such attacks in which the victim nodes are prevented from performing necessary, correct coordinate changes as *coordinate deflation*.

The final type of attack we examine is the *coordinate oscillation* attack, which results in nodes continuously changing their positions and not converging to stable coordinates. The attack consists of each malicious node attempting to maximize system error by randomly reporting erroneous coordinates, low, random local error, and actively delaying probe responses. Depending on the attack, each time a malicious node is used as a reference set node, it can report the same random coordinates or an entirely new position.

While we have described in detail each of the attacks against individual nodes, any attack against the coordinate system may actually target one or more nodes, a subset of nodes, or a region of the coordinate space. Also, as each node reports false information, it attempts to lie within the normal operating characteristics of the system. That is, the attacker reports positions selected over the coordinate area where the majority of nodes are located and it reports a low error in line with low-error benign nodes to minimize the risk of being identified as malicious.

The final goal of manipulating the coordinate system can include isolating subsets of nodes from the network, creating general disorder in the system, or rendering the coordinate system unusable due to high estimation error. While each of these goals serves a different purpose, in the end they all distort the coordinate space and can make using the computed coordinates worse than using randomly assigned coordinates. Even short-lived, localized attacks have a long-lasting effect on the overall system. For example, even when a single victim node is displaced from its correct position, this has an epidemic, detrimental effect on the system as the victim node will push/pull other nodes away from their correct coordinates by reporting its now incorrect coordinates. This occurs since the victim node will serve as a reference

set member for other nodes in the system, negatively influencing their coordinate computation. Besides degrading the accuracy of the coordinate system, the attacks will also adversely impact any application using the coordinate system to estimate network measurements. Additionally, as the attacks exploit the semantics of the information contained on the packet, they do not add a noticeable change in traffic load and thus are difficult to detect by traditional mechanisms.

4.3 Adding Robustness to Virtual Coordinate Systems

We leverage techniques from outlier detection to identify malicious behavior and take defensive actions to mitigate its effects. Instead of allowing malicious coordinate mappings to occur and then trying to detect them, we focus on reducing the likelihood of a node computing incorrect coordinates by filtering out malicious updates using statistical outlier detection. Each node independently performs outlier detection before updating its coordinates in order to identify and filter out outliers in the received metrics.

Since the evidence of malicious activity is distributed across space and time, we detect malicious activity using both temporal and spatial correlations among metrics in the system. *Spatial outlier detection* identifies observations which are inconsistent with their surrounding neighbors, forcing nodes to report metrics consistent with what other reference peers are reporting. *Temporal outlier detection* identifies inconsistencies in the metrics over time, forcing a node to report metrics consistent with what it has reported in the past.

To minimize communication cost and maintain a low-overhead system, we focus on utilizing metrics used by nodes to update their coordinates using Algorithm 3. We use the 3-tuple of $\langle RTT,\ e_{remote},\ \Delta_{remote}\rangle$ to generate the spatial outlier statistics and the 5-tuple of $\langle RTT,\ e_{remote},\ \Delta_{remote},\ e_{local},\ \Delta_{local}\rangle$ to generate the temporal outlier statistics. Here, RTT is the measure of the actual two-way latency between the nodes, e_{remote} is the remote error estimate received in the observation tuple, and e_{local} is the

most recent error estimate calculated at the local node using Algorithm 3. Δ_{remote} is the Euclidean distance the remote node has moved due to its last update. For example, if node j's location prior to an update was located at (4, 5) and moved to (4, 3) after the update, then Δ_{remote} is 2. Δ_{local} is the analogous measure at the local node. Both Δ_{remote} and Δ_{local} are the only metrics used in addition to the original Vivaldi update algorithm. Δ_{remote} is incorporated as part of the observation tuple and Δ_{local} is easily calculated at the local node.

Each metric was chosen on the basis that while each of them represents a different measure of system performance, changes in one measure will result in a correlated change in other metrics. For example, as the system stabilizes to low overall error, the local error reported by each node and magnitude of the change in coordinates will both decrease. An attacker must therefore report a high error with greatly changing coordinates in order to not be identified as malicious. Our solution also forces an attacker to lie consistently with other peers. This is difficult to achieve since an attacker does not have perfect knowledge of the observation space, must accurately predict the random subset of reference nodes that will be queried, and only has a finite amount of time to coordinate with other attackers.

4.3.1 Spatial Outlier Detection

We use spatial outlier detection to examine the consistency of recently received metrics from queried nodes. A node queries a random node from its reference set and receives an *observation tuple* which consists of $\langle RTT,\ e_{remote},\ \Delta_{remote} \rangle$. The node records this response and tracks the most recent u updates in a queue, where the oldest responses are replaced by newer ones. The size of the history queue, u, is equal to the size of the reference set which allows the queue, on average, to contain one entry from each reference set nodes. Unlike more message-intensive distributed systems where a new set of responses from all nodes queried (in this case nodes in the reference set) are collected in response to one query [35], virtual coordinate systems

collect these responses sequentially. Our approach requires a node to perform outlier detection every time it receives a new tuple, considering the most recent u updates. We note that this technique is an instance of spatial outlier detection since we examine metrics across various system nodes and not time.

Once a node receives an observation tuple, the node first computes the centroid of the data set consisting of observation tuples from the stored u updates. The node then computes the Mahalanobis distance between the received observation tuple and the centroid as follows [75]:

$$d(a, \bar{b}) = \sqrt{((a - \bar{b})^T C^{-1} (a - \bar{b}))} \tag{4.1}$$

where a and $\bar{b}$ are two feature vectors whose elements consist of an *observation tuple*. a is the feature vector representing the newly received observation tuple from a remote node and $\bar{b}$ is the averaged feature vector (the centroid) computed from the u most recently received observation tuples which have been used to update the current node's coordinates. C^{-1} is the inverse sample covariance matrix computed from the u most recent tuples. After the distance is calculated, it is compared against a *spatial threshold*. We discuss the spatial threshold selection in Section 4.4.4.

4.3.2 Temporal Outlier Detection

We use temporal correlations to detect inconsistencies in the metrics reported over time by a reference set node. We use the 5-tuple consisting of $\langle RTT,\ e_{remote},\ \Delta_{remote},\ e_{local},\ \Delta_{local} \rangle$. Using incremental learning, we compute a temporal centroid for each of the members of a node's reference set. We assume each of the reported metrics is statistically independent, necessitating the storage of just the mean, standard deviation, and sample count computed from the received query responses over time. The stored values for a reference set member are incrementally updated with metrics received from its query response, similar to Wang and Stolfo [75], using the technique described by Knuth [79]. In order to compare newly received values with

the temporal centroid, we use the "simplified Mahalanobis distance" presented by Wang and Stolfo [75]:

$$d(x, \bar{y}) = \sum_{i=0}^{n-1} \frac{|x_i - \bar{y_i}|}{\bar{\sigma_i} + \alpha} \tag{4.2}$$

where n is the number of metrics, five in our case (remote error, local error, latency, change in remote coordinates, and change in local coordinates), $\bar{\sigma_i}$ is the standard deviation, and α is a smoothing factor empirically set to .001 to help to avoid overfitting and reduce false positives [75]. Once a query response is received, the latest observation tuple is compared with the corresponding temporal centroid using the simplified Mahalanobis distance, based on a *temporal threshold* that decides if the tuple is an outlier or not. We discuss temporal threshold selection in Section 4.4.4.

4.3.3 Spatio-temporal Outlier Detection

As seen in Algorithm 4, we combine the two outlier detection mechanisms described above by using a codebook technique similar to Jiang and Cybenko [74]. As a node receives observation tuples from its reference set members, it checks each one to ensure that it is not a spatial or temporal outlier. If the reference node is found to be an outlier, the query response will not be used in future temporal centroid calculations since it will not be incorporated into the temporal mean, temporal standard deviation, or sample count. Also, it will not be used in future spatial centroid calculations since it will not be added to the queue of the most recent u updates. If the observation tuple is not an outlier, it is used to update the receiver node's coordinates using Algorithm 3.

Algorithm 4: Procedure to exclude malicious observation tuples from the coordinate update process.

Input: x_j, RTT, e_{remote}, Δ_{remote}, e_{local}, Δ_{local}

Output: Updated coordinate if the observation tuple was not malicious

```
// Calculate the centroid of the most recent tuples and check
   spatial consistency using the using the Mahalanobis distance
```

centroid = calcCentroid(u observation tuples);

if *(spatialOutlier(centroid, RTT, e_{remote}, Δ_{remote}) == true)* **then**
return false;

```
// Check temporal consistency using the using the simplified
   Mahalanobis distance
```

if *(temporalOutlier(RTT, e_{remote}, Δ_{remote}, e_{local}, Δ_{local}) == true)* **then**
return false;

```
// Data is not malicious, perform update operations using
   Algorithm 3
```

updateCoordinates(x_j, e_j, RTT_{ij});

store $\langle RTT, e_{remote}, \Delta_{remote} \rangle$ tuple in the queue of u most recent tuples;

update stored temporal statics for RTT, e_{remote}, Δ_{remote}, e_{local}, Δ_{local};

return true;

4.4 Experimental Evaluation

In this section, we demonstrate the impact of attacks against virtual coordinate systems through simulations using actual Internet topologies. In addition, we demonstrate that our proposed mechanisms enhance the robustness of decentralized virtual coordinate systems to such attacks. We examine their effect on the Vivaldi virtual coordinate system which is simulated in the p2psim simulator [86].

Table 4.1
Internet data sets characteristics

Data Set	# Nodes	Avg. RTT	Max. RTT	Std. Dev. RTT
King	1740	180ms	800ms	66ms
Meridian	2500	80ms	1000ms	69ms
AMP	90	70ms	453ms	51ms

4.4.1 Evaluation Methodology

For our simulations, we use three different RTT data sets collected from real-life Internet topologies. Table 4.1 summarizes the characteristics of each data set:

- ***King:*** The King data set contains the pair-wise RTT of 1740 nodes measured using the King method [88].
- ***Meridian:*** The Meridian data set, obtained from the Cornell Meridian project [89], contains the pair-wise RTT of 2500 nodes measured using the King method.
- ***AMP:*** The AMP data set, collected from the NLANR Active Measurement Project [90] on March 1, 2007, contains complete information for 90 high-speed nodes contained mostly in North America.

We selected the King and Meridian data sets because they are representative of larger scale peer-to-peer systems and were used in validating many peer-to-peer and virtual coordinate systems. The King data set contains a variety of link latencies while the Meridian data set is more homogeneous, containing many nodes close to each other in terms of network latency. Due to this homogeneity, the average RTT of the Meridian data set is approximately half that of the King data set. The final data set, AMP, is used since it represents a smaller, high speed system, such as a corporate network. In AMP, links with latency of 100ms or less account for nearly

90% of all of the links. We did not consider synthetic topologies since they do not capture important network properties such as violations of the triangle inequality.

In order to quantitatively compare the effect of attacks on the accuracy of the system, we evaluate two error metrics:

- ***System prediction error*** is defined as

$$Error_{pred} = |RTT_{Act} - RTT_{Est}|$$

  where the RTT_{Act} is the measured RTT between two nodes and RTT_{Est} is the predicted RTT by the virtual coordinate system. This metric provides an intuition of how the overall system is performing. The lower the system prediction error, the more accurate are the predicted RTTs.

- ***Relative error*** is defined as

$$Error_{rel} = \frac{Error_{attack}}{Error_{no\_attack}}$$

  where $Error_{attack}$ is the system prediction error measured in the presence of malicious nodes and $Error_{no\_attack}$ is the system prediction error without malicious nodes. This metric captures the impact an attacker has on the coordinate system. A relative error greater than one indicates a degradation in accuracy and a value less than one indicates a better estimation accuracy than the baseline.

For each of the error measures, the 5^{th}, 50^{th}, and 95^{th} percentile error are analyzed. These values are obtained by selecting the corresponding entries from a sorted array of prediction error and are averaged over multiple simulation runs. Intuitively, the 5^{th} percentile represents low error nodes, the 50^{th} percentile corresponds to average or median error nodes, and the 95^{th} percentile represents high error nodes.

Each simulation was run for 200 time units, where each time unit is 500 seconds in length. The nodes join in a flash-crowd scenario in which all nodes join simultaneously and are each initially placed at the origin of the logical coordinate space. All nodes that join the network are physically stationary and are present for the duration of the

experiment. Each node proceeds independently of other nodes in the network and chooses a reference set of 64 nodes using the Vivaldi method where half of the nodes are selected as the closest nodes based on network latency and the rest are selected at random. All other Vivaldi parameters were initialized to the optimal values discussed by Dabek *et al.* [87] unless otherwise noted. Each of the experiments utilizes a two-dimensional coordinate space $\{(x, y)|x, y \in [-300000, 300000]\}$. Every experiment was run ten times with the reported metrics averaged over all of the simulation. In each of the figures, only the first 50 time unit steps are displayed since the virtual coordinate system has converged to a stable state by that point.

4.4.2 Node Placement in a Virtual Coordinate System

In Figure 4.3, we present graphical representations of the typical operation of the Vivaldi virtual coordinate system using the King, Meridian, and AMP data sets. The King data set contains a variety of link latencies, allowing nodes in the virtual coordinate system to form a structure in which nodes with small RTTs between them converge into clusters, as seen in Figure 4.3(a). As the Meridian data set contains many nodes close to each other in terms of network latency, we visually observe in Figure 4.3(b) that the system forms fewer, but larger clusters. The final data set, AMP forms one main cluster, as seen in Figure 4.3(c).

4.4.3 Impact of Attacks Against Distributed Virtual Coordinate Systems

In this section we demonstrate several attacks against the Vivaldi coordinate system. Vivaldi was designed to tolerate high-error, *benign* nodes, but it has no built-in mechanisms to defend against malicious nodes.

Inflation and Deflation Attacks

We first demonstrate how a coalition of f=30% malicious nodes can target one particular victim node and conduct an inflation or a deflation attack. Figure 4.4 presents the location and associated prediction error of a single victim node under non-attack conditions and under the two attacks. The correct location of the victim node is represented by the triangle in quadrant slowromancapi@. Under the deflation attack, all of the attackers send the victim node coordinates that minimize the difference between the actual RTT and estimated RTT (minimizing the Euclidean distance between the attacker and victim). As a result, since the attackers force the estimated RTT to artificially match the actual RTT, the victim node remains stationary at the origin of the cartesian space while believing it has low estimation error. Figure 4.4(b) also depicts an inflation attack, where all of the attackers send the victim node coordinates from a small chosen area along with RTTs influenced by delaying query responses, causing the node to move rapidly towards the desired area. Note the square in the quadrant slowromancapii@ representing the victim node was forced towards the area chosen by the attackers. As can be seen in the Table 4.4(a), the different attacks greatly increase the prediction error of the victim node from 10ms to 60ms for the deflation attack and 10ms to 70ms for the inflation attack.

Oscillation Attacks

We demonstrate an oscillation attack in Figure 4.5. In this scenario, the malicious nodes work together to cause general disorder in the system by sending the victim nodes erroneous random positions selected over the coordinate space with a low error value, causing the victim nodes to make multiple incorrect coordinate changes. As seen in Figure 4.5(a), the system under non-attack conditions has an easily identifiable structure in which nodes with small RTTs between them converge into clusters in the coordinate space. When the system is under attack as seen in Figure 4.5(b), the virtual coordinate system loses its structure and hence also loses its ability to yield

a *low error* embedding. This attack also exemplifies the epidemic nature of such attacks. As correct nodes computing incorrect coordinates are later used as reference nodes for other nodes, the entire system destabilizes.

Impact of the Percentage of Malicious Nodes

We investigate the effect of the number of malicious nodes on the accuracy of the system, by varying the percentage of malicious nodes. Each queried malicious node returns erroneous metrics in the form of a random position selected over the coordinates $\{(x, y)|x, y \in [-100000, 100000]\}$ and a low, non-zero error value. A malicious node also randomly delays its response between 100ms and 1000ms in order to induce greater variability in its responses in an attempt to expand the coordinate space.

Figure 4.6 presents the prediction error for the King data set for several percentages of malicious nodes. Under non-attack conditions, a node joining the coordinate system is initially placed at the origin of the logical coordinate space. As time passes, each node receives query responses from its reference set and is able to refine its position, allowing the system as a whole to achieve lower prediction error. Once the system stabilizes about halfway through the simulation, the system prediction error remains roughly constant. After this point, each of the nodes continues to refine its position, but the overall sum of these movements yields little change in the prediction error. While the system under attack may initially start with similar prediction errors since nodes are initially placed at the origin, it is never able to effectively refine its coordinates and achieve the desired low estimation error found in the non-attack scenario. As the percentage of attackers increases, the ability of the system to accurately estimate latency significantly degrades.

Similar trends are also evident in Figure 4.7, where the system can be seen to stabilize at a much higher relative error than the baseline of one. Having even a small percentage of attackers incurs *double* or *triple* the estimation error when compared

with the non-malicious scenario. Malicious nodes have a greater negative impact on the lower-error nodes, as can been seen from the higher relative errors in Figure 4.7(a) and Figure 4.7(b) as compared to Figure 4.7(c). When a low error node moves in response to malicious data, it is prone to make large, erroneous changes to its own position and experience a higher estimation error.

The results presented in Figure 4.6 and Figure 4.7 are not a product of system randomness. To validate this claim, we formulate the null hypothesis $H_0 : \mu_{attack} = \mu_{normal}$, which states that system error under non-attack and attack conditions have the same mean and distribution, implying the virtual coordinate system behaves similarly under both conditions. Using a two-sample t-test with pooled variance [103], we disproved H_0, finding with high probability (nearly 100%) that the error results come from distributions with different means. We conclude that malicious nodes can cause the system to have a much higher estimation error and to stabilize at a higher error than a system under non-attack conditions and that the error increases with the size of the malicious set.

Impact of Reference Set Size on System Performance

Next, we explore the effect different reference set sizes has on the overall accuracy of the Vivaldi virtual coordinate system under normal operation and malicious settings using the King data set. While many virtual coordinate systems choose an arbitrary power of two for the reference set size, from Figure 4.8 we can see the impact this selection has on the ability of the system to tolerate malicious nodes and overall system error, with larger reference set sizes minimizing the impact of the malicious node comparatively to the smaller reference sets. To further understand this effect, we look at the effect malicious nodes have on an individual victim node.

It is interesting to note that the actual number of attackers which directly influence a victim node is the number of malicious nodes that are selected to be in the reference set of the victim node. We can analyze the probability of having malicious neighbors

and better understand the design choice of having a smaller or larger reference set using the hypergeometric distribution. If we let k represent the number of malicious nodes in a reference set, N be the number of nodes in the system, D is the total number of malicious nodes, and n is the size of the reference set, then the probability of having exactly k malicious nodes in a reference set is given by

$$f(k; N, D, n) = \frac{\binom{D}{k}\binom{N-D}{n-k}}{\binom{N}{n}} \tag{4.3}$$

By summing the discrete probability distributions for values from 0 to k, we can determine the probability of having a certain percentage of malicious nodes in the reference set. From Table 4.2, we can see the intuitive notion that the greater the size of the neighbor set or greater the percentage of malicious nodes, the more likely a benign node is to have at least one malicious neighbor. For example, we can see that a reference set of 16 nodes has an 82% chance of containing at least one malicious node when 10% of the total nodes are malicious while there is a 97% chance for a reference set of 32 nodes. However, as seen in Figure 4.8, this initial intuition does not hold as the larger reference set sizes are able to minimize the impact of the malicious node comparatively to the smaller reference sets.

Table 4.2
Probability of a reference set having at least one malicious node for the King topology

		Neighbor Set Size				
		4	8	16	32	64
% of	10	34%	57%	82%	97%	99.9%
Nodes	20	59%	83%	97%	99.9%	99.9%
Lying	30	76%	94%	99.7%	99.9%	99.9%

This discrepancy is due to the fact that as the reference set size grows and more nodes can be queried for information, the density of malicious information at the

benign nodes decreases. For example, if we look at Table 4.3, we can see that with 10% of the nodes being malicious with larger reference set sizes (32 and 64), there is minimal chance for a node to have more than thirty percent or more of the nodes to be malicious. However, as the reference set size decreases, the probability of having a significant number of malicious neighbors increases. Once the reference set reaches a significant size (32-64), the additional nodes do not convey extra resiliency.

Table 4.3
Probability of a reference set having at least 30% malicious nodes for the King topology

		Neighbor Set Size				
		4	8	16	32	64
% of	10	34%	19%	2%	.07%	.000006%
Nodes	20	59%	50%	20%	9%	4%
Lying	30	76%	75%	55%	51%	57%

Impact of Attacks on Different Network Topologies

We examine the impact of the attacks on different network topologies with different sizes and variabilities by using three representative data sets. Figure 4.7, Figure 4.9, and Figure 4.10 shows the relative error for these data sets for various percentages of malicious nodes. Each of the topologies is adversely effected, with the King data set (Figure 4.7) showing the greatest degradation in accuracy due to the fact it has more variation in RTT and is prone to excessive over and under estimation in response to an attack. Meridian (Figure 4.9) shows less degradation due to the fact that it has less variation in its link latencies. AMP (Figure 4.10) shows more variability in the relative error due to its small size and frequent, large-scale node coordinate changes. As previously noted, as the percentage of attackers increases, the ability of the system to accurately estimate latency significantly degrades regardless of the topology.

4.4.4 Threshold Selection for Spatial-Temporal Outlier Detection

An important aspect of our approach is selecting the temporal and spatial thresholds that allow for the identification of potentially malicious query responses and eliminate them from the coordinate computation process. We consider the same attack scenario with a percentage of attackers as in Section 4.4.3 to experimentally determine our outlier detection thresholds since this scenario is one of the most difficult in which to identify malicious responses. When a malicious node selects a coordinate to respond with, this coordinate is selected from an area in which many altruistic nodes reside. The malicious nodes also report low but variable error inline with low-error altruistic nodes. These factors help disguise the malicious nodes actions and make them much harder to detect.

We use a slightly modified version of the method proposed in Section 4.3. Specifically, we do not use latency in the outlier detection due to the fact that the latencies are predetermined in the simulator and thus show little variability.

Temporal Threshold Selection

We used a threshold of 4.0 for our temporal outlier detection to allow for the four features: remote error, local error, change in remote coordinates, and change in local coordinates to vary at most one standard deviation over each feature from their temporally developed mean. The value was chosen based on the formula of the simplified Mahalanobis distance as discussed by Wang and Stolfo [75].

Spatial Threshold Selection

The threshold for our outlier detection can be mathematically derived as by Smith and Cheeseman [77] and Ribeiro [78], assuming a multivariate Gaussian distribution for the metrics vector. The contours of equal probability of this distribution create a 2-dimensional ellipse and the outlier threshold reflects the probability of a vector

Table 4.4
False positive rate (percentage) and median prediction error for different spatial outlier thresholds (King data sets)

% Mal. Nodes	*Spatial Outlier Threshold*			
	1.25	1.50	1.75	2.00
0	28, 16ms	21, 16ms	17, 16ms	13, 16ms
10	17, 17ms	13, 18ms	10, 19ms	5, 20ms
20	21, 18ms	15, 21ms	7, 23ms	6, 26ms
30	27, 20ms	11, 22ms	10, 33ms	9, 36ms

being within the ellipse whose semi-axes are determined by k. The probability that a random vector lies within the ellipse increases with the size of k. Thus, for a given value of k the probability that a probed tuple lies within the ellipse can be computed as:

$$P = 1 - e^{\frac{-k^2}{2}} \tag{4.4}$$

We initially analytically selected $k = 1.5$, in theory creating a threshold through which 53% of the coordinate updates would successfully pass. Through empirical testing of over 200,000 coordinate updates over multiple simulations, we found an ellipse determined by this threshold will allow approximately 79% of the updates to pass. This variation from the mathematically derived value can be attributed to the fact that the used metrics do not form a perfect normalized distribution and have a smaller variance than assumed in Equation 4.4. A node may select smaller spatial threshold values for stronger security guarantees, with the drawback that it may find its coordinate less accurate due to discarding valid updates.

Figure 4.11, Figure 4.12, and Figure 4.13 present the relative error for the King data set in which the temporal outlier threshold was set to 4.0 and various spatial outlier detection thresholds were tested. Table 4.4 presents the corresponding false positive rate and median system prediction error for the different thresholds.

Although higher thresholds provide a smaller false positive rate, they do induce a higher error rate. For example, as malicious nodes are introduced into the system, a threshold of 2.00 maintains a low false positive rate with the trade-offs that the prediction error raises to 36ms, with 14ms more than the threshold of 1.5 which maintains a prediction error of 22ms when 30% of the nodes are malicious. We note that virtual coordinate systems are designed to be long-running services and hence the presence of a small percentage of false positive will not hinder the system. Based on the results in the three figures and Table 4.4 we conclude that a spatial threshold of 1.5 worked well for different percentages of attackers while having an acceptable false positive rate.

4.4.5 Mitigating Attacks Against Virtual Coordinate Systems

In this section we demonstrate the effectiveness of our defense mechanisms at mitigating the effects of malicious nodes and sustaining the usability of the system.

Inflation and Deflation Attacks

We begin by reexamining the inflation and deflation attacks against a victim node, this time with a system using our defense mechanisms. The victim node is able to identify and mitigate the effect of the malicious nodes, achieving a prediction error of 11ms, as shown in Figure 4.4. The error is similar to a system under non-attack conditions (10ms), and nearly six times less than the unprotected system.

Different Percentage of Malicious Nodes

Figure 4.11, Figure 4.12, and Figure 4.13 present the relative error for the King data set for different percentages of malicious nodes. Note that for a spatial threshold of 1.5, our solution mitigates the system instability caused by the malicious nodes and even helps the system to stabilize at a more accurate local minimum than the initial

protocol design to tolerate benign errors. While each node may occasionally accept erroneous data from malicious nodes due to a short temporal history or a skewed spatial history with updates from only a few nodes (as can be seen by the brief rise in error before coming back down), over time the system is able to avoid many malicious updates.

Different Network Topologies

Figure 4.14 and Table 4.5 show the results for the King, Meridian and AMP topologies with and without outlier detection, where the attack scenario is the same as the coalition attack in Section 4.4.3. Applying the spatial threshold of 1.5 which was tested on the King data set, we find our solution is able to mitigate the system instability in all three data sets.

The King data set (Figure 4.14(a)), previously seen in Figure 4.12(b) and Figure 4.13(b)), maintains a low relative error for various percentages of the attackers. We also note it is able to maintain a low system prediction error and low number of false positives (Table 4.5). In Table 4.5, the less the system prediction error increased with the number of attackers, the more resiliently the system performed under attack. Similar trends can also be observed for the Meridian data set (Figure 4.14(b)). While our solution is able to offer protection to the smaller scale AMP data set from malicious nodes, it can be seen from Figure 4.14(c) that larger percentages of malicious nodes begin to overwhelm the system. This occurs since the percentage of malicious nodes is high ($\geq 30\%$) and each benign node will have many malicious reference set members. For example, given that 30% of the total nodes are malicious, the probability that at least 30% of the nodes in a reference set of AMP are also malicious is about 67%. This is nearly double the probability for King or Meridian under the same conditions due to AMP's much smaller size.

We validate that the difference between the system under attack using our defense mechanisms and the system operating under non-attack conditions are not statisti-

Table 4.5
False positive rate (percentage) and median prediction error for different data sets using a spatial outlier threshold of 1.5

% Mal. Nodes	***Topology***		
	Meridian	AMP	King
0	23, 30ms	21, 18ms	21, 16ms
10	13, 30ms	15, 20ms	13, 18ms
20	12, 32ms	14, 25ms	15, 21ms
30	11, 40ms	12, 36ms	11, 22ms

cally significant, meaning the attacks had little effect. For a smaller numbers of attackers (<30%), we test the null hypothesis $H_0 : \mu_{defense} = \mu_{normal}$ using the two-sample t-test with pooled variance. Unexpectedly, we disprove H_0, concluding with high probability (nearly 100%) that the error results come from distributions with different means. Upon closer inspection, with high confidence, the actual mean of the system with defense is below that of the system under non-attack conditions, showing our defense mechanisms *improves* the operation of the system even under attack. For larger number of attackers (≥30%), the actual mean of the system with defense is slightly greater than that of the system under non-attack conditions. From this, we conclude our technique degrades gracefully and is still able to prevent much of the damage attackers would be able to cause on an unprotected system.

Malicious Coalition Size Tolerated by Outlier Detection

All defense mechanisms and protocols resilient to insiders have limitations regarding the number of attackers they can tolerate. We analyze the number of malicious colluding nodes that can be tolerated by our outlier detection mechanism using a reference set size of 64. Table 4.6 presents the number of malicious nodes in a reference set which by colluding can influence the spatial centroid calculation enough to allow the attack types discussed in Section 4.2 to bypass the detection mechanism. Nearly twenty malicious nodes (or 30% of the reference set size) are required for nearly all of the identified attack types across the three data sets. The deflation attack is more successful for AMP since the RTTs are less variable and the virtual coordinate system creates one main cluster (Figure 4.3(c)) that contains all of the nodes. This also explains why high percentages of malicious nodes (≥ 30%) were able to overwhelm our solution in the AMP scenarios. In these cases, the benign nodes were likely to have twenty or more malicious nodes in their reference set, which could cause the spatial centroid to shift and allow malicious updates to pass undetected. We conclude that our defense method works well when the size of the malicious coalition is smaller than

Table 4.6
Number of colluding nodes tolerated by spatial outlier detection for different data sets using a spatial outlier threshold of 1.5

	Data Set		
Attack Type	King	Meridian	AMP
Inflation	19.7	21.6	19.8
Deflation	20.2	19.8	12.6
Oscillation	19.6	20.3	19.3

one third of the total number of nodes in the reference set. This bound is inline with the performance of other methods that tolerate malicious insiders (*e.g.*, [43]).

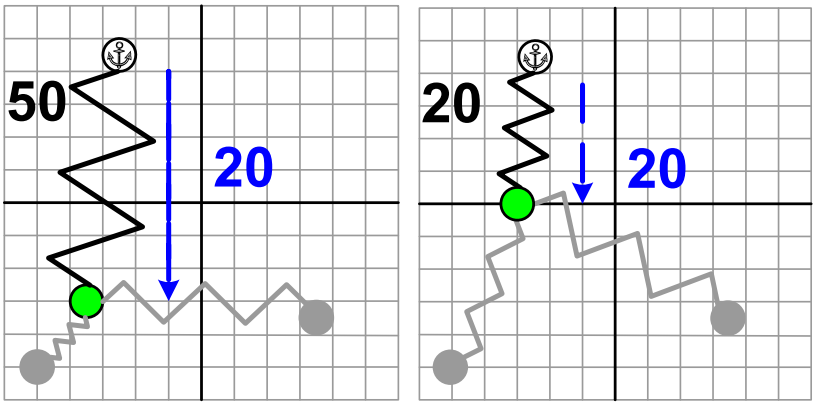
System Overhead

Our defense mechanisms adds minimal link stress as it uses one additional numerical value in addition to the information already being exchanged between nodes. The memory utilization for spatial correlation requires maintaining the most recent u updates. In the case of the temporal outlier detection, the memory usage consists of maintaining the temporal centroid. By incrementally updating the centroid, we do not need to maintain the entire history for each probed node but only need to store the mean, standard deviation, and count for each of the metrics. The additional computational complexity is bounded by the number of nodes in the reference set which is constant. The computation of the temporal and spatial outliers is a constant time calculation performed at each node when it updates its coordinate.

4.5 Summary

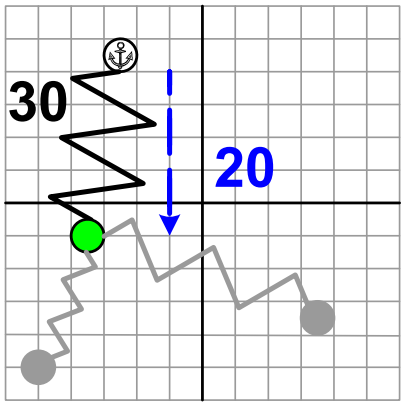
In this chapter, we studied attacks against the accuracy of virtual coordinate systems, classifying three types of attack: coordinate inflation, coordinate deflation,

and coordinate oscillation. We showed that even a small number of attackers can severely degrade coordinate accuracy due to the epidemic nature of the attacks. We proposed spatial-temporal correlation to perform outlier detection on updates received from malicious nodes and eliminate them from the coordinate computation process and experimentally demonstrated the utility of our technique. Finally, we examined the limitations of our defense technique and found that the method starts degrading when more than 30% of the nodes in a reference set form a malicious coalition.



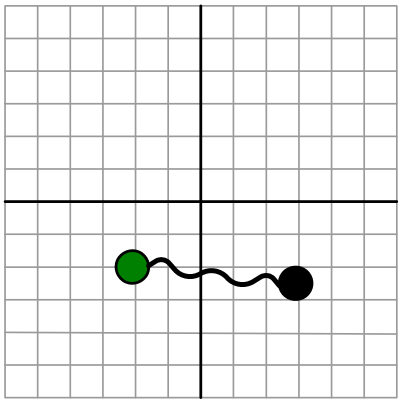
(a) Beginning Coordinate Update

(b) Coordinate Change

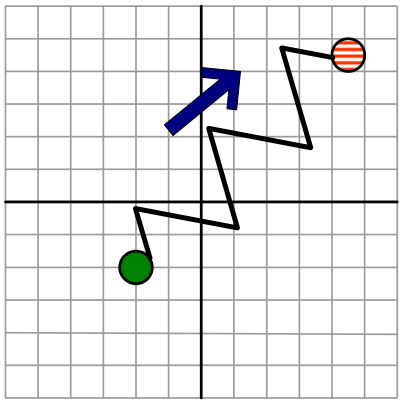


(c) Coordinate Change with Dampening

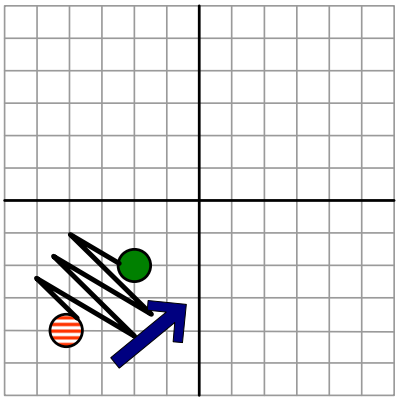
Figure 4.1. Example of the Vivaldi coordinate update process where the highlighted node is updating its location by querying the reference set node denoted by the anchor. The grayed-out nodes and links are part of the coordinate system but not the current update procedure. In Figure 4.1(a) the highlighted node exchanges information with anchor reference node. Figure 4.1(b) and Figure 4.1(c) depict the change in location of the highlighted node (and the estimated latency) when the node updates its coordinate using Algorithm 3. Figure 4.1(c) highlights the use of the dampening factor to minimize excess movement.



(a) No Attack

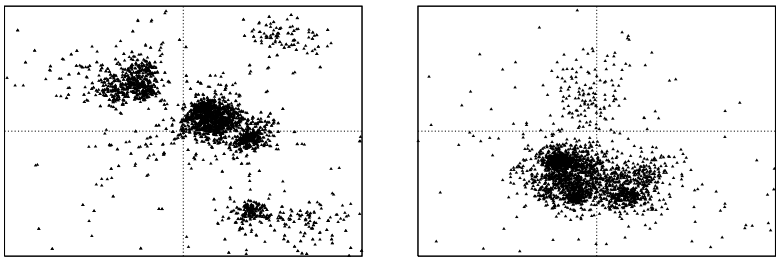


(b) Coordinate Inflation, Pull

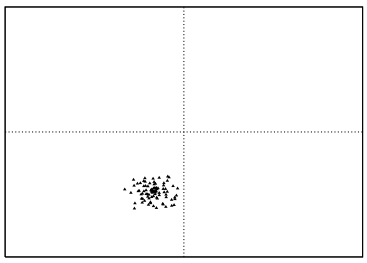


(c) Coordinate Inflation, Push

Figure 4.2. Example inflation attack scenarios against an individual victim node. Figure 4.2(a) represents the system in a benign scenario where the tension on the logical spring connecting the two nodes is at a minimum and not inducing movement. Figure 4.2(b) represents the coordinate inflation attack in which the malicious node is pulling the victim away from its correct position. Similarly, Figure 4.2(c) represents the coordinate inflation attack in which the malicious node is pushing the victim away from its correct coordinates.



(a) King (b) Meridian

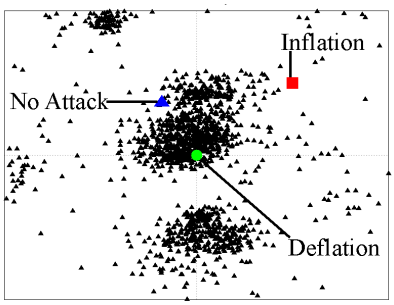


(c) AMP

Figure 4.3. Node placement chosen by Vivaldi for various data sets

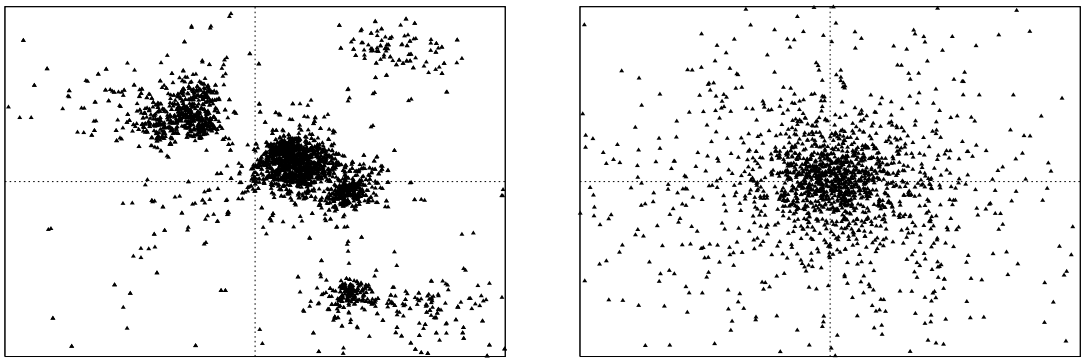
Attack	Pred. Error
None	10 ms
Deflation	60 ms
Inflation	70 ms
w/defense	11 ms

(a) Prediction Error



(b) Node Placement

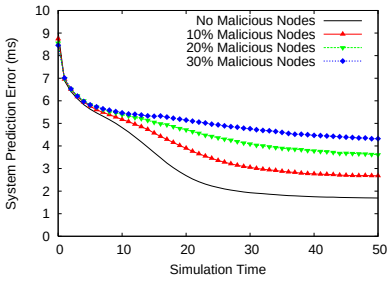
Figure 4.4. Victim node error and placement for a deflation and inflation attack (King)



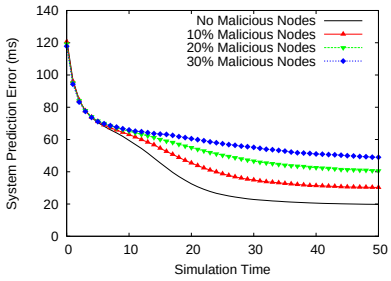
(a) No attack

(b) Oscillation attack

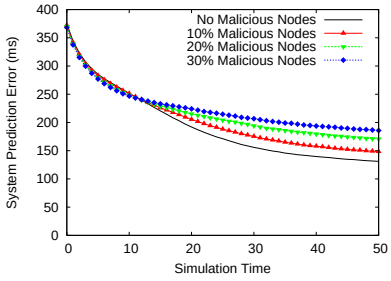
Figure 4.5. Virtual coordinate system node placement under an oscillation attack (King)



(a) 5^{th} Percentile Prediction Error

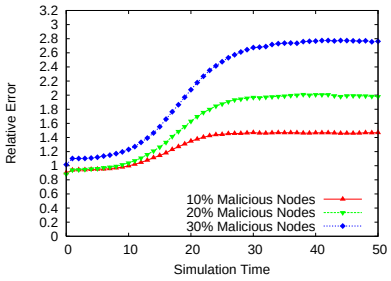


(b) 50^{th} Percentile Prediction Error

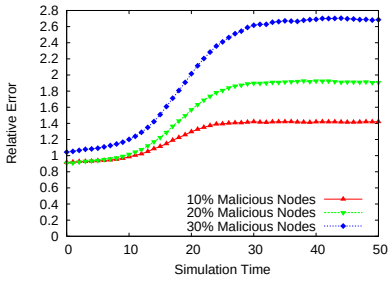


(c) 95^{th} Percentile Prediction Error

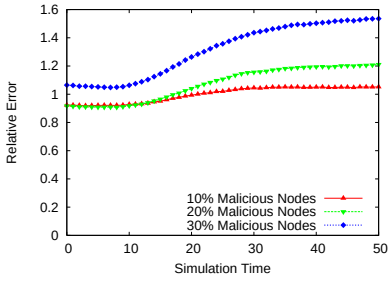
Figure 4.6. System prediction error under different percentages of attackers (King)



(a) 5^{th} Percentile Relative Error

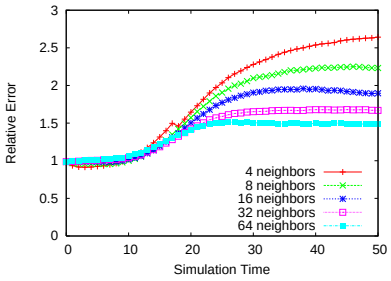


(b) 50^{th} Percentile Relative Error

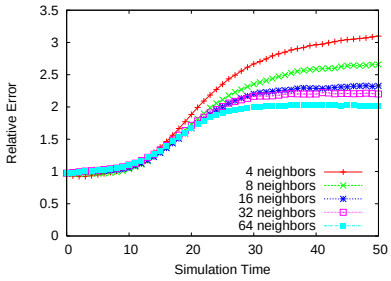


(c) 95^{th} Percentile Relative Error

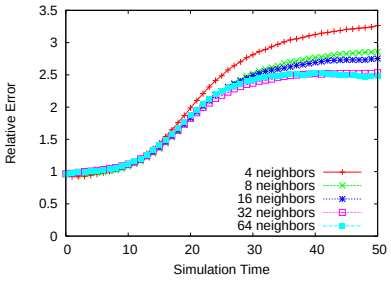
Figure 4.7. Relative error under under different percentages of attackers (King)



(a) 10% Malicious

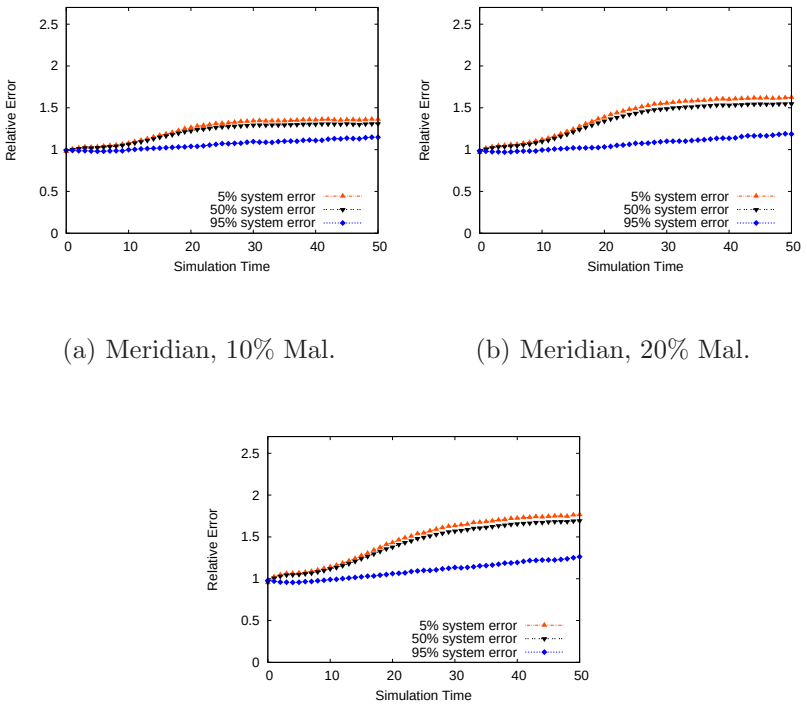


(b) 20% Malicious



(c) 30% Malicious

Figure 4.8. Median relative error under different reference set size with varying percentages of malicious nodes (King)

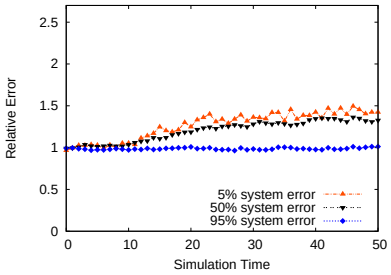


(a) Meridian, 10% Mal.

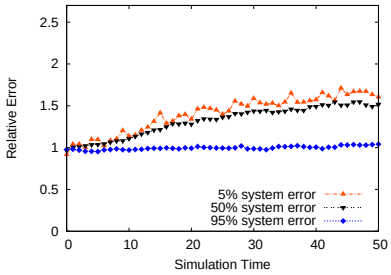
(b) Meridian, 20% Mal.

(c) Meridian, 30% Mal.

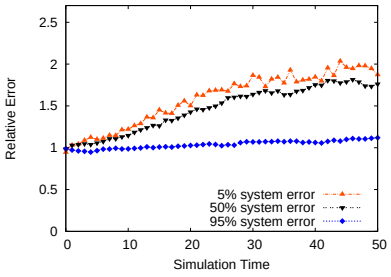
Figure 4.9. Relative error under under different percentages of attackers (Meridian)



(a) AMP, 10% Mal.

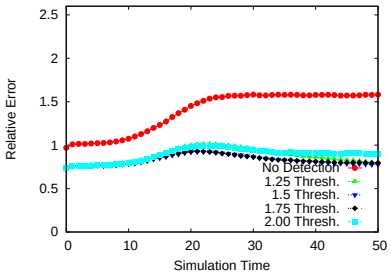


(b) AMP, 20% Mal.

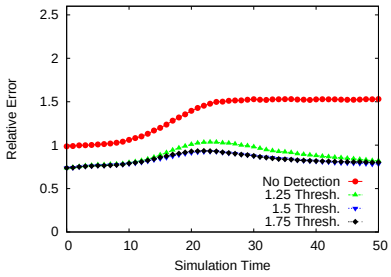


(c) AMP, 30% Mal.

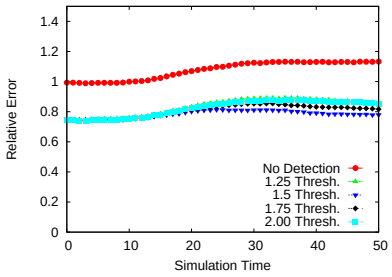
Figure 4.10. Relative error under under different percentages of attackers (AMP)



(a) 5^{th} Pct. Relative Error

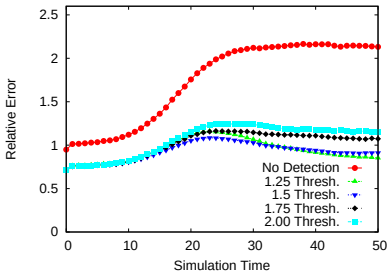


(b) 50^{th} Pct. Relative Error

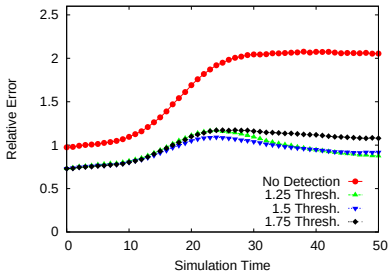


(c) 95^{th} Pct. Relative Error

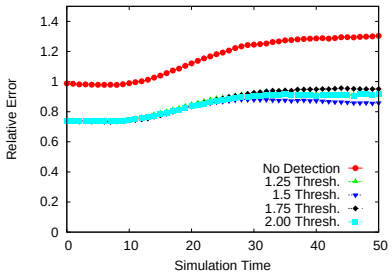
Figure 4.11. Relative error using different spatial outlier thresholds when 10% of the network is malicious (King)



(a) 5^{th} Pct. Relative Error

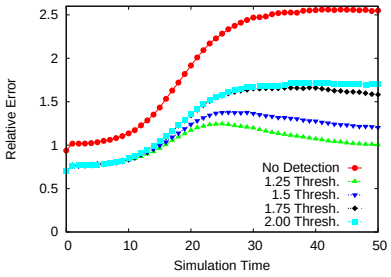


(b) 50^{th} Pct. Relative Error

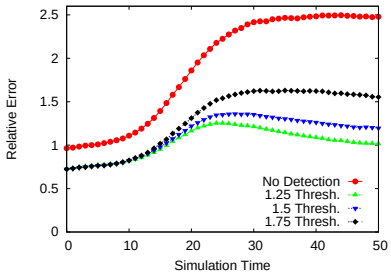


(c) 95^{th} Pct. Relative Error

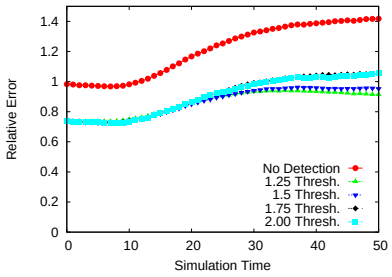
Figure 4.12. Relative error using different spatial outlier thresholds when 20% of the network is malicious (King)



(a) 5^{th} Pct. Relative Error

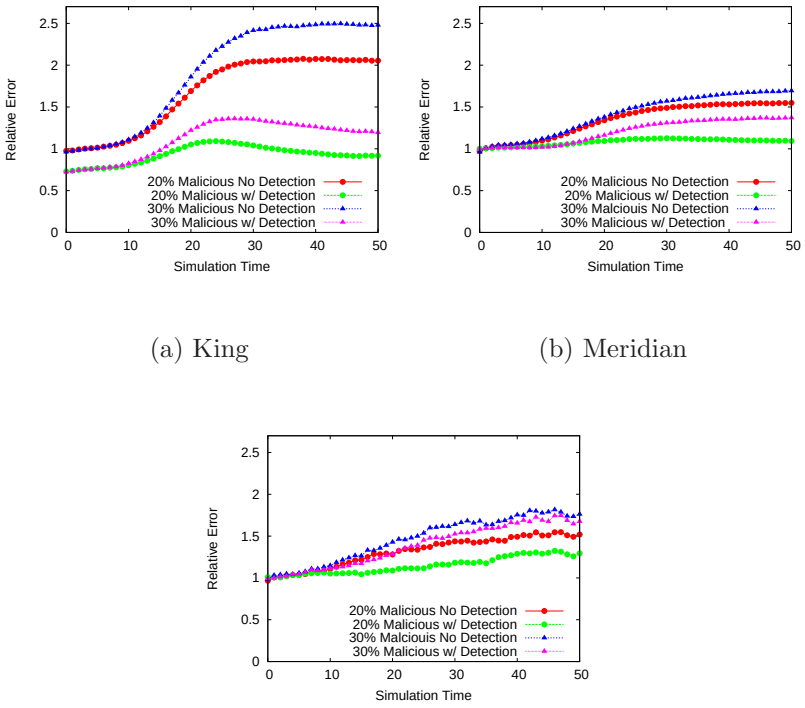


(b) 50^{th} Pct. Relative Error



(c) 95^{th} Pct. Relative Error

Figure 4.13. Relative error using different spatial outlier thresholds when 30% of the network is malicious (King)



(a) King

(b) Meridian

(c) AMP

Figure 4.14. Relative error under different percentage of attackers using a spatial outlier threshold of 1.5 with three real-life Internet latency data sets

5 RESPONDING TO IDENTIFIED THREATS

As peer-to-peer applications are deployed in unsecured environments such as the Internet, they will often be the target of malicious activity. Even though designing peer-to-peer protocols based on security principles such as those identified by Saltzer and Schroeder [34] will prevent many attacks from being successful, some attackers will still prevail and degrade the quality of service provided by the application. With this mind, peer-to-peer applications need an effective and efficient mechanism to respond to malicious nodes that interfere with the applications. A considerable amount of research has focused on the development of trust and reputation systems for peer-to-peer systems that help users make beneficial decisions assuming the existence of detection mechanisms for malicious behavior [104, 105]. It has been shown by Aberer and Despotovic [106], Damiani *et al.* [107], and Xiong and Liu [108] that these systems can provide an effective way to mitigate the effects of malicious nodes in decentralized distributed systems.

In order for reputation systems to become an important component of current and next generation peer-to-peer systems, it is critical that we understand the fundamental strengths and weaknesses of different reputation system design choices and how to integrate them with other system components. These choices will dictate the resiliency of the reputation system to attack and its ability to provide an accurate representation of trust in the network.

In this section, we provide an overview of reputation systems, analyze their possible vulnerabilities to malicious behavior, and provide a concrete solution utilizing the EigenTrust [105] reputation system to respond to malicious behavior in the context of the adaptive multicast system, ESM [35]. We summarize our key contributions:

- We provide an overview of an analytical framework by which reputation systems can be decomposed, analyzed, and compared using a common set of metrics. This framework facilitates insights into the strengths and weaknesses of different systems and comparisons within a unified framework. Additionally, we classify attacks against reputation systems, analyzing what system components are exploited by each attack type. We elucidate the relevance of the framework and the attacks by providing specific examples to the EigenTrust reputation system.

- We propose techniques to isolate malicious nodes by aggregating the local suspicious behavior derived from local observations (based on outlier detection) to build a global reputation for each overlay node using the EigenTrust reputation system [105]. We demonstrate the benefits of our response mechanism through deployments of ESM on the PlanetLab [58] Internet testbed.

The rest of the chapter is organized as follows: We provide an overview of reputation systems in Section 5.1 and their vulnerability to attack in Section 5.2. We present our two-part approach for responding to malicious behavior which uses local observed behavior to generate an immediate local bias against misbehaving nodes and subsequently allows the overlay to construct and share global knowledge about the malicious nodes in Section 5.3. We present experimental results demonstrating the attacks and the utility of the response technique in Section 5.4 and conclude this chapter in Section 5.5.

5.1 Reputation Systems

Reputation and trust play a pivotal role in peer-to-peer applications by enabling multiple parties, whether human or automated, to establish relationships that achieve mutual benefit. In general, ***reputation*** is the opinion of the public toward a person, a group of people, or an organization. In the context of peer-to-peer applications, reputation represents the opinions nodes in the system have about their peers. Repu-

tation allows parties to build ***trust***, or the degree to which one party has confidence in another within the context of a given purpose or decision. By harnessing the community knowledge in the form of feedback, reputation-based trust systems help participants decide who to trust, encourage trustworthy behavior, and deter dishonest participation by providing a means through which reputation and ultimately trust can be quantified and disseminated [109]. Without such mechanisms, opportunism can erode the foundations of these collaborative applications and lead to peer mistrust and eventual system failure [110].

Due to their common purpose, reputation systems naturally share similar structural patterns. Understanding these similarities and developing an analysis framework serves a twofold purpose. First, it provides greater insight into prior research, facilitating common ground comparison between different systems. Second, it provides insights into the fundamental strengths and weaknesses of certain design choices, contributing to the future design of attack-resilient reputation systems. We identify the following three dimensions as being fundamental to any reputation system:

- ***Formulation*** The ideal mathematical underpinnings of the reputation metric and the sources of input to that formulation. For example, a system may accept positive and negative feedback information, weighted as $+1$ and -1 and define an identity's reputation to be the summation of all of its corresponding feedback.

- ***Calculation*** The algorithm to calculate the mathematical formulation for a given set of constraints (physical distribution of participants, type of communication substrate, *etc.*). For example, the algorithm to calculate the formulation could specify that a random set of peers is queried and the feedback received for each identity tallied.

- ***Dissemination*** The mechanism that allows system participants to obtain the reputation metrics resultant from the calculation. Such a mechanism may involve storing the values and disseminating them to the participants. For exam-

ple, a system might choose to use a distributed hash table to store the calculated reputation values and a gossip protocol to distribute new information.

Realizing each reputation system is composed of these dimensions provides a basis for comparison between different systems and provides insights into the fundamental strengths and weaknesses of design choices with respect to malicious behavior. By analyzing a reputation system based on the composition of these dimensions, we can determine the vulnerabilities to which it is susceptible and which system will operate the best in a given environment.

5.1.1 The Dimensions of the EigenTrust Reputation System

One of the most well known and widely studied reputation systems is EigenTrust [105]. The EigenTrust reputation system was motivated by the need to filter out inauthentic content in peer-to-peer file sharing networks. EigenTrust calculates a global reputation value for each peer in the system based on the local opinions of all of the other peers.

Formulation

The input to the EigenTrust formulation consists of the information derived from the direct experience a peer has with other peers in the network and indirect information about the perception of neighboring peers about each other. To acquire the direct information, users provide manual feedback about each peer-to-peer transaction. A user ranks each transaction using the binary scale of positive or negative and the summation of these values is used as input into the formulation. The indirect information is automatically exchanged between peers and is what gives the system the ability to develop transitive trust. The system considers both positive and negative information and is biased towards positive information.

The formulation does not take into consideration the effects of how reputations change over time. While it is true that the local trust values are a summation of all votes ever cast for a particular identity, the formulation itself makes no attempt to distinguish between votes cast today vs. votes cast a year ago.

The intuition behind EigenTrust formulation is that a node i forms a broader trust value (t_{ik}) in node k by asking its neighbors to report their trust in k and weighting those opinions by i's trust in its neighbor: $t_{ik} = \sum_j c_{ij}c_{jk}$, where c_{ij} is the normalized local trust value of node i in node j. The value c_{ij} is calculated as follows:

$$c_{ij} = \begin{cases} \frac{max(s_{ij},0)}{\sum_j max(s_{ij},0)} & \text{if } \sum_j max(s_{ij},0) > 0 \\ p_j & \text{otherwise} \end{cases} \tag{5.1}$$

where s_{ij} represents the non-normalized local trust value at node i in node j and p_j represents the default trust value for the node j. As the only trusted node for our application is the source, $p_j = 1$ if the node is the source and $p_j = 0$, otherwise. A node will have a local trust value of zero with all nodes it has not interacted with and initially only trust the source. When a node i first interacts with another node j, it forms a new non-zero local trust value, s_{ij}, based on the "goodness" of the interaction. The reputation metric is formulated deterministically and produces values on a continuous spectrum between 0.0 and 1.0.

Calculation

Algorithm 5: Basic EigenTrust Algorithm

$\vec{t}^0 = \vec{p}$;

while $\delta > \epsilon$ **do**

  $\vec{t}^{k+1} = C^T\vec{t}^k$;

  $\vec{t}^{k+1} = (1-\lambda)\vec{t}^{k+1} + \lambda\vec{p}$;

  $\delta = \left\|\vec{t}^{(k+1)} - \vec{t}^{(k)}\right\|$

end

Given the formulation of trust in EigenTrust is based on the summation of observations and indirect data, it naturally lends itself to calculation using matrix operations. Using the basic EigenTrust Algorithm presented by Kamvar *et al.* [105] and reproduced in Algorithm 5, the idea of transitive trust can be extended to formulate a system wide trust ranking by formulating the summation of local trust values as a matrix multiplication. Through each iteration of multiplying the global trust vector $\vec{t}$ by the aggregated local trust values contained in C, the algorithm intuitively represents asking successively further nodes opinions of their neighbors. After each iteration, each of the trust values stored at source in $\vec{t}$ is normalized to guarantee that meaningful comparisons between values can be performed, but not that all trust values add up to one. The calculation continues until the convergence of $\delta = \left\|\vec{t}^{(k+1)} - \vec{t}^{(k)}\right\| < \epsilon$, where ϵ is empirically set to .0001. To guarantee that the calculation will converge, the pre-trusted nodes trust vector, $\vec{p}$, is used as the starting vector ($t^0 = \vec{p}$). To mitigate the effects of malicious coalitions of nodes cooperating to subvert the reputation system, the pre-trusted nodes are favored with a certain weight, λ, after each iteration.

While the formulation lends itself naturally to a centralized calculation based upon matrix operations, this is not desirable in the peer-to-peer file sharing environment. Instead, each peer calculates the global trust values by using a randomized algorithm which guarantees that each participant will converge to the same value within some error bounds. For the original distributed algorithm, the cost to calculate the global trust value for one identity is $O(n)$ in the worst case since (a) the number of iterations needed to converge can be viewed as constant and (b) it will need to potentially communicate with all other identities. Through optimization this bound can be reduced to $O(\log n)$ without compromising accuracy [105].

Dissemination

EigenTrust uses a deterministic dissemination framework which is dictated by the underlying application.

5.2 Vulnerabilities of Reputation Systems in Adversarial Environments

The success of a reputation system is measured by how accurately the calculated reputations predict the quality of future interactions. This is difficult to achieve in an environment where any party can attempt to exploit the system for its own benefit. The impact of attacks against reputation systems reaches beyond just the manipulation of virtual numbers and turns into dollars fraudulently lost and ruined business reputations [111].

We classify attacks against reputation systems based on the goals of the reputation systems targeted by attacks. The goal of a reputation system is to ensure that the reputation metrics correctly reflect the actions taken by participants in the system and cannot be maliciously manipulated. This is not achieved if participants can falsely improve their own reputation or degrade the reputations of others. As a result of the attacks, misbehaving participants can obtain unwarranted service or honest participants can be prevented from obtaining service.

We identify several classes of attacks:

- ***Self-Promoting*** - Attackers manipulate their own reputation by falsely increasing it.
- ***Self-Serving or Whitewashing*** - Attackers escape the consequence of abusing the system by using some system vulnerability to repair their reputation. Once they restore their reputation, the attackers can continue the malicious behavior.
- ***Slandering*** - Attackers manipulate the reputation of other nodes by reporting false data to lower their reputation.

- ***Orchestrated*** - Attackers orchestrate their efforts and employ several of the above strategies.

- ***Denial of Service*** - Attackers may cause denial of service by either lowering the reputation of victim nodes so they cannot use the system or by preventing the calculation and dissemination of reputation values.

Each of these attacks can damage the reputation system and must be considered when selecting a system to secure and utilize. Besides degrading the utility of the reputation system, the attacks adversely impact any application using the reputation system to augment decisions. We must carefully analyze the goals of the peer-to-peer application and the environment in which the reputation system will be deployed in order to choose the best reputation system for the purpose.

5.2.1 Self-promoting

In self-promoting attacks, attackers seek to falsely augment their own reputation. Such attacks are only possible in systems that consider positive feedback in the formulation. Fundamentally, this is an attack against the formulation, but attackers may also exploit weaknesses in the calculation or dissemination dimensions to falsely increase reputation metric values.

Self-promotion attacks can be performed by a lone identity or organized in groups of collaborating identities. One very basic form of the attack occurs when an attacker fabricates fake positive feedback about itself or modifies its own reputation during the dissemination. Systems lacking mechanisms to provide data authentication and integrity are vulnerable to such attacks as they are not able to discern between fabricated and legitimate feedbacks.

However, even if source data is authenticated using cryptographic mechanisms, self-promotion attacks are possible if disparate identities or a single physical identity acquiring multiple identities through a Sybil attack [112] collude to promote each other. Systems that do not require participants to provide proof of interactions which

result in positive reputations are particularly vulnerable to this attack. To perform the attack, colluding identities mutually participate in events that generate real feedback, resulting in high volumes of positive feedback for the colluding participants. Because the colluders are synthesizing events that produce verifiable feedback at a collective rate faster than the average, they are able to improve their reputations faster than honest participants or counter the effects of possible negative feedback. Such patterns of attack have been observed in the Maze file sharing system [113]. Attackers that are also interacting with other identities in honest ways are known as *moles* [114]. Colluding attackers can also contribute further to the self-promotion of each other by manipulating the computation dimension when aggregating reputation values.

5.2.2 Whitewashing

Whitewashing attacks occur when attackers abuse the system for short-term gains by letting their reputation degrade and then escape the consequences of abusing the system by using some system vulnerability to repair their reputation. Often attackers will attempt to re-enter the system with a new identity and a fresh reputation [115]. The attack is facilitated by the availability of cheap pseudonyms and the fact that reciprocity is much harder to maintain with easily changed identifiers [116].

This attack fundamentally targets the reputation system's formulation. Formulations that are based *exclusively on negative* feedback are especially vulnerable to this type of behavior since newcomers have equal reputation metric values to participants which have shown good long-term behavior. Separately, a reputation system using either type of feedback (positive and/or negative) is vulnerable if the formulation relies exclusively on long-term history without discriminating between old and recent actions. If an attacker is able to generate a beneficial reputation based solely on history, it can perform short duration malicious attacks with little risk of negative consequences as the previous history will heavily outweigh current actions. This can have a large impact on the system as the malicious node will continue to have a

high reputation for a substantial period of time during which the system is slow to identify the malicious behavior and unable to sufficiently lower the reputation of the malicious node. In systems with formulations that include positive feedback, attackers may have to behave honestly for an initial period of time to build up a positive reputation before starting the self-serving attack. Attackers that follow this pattern are also known as *traitors* [117].

Whitewashing attacks may be combined with other types of attacks to make each attack more effective. For example, in systems with both positive and negative feedback, concurrently executing a self-promoting attack will lengthen the duration of effectiveness of a whitewashing attack. Likewise, whitewashing identities may slander those identities that give negative feedback about the attacker so that their negative feedback will appear less reputable since many systems weight the opinions of an identity by its current level of trustworthiness. In this case, slandering minimizes the amount of whitewashing an attacker must perform to maintain a good reputation.

5.2.3 Slandering

In *slandering attacks*, one or more identities falsely produce negative feedback about other identities. As with self-promoting attacks, systems that do not authenticate the origin of the feedback are extremely vulnerable to slander. In general, these attacks target the formulation dimension of a reputation system.

The attack can be conducted both by a single attacker and a coalition of attackers. As typically the effect of a single slandering node is small, especially if the system limits the rate at which valid negative feedback can be produced, slandering attacks primarily involve collusion between several identities. Depending on the application of the system, slandering attacks may be more or less severe than self-promotion attacks. For example, in high-value monetary systems, the presence of even small amounts of negative feedback may severely harm an identity's reputation and ability to conduct business [118].

The lack of authentication and high sensitivity of the formulation to negative feedback are the main factors that facilitate slandering attacks. Reputation systems must consider the inherent trade-offs in the sensitivity of the formulation to negative feedback. If the sensitivity is lower, then the formulation is robust against malicious collectives falsely slandering a single entity, but it allows entities to exhibit bad behavior for a longer time, for the same decrease in reputation. On the other hand, if sensitivity is higher, the bad behavior of a single identity can be punished quickly, but honest identities are more susceptible to attacks from malicious collectives. If malicious collectives are well-behaved except to slander a single identity it may be difficult to distinguish that slander from the scenario where the single identity actually deserved the bad feedback that was received.

5.2.4 Orchestrated

Unlike the previously described attacks that employ primarily one strategy, in *orchestrated* attacks, colluders follow a multifaced, coordinated attack. These attacks utilize multiple strategies, where attackers employ different attack vectors, change their behavior over time, and divide up identities to target. While orchestrated attacks fundamentally target a system's formulation, these attacks also may target the calculation and dissemination dimensions. If the colluding attackers become a significant part of the calculation or dissemination of reputation within an area of the system, they can potentially alter reputation metric values to their benefit.

One example of an orchestrated attack, known as an oscillation attack [119], is where colluders divide themselves into teams and each team plays a different role at different times. At one point in time, some teams will exhibit honest behavior while the other teams exhibit dishonest behavior. The honest teams serve to build their own reputations as well as decrease the speed of decline of the reputation of the dishonest teams. The dishonest teams attempt to gain the benefits of dishonest behavior for as long as possible, until their reputation is too low to obtain benefit from the system. At

this point, the roles of the teams switch, so that the dishonest teams can rebuild their reputation and the previously honest teams can begin exhibiting dishonest behavior. Even more complex scenarios are possible where there are more than two roles. For example, one team of nodes may self-promote, another may slander benign nodes, and the final team misbehaves in the context of the peer-to-peer system, such as dropping maintenance messages used to maintain the system structure and connectivity.

Orchestrated attacks are most effective when there are several colluders for each role. Larger numbers allow each colluder to be linked less tightly to other colluders, which makes detection much more difficult. Colluders performing orchestrated attacks balance between maximizing selfish or malicious behavior and avoiding detection. Robust formulations increase the number of colluders that must participate in order to achieve the desired effect.

5.2.5 Denial of Service

Finally, attackers may seek to subvert the mechanisms underlying the reputation system itself, causing a denial of service. Such attacks are conducted by malicious nonrational attackers, making them difficult to defend against. Systems using centralized approaches and lacking any type of redundancy are typically vulnerable to denial of service attacks. Attackers can attempt to cause the central entity to become overloaded (e.g. by overloading its network or computational resources). These attacks target the calculation and dissemination dimensions of a system and are performed by groups of colluding attackers.

Preventing a reputation system from operating properly with a denial of service attack may be as attractive to attackers as corrupting the reputation values, especially if the application employing the reputation system is automated and needs to make decisions in a timely fashion. For example, consider a peer-to-peer data dissemination application where data is routed along the most trustworthy paths. If the reputation system is inoperable, the system relying on reputation may need to continue to route

data even if reputation information is unavailable, allowing malicious identities to participate for periods of time without their negative reputations being known (or without being punished for their negative behavior).

5.2.6 EigenTrust Defense Mechanisms

The EigenTrust formulation has foundations in statistics, as the global trust vector can be formulated as the stationary distribution of a Markov chain. The formulation is designed so nodes give greater weight to information obtained from their neighbors or nodes they have interacted with in the past to mitigate malicious manipulations. Pre-trusted identities are used to bias reputation values towards known good nodes and ensure that the randomized calculation will converge quickly. Redundancy is employed during the calculation and dissemination stages to prevent benign data loss and malicious data tampering. Each of the score managers for an identity is randomly selected, making it less likely that a single malicious collective will be responsible for the reputation value for any one identity. The combination of these defense techniques makes EigenTrust robust to all of the previously mention attacks except for the whitewashing attacks.

5.3 Mitigating Identified Threats Based on Local and Global Reputation

In order to demonstrate how to analyze and integrate reputation into peer-to-peer applications, we designed a response mechanism based on a reputation system for the adaptive multicast system we explored in Chapter 3, ESM [35]. By analyzing local information at each node in ESM, we are able to detect possible malicious activity. Once this activity has been detected, corrective measures must be taken to isolate the malicious nodes and minimize their effect on the overlay network. Without appropriate response mechanisms, the overall system performance may suffer as the malicious nodes continue to interfere with the system. We propose a two-prong approach which uses local observed behavior to generate an immediate local bias against misbehaving

nodes and subsequently allows the overlay to construct and share global knowledge about the malicious nodes.

5.3.1 Isolating Malicious Nodes in ESM

Once malicious nodes are detected, corrective measures must be taken to isolate them and minimize their effect on the overlay network. Without appropriate response mechanisms, the overall system performance may suffer as the malicious nodes continue to interfere with the system. We propose a two-prong approach which uses local observed behavior to generate an immediate local bias against misbehaving nodes and subsequently allows the overlay to construct and share global knowledge about the malicious nodes. Each node creates and maintains a local suspects list. The list is periodically sent to the trusted source which uses the collected information to construct a global list of nodes that must be banned due to their malicious behavior. The source periodically sends the generated global black list to all nodes in the overlay structure via a gossip-based protocol. Building and disseminating the global list has a higher cost than maintaining just the local list at each node. However, it has the advantage that it allows the overlay to quickly converge to a stable equilibrium point and achieve higher throughput as malicious nodes are more quickly removed from the overlay. This is because each node does not have to experience the malicious nodes' actions before being able to avoid them. Nodes are informed through the global black list and consequently are able to select beneficial parents.

Local Response

Each node takes immediate action based on a local suspects list created by tracking the behavior of neighbor nodes. This is achieved by recording any inconsistent metrics detected when performing outlier analysis on the information from the probed nodes. Every node computes a *suspicion value* for each neighbor based upon how far away the reported metrics were from the spatial and temporal centroids. The computation

of the suspicion value also takes into account each node locally sharing information with the other nodes about its suspects list.

$$q_{ij} = \begin{cases} q_{ij} + \alpha(\frac{|U_j - \overline{U}|}{\sigma_U}) + \beta(\frac{|V_j - \overline{V}|}{\sigma_V}) + N_r & \text{if } j \text{ is outlier} \\ \frac{3q_{ij}}{4} - 1 & \text{otherwise} \end{cases} \tag{5.2}$$

The suspects list is updated at the end of each probe cycle after the system has performed outlier detection. Equation 5.2 presents the computation of a single suspicion value during a probe cycle: q_{ij} is the suspicion value of node i for peer j, U is the list containing the Mahalanobis distances measuring spatial distance for i's peers, V is the list containing the temporal distance for i's peers, $\overline{U}$ and $\overline{V}$ are the averages of each list, σ_U and σ_V are the standard deviations of U and V respectively, and N_r is a counter representing how many other nodes reported j as suspicious. Each measure is weighted independently (α, β, and γ) to allow the response to be tailored to the application or network conditions. Intuitively, our scheme assigns suspicion such that the greater the distance between the observed data for a node and the centroid of the entire data set, the greater the local suspicion value a node is assigned. We also assign more weight to the direct observations (*i.e.*, how far a node's distance is from the centroid) than to indirect observation (*i.e.*, how many nodes reported a node as suspicious). The suspicion value is increased every probe cycle if the probed node is an outlier. Otherwise, as seen in the second part of Equation 5.2, the suspicion values undergoes decay to accommodate transient network conditions and allow nodes to be removed from the suspects list, eventually enabling a node that behaves well to be reconsidered as a parent.

Once a node is placed on the suspects list, it can be displaced as a child node, it will not be chosen as a parent, its gossiped information will not be propagated, and it may eventually be reported to the source as being *malicious*. A node will decide if it considers a neighbor node *malicious* by comparing the suspicion value against a threshold, Δ. If the value is higher than the threshold, then the node is reported as malicious to the other nodes and the source.

Malicious nodes may collude and send false information causing non-malicious nodes to incorrectly suspect their peers. To prevent this, even if node i has a positive suspicion value for node j, i will not report a negative reputation for j to the source unless i has directly experienced suspicious/malicious behavior from j. This approach still allows honest nodes to build up a strong negative reputation through indirect observations, but holds global-response at bay until the malicious node directly treats the honest node badly. Even if a node is marked as suspicious, it will still receive service from the overlay and will remain a member of the topology. In this manner an honest node cannot be isolated from the overlay unless *every* overlay node peer it attempts to use as a parent is malicious and drops all traffic to the honest node.

5.3.2 Global Response

Our global response mechanism creates a global representation of the trust in each node in the overlay by using a reputation system to aggregate the individual suspicion values from the local suspects list at the source. The nodes with a trust value below a specified threshold are added to a global black list disseminated to all nodes. We adapt a well-known distributed reputation system, EigenTrust [105], to the trust model of our application in which the source node is trusted. We selected the EigenTrust algorithm because its trust value aggregation method is robust to malicious nodes and coalitions as discussed in Section 5.2.6. We make several modifications to the EigenTrust algorithm to tailor it to our application.

Using the EigenTrust algorithm presented in Algorithm 5 our solution creates a formulation of the system wide trust ranking for each node. Since the source is the only trusted node in the system, it is the only node to start out with a positive reputation. The source's pre-trusted weight, λ, was empirically set at 0.3 to minimize the convergence time of the EigenTrust algorithm while still providing valuable feedback. Once the trust values drop below a specified threshold, Ψ, the system will consider the identified node as being malicious.

In addition to tailoring the EigenTrust algorithm to our environment, we also take into account that while EigenTrust was designed to use and create positive reputation, our outlier detection produces only negative reputation about a node. We convert the local suspicion values into a positive form suitable for the EigenTrust algorithm.

5.4 Experimental Evaluation

We demonstrate the effects of the identified attacks and our response techniques under real-world conditions by conducting experiments on the PlanetLab [58, 71] Internet testbed. We conducted multiple experiments at different times of the day and different days of the week. Further, experimental nodes were selected randomly for different experiments to validate the statistical significance of results and nodes were chosen to span multiple operational and administrative domains. Each experiment was conducted multiple times and the results were averaged.

The baseline configuration for the following experiments consist of 60 minute long ESM deployments of 100 nodes in which the nodes join after the experiment begins and leave before it ends, with an average participation time of 45 minutes. Each node is probed every seven seconds, each node probes 30 peers, the saturation degree of benign nodes is six, and the source streaming rate is 480Kbps. We use a bit rate of 480 Kbps as it is sufficient to transmit video at two different qualities of audio. All experiments use these parameters unless otherwise noted.

5.4.1 Mitigating Identified Threats

Algorithm 6: Procedure to exclude malicious nodes as possible parents. The code is invoked after line 8 in the ESM parent selection pseudo-code presented in Algorithm 1

Input: Potential Parent Candidates List (PCL)

Output: Updated PCL

```
1  foreach rnode in PCL do
       // Ignore Known Malicious Nodes
       // Resulted from Global Response
2      if (rnode is on BlackList) then
3          remove rnode from PCL;
4      else
5          keep rnode in PCL;
6      end
       // Detect Malicious Behavior
       // Resulted from Local Response
7      if (outlierDetection(rnode) == false) then
8          keep rnode in PCL;
9      else
10         remove rnode from PCL;
11     end
12 end
```

To demonstrate the effectiveness of our response mechanisms at mitigating the effects of malicious nodes and sustaining the average bandwidth of the system, we conducted experiments in which a percentage of the nodes were malicious and recorded the average bandwidth for the duration of the experiment. The system was using both spatial and temporal outlier detection discussed in Section 3.5 to generate the local suspicion values. The additional steps which occur during the the parent selection process to mitigate the effects of malicious nodes are presented in Algorithm 6, which is executed between lines 8 and 9 of the original parent selection algorithm presented in Algorithm 1.

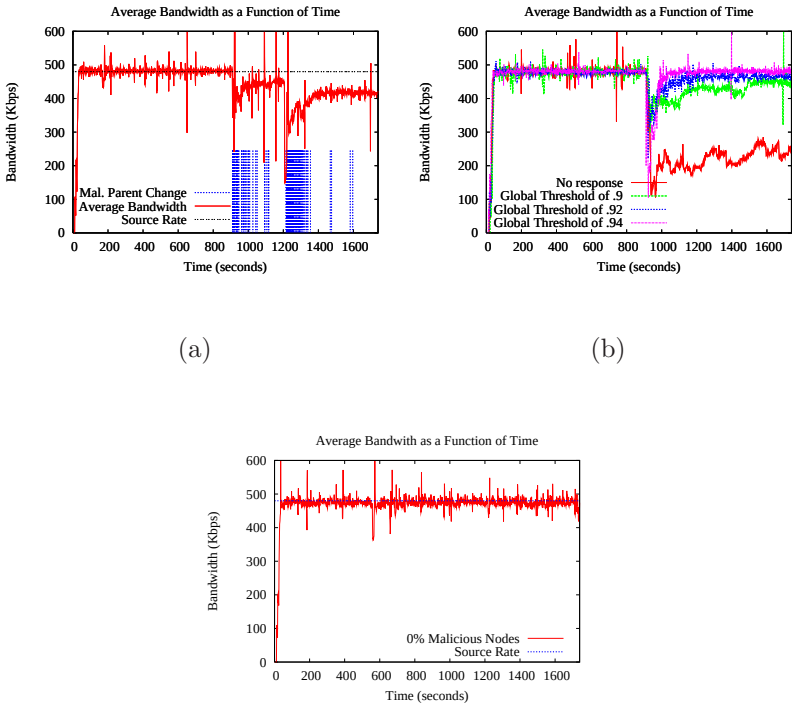


Figure 5.1. (a) The effectiveness of local response only in mitigating attacks on an ESM overlay of 100 nodes on PlanetLab for a duration of 30 minutes. (b) The effectiveness of local and global response in mitigating attacks on an ESM overlay of 100 nodes on PlanetLab for a duration of 30 minutes. (c) The average bandwidth over time for an ESM overlay of 100 nodes on PlanetLab for a duration of 30 using both response mechanisms under non-attack conditions.

Figure 5.1(a) and Figure 5.1(b) present results for the local and global response mechanisms when 30% of the nodes are malicious. Each malicious node joins the network and lies about having the best bandwidth (480Kbps), latency (0ms), and no saturation. Once the malicious nodes have had a chance to optimize their position in the overlay, fifteen minutes after they joined the overlay, they start dropping 90% of the data traffic received through the data dissemination tree. We also present as a reference the average bandwidth under non-attack conditions, with the response

Table 5.1
System settings for the reputation-based response mechanism

Variable	Description	Value
α	Spatial (Horizontal) Outlier Weighting	10
β	Temporal (Vertical) Outlier Weighting	7
γ	Gossip Response Weighting	1
Δ	Local Reporting Threshold	14
Ψ	Global Trust Value Threshold	Variable

mechanisms enabled in Figure 5.1(c). Figure 5.1(a) shows that using only a local response does decrease the effect of the malicious nodes. However, the average bandwidth of the system converges to a value below the one obtained in the absence of malicious nodes and the system takes longer to stabilize. Next, we explored the effect of the global response mechanism on the bandwidth and system stability. Figure 5.1(b) demonstrates that the addition of the global response mechanism in combination with the local response further decreases the effect of the malicious nodes, bringing the average bandwidth of the system close to the value when no malicious nodes exist in the system. When using only the local response, over one third of the identified malicious nodes were actually false positives. By using the global reputation system, we were able to reduce the number of false positives to zero. We also evaluated the use of only the global response mechanism, which we can see in Figure 5.2 resulted in performance similar to the combination of the local and global response mechanisms but showed greater variation in bandwidth.

The system settings for the response mechanism can be found in Table 5.1. Intuitively, the settings were designed to place more weight on being consistent with the current state of the system as measured by the spatial outlier while allowing nodes to have small inconsistencies in reported metrics and not be considered malicious. Each value was determined empirically through experimentation. As shown

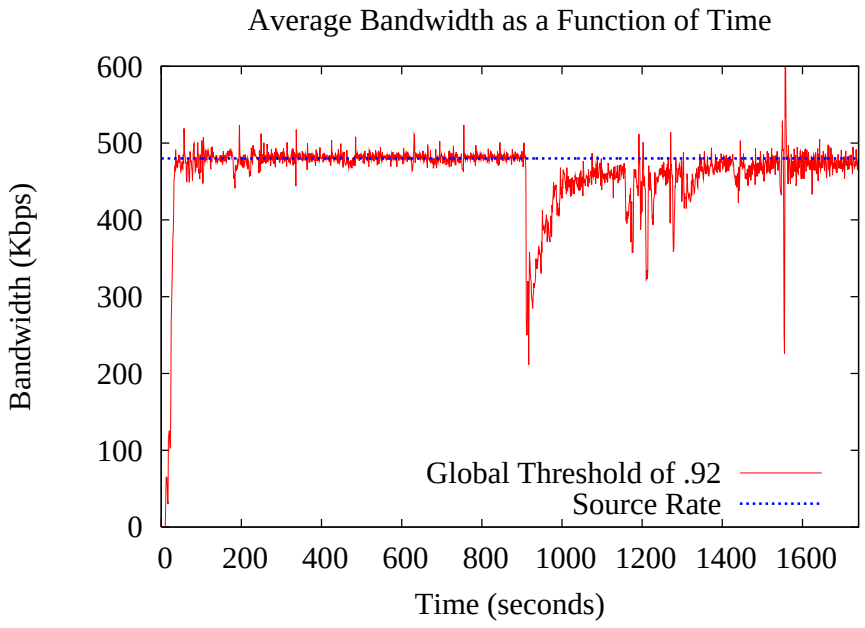


Figure 5.2. The effectiveness of global response only in mitigating attacks on an ESM overlay of 100 nodes on PlanetLab for a duration of 30 minutes.

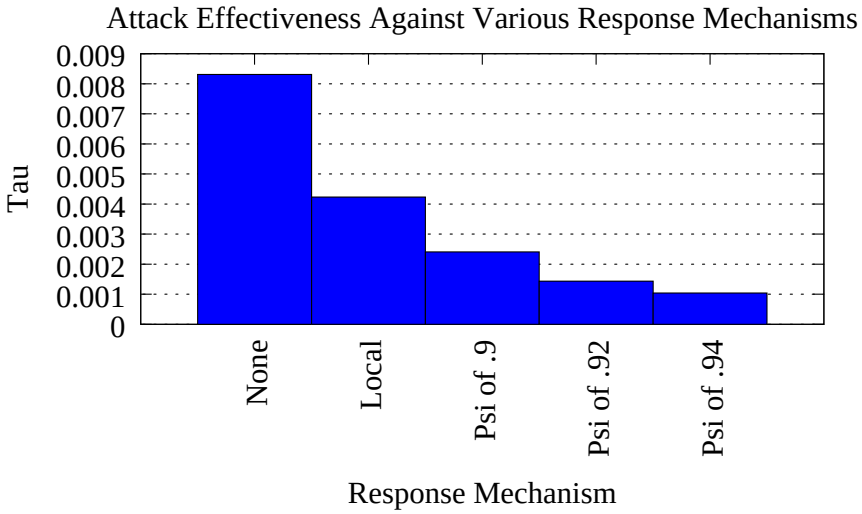


Figure 5.3. Attack effectiveness against different response mechanisms on an ESM overlay of 100 nodes. Tau represents the amount of damage an attack created in the system.

in Figure 5.1(b), even a very conservative response mechanism greatly increases the resiliency of the network to a large percentage of malicious attackers. While the conservative approach was only able to black list approximately one third of the malicious nodes (no non-malicious nodes were black listed), the remaining malicious nodes were pushed towards the fringes of the overlay, allowing the system to sustain near-optimal bandwidth. As the response mechanism is tuned to the application and network conditions, it is able to identify and quarantine higher percentages of malicious nodes. It was identified during testing that the local suspects list could be optimized to only include nodes that had been considered as a parent. This allowed each node to track a much smaller set of suspicious nodes and resulted in a homogenous set of suspicion values at the source. We were able to set cutoff thresholds much tighter without black listing poor performing non-malicious nodes, thereby improving the effectiveness of our response mechanism.

Figure 5.3 presents the effect of the response mechanisms on the relative strength of the attacks, τ, as defined in previously in Equation 3.1. It confirms the intuition

that when the response mechanisms act quickly, the strength of the attack will be diminished as the malicious nodes are more quickly eliminated from the list of potential parents. As expected, the local response mechanism can partially mitigate the effects of the attackers, while the faster convergence of the global mechanism results in a smaller damage on the overall system. The higher the cut-off threshold, the smaller the damage on the system.

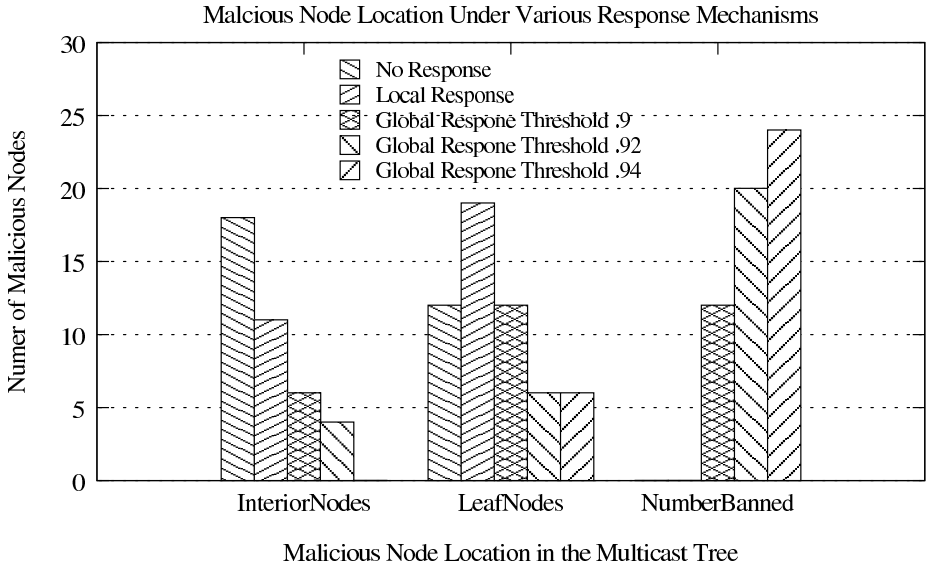


Figure 5.4. Malicious node location for an ESM overlay of 100 nodes with 30% malicious nodes on PlanetLab under different response mechanisms

Figure 5.4 presents the effectiveness of the response mechanisms at moving the malicious nodes towards the fringes of the tree to locations of less influence. Without a response mechanism, the majority of malicious nodes (18 out of 30) occupy interior positions in the multicast tree, with the rest being leaves in the tree. The greater the number of interior positions the malicious nodes control, the greater the effect they will have on the system performance since they can affect all nodes downstream of them. By utilizing the local response only, the system is able to push more of the malicious nodes towards the fringe of the network, with only 11 being interior nodes and 19 now being leaf nodes. Note that the local response does not ban nodes from the

network since they can only be banned when the global mechanism is activated. With the addition of the global response and the ability to remove malicious nodes from the network, the system is more effective in responding to the malicious nodes. For example, with a global response threshold of .92, only 4 malicious nodes are interior nodes, 6 nodes are leaf nodes, and the remaining 20 are removed from the network entirely. By removing malicious nodes from locations of influence, our solution is able to maintain the performance of the system.

5.4.2 Overhead and System Performance

Our response mechanisms introduce minimal link stress since the reputation information for each individual node is combined with the pre-existing membership protocol. Additional messages are required to disseminate the global black list to member nodes through the multicast tree and suspect information from each member node to the source. On average, the source receives an extra $83(\frac{N}{t})$ Bytes of network traffic per second, where N is the size of the overlay and t is the reporting interval. In our experiments, t was set to 20 seconds, resulting in the source receiving approximately 413 Bytes of suspect data per second. Every node in the tree receives $4B$ Bytes of black list information every t seconds, where B is the number of nodes on the black list. The memory utilization per node is up to $8N$ Bytes for the suspects list and up to $4N$ for the black list. The difference is due to each node storing a local reputation for every other node on the suspects list. During calculations of trust at the source, an extra $4N^2$ Bytes for matrix operations are temporarily required. As can be seen, the additional messaging and storage overhead for our response mechanism is minimal.

5.5 Summary

In this chapter, we identified reputation systems as one mechanism for responding to malicious threats in peer-to-peer systems. We provided an overview of reputation

systems, detailing their basic structural components and potential vulnerabilities to malicious behavior. We created a concrete solution to respond to identified threats utilizing the EigenTrust reputation system to respond to malicious behavior identified in ESM. The solution has a two-prong approach, using local observed behavior to generate an immediate local bias against misbehaving nodes and subsequently aggregating these local observations into a global trust value for each node through the use of an augmented version of the EigenTrust reputation system. Our experiments demonstrated that our technique improves the resiliency and overall stability of the system by isolating the malicious nodes.

6 MAINTAINING APPLICATION PERFORMANCE USING ROBUST SYSTEM COMPONENTS

The ability to efficiently manage large amounts of data, now commonly stored across multiple locations, has led to increased interest in *distributed information retrieval*. A common form of a distributed information retrieval system is the distributed hash table (DHT), which provides the nodes in a network with the ability to efficiently store and locate ⟨*key, value*⟩ pairs. Given the general applicability of DHTs, recent research has focused on improving the lookup performance of these systems [54, 120–124]. Many of these improvements hinge on optimizing DHT routing and node selection based on measured network latency (*e.g.*, contacting the peer node with the lowest latency). One popular method for obtaining estimates of network latency while avoiding the cost associated with direct measurement is through the use of decentralized virtual coordinate systems [54, 124–126].

While these optimizations increase the performance of the system, they also open new attack vectors to malicious nodes. The optimization techniques use system components, such as virtual coordinate systems, under the assumption that the information provided by the underlying system is correct. As has been noted in previous work [127], this assumption is not valid for the open environments in which peer-to-peer systems are routinely deployed. If malicious attackers are able to subvert the underlying virtual coordinate system, then they can adversely impact the performance of the overlying application. In this chapter, we make the following contributions:

- We demonstrate the impact malicious attacks against lower-level system components have on higher-level applications through p2psim [86] simulations of the Kademlia DHT [128] using the Vivaldi virtual coordinate system [87] to provide latency estimations for the King topology [88]. We show the performance

of Kademlia is detrimentally effected by various attacks against the underlying virtual coordinate system and continues to degrade as the number of malicious nodes increases.

- We elucidate the hidden tradeoff between optimizing the DHT performance and stability versus the resiliency of the DHT to attack based on real-world implementation and parameters [129]. For example, we find that a Kademlia bucket size of 10 peers provides improved lookup performance and maintains robustness to attack, even when 30% of the network is malicious.

- We demonstrate how the identified attacks detrimentally effect the overlying application, regardless of its construction, through p2psim simulations of the Chord DHT [130] utilizing the Vivaldi virtual coordinate system to provide latency estimations for the King topology. We show that the higher-level application performance severely degrades as the number of attackers increases. We also demonstrate the effectiveness of deploying a robust network awareness component (a robust virtual coordinate system) at mitigating the effects of the attack. We found through empirical studies that our robust implementation of the Vivaldi virtual coordinate system is able to preserve the upper-level application performance, even when 30% of the network is malicious.

The rest of the chapter is organized as follows: We provide an overview of DHTs in Section 6.1. In Section 6.2, we examine the Kademlia DHT and experimentally validate the impairment of the overlying DHT caused by attacks against the Vivaldi virtual coordinate system used to provide latency estimation in Section 6.2. Next, in Section 6.3, we examine the attacks in the context of the Chord DHT and demonstrate how these effects can be mitigated by incorporating a robust virtual coordinate system conferring resiliency to attack. Finally, Section 6.4 concludes this chapter.

6.1 Distributed Hash Tables

Distributed hash tables are a class of distributed application which provide the nodes in a network with the ability to efficiently store and retrieve $\langle key,\ value \rangle$ pairs. Over the past few years, DHTs have generated a great deal of research interest, resulting in the creation of a myriad of systems such as Chord [130], Kademlia [128], Pastry [131], and Tapestry [82]. Along with the creation of multiple DHTs with differing properties, they are employed as building blocks in a variety of systems, ranging from file distribution protocols such as BitTorrent, secure communications platform such as CSpace [132], to content distribution networks such as Coral [133].

6.2 The Kademlia Distributed Hash Table

The Kademlia DHT [128] is a $\langle key,\ value \rangle$ store in which each node (and value) is assigned a 160-bit identifier $i \in \{0,1\}^{160}$ when it enters the system. In order to determine the distance between two identifiers x and y, Kademlia defines an XOR metric where the distance is simply the integer representation of the bitwise exclusive or $(x \oplus y)$. Conceptually, the length of the common prefix of the identifier increases as the distance between identifiers decreases.

As part of the Kademlia protocol, for each bit b of the node id i, a node stores a list, known as a *bucket*, of contact information about peers who have an XOR distance between 2^i and 2^{i+1} from i. Up to k peers are stored in each bucket in order to provide resiliency to node failure. As nodes receive Kademlia routing and lookup messages, they update their buckets with the sender's identifier until each bucket has k peers. If there are already k peers in the bucket, a non-responsive node (or the least-recently seen node if all nodes are responsive) is removed from the bucket and the new identifier is appended, keeping the list ordered by the time last seen.

The most important component of Kademlia is the $\langle key,\ value \rangle$ lookup. While it has been shown that recursive lookups can provide performance gains in terms of latency [54], Kademlia uses iterative lookups as they provide greater fault tolerance

and easier debugging. To find the value associated with a key *key*, a node n selects k nodes that are closest to *key* and sends α of these nodes concurrent request for *key*. If the receivers of the request possesses the value, it is returned to the n. Otherwise, the receivers reply with the k nodes that are closet to *key*. If any replies n receives contain the value, n terminates the lookup process. Otherwise, n will aggregate the peers received from the remote peers, selecting a new set of α closet peers to query. If at any point one of the concurrent requests experiences a timeout, the receiver is removed from the $n's$ bucket and a new request to the next closest node is sent. The procedure to iterate through the identifier space terminates when the value is found or n has queried all of the closest neighbors it received during its search.

6.2.1 Utilizing Latency Estimation to Improve Kademlia Performance

In the original design of Kademlia, each of the buckets contains up to k values order by the time a node was last seen. While maintaining the list in terms of stay time is beneficial in terms of resiliency to churn, the Kademlia protocol can also be augmented to utilize latency information to improve the lookup performance while maintaining resiliency and preserving the DHT invariants. This flexibility allows the system to use latency estimation provided by network services such as the Vivaldi virtual coordinate systems as depicted in Figure 6.1 to optimize peer selection and retention, reducing the average lookup time by fifty percent [123].

This latency information provided by the virtual coordinate system is utilized in two manners. First, as each node fills the buckets associated with each bit b of the identifier space with peers, it also tracks the network latency to each peer. In order to estimate the latency between any two nodes in the network, each node maintains a coordinate in a decentralized virtual coordinate system. These coordinates are appended to each message a node disseminates and are used to estimate the latency between nodes. If an identifier is being added to a bucket with $\geq k$ responsive peers, the peer with the highest (estimated) latency is evicted.

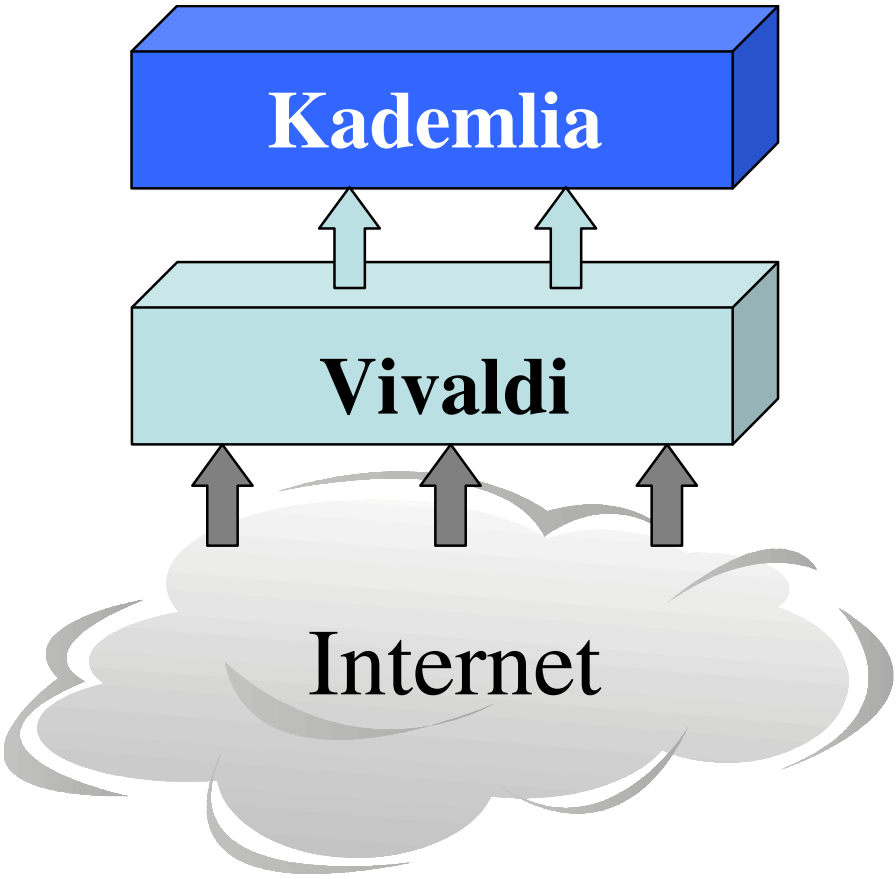


Figure 6.1. An depiction of Kademlia DHT using the latency estimation provided by a virtual coordinate system to optimize its performance. Low-rate network latency measurements are provided by the virtual coordinate system component, which provides virtual coordinates to the higher-level Kademlia DHT to be used to estimate the latency between arbitrary nodes.

In addition to augmenting the maintenance of the buckets, the $\langle key, value \rangle$ lookup is updated to utilize the latency information. Normally, during each lookup, Kademlia selects the k nodes that are closest in terms of the XOR distance to the key key and subsequently randomly selects α of these nodes to contact. This selection of α random nodes is performed multiple times as the lookup initiator iterates through the identifier space in the search for the desired value. Instead of randomly selecting

α out of k peers, the lookup protocol is updated to contact the α nodes closest in terms of estimated latency. This allows the protocol to maintain a worst case lookup performance of $\log(n)$ hops while decreasing the average time required for lookups.

6.2.2 Draining the Performance of Kademlia

If new system components such latency estimations provided by virtual coordinate systems are not robust to malicious compromise, the operation of the overlying DHT can be significantly impaired. We demonstrate the impact various attacks against virtual coordinate systems have on the overlying Kademlia DHT by varying the percentage of malicious nodes and attack type.

Experimental Methodology

We use the King data set [88] in conjunction with the p2psim simulator [86]. We utilize the King data set since it is representative of larger scale peer-to-peer systems for which most DHTs are designed and the data set was used to validate several virtual coordinate system and DHT designs. We ran each simulation for 200 time units, where each time unit is 500 seconds in length. Every simulation was run ten times with the lookup latency averaged over all of the simulation. We enable the system to quickly form and stabilize by having the nodes join the DHT in a flash-crowd scenario in which all nodes join simultaneously, allowing us to focus on the lookup performance of the DHT. While previous research [129, 134] has shown that $\alpha = 3$ concurrent lookups and buckets containing $k = 20$ peers provide good tradeoffs between performance, consistency, and overhead, we analyze a range of lookup options and bucket sizes to determine their impact on the susceptibility of the system to attack. We let the number of concurrent lookups range from $1 \leq \alpha \leq 8$ and found $\alpha \in \{2, 4\}$ to be representative of the system performance in our simulations and present the corresponding results below. All other Kademlia parameters were

initialized to the optimal values discussed by Maymounkov *et al.* [128] or established empirically by the authors of p2psim [86].

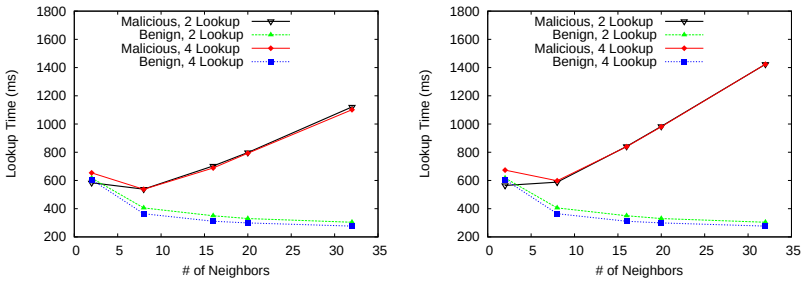
In order to quantitatively compare the effect of attacks on the overlying DHT, we evaluate the system metric:

- ***Lookup Latency*** is defined as the time required to find the address of the node holding a key k in the DHT and return the location to the requestor. Note, the time required to actually retrieve the data from the remote node is not considered.

Coordinate Deflation Attacks

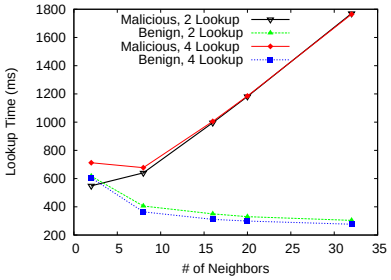
The first set of attacks we examine are those based on coordinate deflation, as described in Section 4.2. In such attacks, the malicious nodes induce the victim to move towards the origin of the coordinate space and then remain immobile by reporting positions near the origin with low error. In this manner, malicious node are able to trick the benign node into assuming their positions are correct and have low error, causing the correct nodes to avoid updating their coordinates and resisting future coordinate adjustment attempts.

As we can see from Figure 6.2, the deflation attack on the underlying virtual coordinate system greatly increases the lookup latency for Kademlia. If we examine the case when $k = 20$ nodes (*e.g.*, the most commonly suggested value), we can see that the lookup performance is approximately 2.5 time higher with 10% malicious nodes in the network, 3 times higher for 20% malicious, and 3.5 times higher with 30%. We also note that the effect of the attack is observable even for smaller values of k. This is due to the fact that many of the coordinates converge to points near the origin, with the malicious nodes being some of the exception. Differentiation between nodes based on latency becomes increasingly random, with nodes selecting high latency peers that would normally reside in distant parts of the coordinate space.



(a) 10% Malicious

(b) 20% Malicious

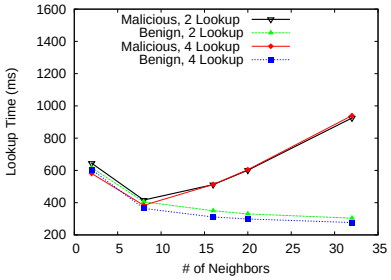


(c) 30% Malicious

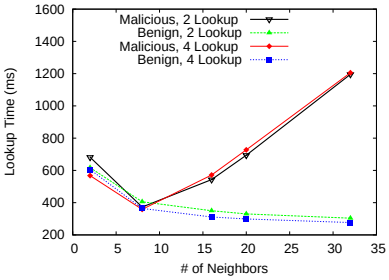
Figure 6.2. Kademlia lookup latency under different percentages of attackers performing a deflation attack on the underlying virtual coordinate system.

Coordinate Inflation Attacks

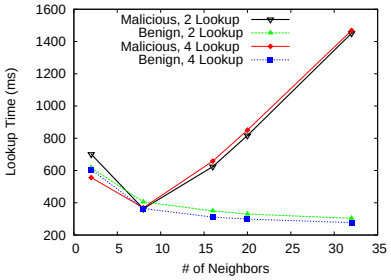
The next set of attacks we analyze are those based on coordinate inflation described in Section 4.2. In order to influence the coordinate construction of the nodes, the malicious nodes utilize two tactics. First, the malicious nodes attract victim nodes towards random positions away from the victims' correct position by reporting coordinates in remote parts of the coordinate space and a low error. Additionally, if the attacker node is located near the origin of the coordinate space, the attacker will induce delay by delaying query responses, pushing the victim node further out in the coordinate space.



(a) 10% Malicious



(b) 20% Malicious



(c) 30% Malicious

Figure 6.3. Kademlia lookup latency under different percentages of attackers performing an inflation attack on the underlying virtual coordinate system.

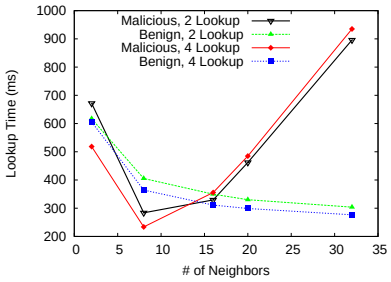
As can be seen in Figure 6.3, the combination of these tactics leads to a reduction in the accuracy of the virtual coordinate system and can severely degrade the performance of the overlying Kademlia DHT. Initially, for smaller values of k, the performance of the system is minimally affected, maintaining performance similar to that of the system under no attack. However, as the value of k increases, the lookup time begins to degrade. This is due to the fact that as the number of potential nodes at each step in the routing processes increases, a benign node has more possible chances to inaccurately select nodes with faulty coordinates by selecting malicious nodes or nodes who have been induced to move away from their correct position. When the values of k are smaller, the routing process utilizes fewer optimization decision based

on latency, minimizing the impact of the attack on the underlying virtual coordinate system. Additionally, we note that as the percentage of attackers increases, the degradation of the lookup latency time for larger number of neighbors also increases. For example, with $k = 32$, the lookup time increases by approximately 200ms for each additional 10% malicious nodes in the network over the initial 10%.

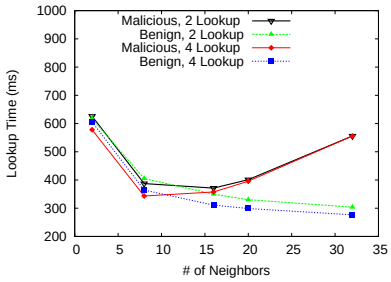
Coordinate Oscillation Attacks

The final set of attacks from Section 4.2 are the coordinate oscillation attacks, in which the attacker attempts to prevent the coordinate system from converging to stable embedding by creating general disorder in the system. In order to conduct this attack, the attackers reports different, random coordinates associated with a random low estimation error for each new query. Additionally, the attackers will delay query responses by $t_d$ms, where $t_d \in [100, 1000]$. The combination of these tactics causes benign nodes to continually update their coordinates, often moving large distances in a random direction in the coordinate space, maintaining a high estimated error. The fact that nodes maintain high estimated errors actually aggravates the attack as the high error, benign updates are given less importance when updating the coordinates, giving malicious updates more weight.

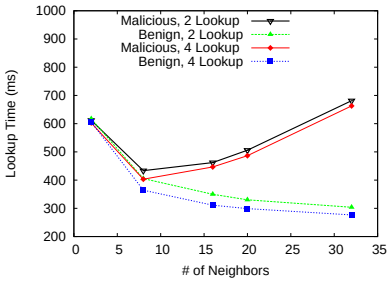
While it is not as severe as the previous attacks, we can see from Figure 6.4 that the oscillation attack still impacts the performance of the optimized Kademlia DHT. The reduction in severity is due to the random nature of the coordinate updates caused by attack. Over the lifetime of the system, the benign nodes will receive random updates from malicious nodes that actually move them towards their appropriate locations in the coordinate space. As the number of peers (k) in each bucket increases, so does the degradation in the lookup time. For instance, when each bucket has $k = 20$ peers, regardless of the percentage of malicious nodes, the lookup time is approximately double that of a system under no attack. Similar to the inflation attacks, the smaller values are more resilient to the attack as there are less choices for each optimization



(a) 10% Malicious



(b) 20% Malicious



(c) 30% Malicious

Figure 6.4. Kademlia lookup latency under different percentages of attackers performing an oscillation attack on the underlying virtual coordinate system.

decision, with a node likely to choose the same path as when the system simply used the XOR metric. It is also interesting to note that unlike the other attacks, the most damaging oscillation attacks occur when a smaller percentage (10%) of the nodes are malicious. This is also due in part to the random nature of the oscillation attack. As more malicious nodes are added to the network, the greater the number of changes the benign nodes will make based on the malicious updates, with a percentage of these updates send the node in the correct direction and effectively canceling out other malicious updates.

6.2.3 Kademlia Discussion

As shown in the previous subsection (Section 6.2.2, each of the attacks against the underlying Vivaldi virtual coordinate system greatly affects the performance of the overlying Kademlia DHT. While previous work [122, 129] has suggested a larger bucket size improves the performance and stability of Kademlia, there is a hidden tradeoff between the bucket size and the resiliency of the system to attack. We can see from Figure 6.2, Figure 6.3, and Figure 6.4 that smaller bucket sizes ($k \leq 10$), are actually *more* resilient to attack. As Kademlia is an important component of many peer-to-peer systems including BitTorrent [1], system architects must take this tradeoff into consideration. For deployed systems, a bucket size of 10 provides a good tradeoff between performance enhancement and resiliency to attack.

We also note that while attacking the virtual coordinate system has a significant impact on the lookup latency of Kademlia, other system performance metrics such as lookup success rate are unaffected by these attacks. At the level of the overlying system, the attacks against the subcomponent will produce very little visible effects, making them difficult to detect. This insight motivates the need to deploy robust virtual coordinate systems to protect the performance of both the virtual coordinate system and the overlying DHT.

6.3 The Chord Distributed Hash Table

In the Chord DHT, every node is assigned a n-bit identifier based on the hash of its IP address and every $\langle$*key, value*$\rangle$ pair is assigned a n-bit identifier based on the hash of the key. When a node joins the DHT, it is placed into a logical ring based upon its identifier, connected to the node with the pervious identifier and the next highest identifier (the predecessor and successor, respectively). For example, in Figure 6.5, the predecessor of node 211 is 205 and the successor of node 211 is 230. In addition, a node with an identifier a maintains a table of neighbors known as a finger table, where the i^{th} finger points to the node closest to the identifier $a + 2^i$. These fingers can logically be thought of as "shortcuts" across the ring, allowing for shorter, more efficient lookup paths. As these values are assigned at join time and may change due to churn, a node will periodically run a maintenance protocol to ensure its predecessor, successor, and fingers are accurate and reachable.

When a $\langle$*key, value*$\rangle$ pair is added to the system, it is assigned to the first node with an identifier greater than or equal to the hash of the key k. When a client attempts to locate the key k in the DHT, it can query any node in the DHT to start the lookup process. In Figure 6.5, node 100 has been queried for a key $k = 228$. The query will be iteratively forwarded through the finger and successor links around the ring until the query reaches the node 230 which holds the key k. For further details on the protocol, we refer the reader to [130].

6.3.1 Utilizing Latency Estimation to Improve Chord Performance

In the original design of Chord, the selection of the node with which to fill the i^{th} position of the finger table of node a is restricted to the node closest to the identifier $a+2^i$. However, it has been realized that this "requirement" is not fundamental to the correct operation of the protocol and any node which falls in the range $[a+2^i, a+2^{i+1}]$ will satisfy the system requirements [135]. As seen in Figure 6.5, the lookup originator has multiple choices for each finger node. This flexibility allows the system to utilize

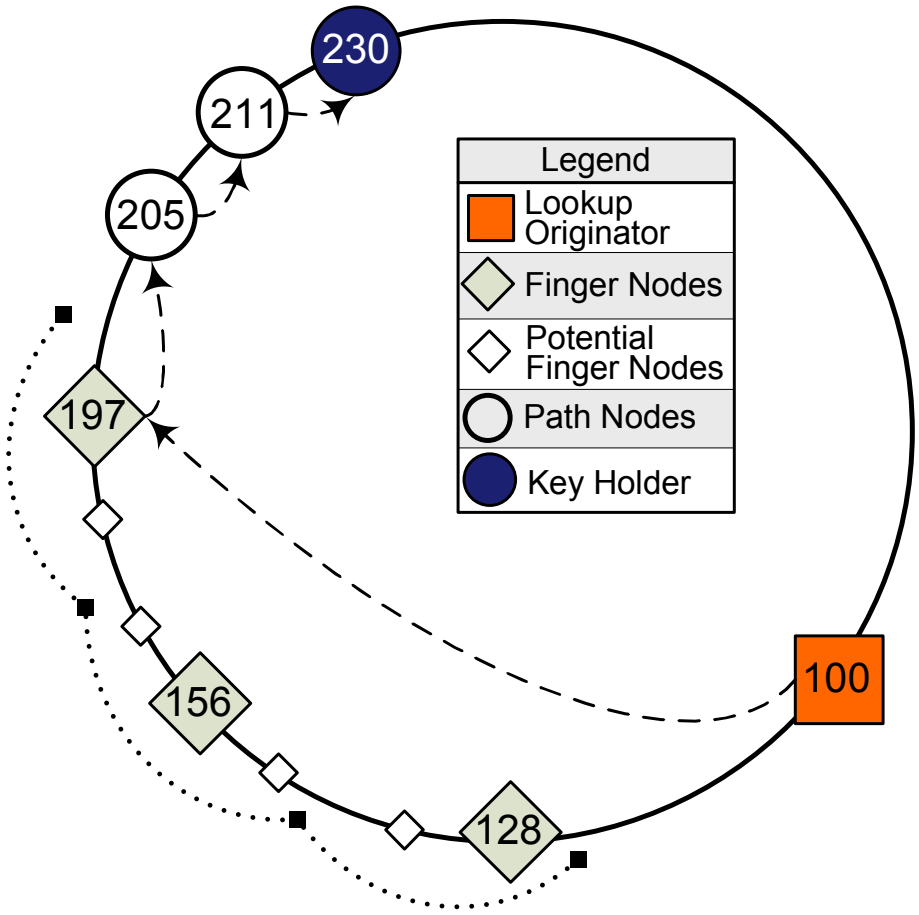


Figure 6.5. An example lookup for the key 228 in the Chord DHT using an 8-bit ($2^8 = 256$) identifier space. The square represents the origin of the lookup while the large, filled diamond nodes are entries in its finger table. The smaller, unfilled diamonds represent possible finger choices in the identifier range denoted by the dotted arcs, which are both described in further detail in Section 6.3.1. The unfilled circles represent nodes on the lookup path and the final filled circle represents the node that holds the desired key k. The lookup path is denoted by the arrows.

the latency estimation provided by virtual coordinate systems such as Vivaldi to optimize the finger selections, reducing the average lookup time by nearly one-half [54].

When a node updates the i^{th} finger, it will check up to x nodes, where x is some small constant defined by the system, in the desired range of $[a + 2^i,\ a + 2^{i+1}]$ to determine which one has the lowest latency and will thus result in the best performance for lookups. It has been shown in previous work that $x = 16$ provides near optimal results [54].

6.3.2 Experimental Evaluation

While the inclusion of the Vivaldi virtual coordinate system as a component of Chord can significantly improve the performance of the system, it also opens it up to a new avenue of attack. If the latency estimations provided by the virtual coordinate system are not robust to malicious compromise, the operation of the DHT can be significantly impaired. In essence, the system is only as strong as its "weakest link". We demonstrate the impact attacks against the virtual coordinate system have on an overlying DHT and how these effects can be mitigated by adding robustness to the underlying virtual coordinate system.

Evaluation Methodology

In order to quantitatively compare the effect of attacks on the overlying DHT, we evaluate two system metrics:

- ***Lookup Latency*** is defined as the time required to find the address of the node holding a key k in the DHT and return the location to the requestor.

- ***Normalized Lookup Delay*** is defined as

$$Lat_{norm} = \frac{Lat_{attack}}{Lat_{no\_attack}}$$

  where Lat_{attack} is the difference between the optimal and actual lookup latency measured in the presence of malicious nodes and Lat_{no_attack} is the difference between the optimal and actual lookup latency measured without malicious nodes.

The optimal lookups occurs when nodes fill their finger table with the lowest latency nodes from the specified range, as discussed in Section 6.3.1. This metric captures the impact of malicious activity on the performance of the DHT. A normalized lookup delay greater than one indicates a degradation in performance (larger lookup times) while a value less than one indicates performance closer to optimal.

We use the King data set [88] in conjunction with the p2psim simulator [86]. We present the results for the King data set as they are representative of the results experienced with the other data sets. We ran each simulation for 200 time units, where each time unit is 500 seconds in length. Every simulation was run ten times with the lookup latency averaged over all of the simulation. We enable the system to quickly form and stabilize by having the nodes join the DHT in a flash-crowd scenario in which all nodes join simultaneously, allowing us to focus on the lookup performance of the DHT. We used a finger table with the optimal size of $x = 16$, as shown by Dabek *et al.* [54]. All other Chord parameters and were initialized to the optimal values discussed by Gummadi *et al.* [135] and Dabek *et al.* citeDabek2004.

Utilizing a Non-Robust Virtual Coordinate System for Latency Estimation

We investigate the effect of malicious nodes attacking the underlying Vivaldi virtual coordinate system has on the lookup latency of Chord by varying the percentage of malicious nodes using the attacks described in Section 4.2.

Figure 6.6 presents the average lookup latencies for the Chord DHT over Vivaldi being attacked by several percentages of malicious nodes. As the number of malicious nodes increase, the average lookup performance of the system degrades. Under non-attack conditions, the DHT has an average lookup latency of 418ms. When 30% of the nodes in the network are attacking the virtual coordinate systems, the lookup latency increase to 600ms. In this case, an average lookup for the DHT utilizing a non-robust virtual coordinate system took nearly 50% longer when attackers were

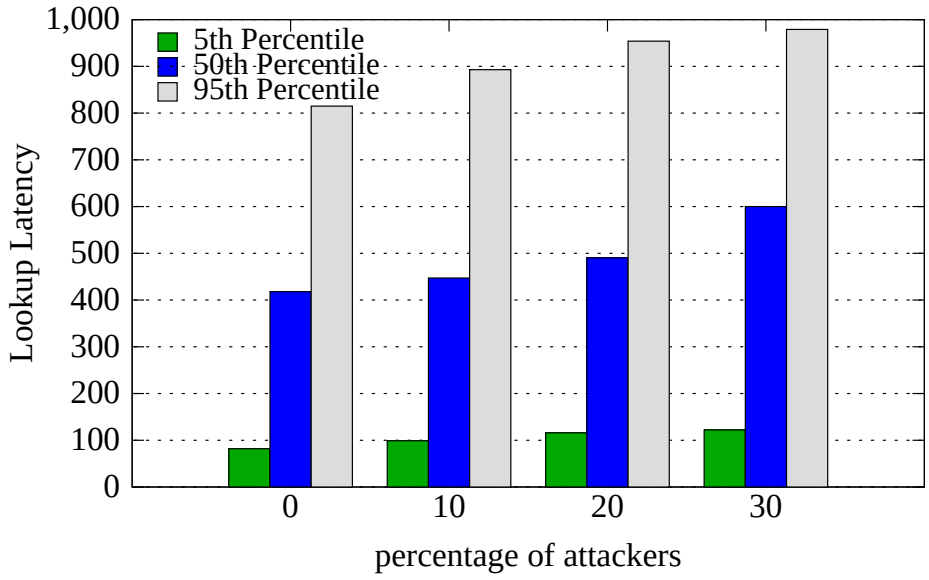


Figure 6.6. Chord lookup latency under different percentages of attackers targeting the Vivaldi virtual coordinate system over the King topology

present. Lookups in the 5^{th} percentile are less affected as these lookups have shorter lookup path lengths (< 3 hops) while lookups in the average and 95^{th} percentile cases have longer paths (≥ 5 hops), increasing the chances the path will traverse a higher latency hop.

Adding Robustness to the Virtual Coordinate System to Maintain DHT Performance

As noted previously in Section 6.2.3, a robust virtual coordinate system is needed to protect the performance of both the virtual coordinate system and the overlying DHT. We further motivate the need to protect the virtual coordinate system subsystem and not just the higher level DHT. We also demonstrate the effectiveness of our defense mechanisms employed at the level of the Vivaldi virtual coordinate system at mitigating the effects of malicious nodes as seen by the higher-level application, the Chord DHT.

If defense mechanism are only placed at the high-level application, both the detection and efficient response to attacks targeting specific subsystems become much more difficult. First, while attacking the virtual coordinate system has a significant impact on the lookup latency of Chord, other system performance metrics such as lookup success rate and hops per lookup are unaffected by these attacks. Under both benign and attack conditions, a random key lookup succeeded on average 97% of the time with an average path length of 5 hops, with almost no variation seen between different scenarios. This suggests that attacks against the underlying components produce little "visible" effect at the upper-level of the system, making them difficult to identify. Secondly, even if an attack is detected, the upper-level system is unable to take preventative actions against the malicious nodes since the system implicitly trusts the underlying components and is unable to determine which nodes are incorrectly using the virtual coordinate system. These reasons necessitate that the virtual coordinate system itself be robust to attack.

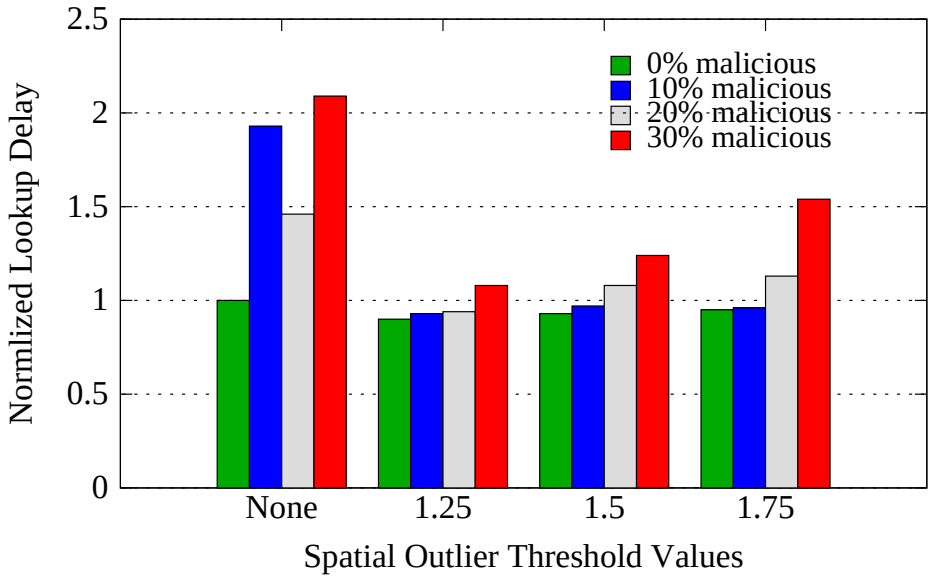


Figure 6.7. Normalized lookup delay for Chord under different percentages of attackers

Figure 6.3.2 presents the normalized lookup delay for different percentages of malicious nodes. As the underlying system is attacked, the ability of the overlying DHT to choose optimal finger nodes degrades, causing the normalized lookup delay to greatly increase. For example, with just 10% malicious nodes, the normalized lookup delay is 1.93 times greater than that of the system not under attack. A detailed explanation of the drop in attack strength under 20% malicious attackers is provided later in this section. Any user or application storing and fetching data from the DHT will notice a marked increase in the average response time no matter the percentage malicious, thus degrading the performance of the system and user experience. However, with the addition of robustness through the use of outlier detection to the virtual coordinate system, the effect of the attack on the DHT has been greatly decreased. For example, utilizing a spatial outlier threshold of 1.5, the normalized lookup delay for a system experiencing attack by 10% of its nodes is .93, or 7% better than prior to our solution. his allows the DHT to more accurately assess and choose the optimal finger nodes.

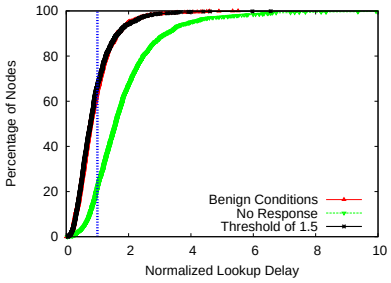
In order to determine if the decline in performance of the DHT under attack is due to a select group of high error nodes or a general system decline, we look at the effect of the attack on individual nodes in the network. We can see from the CDFs presented in Figure 6.8, analogous to the increase in the average lookup latency error, as the number of attackers increases, the number of nodes able to maintain low lookup latency error decreases. For example, under non-attack conditions, 67% of the nodes have a lookup latency error of less than or equal to one (which indicates performance equal to or better than average) while over 95% have a lookup latency error less than 2. However, as seen in Figure 6.8(a), with a network containing just 10% malicious nodes, only 22% of the nodes are able to maintain a lookup latency error of less than one and only 66% have a lookup latency error less than 2. Using a spatial outlier detection threshold of 1.5 and a temporal threshold of 4.0, the system is able to return to functioning at non-attack scenario levels, while 66% of the nodes have a lookup latency error of less than one and over 94% have a lookup latency error less than 2.

For both Figure 6.3.2 and Figure 6.8, it at first appears that the attacks with 20% malicious nodes are detrimental but less effective than other percentages of malicious attackers. Upon further inspection, we determined that the random node movement caused by the attack forced a number of optimal finger node selections to have artificially low latency estimations and thus be chosen as fingers while many higher latency nodes (which would be avoided by the optimal selection) had markedly higher latency estimations. In other attack scenarios, these two cases were reversed, with many higher latency nodes having their latency estimation artificially reduced and thus being chosen as finger nodes. This artifact of virtual coordinate system caused the normalized lookup delay in the 20% malicious nodes scenario to be lower than the other attack percentages.

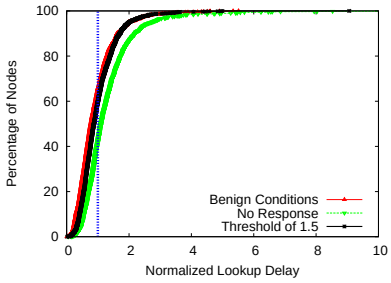
By using a robust virtual coordinate system to estimate network latency, the DHT is able to optimize its performance in *both* non-attack and attack scenarios.

6.4 Summary

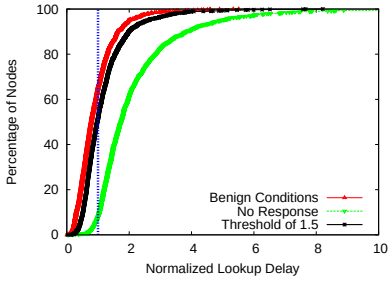
In this chapter, we demonstrated the susceptibility of higher-level applications to malicious attack against their supporting components. Through simulations based on real-life topologies, we showed that multiple attack types performed against the Vivaldi virtual coordinate system severely impact the performance of an overlying DHT, irrespective of the DHT formulation. In some cases, the lookup latency of the system was triple that of the system not under attack. We elucidated the need to understand how the design parameters of protocols like Chord and Kademlia, such as the number of peers per bucket, impacts their resiliency to malicious attack. For example, we found that a bucket size of 10 peers provides a good balance between performance enhancement and resiliency to attack, even when 30% of the network is malicious. Finally, our simulation demonstrated how the use of a robust virtual coordinate system to estimate network latency allows the DHT to optimize its performance in *both* non-attack and attack scenarios.



(a) 10% Malicious



(b) 20% Malicious



(c) 30% Malicious

Figure 6.8. Cumulative distribution functions of the normalized lookup delay for Chord under different percentages of malicious nodes in three different scenarios. The red upright triangle line plots the conditions experienced when *no* malicious nodes are present, the green downward triangle plots the conditions experienced when using a non-robust virtual coordinate system, and the black square line plots the conditions experienced when using a robust virtual coordinate system. The solid blue vertical line represents a normalized lookup delay of 1, at which point nodes are experiencing performance similar to the average optimal delay.

7 RELATED WORK

This chapter provides a review of relevant related work that has influenced the research we have presented. We look at previous attacks exploiting adaptive systems and how techniques from anomaly detection such as using spatial and temporal correlation in conjunction with the Mahalanobis distance could be used to effectively constrain attackers. We review previous research in areas such as BGP, which also utilize data-plane information similar to path graph solution and helped to motivate some of our solutions. We delve into attacks against virtual coordinate systems and demonstrate how previous methods to mitigate them differ from our solution. We review interesting work in error minimization techniques which could be used in conjunction with our solutions to create even more robust systems. We examine previous solutions from the domains of secure localization in sensor networks and secure routing which provide insights into how to secure peer-to-peer systems. Finally, we present an overview of previous work in reputation systems that helped to motivate our response mechanism.

7.1 Attacks Exploiting Adaptivity

Previous work showed the vulnerability of the TCP adaptation mechanisms (*i.e.*, the congestion control mechanism) to malicious attacks [136]. 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. The attack was generalized in [137], as a form of low-rate ROQ attack targeting point-to-point adaptive control loops that drive resource allocation and affect perceived service of a system (bandwidth, jitter, etc). Our work assumes a stronger adversarial model in an overlay network. The nature of the attacks, appli-

cation and deployment environment allows us to use a context sensitive observation space and correlated information associated with the same information that drives the adaptation to detect and limit the effect of malicious behavior.

7.2 Use of Spatial and Temporal Correlations

Recently, the benefits of the Mahalanobis distance for statistical anomaly detection have been demonstrated in the context of network intrusion detection [75,138]. In the work by Lazarevic *et al.* [138], the authors present a comparative study of detection schemes based on data mining techniques for network based intrusion detection. Wang and Stolfo [75] discuss an unsupervised, payload-based network anomaly detector based on the Mahalanobis distance which was used to detect attacks like worms.

Spatial and temporal correlations were previously used in the context of network security. A notable work in this aspect by Jiang and Cybenko [74] uses temporal and spatial correlations to detect attack scenarios using a large amount of information from intrusion detection systems, firewalls, and different software logs. Unlike this more general approach, our work focuses on virtual coordinate systems and overlay networks and does not look for correlations, but exploits the fact that they exist to detect inconsistent metrics and find suspicious nodes.

Correlations have also been used in wireless networks for the detection of attacks [139, 140]. The work by Huang *et al.* [139] uses correlations between different features to identify attacks against wireless ad hoc routing protocols while the work by Tanachaiwiwat and Helmy [140] shows how to augment sensor networks with spatio-temporal correlations to detect misinformation being injected into the sensor streams. In our work, the correlation is incorporated with each system and analysis is performed on real deployments and Internet data sets.

7.3 Robust BGP Using Data-Plane Information

A popular area of research has been securing BGP routing. Several proposed security mechanisms make use of the data-plane information present in BGP to make it more robust to misconfiguration and attack. Listen and Whisper [141] are protocols based on anomaly-detection designed to secure BGP by detecting inconsistences in the reported data-plane information. Hu and Mao [142] examine methods for detecting prefix hijacking attacks in real-time by examining collected routing updates for inconsistencies and comparing these with data-plane fingerprints of suspicious prefixes to reduce false positives. Stealth Probing [143] uses data-plane information combined with encryption and multipath routing to detect and prevent attacks on BGP routing. While each of these solutions for threats to BGP are not directly applicable to the peer-to-peer environment, they helped inspire some of this work and lend credibility to our belief in using lightweight information-driven techniques to add robustness to peer-to-peer systems.

7.4 Defense Mechanisms in Virtual Coordinate Systems

Research has demonstrated the susceptibility of Vivaldi to attacks [83, 84]. To address these vulnerabilities, there have been several proposed methods to maintain virtual coordinate system accuracy [95, 144–146]. The PIC virtual coordinate system [95] uses a security test based on the triangle inequality in which any node that violates the triangle inequality above some margin of error is ignored and designated as malicious. However, it has been shown that RTT measurements often violate this inequality [147–149] and thus solutions based solely on such inequalities may degrade system performance when no attack is occurring.

Recent work by Kaafar *et al.* [144] utilizes a solution which employs a set of trusted nodes as a reference set by which to analyze all node behavior for malicious patterns and behavior. In a similar vein, the reputation-based work by Saucez *et al.* [145] uses a priori trusted nodes to rank and detect malicious nodes. The key difference

between these techniques and our method is we do not necessitate the need for a trusted component in the network. The work by Sherr *et al.* [146] uses a verification set of nodes for a node n, where n's update is considered malicious if a certain percentage of the verification set perceives the error of the update to be greater than a predetermined threshold. The main differences between this technique and our solution for securing virtual coordinate systems is we do not necessitate the need for extra node sets nor network communication and we utilize outlier detection over multiple metrics. One interesting research avenue to further increase the robustness of virtual coordinate systems is through the combination of techniques such as those suggested by Sherr *et al.* [146] with our solution. Under such a system, each node would reach independent decisions about the data it is receiving which are augmented using the group decisions reached by the verification set.

7.5 Coordinate System Error Minimization and Landmark Selection

An important area of research orthogonal to the security of the virtual coordinate system is the minimization of error in the system. The accuracy of such systems is greatly effected by landmark placement for centralized schemes and neighbor selection in decentralized schemes. In the work by Zhang *et al.* [150], it is shown that a hierarchical approach can lead to better performance over non-hierarchical solutions. Works by Lua *et al.* [148], Zhang *et al.* [151], and Narayanan and Shim [152] demonstrate shortcomings of current systems and propose possible new metrics and measurements to more accurately embed the latency in the coordinate system. These areas provide interesting opportunities for further research since our work could possibly leverage these new metrics to place further constraints on the attackers and create a more robust, accurate, and fault-tolerant system.

7.6 Secure Localization in Sensor Networks

There have been many solutions proposed to secure localization in sensor networks. While these provide insight into securing virtual coordinate systems, the majority of the solutions rely on assumptions that are not applicable in the peer-to-peer domain. Defense mechanisms such as SPINE [153], LAD [154], and TCSD [155] as well as those proposed by Lazos *et al.* [156] and Mathews *et al.* [157] rely heavily on pre-defined set of reference nodes to identify potentially malicious activity. ROPE [158], SLA [159], and the work by Mathews *et al.* [157] require the reference nodes to be trusted. As decentralized virtual coordinate systems do not have such infrastructure components, the proposed solutions are not applicable. Additionally, solutions such as ROPE [158] and HiRLoc [160] rely on secure communication based on cryptography to mitigate outside attackers. Our defense mechanisms are designed to mitigate insider attacks which are not prevented by such schemes.

Recently, there has been growing interest in utilizing anomaly detection to mitigate attacks in sensor network localization [155, 161, 162]. DRBTS [161] creates a distributed reputation-based trust protocol to detect malicious reference nodes and TCSD [155] detects temporal and spacial inconsistences in reported data caused by malicious tampering. Based on the broadcast nature of the wireless medium, both solutions require nodes to overhear each other which is not possible in our environment. Li *et al.* [162] utilize statistical methods to secure sensor localization. The authors developed an attack resilient location estimator based on Least Median of Squares (LMS) designed to tolerate malicious nodes as opposed to removing them. Their technique generates random subsets of data from the original data pool for individual estimations and then combines these estimations based on their quality. While Li *et al.* [162] holds similarities to ours in that it filters out outliers in the range estimates to improve sensor location estimates, our technique utilizes multiple, correlated attributes over the observation space and over time to improve the performance of the system.

7.7 Secure Routing

Our research has ties to secure wired and wireless routing. A large number of solutions have been proposed to secure wired routing, with particular focus on securing interdomain routing [142, 163–167]. Many of these solution rely solely on the use of cryptographic constructions [163–165]. These techniques are often expensive in terms of complexity and do not offer protection from malicious insider who have compromised a machine and have access to all information on it, including the cryptographic keys.

Previous work in wireless routing has examined defending against malicious insiders [168–171]. Ariadne [168] and SDT [169] utilize multiple routes through the network to prevent malicious nodes from dropping data. Watchdog [170] relies on properties of the wireless medium to overhear packets sent between nodes to ensure nodes are functioning correctly. ODBSR [171] identifies malicious links through the use of probes and acknowledgments along the faulty path. These works use different environmental assumptions (a broadcast medium) and have much higher communication overhead then our proposed solution.

Other research has examined the use of anomaly detection to identify and avoid suspicious routing behavior [142, 166, 167, 172, 173]. PHAS [166] allows the owners of routing prefixes to collect information about current and past BGP routing updates and check the validity of anomalous route updates based on expected patterns, similar to signature-based anomaly detection. Hu and Mao [142] combine anomaly detection based on control-plane information with host fingerprints to detect route hijacking attempts. Zheng *et al.* [167] utilize path monitors to determine if the current routing path is within some a threshold distance of the expected routing path through the network. Huang and Lee [172] and Patwardhan *et al.* [173] detect malicious actions in wireless networks based on deviances from predefined patterns. Our work differs in the fact it does not require extra infrastructure components and is based on statistical anomaly detection which does not require sequences of events to detect malicious

activity. Additionally, we use a variant of statistical anomaly detection that utilizes multiple, correlated attributes and is not based on differences determined by finite automata.

Another interesting approach for securing interdomain routing is SBone [174], in which autonomous systems improve their security by creating collaborative overlays of a small numbers of systems. One possible avenue for further research is to enable the collaboration of virtual coordinate system nodes similar to that of SBone to increase the efficacy of the system.

7.8 Reputation Systems

A considerable amount of research has focused on the development of trust and reputation systems for peer-to-peer systems [104, 105] and ad hoc networks [175], which help users make beneficial decisions assuming the existence of detection mechanisms for malicious behavior. It has been shown in [106–108] that these systems provide an effective way to mitigate the effects of malicious nodes in a decentralized distributed system. In [119], the authors provide techniques for improving the security of reputation systems which could be used in conjunction with our approach. Our work presents concrete solutions for the detection mechanisms that reputation can be built on.

To the best of our knowledge, our work is one of the few implementations of a reputation system in overlay applications that has been tested in a operational system over the Internet and not just simulated. Credence [176] is one other system that we are aware of which uses a reputation system to deal with the file pollution problem in a file-sharing application. Unlike Credence, we focus on multicast applications and consider malicious insiders. Although our work has the advantage of having a point of trust (*i.e.*, the source), it faces the challenge that it can not rely on human interaction to generate reputation. Instead, it achieves its goals in an autonomic fashion.

8 CONCLUSIONS

Peer-to-peer applications will only continue to grow in importance as the world becomes increasingly connected. Given the open nature of these applications and their susceptibility to attack, creating systems that maintain their performance while being resilient to attack will be key to their continued success. In this thesis, we have developed a framework of robust system components which can be used to construct the next generation of peer-to-peer systems. We identify and characterize three critical components (robust adaptability, robust network awareness, and responsiveness to identified threats), identify their associated security risks, provide techniques to make each component robust to malicious activity, and demonstrate how to effectively integrate them into current systems.

For robust adaptation, we focused on insider attacks against adaptation mechanisms. First, we proposed a lightweight solution which aggregates and correlates network topology and application metrics at each node, allowing the nodes to use the derived information to make better adaptation decisions and to constrain the actions of malicious nodes. Second, we proposed the use of context sensitive anomaly detection to detect spatial and temporal outliers of measured and probed metrics, allowing nodes to avoid malicious data and unnecessary adaptations. Our experiments conducted using the adaptive, unstructured overlay multicast system ESM showed that although ESM employs an advanced set of adaptation mechanisms, it was unable to mitigate the attacks posed by a malicious adversary. The experiments also demonstrated that our techniques improved the adaptation process and the overall stability of the system while limiting the effect of malicious nodes.

To enable efficient network awareness, we turned to virtual coordinate systems. We studied attacks against the accuracy of virtual coordinate systems, classifying three types of attack: coordinate inflation, deflation, and oscillation. We showed that

even a small number of attackers can severely degrade coordinate accuracy due to the epidemic nature of the attacks. We proposed to use spatial-temporal correlation to perform outlier detection on updates received from malicious nodes and eliminate them from the coordinate computation process and experimentally demonstrated the utility of our technique. Finally, we examined the limitations of our defense technique and found that the method starts degrading when more than 30% of the nodes in a reference set form a malicious coalition.

In order to create solutions that are responsiveness to identified threats, we examine the incorporation of a reparation system as a basic system building block. We identified their basic structural components and their possible vulnerabilities to malicious behavior. We provided a concrete solution utilizing the EigenTrust reputation system to respond to malicious behavior identified in ESM. We proposed a two-prong approach which uses local observed behavior to generate an immediate local bias against misbehaving nodes and subsequently allows the overlay to construct global knowledge about the malicious nodes through the use of a global reputation system. Our experiments demonstrated that our technique improves the resiliency and overall stability of the system while limiting the effect of malicious nodes.

Reinforcing the need for creating secure system components, we looked at the susceptibility of higher level applications to malicious attack against their supporting components. Through simulations based on real-life topologies, we showed that multiple attack types performed against the Vivaldi virtual coordinate system severely impact the performance of an overlying DHT, irrespective of the DHT formulation. In some cases the lookup latency of the system was triple that of the system not under attack. We demonstrated through simulation how the use of a robust virtual coordinate system to estimate network latency allows the DHT to optimize its performance in *both* non-attack and attack scenarios.

LIST OF REFERENCES

LIST OF REFERENCES

[1] BitTorrent. `http://www.bittorrent.com/`.

[2] Yoram Kulbak and Danny Bickson. The eMule Protocol Specification. eMule project, `http://sourceforge.net/`.

[3] Gnutella. `http://www.gnutella.com/`.

[4] Direct Connect. `http://adc.sourceforge.net/ADC.html`.

[5] OpenNap. `http://opennap.sourceforge.net/`.

[6] Skype. `http://www.skype.com/`.

[7] Yang Chu, Aditya Ganjam, T.S. Eugene Ng, Sanjay G. Rao, Kunwadee Sripanidkulchai, Jibin Zhan, and Hui Zhang. Early Experience with an Internet Broadcast System Based on Overlay Multicast. In *Proceedings of the USENIX Annual Technical Conference (ATC)*, 2004.

[8] Miguel Castro, Peter Druschel, Anne-Marie Kermarrec, and Antony Rowstron. SCRIBE: A Large-scale and Decentralized Application-level Multicast Infrastructure. *IEEE Journal on Selected Areas in Communications*, 20:1489–1499, 2002.

[9] Miguel Castro, Peter Druschel, Anne-Marie Kermarrec, Animesh Nandi, Antony Rowstron, and Atul Singh. SplitStream: High-bandwidth Multicast in Cooperative Environments. In *Proceedings of the ACM Symposium on Operating Systems Principles (SOSP)*, 2003.

[10] Suman Banerjee, Bobby Bhattacharjee, and Christopher Kommareddy. Scalable Application Layer Multicast. In *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, 2002.

[11] John Jannotti, David K. Gifford, Kirk L. Johnson, M. Frans Kaashoek, and James W. O'Toole, Jr. Overcast: Reliable Multicasting with an Overlay Network. In *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2000.

[12] D. Kostic, A. Rodriguez, J. Albrecht, and A. Vahdat. Bullet: High Bandwidth Data Dissemination Using an Overlay Mesh. In *Proceedings of the ACM Symposium on Operating Systems Principles (SOSP)*, 2003.

[13] Vinay Pai, Kapil Kumar, Karthik Tamilmani, Vinay Sambamurthy, and Alexander E. Mohr. Chainsaw: Eliminating Trees from Overlay Multicast. In *Proceedings of the International Workshop on Peer-to-peer Systems (IPTPS)*, 2005.

[14] Xinyan Zhang, Jiangchuan Liu, Bo Li, and Tak-Shing Peter Yum. CoolStreaming/DONet: A Data-driven Overlay Network for Peer-to-peer Live Media Streaming. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, 2005.

[15] SOPCast. `http://www.sopcast.org/`.

[16] PPLive. `http://www.pplive.com/`.

[17] PPStream. `http://www.ppstream.com/`.

[18] TVAnts. `http://www.tvants-ppstream.com/`.

[19] QQLive. `http://tv.qq.com/`.

[20] Feidian. `http://www.feidian.com/`.

[21] Mysee. `http://www.mysee.com/`.

[22] Pdbox. `http://www.pdbox.co.kr/`.

[23] PPMate. `http://www.ppmate.com/`.

[24] UUSee. `http://www.uusee.com/`.

[25] VGO. `http://vgo.21cn.com/`.

[26] CTV. `http://www.tvoon.de/ctv/`.

[27] StreamerOne. `http://www.streamerone.com/`.

[28] TVUnetworks. `http://www.tvunetworks.com/`.

[29] Joost. `http://www.joost.com/`.

[30] Veoh. `http://www.veoh.com/`.

[31] Zattoo. `http://zattoo.com/`.

[32] Folding@home. `http://folding.stanford.edu/`.

[33] International Data Corporation. `http://www.idc.com/`.

[34] Jerome Saltzer and Michael Schroeder. The protection of information in computer systems. *Proceedings of the IEEE*, 63:1278–1308, 1975.

[35] Yang Chu, Sanjay G. Rao, and Hui Zhang. A Case For End System Multicast. In *Proceedings of the International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, 2000.

[36] Miguel Castro, Manuel Costa, and Antony Rowstron. Should we Build Gnutella on a Structured Overlay? In *Proceedinsg of the ACM Workshop on Hot Topics in Networks (HotNets)*, 2003.

[37] Eng Keong Lua, Jon Crowcroft, Marcelo Pias, Ravi Sharma, and Steven Lim. A Survey and Comparison of Peer-to-peer Overlay Network Schemes. *IEEE Communications Surveys & Tutorials*, 7:72–93, 2005.

[38] Miguel Castro, Manuel Costa, and Antony Rowstron. Debunking Some Myths About Structured and Unstructured Overlays. In *Proceedings of the USENIX Symposium on Networked Systems Design & Implementation (NSDI)*, 2005.

[39] E-Crime Watch Survey. `http://www.cert.org/archive/pdf/ecrimesurvey07.pdf`, 2007.

[40] Jeffrey Shneidman, David C. Parkes, and Laurent Massoulié. Faithfulness in Internet Algorithms. In *Proceedings of the SIGCOMM Workshop on Practice and Theory of Incentives and Game Theory in Networked Systems (PINS)*, 2004.

[41] Jeffrey Shneidman and David C. Parkes. Specification Faithfulness in Networks with Rational Nodes. In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, 2004.

[42] Leslie Lamport, Robert Shostak, and Marshall Pease. The Byzantine Generals Problem. *ACM Transactions on Programming Languages and Systems*, 4:382–401, 1982.

[43] Miguel Castro and Barbara Liskov. Practical Byzantine Fault Tolerance. In *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 1999.

[44] Seth James Nielson, Scott A. Crosby, and Dan S. Wallach. A taxonomy of rational attacks. *Lecture Notes in Computer Science*, 3640:36–46, 2005.

[45] Bram Cohen. Incentives Build Robustness in BitTorrent. In *Proceedings of the ACM SIGCOMM Workshop on Economics of Peer-to-peer Systems (P2PEcon)*, 2003.

[46] Amitanand S. Aiyer, Lorenzo Alvisi, Allen Clement, Mike Dahlin, Jean-Philippe Martin, and Carl Porth. BAR Fault Tolerance for Cooperative Services. In *Proceedings of the ACM Symposium on Operating Systems Principles (SOSP)*, 2005.

[47] Thomas Moscibroda, Stefan Schmid, and Roger Wattenhofer. When Selfish Meets Evil: Byzantine Players in a Virus Inoculation Game. In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, 2006.

[48] Allen Clement, Harry Li, Jeff Napper, Jean-Philippe Martin, Lorenzo Alvisi, and Mike Dahlin. BAR Primer. In *Proceedings of the IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2008.

[49] Miguel Castro, Peter Druschel, Ayalvadi Ganesh, Antony Rowstron, and Dan S. Wallach. Secure Routing for Structured Peer-to-peer Overlay Networks. In *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2002.

[50] Danny Dolev and Andrew C. Yao. On the Security of Public Key Protocols. *IEEE Transactions on Information Theory*, 29:198–208, 1983.

[51] Kenjiro Cho, Kensuke Fukuda, Hiroshi Esaki, and Akira Kato. The Impact and Implications of the Growth in Residential User-to-user Traffic. In *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, 2006.

[52] Long Vu, Indranil Gupta, Jin Liang, and Klara Nahrstedt. Measurement and Modeling of a Large-scale Overlay for Multimedia Streaming. In *Proceedings of the International Conference on Quality of Service in Heterogeneous Wired/Wireless Networks (QShine)*, 2007.

[53] Shahzad Ali, Anket Mathur, and Hui Zhang. Measurement of Commercial Peer-to-peer Live Video Streaming. In *Proceedings of the ACM Workshop on Recent Advances in Peer-to-peer Streaming (WRAIPS)*, 2006.

[54] Frank Dabek, Jinyang Li, Emil Sit, James Robertson, M. Frans Kaashoek, and Robert Morris. Designing a DHT for Low Latency and High Throughput. In *Proceedings of the USENIX Symposium on Networked Systems Design & Implementation (NSDI)*, 2004.

[55] Muhammad Sher and Thomas Magedanz. A vulnerabilities analysis and corresponding middleware security extensions for securing NGN applications. *Computer Networks*, 51:4697–4709, 2007.

[56] Codenomicon. IPTV Security and Reliability Challenge. Technical report, Codenomicon, 2008.

[57] Jim Louderback. Inside the Attack that Crippled Revision3, May 2008. `http://revision3.com/blog/2008/05/29/`.

[58] PlanetLab. `http://www.planet-lab.org/`.

[59] Liang Xie and Sencun Zhu. Message Dropping Attacks in Overlay Networks: Attack Detection and Attacker Identification. *ACM Transactions on Information and System Security*, 11:1–30, 2008.

[60] Stephen Deering. Multicast Routing in Internetworks and Extended LANs. In *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, 1988.

[61] David G. Andersen. Resilient Overlay Networks. Master's thesis, Department of EECS, MIT, 2001. `http://nms.lcs.mit.edu/projects/ron/`.

[62] Daniel Bauer, Sean Rooney, Paolo Scotton, Sonja Buchegger, and Ilias Iliadis. The Performance of Measurement-based Overlay Networks. In *Proceedings of the International Workshop on Quality of Future Internet Services (QofIS)*, 2002.

[63] Yang Chu, Sanjay G. Rao, Srinivasan Seshan, and Hui Zhang. Enabling Conferencing Applications on the Internet Using an Overlay Multicast Architecture. In *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, 2001.

[64] Y. Rekhter and T. Li. A Border Gateway Protocol 4 (BGP 4), March 1995.

[65] Craig Labovitz, Abha Ahuja, Abhijit Bose, and Farnam Jahanian. Delayed Internet Routing Convergence. *IEEE/ACM Transactions on Networking*, 9:293–306, 2001.

[66] Zhuoqing Morley Mao, Ramesh Govindan, George Varghese, and Randy H. Katz. Route Flap Damping Exacerbates Internet Routing Convergence. In *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, 2002.

[67] Mukund Seshadri and Randy H. Katz. Dynamics of Simultaneous Overlay Network Routing. Technical Report UCB//CSD-03-1291, University of California, Berkeley, 2003.

[68] Chunqiang Tang and Christopher Ward. GoCast: Gossip-enhanced Overlay Multicast for Fast and Dependable Group Communication. In *Proceedings of the IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2005.

[69] Ben Y. Zhao, Ling Huang, John D. Kubiatowicz, and Anthony D. Joseph. Exploiting Routing Redundancy Using a Wide-area Overlay. Technical Report UCB//CSD-02-1215, University of California, Berkeley, 2002.

[70] AAron Walters, David Zage, and Cristina Nita-Rotaru. A Framework for Securing Measurement-based Adaptation Mechanisms in Unstructured Multicast Overlay Networks. *IEEE/ACM Transactions on Networking*, 16:1434–1446, 2008.

[71] Brent Chun, David Culler, Timothy Roscoe, Andy Bavier, Larry Peterson, Mike Wawrzoniak, and Mic Bowman. PlanetLab: An Overlay Testbed for Broad-coverage Services. *SIGCOMM Computer Communication Review*, 33:3–12, 2003.

[72] Neil Spring, Larry Peterson, Andy Bavier, and Vivek Pai. Using PlanetLab for Network Research: Myths, Realities, and Best Practices. *ACM SIGOPS Operating Systems Review*, 40:17–24, 2006.

[73] AAron Walters, David Zage, and Cristina Nita-Rotaru. Mitigating Attacks Against Measurement-based Adaptaion Mechanisms in Unstructured Multicast Overlay Networks. In *Proceedings of the IEEE International Conference on Network Protocols (ICNP)*, 2006.

[74] Guofei Jiang and George Cybenko. Temporal and Spatial Distributed Event Correlation for Network Security. In *Proceedings of the American Control Conference (ACC)*, 2004.

[75] Ke Wang and Salvatore J. Stolfo. Anomalous Payload-based Network Intrusion Detection. In *Proceedings of the International Symposium on Recent Advances in Intrusion Detection (RAID)*, 2004.

[76] Chang-Tien Lu, Dechang Chen, and Yufeng Kou. Multivariate Spatial Outlier Detection. *International Journal on Artificial Intelligence Tools*, 13:801–812, 2004.

[77] Randall C. Smith and Peter Cheeseman. On the Representation and Estimation of Spatial Uncertainty. *International Journal of Robotics Research*, 5:56–68, 1986.

[78] Maria Isabel Ribeiro. Gaussian Probability Density Functions: Properties and Error Characterization. Technical Report 1049-001, Instituto Superior Tcnico, Lisboa, Portugal, 2004.

[79] Donald Ervin Knuth. *The Art of Computer Programming, 2nd Ed.* Addison-Wesley Longman Publishing Co., Inc., 1978.

[80] Marco Barreno, Blaine Nelson, Russell Sears, Anthony D. Joseph, and J. D. Tygar. Can Machine Learning be Secure? In *Proceedings of the ACM Symposium on Information, Computer & Communication Security (ASIA CCS)*, 2006.

[81] Eric Chan-Tin, Daniel Feldman, Nicholas Hopper, and Yongdae Kim. The Frog-boiling Attack: Limitations of Anomaly Detection for Secure Network Coordinate Systems. In *Proceedings of the International Conference on Security and Privacy in Communication Networks (SecureComm)*, 2009.

[82] Ben Y. Zhao, Ling Huang, Jeremy Stribling, Sean C. Rhea, Anthony D. Joseph, and John Kubiatowicz. Tapestry: A Resilient Global-scale Overlay for Service Deployment. *IEEE Journal on Selected Areas in Communications*, 22:41–53, 2004.

[83] Mohamed Ali Kaafar, Laurent Mathy, Thierry Turletti, and Walid Dabbous. Virtual Networks Under Attack: Disrupting Internet Coordinate Systems. In *Proceedings of the ACM International Conference on emerging Networking EXperiments and Technologies (CoNext)*, 2006.

[84] Mohamed Ali Kaafar, Laurent Mathy, Thierry Turletti, and Walid Dabbous. Real attacks on virtual networks: Vivaldi out of tune. In *Proceedings of the ACM SIGCOMM Workshop on Large-scale Attack Defense (LSAD)*, 2006.

[85] Jonathan Ledlie, Peter Pietzuch, Michael Mitzenmacher, and Margo Seltzer. Wired Geometric Routing. In *Proceedings of the International Workshop on Peer-to-peer Systems (IPTPS)*, 2007.

[86] p2psim: A Simulator for Peer-to-peer Protocols. `http://pdos.csail.mit.edu/p2psim/`.

[87] Frank Dabek, Russ Cox, Frans Kaashoek, and Robert Morris. Vivaldi: A Decentralized Network Coordinate System. In *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, 2004.

[88] Krishna P. Gummadi, Stefan Saroiu, and Steven D. Gribble. King: Estimating Latency between Arbitrary Internet End Hosts. In *Proceedings of the ACM SIGCOMM Workshop on Internet Measurment (IMW)*, 2002.

[89] Bernard Wong, Aleksandrs Slivkins, and Emin Sirer. Meridian: A Lightweight Network Location Service without Virtual Coordinates. In *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, 2005.

[90] NLANR Active Measurement Project. `http://amp.nlanr.net/`.

[91] T.S. Eugene Ng and Hui Zhang. A Network Positioning System for the Internet. In *Proceedings of the USENIX Annual Technical Conference (ATC)*, 2004.

[92] Ananth Rao, Sylvia Ratnasamy, Christos Papadimitriou, Scott Shenker, and Ion Stoica. Geographic Routing Without Location Information. In *Proceedings of the ACM International Conference on Mobile Computing and Networking (MobiCom)*, 2003.

[93] Liying Tang and Mark Crovella. Virtual Landmarks for the Internet. In *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, 2003.

[94] Eugene Ng and Hui Zhang. Predicting Internet Network Distance with Coordinates-based Approaches. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, 2002.

[95] Manuel Costa, Miguel Castro, Antony Rowstron, and Peter Key. PIC: Practical Internet Coordinates for Distance Estimation. In *Proceedings of the International Conference on Distributed Computing Systems (ICDCS)*, 2004.

[96] Hyuk Lim, Jennifer C. Hou, and Chong-Ho Choi. Constructing Internet Coordinate System Based on Delay Measurement. In *Proceedings of the ACM SIGCOMM Conference on Internet Measurement (IMC)*, 2003.

[97] LiWei Lehman and Steven Lerman. A Decentralized Network Coordinate System for Robust Internet Distance. In *Proceedings of the International Conference on Information Technology: New Generations (ITNG)*, 2006.

[98] Paul Francis, Sugih Jamin, Cheng Jin, Yixin Jin, Danny Raz, Yuval Shavitt, and Lixia Zhang. IDMaps: A Global Internet Host Distance Estimation Service. *IEEE/ACM Transactions on Networking*, 9:525–540, 2001.

[99] Marcelo Pias, Jon Crowcroft, Steve Wilbur, Tim Harris, and Saleem Bhatti. Lighthouses for Scalable Distributed Location. In *Proceedings of the International Workshop on Peer-to-peer Systems (IPTPS)*, 2003.

[100] LiWei Lehman and Steven Lerman. PCoord: Network Position Estimation Using Peer-to-peer Measurements. In *Proceedings of the IEEE International Symposium on Network Computing and Applications (NCA)*, 2004.

[101] Yuval Shavitt and Tomer Tankel. Big-bang Simulation for Embedding Network Distances in Euclidean Space. *IEEE/ACM Transactions on Networking*, 12:993–1006, 2004.

[102] Cristian Lumezanu and Neil Spring. Playing Vivaldi in Hyperbolic Space. In *Proceedings of the ACM SIGCOMM Conference on Internet Measurement (IMC)*, 2006.

[103] David S. Moore and George P. McCabe. *Introduction to the practice of statistics*. WH Freeman and Company, 2003.

[104] Roberto Aringhieri, Ernesto Damiani, Sabine De Capitani Di Vimercati, Stefano Paraboschi, and Pierangelo Samarati. Fuzzy Techniques for Trust and Reputation Management in Anonymous Peer-to-peer Systems. *Journal Of The American Society For Information Science And Technology*, 57:528–537, 2006.

[105] Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. The Eigentrust Algorithm for Reputation Management in P2P Networks. In *Proceedings of the International Conference on World Wide Web (WWW)*, 2003.

[106] Karl Aberer and Zoran Despotovic. Managing Trust in a Peer-to-peer Information System. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*, 2001.

[107] Ernesto Damiani, De Capitani di Vimercati, Stefano Paraboschi, Pierangela Samarati, and Fabio Violante. A Reputation-based Approach for Choosing Reliable Resources in Peer-to-peer Networks. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, 2002.

[108] Li Xiong and Ling Liu. A Reputation-based Trust Model for Peer-to-peer Ecommerce Communities. In *Proceedings of the IEEE Conference on Commerce and Enterprise Computing (CEC)*, 2003.

[109] Paul Resnick, Ko Kuwabara, Richard Zeckhauser, and Eric Friedman. Reputation Systems. *Communications of the ACM*, 43:45–48, 2000.

[110] George Akerlof. The Market for "Lemons": Quality Uncertainty and the Market Mechanism. *The Quarterly Journal of Economics*, 84:488–500, 1970.

[111] Tapan Khopkar, Xin Li, and Paul Resnick. Self-selection, Slipping, Salvaging, Slacking, and Stoning: the Impacts of Negative Feedback at eBay. In *Proceedings of the ACM Conference on Electronic Commerce (EC)*, 2005.

[112] John R. Douceur. The Sybil Attack. In *Proceedings of the International Workshop on Peer-to-peer Systems (IPTPS)*, 2002.

[113] Qiao Lian, Zheng Zhang, Mao Yang, Ben Zhao, Yafei Dai, and Xiaoming Li. An Empirical Study of Collusion Behavior in the Maze P2P File-sharing System. In *Proceedings of the International Conference on Distributed Computing Systems (ICDCS)*, 2007.

[114] Michal Feldman, Kevin Lai, Ion Stoica, and John Chuang. Robust Incentive Techniques for Peer-to-peer Networks. In *Proceedings of the ACM Conference on Electronic Commerce (EC)*, 2004.

[115] Kevin Lai, Michal Feldman, Ion Stoica, and John Chuang. Incentives for Cooperation in Peer-to-peer Networks. In *Proceedings of the ACM SIGCOMM Workshop on Economics of Peer-to-peer Systems (P2PEcon)*, 2003.

[116] Eric J. Friedman and Paul Resnick. The Social Cost of Cheap Pseudonyms. *Economics and Management Strategy*, 10(2):173–199, 2001.

[117] Sergio Marti and Hector Garcia-Molina. Taxonomy of Trust: Categorizing P2P Reputation Systems. *Computer Networks*, 50:472–484, 2006.

[118] Sulin Ba and Paul Pavlou. Evidence of the Effect of Trust Building Technology in Electronic Markets: Price Premiums and Buyer Behavior. *Management Information Systems Research Center Quarterly*, 26:243–268, 2002.

[119] Mudhakar Srivatsa, Li Xiong, and Ling Liu. TrustGuard: Countering Vulnerabilities in Reputation Management for Decentralized Overlay Networks. In *Proceedings of the International Conference on World Wide Web (WWW)*, 2005.

[120] Sylvia Ratnasamy, Mark Handley, Richard Karp, and Scott Shenker. Topologically-aware Overlay Construction and Server Selection. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, 2002.

[121] Sushant Jain, Ratul Mahajan, and David Wetherall. A Study of the Performance Potential of DHT-based Overlays. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems (USITS)*, 2003.

[122] Daniel Stutzbach and Reza Rejaie. Understanding Churn in Peer-to-peer Networks. In *Proceedings of the ACM SIGCOMM Conference on Internet Measurement (IMC)*, 2006.

[123] Sebastian Kaune, Tobias Lauinger, Aleksandra Kovacevic, and Konstantin Pussep. Embracing the Peer Next Door: Proximity in Kademlia. In *Proceedings of the International Conference on Peer-to-peer Computing (P2P)*, 2008.

[124] Sean Rhea, Dennis Geels, Timothy Roscoe, and John Kubiatowicz. Handling Churn in a DHT. In *Proceedings of the USENIX Annual Technical Conference (ATC)*, 2004.

[125] Azureus BitTorrent Client. `http://azureus.sourceforge.net/`.

[126] Haiyong Xie, Y. Richard Yang, Arvind Krishnamurthy, Yanbin Grace Liu, and Abraham Silberschatz. P4P: Provider Portal for Applications. In *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, 2008.

[127] James Cowling, Dan R. K. Ports, Barbara Liskov, Raluca Ada Popa, and Abhijeet Gaikwad. Census: Location-aware Membership Management for Large-scale Distributed Systems. In *Proceedings of the USENIX Annual Technical Conference (ATC)*, 2009.

[128] Petar Maymounkov and David Mazieres. Kademlia: A Peer-to-peer Information System Based on the XOR Metric. In *Proceedings of the International Workshop on Peer-to-peer Systems (IPTPS)*, 2002.

[129] Jarret Falkner, Michael Piatek, John P. John, Arvind Krishnamurthy, and Thomas Anderson. Profiling a Million User DHT. In *Proceedings of the ACM SIGCOMM Conference on Internet Measurement (IMC)*, 2007.

[130] Ion Stoica, Robert Morris, David Liben-Nowell, David Karger, M. Frans Kaashoek, Frank Dabek, and Hari Balakrishnan. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. *IEEE/ACM Transactions on Networking*, 11:17–32, 2003.

[131] Antony Rowstron and Peter Druschel. Pastry: Scalable, Decentralized Object Location and Routing for Large-scale Peer-to-peer Systems. *Lecture Notes In Computer Science*, 2218:329–350, 2001.

[132] CSpace. `http://cspace.in/`.

[133] Michael Freedman, Eric Freudenthal, and David Mazières. Democratizing Content Publication with Coral. In *Proceedings of the USENIX Symposium on Networked Systems Design & Implementation (NSDI)*, 2004.

[134] Daniel Stutzbach and Reza Rejaie. Improving Lookup Performance Over a Widely-deployed DHT. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, 2006.

[135] Krishna P. Gummadi, Richard J. Dunn, Stefan Saroiu, Steven D. Gribble, Henry M. Levy, and John Zahorjan. Measurement, Modeling, and Analysis of a Peer-to-peer File-sharing Workload. In *Proceedings of the ACM Symposium on Operating Systems Principles (SOSP)*, 2003.

[136] Aleksandar Kuzmanovic and Edward W. Knightly. Low-rate TCP-targeted DOS attacks: The Shrew vs. the Mice and Elephants. In *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, 2003.

[137] Mina Guirguis, Azer Bestavros, and Ibrahim Matta. Exploiting the Transients of Adaptation for RoQ Attacks on Internet Resources. In *Proceedings of the IEEE International Conference on Network Protocols (ICNP)*, 2004.

[138] Aleksandar Lazarevic, Levent Ertoz, Vipin Kumar, Aysel Ozgur, and Jaideep Srivastava. A Comparative Study of Anomaly Detection Schemes in Network Intrusion Detection. In *Proceedings of the SIAM International Conference on Data Mining (SDM)*, 2003.

[139] Yian Huang, Wei Fan, Wenke Lee, and Philip S. Yu. Cross-feature Analysis for Detecting Ad-hoc Routing Anomalies. In *Proceedings of the International Conference on Distributed Computing Systems (ICDCS)*, 2003.

[140] Sapon Tanachaiwiwat and Ahmed Helmy. Correlation Analysis for Alleviating Effects of Inserted Data in Wireless Sensor Networks. In *Proceedings of the International ICST Conference on Mobile and Ubiquitous Systems (MobiQuitous)*, 2005.

[141] Lakshminarayanan Subramanian, Volker Roth, Ion Stoica, Scott Shenker, and Randy Katz. Listen and Whisper: Security Mechanisms for BGP. In *Proceedings of the USENIX Symposium on Networked Systems Design & Implementation (NSDI)*, 2004.

[142] Xin Hu and Z. Morley Mao. Accurate Real-time Identification of IP Prefix Hijacking. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, 2007.

[143] Ioannis Avramopoulos and Jennifer Rexford. Stealth Probing: Efficient Data-plane Security for IP Routing. In *Proceedings of the USENIX Annual Technical Conference (ATC)*, 2006.

[144] Mohamed Ali Kaafar, Laurent Mathy, Chadi Barakatand Kave Salamatian, Thierry Turletti, and Walid Dabbous. Securing Internet Coordinate Embedding Systems. In *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, 2007.

[145] Damien Saucez, Benoit Donnet, and Olivier Bonaventure. A Reputation-based Approach for Securing Vivaldi Embedding System. *Lecture Notes In Computer Science*, 4606:78–85, 2007.

[146] Micah Sherr, Boon Thau Loo, and Matt Blaze. Veracity: A Fully Decentralized Service for Securing Network Coordinate Systems. In *Proceedings of the International Workshop on Peer-to-peer Systems (IPTPS)*, 2008.

[147] Han Zheng, Eng Keong Lua, Marcelo Pias, and Timothy G. Griffin. Internet routing policies and round-trip-times. In *Proceedings of the Passive and Active Measurement Workshop (PAM)*, 2005.

[148] Eng Keong Lua, Timothy Griffin, Marcelo Pias, Han Zheng, and Jon Crowcroft. On the Accuracy of Embeddings for Internet Coordinate Systems. In *Proceedings of the ACM SIGCOMM Conference on Internet Measurement (IMC)*, 2005.

[149] Jonathan Ledlie, Paul Gardner, and Margo Seltzer. Network Coordinates in the Wil. In *Proceedings of the USENIX Symposium on Networked Systems Design & Implementation (NSDI)*, 2007.

[150] Rongmei Zhang, Charlie Hu, Xiaojun Lin, and Sonia Fahmy. A Hierarchical Approach to Internet Distance Prediction. In *Proceedings of the International Conference on Distributed Computing Systems (ICDCS)*, 2006.

[151] Rongmei Zhang, Chunqiang Tang, Y. Charlie Hu, Sonia Fahmy, and Xiaojun Lin. Impact of the Inaccuracy of Distance Prediction Algorithms on Internet Applications – An Analytical and Comparative Study. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, 2006.

[152] Sathya Narayanan and Eunsoo Shim. Performance Improvement of a Distributed Internet Coordinates System. In *Proceedings of CCNC*, 2007.

[153] Srdan Čapkun and Jean-Pierre Hubaux. Secure Positioning of Wireless Devices with Application to Sensor Networks. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, 2005.

[154] Wenliang Du, Lei Fang, and Peng Ning. LAD: Localization Anomaly Detection for Wireless Sensor Networks. *Journal of Parallel and Distributed Computing*, 66:874–886, 2006.

[155] Honglong Chen, Wei Lou, Junchao Ma, and Zhi Wang. TSCD: A Novel Secure Localization Approach for Wireless Sensor Networks. In *Proceedings of the International Conference on Sensor Technologies and Applications (SENSORCOMM)*, 2008.

[156] Loukas Lazos and Radha Poovendran. SeRLoc: Robust Localization for Wireless Sensor Networks. *ACM Transactions on Sensor Networks*, 1:73–100, 2005.

[157] Mary Mathews, Min Song, Sachin Shetty, and Rick McKenzie. Detecting Compromised Nodes in Wireless Sensor Networks. In *Proceedings of the ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, 2007.

[158] Loukas Lazos, Radha Poovendran, and Srdjan Čapkun. ROPE: Robust Position Estimation in Wireless Sensor Networks. In *Proceedings of the International Conference on Information Processing in Sensor Networks (IPSN)*, 2005.

[159] Farooq Anjum, Santosh Pandey, and Prathima Agrawal. Secure Localization in Sensor Networks using Transmission Range Variation. In *Proceedings of the IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS)*, 2005.

[160] Loukas Lazos and Radha Poovendran. HiRLoc: High-resolution Robust Localization for Wireless Sensor Networks. *IEEE Journal on Selected Areas in Communications*, 24:233–246, 2006.

[161] Avinash Srinivasan, Joshua Teitelbaum, and Jie Wu. DRBTS: Distributed Reputation-based Beacon Trust System. In *Proceedings of the International Conference on Dependable, Autonomic and Secure Computing (DASC)*, 2006.

[162] Zang Li, Wade Trappe, Yanyong Zhang, and Badri Nath. Robust Statistical Methods for Securing Wireless Localization in Sensor Networks. In *Proceedings of the International Conference on Information Processing in Sensor Networks (IPSN)*, 2005.

[163] Russ White. Securing BGP through secure origin BGP (soBGP). *Business Communications Review*, 33:47–53, 2003.

[164] Yih-Chun Hu, Adrian Perrig, and Marvin Sirbu. SPV: Secure Path Vector Routing For Securing BGP. *SIGCOMM Computer Communication Review*, 34:179–192, 2004.

[165] P.C. van Oorschot, Tao Wan, and Evangelos Kranakis. On Interdomain Routing Security and Pretty Secure BGP (psBGP). *ACM Transactions on Information and System Security*, 10:11, 2007.

[166] Mohit Lad, Dan Massey, Dan Pei, Yiguo Wu, Beichuan Zhang, and Lixia Zhang. PHAS: A Prefix Hijack Alert System. In *Proceedings USENIX Security*, 2006.

[167] Changxi Zheng, Lusheng Ji, Dan Pei, Jia Wang, and Paul Francis. A Lightweight Distributed Scheme for Detecting IP Prefix Hijacks in Real-time. *SIGCOMM Computer Communication Review*, 37:277–288, 2007.

[168] Yih-Chun Hu, Adrian Perrig, and David B. Johnson. Ariadne: A Secure On-demand Routing Protocol for Ad Hoc Networks. *Wireless Networks*, 11:21–38, 2005.

[169] Panagiotis Papadimitratos and Zygmunt J. Haas. Secure Data Transmission in Mobile Ad Hoc Networks. In *Proceedings of the ACM workshop on Wireless security (WiSe)*, 2003.

[170] Sergio Marti, T. J. Giuli, Kevin Lai, and Mary Baker. Mitigating Routing Misbehavior in Mobile Ad Hoc Networks. In *Proceedings of the ACM International Conference on Mobile Computing and Networking (MobiCom)*, 2000.

[171] Baruch Awerbuch, Reza Curtmola andDavid Holmer, Herbert Rubens, and Cristina Nita-Rotaru. On the Survivability of Routing Protocols in Ad Hoc Wireless Networks. In *Proceedings of the International Conference on Security and Privacy in Communication Networks (SecureComm)*, 2005.

[172] Yian Huang and Wenke Lee. Attack analysis and detection for ad hoc routing protocols. *Lecture Notes in Computer Science*, 3224:125–145, 2004.

[173] Anand Patwardhan, Jim Parker, Anupam Joshi, Michaela Iorga, and Tom Karygiannis. Secure routing and intrusion detection in ad hoc networks. In *Proceedings of the IEEE International Conference on Pervasive Computing and Communications (PerCom)*, 2005.

[174] I. Avramopoulos, M. Suchara, and J. Rexford. How small groups can secure interdomain routing. Technical Report 808-07, Princeton CS Department, 2007.

[175] Sonja Buchegger and Jean-Yves Le Boudec. Robust Reputation System for P2P and Mobile Ad-hoc Networks. In *Proceedings of the ACM SIGCOMM Workshop on Economics of Peer-to-peer Systems (P2PEcon)*, 2004.

[176] Kevin Walsh and Emin Gn Sirer. Experience with an Object Reputation System for Peer-to-peer Filesharing. In *Proceedings of the USENIX Symposium on Networked Systems Design & Implementation (NSDI)*, 2006.

VITA

VITA

Contact Information

Voice: +1 (765) 228-7469

E-mail: zage@purdue.edu

WWW: http://www.cs.purdue.edu/homes/zagedj/

Citizenship

United States of America

Research Interests

Secure and fault-tolerant protocols, insider-resilient overlay networks, routing, geographical routing, peer-to-peer systems, adaptive systems, wireless mesh networks, virtual coordinate systems

Education

Purdue University, West Lafayette, IN USA

Ph.D, Computer Science, May 2010

- Advisor: Professor Cristina Nita-Rotaru
- Thesis Topic: A Platform for Creating Efficient, Robust, and Resilient Peer-to-Peer Systems

  The rapid growth of communication networks such as the Internet and ad hoc wireless mesh networks has spurred the development of numerous collaborative peer-to-peer (P2P) systems. It is necessary to provide components and methodologies for creating efficient P2P systems that can tolerate changes in network conditions and are resilient to malicious activity. A characterization of

the necessary components, their associated security risks, techniques to make each component robust to malicious activity, and a demonstration of how to effectively integrate the components into current systems will be provided.

B.S., Honors Computer Science and Mathematics, May 2004

- *Graduated with distinction*

Work Experience

Research Assistant **January 2005 - May 2010**

Center for Education and Research in Information Assurance and Security (CERIAS) and Department of Computer Science, Purdue University, West Lafayette, IN

- Member of the Dependable and Secure Distributed Systems Laboratory. Currently conducting research in distributed systems and security. Published research in scaling Byzantine fault-tolerant protocols, securing position services for wireless routing, securing adaptive mechanisms of distributed protocols, robust virtual coordinate systems, and using trust in distributed networks.

Research Intern **June-September 2008, January- September 2009**

Intel Corporation, Santa Clara, CA

- Research and development of a framework to provide scalability, robustness, and security for distributed detection and inference systems. Proposed and demonstrated the utility of a network-aware, distributed membership protocol to improve the performance of overlay networks by biasing neighbor selection towards beneficial nodes based on multiple system metrics. The work includes the analysis of real enterprise data sets and topologies, including node churn statistics, offering a new distribution to accurately model enterprise churn.

Teaching Assistant **August-December 2004**

Department of Computer Science, Purdue University, West Lafayette, IN

- Assisted and evaluated students in undergraduate security course

Tool Developer **May-August 2004**

Baker Hill Corporation, Indianapolis, IN

- Research and development of next generation banking data conversion tool using the Visual Studio .Net that is currently used by all support staff and customers needing data migration from old databases to SQL2000.

Development Intern **May-August 2003**

Motorola Corporation, Schaumburg, IL

- Research and development of end-user applications to seamlessly fuse data from multiple security sensors (video, motion, vibration) into three applications: an easy to use standalone application, a web interface, and a mobile interface to be used by security personnel to monitor remote radio towers. Remote data transfer and sensor manipulation tools were implemented to support the end-user application

Software Development Engineer in Test **May-August 2002**

Microsoft Corporation, Redmond, WA

- Development of a testing framework and automated tests for a new Visual Studio .Net add-in designed to allow the use of .Net code logic behind office documents. Research and creation of initial test plan and test case outlines used by the development group.

Software Development Engineer in Test **May-August 2001**

Microsoft Corporation, Redmond, WA

- Development of regression tests for Office Developer, development of test cases and automated tests for Visual Studio .Net, and product research into possible desired extensions facilitating ease of use in Visual Studio .Net.

Development Intern **May-August 2000**

Tivoli Systems (subsidiary of IBM), Indianapolis, IN

- Development of an internal error reporting tool using Java Servlets

Publications

Journal Articles:

1. "Robust Decentralized Virtual Coordinate Systems in Adversarial Environment", David Zage and Cristina Nita-Rotaru. *To appear in the ACM Transactions on Information and System Security.*
2. "A Framework for Securing Measurement-Based Adaptation Mechanisms in Unstructured Multicast Overlay Networks", AAron Walters, David Zage, Cristina Nita-Rotaru. *IEEE/ACM Transactions on Networking, Volume 16, Issue 6, December 2008.*
3. "A Survey of Attack and Defense Techniques for Reputation Systems", Kevin Hoffman, David Zage and Cristina Nita-Rotaru. *To appear in ACM Computing Surveys, Volume 41, Issue 4, December 2009. Also Technical Report CSD TR 07-013.*
4. "STEWARD: Scaling Byzantine Fault-Tolerant Replication to Wide Area Networks", Yair Amir, Claudiu Danilov, Danny Dolev, Jonathan Kirsch, John Lane, Cristina Nita-Rotaru, Josh Olsen, David Zage. *To appear in IEEE Transactions on Dependable and Secure Computing.*

Conference Papers:

1. "A Network-Aware Distributed Membership Protocol for Collaborative Defense", David Zage, Carl Livadas, and Eve Schooler. *Proceedings of the 12th IEEE International Conference on Computational Science and Engineering (CSE), 2009.*
2. "Experimental Comparison of Peer-to-Peer Streaming Overlays: An Application Perspective", Jeff Seibert, David Zage, Sonia Fahmy, Cristina Nita-Rotaru. *Proceedings of the 33rd IEEE Conference on Local Computer Networks (LCN), 2008.*
3. "On the Accuracy of Decentralized Network Coordinate Systems in Adversarial Networks", David Zage and Cristina Nita-Rotaru. *Proceedings of the*

14th ACM Conference on Computer and Communications Security (CCS), 2007.

4. "Mitigating Attacks Against Measurement-Based Adaptation Mechanisms in Unstructured Multicast Overlay Networks", AAron Walters, David Zage and Cristina Nita-Rotaru. *Proceedings of the IEEE International Conference on Network Protocols (ICNP), 2006.*
5. "Scaling Byzantine Fault-Tolerant Systems to Wide Area Networks", Yair Amir, Claudiu Danilov, Danny Dolev, Jonathan Kirsch, John Lane, Cristina Nita-Rotaru, Josh Olsen, David Zage. *Proceedings of the International Conference on Dependable Systems and Networks (DSN), 2006.*

Extended Abstracts:

1. "Robust Virtual Coordinate Systems with Byzantine Participants", David Zage *PhD Student Forum - Proceedings of the 37th International Conference on Dependable Systems and Networks (DSN), June 2007.*
2. "Byzantine Resilient Distributed Position Service", Josh Olsen, David Zage and Cristina Nita-Rotaru. *Extended Abstract in the Proceedings of the International Conference on Dependable Systems and Networks (DSN), 2006.*

Software

- ADAPT: A security module than enables attacker-resilient parent selection in ESM, a well-known broadcast overlay system. The mechanism combines spatial-temporal outlier detection with reputation mechanisms to prevent malicious nodes from controlling the overlay topology through the membership and parent selection protocols.
- S-VIVALDI: An extension to the P2P simulator implementation of the Vivaldi virtual coordinate systems. The code provides the capability to simulate Byzantine attackers in the P2P simulator and enables each system node to track the current performance and the performance history of its neighboring node. Com-

bining this information with outlier detection, nodes are able to filter out potentially harmful virtual coordinates, preserving the integrity of the system.

Professional Services

Conference Chair Positions:

- Web and Proceedings Chair, STC 2009
- Web and Proceedings Chair, NPSec 2007

Journal Referee:

- ACM Transactions on Computing (2009)
- ACM Transactions on Information and System Security (2009)
- Elsevier Journal of Systems and Software (2009)
- Elsevier Journal of Computer Networks (2009)

Conference Referee:

- International Symposium on Reliable Distributed Systems (SRDS 2009)
- IEEE Conference on Computer Communications (INFOCOM 2006, 2008)
- ACM Conference on Computer and Communications Security (CCS 2008, 2009)
- IEEE International Conference on Security and Privacy for Emerging Areas in Communication and Networks (SecureComm 2005, 2006)
- Workshop on Secure Network Protocols (NPSec 2006)
- IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2005, 2006)
- IEEE International Conference on Distributed Computing Systems (ICDCS 2007, 2009)
- ACM Workshop on Scalable Trusted Computing (STC 2007)
- IEEE High Assurance Systems Engineering Symposium (HASE 2007)
- ACM Conference on Wireless Network Security (WiSec 2008)
- ACM International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV 2009)

Memberships

- Association for Computing Machinery (ACM)
- Institute of Electrical and Electronics Engineers (IEEE)
- Upsilon Pi Epsilon
- Phi Beta Kappa
- Alpha Lambda Delta

Awards

- Travel grant to participate in the Wireless Summer School held at UIUC (2009)
- Second place poster at the 10^{th} annual CERIAS Information Security Symposium, "Removing the Blinders: Utilizing Data-Plane Information to Mitigate Adversaries in Unstructured Multicast Networks" (2009)
- Travel grant from ACM to participate in the ACM Conference on Computer and Communications Security (2007)
- Third place poster at the 8^{th} annual CERIAS Information Security Symposium, "On the Accuracy of Decentralized Network Coordinate Systems in Adversarial Networks" (2007)
- Travel grant from IEEE and NSF to participate in the IEEE International Conference on Network Protocols (2006)
- Scholarship from Northrop Grumman for high academic achievement (2003)
- Scholarship from Raytheon for high academic achievement (2003)
- Purdue Outstanding Freshman (2001) and Sophomore (2002) in Computer Science
- Lilly Endowment Community Scholarship (2000)
- Corporate Partners Council Scholarship from the Purdue University Computer Science Department (2000)
- Scholarship from Purdue University for academic excellence (2002-2004)