Towards Securing Data Delivery in Peer-to-Peer Streaming

Jeff Seibert, Xin Sun, Cristina Nita-Rotaru, Sanjay Rao
Purdue University
jcseiber@cs.purdue.edu, sun19@ecn.purdue.edu, crisn@cs.purdue.edu, sanjay@ecn.purdue.edu

***Abstract*—The goal of enabling ubiquitous video broadcasting on the Internet has been a long cherished vision in the networking community. Prior efforts aimed at achieving this goal based on the IP Multicast architecture have been unsuccessful. In recent years, peer-to-peer (P2P) streaming has emerged as a promising alternative technology, which has matured to the point that there are several commercial offerings available to users. While these developments are encouraging, P2P streaming systems are susceptible to attacks by malicious participants, and their viability depends on how effectively they can perform under such attacks. In this paper, we explore this issue in the context of mesh-based designs, which have emerged as the dominant architecture for P2P streaming.**

We provide a taxonomy of the implicit commitments made by nodes when peering with others. We show that when these commitments are not enforced explicitly, they can be exploited by malicious nodes to conduct attacks that degrade the data delivery service. We consider an important class of attacks where malicious nodes deliberately become neighbors of a large number of nodes and do not upload data to them. We focus on these attacks given the limited attention paid to them, and the significant impact they can have on overall data delivery. We present mechanisms that can enhance the resilience of mesh-based streaming against such attacks. A key part of the solution is a novel reputation scheme that combines feedback from both the control and data planes of the overlay. We evaluate our design with real-world experiments on the PlanetLab testbed and show that our design is effective. Even when there are 30% attackers, nodes can receive 92% of the data with our schemes compared to 10% of the data without our schemes. Overall these results indicate the feasibility of enabling effective P2P streaming even under the presence of malicious participants.

I. Introduction

The vision of enabling simultaneous video broadcast as a common Internet utility in a manner that any publisher can broadcast content to any set of receivers has been driving the research agenda in the networking community for over two decades. For much of the 1990's, the research and industrial community investigated support for such applications using the IP Multicast architecture [1]. However, serious concerns regarding its scaling, support for higher level functionality, and deployment have dogged IP Multicast. The sparse deployment of IP Multicast, and the high cost of bandwidth required for server-based solutions or Content Delivery Networks (CDNs) are two main factors that have limited broadcast to only a subset of Internet content publishers. While many network service providers have enabled IPTV services that deliver quality video to their own subscribers using packet switching, there remains a need for cost-effective, ubiquitous support for Internet-wide video broadcast.

Over the last decade, there has been significant interest in the use of peer-to-peer (P2P) technologies for Internet video broadcast [2]–[7]. There are two key drivers making the approach attractive. First, such technology does not require support from Internet routers and network infrastructure, and consequently is extremely cost-effective and easy to deploy. Second, in such a technology, a participant that tunes into a broadcast is not only downloading a video stream, but also uploading it to other participants watching the program. Consequently, such an approach has the potential to scale with group size, as greater demand also generates more resources.

The extensive research in the design of P2P streaming systems [2], [3], [8]–[12] has matured to the extent that we are today seeing several efforts aimed at commercializing the technology [4], [5], [13], [14]. High user demand for these systems has been shown by their increasingly large user base [6], [7]. Not surprisingly, recent studies indicate that over 60% of Internet traffic is generated by P2P systems [15], with video accounting for more than one-third of all Internet traffic today [16].

P2P streaming can be divided into two main approaches, tree-based [8], [10] and mesh-based [9], [11] architectures. (see [17] for a survey). Tree-based overlays construct a tree, rooted at the source, which broadcasts the stream. Mesh-based overlays disseminate data in a less structured manner, where nodes exchange data with a subset of the nodes in the network without using any predefined route. Mesh-based approaches have received a lot of attention in recent times because they are more resilient to churn and node failures, and have been shown to perform better than tree-based approaches [18], [19].

While mesh-based approaches have several attractive properties, the performance of these systems in the presence of malicious participants has received little attention. The only known work to date [20] shows the vulnerability of such systems to attacks where malicious nodes upload polluted data to other nodes in the overlay. Work has been done on the problem of peers which download data from other nodes but do not in turn upload data [21], however these works focus on selfish rather than malicious node behavior.

In this paper, we systematically analyze the vulnerabilities of the components of mesh-based streaming overlays, and provide a taxonomy of the implicit commitments made by nodes when peering with others. We show that when these commitments are not enforced explicitly, they can be exploited

978-1-4244-5489-1/10/ $26.00

by malicious nodes to conduct attacks that degrade the data delivery service. To our knowledge, this is the first effort at taxonomizing attacks on mesh-based streaming protocols.

We focus on an important and broad class of attacks where malicious nodes deliberately become neighbors of a very large number of nodes in the system and do not upload data to them. We focus on these attacks given they have received limited attention, the significant disruption they can have on data delivery, and their applicability to many mesh-based systems. For instance, our evaluation with a state-of-the-art mesh-based streaming system shows that when the attacks are conducted with just 10% of nodes in the system being malicious, the average data rate received across all nodes is only 45% of the source rate.

We wish to emphasize that our focus in this paper is on mesh-based approaches for live video streaming, rather than file-download systems like BitTorrent [22]. While some of the attacks we consider may also be relevant to file-download systems, the impact on application performance is far more serious for streaming applications given that they are associated with stringent real-time deadlines. Consequently, the solutions must also be tailored to the unique demands of streaming applications.

We present solutions to enhance the resilience of mesh-based overlays to the attacks we consider. Our solution is centered around a novel reputation scheme that combines feedback from the data plane (based on data received from the nodes) and the control plane (based on who a node has as neighbors). Through detailed security analysis, we show that our scheme is resistant to of attacks commonly associated with reputation schemes such as self-promotion and slandering [23]. In particular, we show that our scheme ensures that a malicious node must contribute a minimum amount of data to acquire a certain reputation, and a benign node that contributes data is assured a certain minimum reputation and cannot be slandered.

While the reputation scheme is a core part of our attack prevention mechanisms, to further augment the system, a more comprehensive approach is required that also addresses potential vulnerabilities in the bootstrap mechanism, and with the source of the broadcast. We present a set of simple mechanisms to achieve this goal. In particular, we present a source protection scheme that disallows malicious nodes to be overly connected to it, and a scheme that prevents malicious nodes from influencing the membership bootstrap service.

We evaluate our design using experiments on the PlanetLab testbed. Our results show that our schemes are extremely effective in ensuring good performance under attacks. With the local-reputation scheme, with 10% of the nodes being malicious, the average data-rate received across nodes increases from 45% of the source-rate to 65%. However, when the scheme is augmented with source and bootstrap mechanisms, nodes receive 95% of the source-rate on average. Our scheme also works well in extreme regimes, in fact even with 30% of the nodes being malicious, more than 85% of the peers receive over 90% of the data. Overall, our results show the feasibility of augmenting mesh-based P2P streaming schemes to be resistant to attacks that target data delivery.

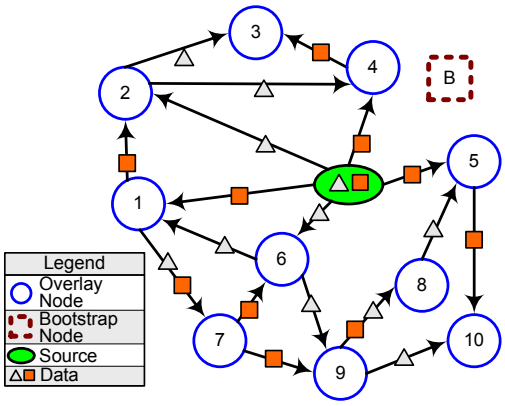


Fig. 1. Example of a unidirectional mesh-based streaming overlay in which the source sends two different data chunks as denoted by the gray triangle and orange square. Each node has an in-neighbor set and an out-neighbor set. For example, for node 6, the in-neighbor set consists of node 7 and the source, while the out-neighbor set consists of nodes 1 and 9.

II. Mesh-Based Peer-to-Peer Streaming

We consider a unidirectional mesh-based P2P overlay consisting of a bootstrap node, a source node and peer nodes. As seen in Figure 1, the mesh allows peers to download a stream generated by the source, while the bootstrap maintains a list of alive peers used to assist peers to join the network. We consider a unidirectional mesh since it is more general than a bidirectional mesh. Also, unidirectional meshes have been shown to perform better than bidirectional meshes [24].

Every peer node maintains two sets of nodes, in-neighbors and out-neighbors. The in-neighbors represent the nodes that the peer node is receiving data from. The size of the in-neighbors is a system parameter. The out-neighbors represent the nodes that the peer node is sending data to. Each node decides independently the number of out-neighbors to support which will be proportional to its bandwidth. The source has no in-neighbors, only an out-neighbors set, whose size is usually larger than the size of an out-neighbor set of a peer node.

At join time, a peer node j first contacts the bootstrap node to receive a set of candidate nodes to serve as its neighbors in the overlay. Node j then contacts each candidate node and requests to become one of its out-neighbors. If a candidate node c accepts the request, then in turn, j will add c to its in-neighbor set. Each node pro-actively looks for several out-neighbors to connect to as well.

After it joins the overlay, a node discovers other peers by occasionally contacting its neighbors to learn about their own neighbors. This gossip protocol allows a node to update its in-neighbor set when neighbors leave or crash. A node also registers with the bootstrap node occasionally to allow the bootstrap node to have an up-to-date list of alive nodes. We will refer to these protocols as the *control plane* of the overlay.

The source node splits the stream into data chunks of a fixed size, each uniquely identified by a sequence number. To

receive a chunk a node will send a request to an in-neighbor with that chunk's sequence number. If the requested node does not respond before a deadline then the requesting peer will consider that request lost. Each peer node maintains a buffer that it is trying to fill with data chunks. The buffer corresponds to a playback deadline, such that if a block of the stream is not received before that deadline, the data is considered lost and thus the quality of the playback stream is diminished. We will refer to this protocol as the *data plane* of the overlay.

This model is general enough to capture the characteristics of several mesh-based previously deployed systems [4], [13].

III. Attacks Against Data Delivery

We state the assumptions we make about the attacker and provide a taxonomy of attacks against mesh-based P2P streaming systems.

A. Attack Model

We assume that a fraction f of peers are compromised and can behave arbitrarily. Their main goal is preventing the overlay from delivering data to each peer in a timely fashion. An attacker can disrupt the data delivery *directly* by attacking the data plane, or *indirectly* by attacking first the control plane to gain control over the data delivery path and then disrupting the data delivery.

We assume a defense against Sybil attacks [25] is in place, such as binding IP addresses to certificates or one that leverages social networks [26]. We also assume that data integrity is ensured and data is protected from pollution [6], [20]. We assume that the source and the bootstrap node are trusted and always behave correctly.

B. Attacks on the Data Plane

When two nodes A and B accept each other as out-neighbor, and in-neighbor, respectively, they assume several implicit commitments from each other:

- **Data delivery commitment:** A commits to B that it is going to deliver a certain amount of data to B.
- **Data download commitment:** B commits to A that it is going to download a certain amount of data from A.
- **Data upload commitment:** B is going to upload to the overlay what it downloaded from A.
- **Source upload commitment:** B will upload the data downloaded from the source to others in the overlay. This is similar to the data upload commitment, however we list it separately given that the source is a special entity where all the data originates.
- **Data integrity commitment:** A commits to B that it is not going to upload to B meaningless data.

However, in many mesh systems, not all of these commitments are explicitly enforced by the system. As a result, malicious nodes can exploit them to attack the data plane. We identify the following attacks:

- **Data dropping attacks:** If the data delivery commitment is not met, a malicious node can accept benign nodes as its out-neighbors, but not deliver data to them. The attacks are effective because each data chunk has a strict deadline. A node only has time to make a few downloading attempts for a chunk, and will miss it once the deadline is passed.
- **Neighbor exhaustion attacks:** If the data download commitment is not met, a malicious node can become out-neighbors of benign nodes, but not download data from them. As many meshes limit the number of out-neighbors to ensure that nodes can honor the bandwidth requirements, by being included in the out-neighbors a malicious node exhausts the slots in that set thus denying access to other benign nodes.
- **Source attack:** If the source upload commitment is not met, malicious nodes do not forward data given to it by the source. Thus if a particular chunk is only received by malicious nodes it will not be available to any benign nodes.
- **Free-riding attacks:** If the data upload commitment is not met, malicious nodes could also download data but not upload them to other peers, and basically obtain free service without contributing to the system.
- **Data pollution attacks:** If the data integrity commitment is not met, malicious nodes can upload meaningless data, thus polluting the information in the overlay.

C. Attacks on the Control Plane

The above data plane attacks are more effective when they impact many nodes in the overlay. A malicious node can increase the impact of its attack by first attacking the control plane. The control plane provides nodes with two mechanisms to discover peers. The first consists of the list of alive peers provided by the bootstrap node when a node joins the overlay. The second consists of exchanging membership information between the node and known peers. The bootstrap list is up-to-date if peers periodically register with the bootstrap node to inform it that they are alive. Assuming the bootstrap node is trusted, the control plane achieves its goals if the following commitments are met:

- **Registration with the bootstrap node commitment:** A peer commits that it will register occasionally with the bootstrap node, at a rate specified by the protocol.
- **Referral list commitment:** A node commits to provide a neighbors list that does not purposely contain malicious nodes and is not biased towards some nodes.

We identify the following attacks that have an impact on neighbor selection:

- **Bootstrap list pollution attacks:** If the registration with the bootstrap node commitment is not met, malicious nodes can register fast and often with the bootstrap node filling the bootstrap node's list of alive peers. Thus, although the bootstrap node is trusted, the list that it will provide to the joining peers will be polluted with malicious nodes. Note that malicious nodes can also register infrequently or not at all, but in this case they will not impact the list of the bootstrap node.
- **Neighbor selection attacks:** If the referral list commitment is not met an attacker can collude with other malicious nodes and when contacted about its own neighbors, refers only other malicious nodes. This attack is epidemic in nature since soon

benign nodes will also be referring the malicious nodes they know to other benign nodes.

D. Our Focus

We focus on the attacks that we believe can be the most effective strategy for an attacker to disrupt the data delivery, and allow him to inflict maximum damage on the system with minimal resources. The most effective strategy for a malicious node is to (i) become neighbors of as many nodes as possible, and (ii) deliver as little data as possible. Hence we focus on control plane attacks (i.e. bootstrap list pollution and neighbor selection) that seek to increase the connectivity of malicious nodes and also on several data plane attacks (i.e. dropping, neighbor exhaustion and source) as they can create considerable damage in the network.

We note that many of these attacks are specific to streaming, as file-distribution systems do not have real-time deadlines of data, nor need to download at a particular streaming rate, and often have centralized membership protocols (e.g. BitTorrent).

We do not consider attacks such as free-riding or data pollution as they relate to selfish behavior and data integrity but not attacks on data delivery. Furthermore, several solutions to free-riding have been proposed in previous work [21], [22]. Also, to prevent data pollution, Dhungel et al. [20] have shown that a suitable means to accomplishing this is the source digitally signing hashes of the chunks. We note that solutions to these attacks can be used to complement our work.

IV. A Design For Securing Data Delivery

In this section we describe our design for securing the data delivery for a P2P mesh-based streaming overlay. We first outline the design goals, then describe the details of our design.

A. Design Goals and Overview

Our focus is on ensuring that the P2P system achieves its intended goal which is continuous data delivery, even when under attack. However, achieving the same level of service in the presence of insider attacks as in the benign case is not always possible. As a result, our specific goals are:

(G1) Limit the impact of the attack: We seek to raise the bar for the attacker and bound the amount of damage per attacker. The damage created is directly proportional with the number of attackers and the amount of data dropped by the attacker nodes. Our goal is to allow nodes to quickly find other trustworthy nodes to send and receive data from, thus limiting the adverse affects malicious nodes can have on the system.

(G2) Limit the overhead of the defense mechanisms: As malicious behavior is not a priori known, the components of our design are proactive, thus they must be enabled regardless of the presence of attacks. One specific concern is the overhead of the defense mechanisms. Our goal is that when no attack takes place, the system performance with the defense mechanisms enabled is the same as if they were not used.

Peer protection: To limit the impact of attacks and the overhead of the defense solution, we use decentralized mechanisms deployed at each individual peer that allow it to make local decisions about accepting, rejecting or excluding other peers from its set of neighbors. Each individual node derives reputation scores for the other peers it is aware of in the overlay. The use of reputation is a natural choice in a distributed system with malicious participants. Since many existing reputation systems require additional overlays or have high computational or bandwidth overhead [27], we design schemes that are tailored to streaming overlays. The novelty of our scheme lies in combining feedback from the data plane and control plane to build reputations for each peer.

Protecting Other Components: As the source is a producer but not a consumer of data, the protection mechanisms used for peers are not applicable to the protection of the source. We use mechanisms that limit the impact of source attacks by controlling the duration at which nodes can become neighbors of the source. The bootstrap node plays a critical role in the control plane. Attacks against the control plane can be amplified if the bootstrap is not a reliable and unbiased source of information on who is currently in the overlay. Our scheme discourages nodes from registering at a fast rate and thus limits the percentage of malicious nodes in the bootstrap list.

B. Protecting Peers through Local Reputation

We propose a mechanism that allows peers to select as neighbors the nodes that provide the best performance while being resilient to data dropping and neighbor selection attacks. A node uses locally observed data and control plane information to compute scores for each of its neighbors. The lower the score, the higher the chance that a node is malicious. Nodes that have a score lower than a threshold T_d are evicted from the in-neighbors set. The local reputations are also sent across one hop to neighbors, so that they can avoid accepting malicious nodes as in-neighbors. The score consists of two components:

• *Data score:* This score is a *positive reputation* (it rewards good behavior) and it is calculated based on how much data a node has received from a particular neighbor. The goal of the data score is to capture regular performance degradation and data dropping attacks. Nodes who do not deliver sufficient data will have a lower data score. Nodes with a data score below a threshold T_s are considered to be *suspicious*. This approach forces malicious neighbors to deliver a certain amount of data. Note that for a node to be evicted from the neighbors set, his total score has to be smaller than T_d ($T_d < T_s$).

• *Graph connectivity score:* This score is a *negative reputation* (it penalizes bad behavior) and is calculated based on how connected a node is to other nodes. The goal of the graph connectivity score is to target neighbor selection attacks. This score is relevant only for suspicious nodes because if nodes deliver enough data (i.e. have a data score $> T_s$) they do not disturb the overlay. A high graph connectivity score indicates that a node is potentially conducting a neighbor selection attack. This score is used because if the data score is neither high nor low, it may not be obvious if a node is malicious.

Below we provide details about the data and graph connectivity score computation, about the way they are combined

into a reputation score, and about how the reputation score is used to make decisions on what nodes to allow as neighbors.

Data score computation. Every node i calculates a data score for every in-neighbor j as follows:

$$L_{ij}(t) = min\left(1, \frac{G_{ij}(t)}{E(t)}\right) \quad (1)$$

where $G_{ij}(t)$ is the number of chunks received by node i from j before deadline D_r in a time period. Note that when schemes to enforce data integrity are in place [6], [20], only chunks that passed integrity checks and are not polluted are included. D_r is the amount of time the requesting peer will wait before considering that the request was dropped. If a request for a chunk is honored after the D_r deadline has passed, it is not included in $G_{ij}(t)$. We note that the above equation can be augmented to give partial credit to the sender for chunks that arrive after the D_r deadline. $E(t)$ is the expected number of chunks to be received by a node in a time period. Typically, the expected value is the same for all nodes and if it is received from all in-neighbors the full streaming rate will be received. We take the minimum of $\frac{G_{ij}(t)}{E(t)}$ and 1 so that if a node performs better than expected the end result more data j delivers to i, the bigger the $L_{ij}(t)$. If $L_{ij}(t)$ is less than a threshold T_s, then i marks j as *suspicious*.

A score for a node's out-neighbors is calculated by replacing $G_{ij}(t)$ with the number of requests fulfilled for that node in a time period. Such a score allows nodes to mitigate neighbor exhaustion attacks.

Graph connectivity score computation. Every node also calculates a graph connectivity score for each of its neighbors that were marked suspicious. This score relies on the observation that a malicious node conducting a neighbor selection attack will be an in-neighbor for many honest nodes. In particular, if the malicious node is an in-neighbor of a benign node i, it is likely to be an in-neighbor of i's neighbors as well. We propose the following graph connectivity equation for each node i to calculate the likehood of each of its in-neighbor j being malicious:

$$C_{ij}(t) = \frac{K_{ij}(t)}{N_i(t)} \quad (2)$$

where $N_i(t)$ is the total number of non-suspicious neighbors of i (i.e. neighbors whose data score L is greater than T_s), and $K_{ij}(t)$ is the number of non-suspicious neighbors for whom j is also an in-neighbor. Intuitively, the equation calculates a score equal to the percentage of non-suspicious neighbors that a neighbor j is currently an in-neighbor for. The score will be high if a neighbor is in many neighbor sets, indicating that it is malicious. We consider only non-suspicious nodes so that in the case a malicious node wants to falsely advertise other nodes in its in-neighbor set, it has to perform some work for the system.

Reputation score computation. Every node combines the data and graph connectivity score as follows:

$$R'_{ij}(t) = \begin{cases} L_{ij}(t) - \alpha * C_{ij}(t) & \textit{if j is suspicious} \\ L_{ij}(t) & \textit{otherwise} \end{cases} \quad (3)$$

If a node had a low data score and was marked as suspicious, then we subtract from the L_{ij} data score the C_{ij} graph connectivity score weighted by a parameter α. This choice was made based on the observation that if the nodes deliver enough data, it does not matter how connected they are as they do not disturb the honest nodes.

Incorporating history. Every node takes into account the history of its neighbors by calculating for each neighbor the following equation:

$$R_{ij}(t) = \lambda * R'_{ij}(t) + (1 - \lambda) * R_{ij}(t-1) \quad (4)$$

where λ is a value less than 1. We take into account history to accommodate transient network conditions. All nodes start with a reputation equal to T_s.

Reputation based neighbor selection. A node uses reputation scores to decide when to drop or add neighbors. To decide if he keeps a node j as a neighbor, node i compares the reputation score R_{ij} for node j with a threshold T_d. If j's score becomes less than T_d, then i will drop j from its neighbor set and will not allow j to be in either its in-neighbor or out-neighbor sets from then on.

A node also uses the reputation score to determine if a node is non-malicious when deciding to add a neighbor. Consider the case when a node s refers a neighbor k to node i, s will also send the reputation score of k. To decide if he adds k as a neighbor, i computes $R_{is} * R_{sk}$. Node i will then add node k as a neighbor if the resulting number is greater than the suspicion threshold, T_s.

C. Source and Bootstrap Protection

The source is a critical component of the overlay. As will be shown in Section VII attacks against the source can significantly degrade the performance of the system. We note that as time progresses and benign nodes churn in and out of the system, malicious nodes can continue to stay and eventually eclipse the source as its neighbors. To address this problem we induce churn [28] on the source. We allow a single node to stay as an out-neighbor for only a certain amount of time and then disconnect it. To further stagger the disconnection times of nodes, we only allow one node to be disconnected in a time period. We refer to this scheme as *Drop Periodically*. Assuming the bootstrap node is also protected and the source only obtains referrals from it, we expect that with this mechanism the percentage of malicious nodes in the source's out-neighbor set will be no greater than the percentage of malicious nodes in the system.

Our bootstrap node solution relies on the observation that nodes that register with it many times in a short period are most likely malicious. Thus to penalize this behavior, if nodes register too fast, they will not be put into the bootstrap list and then will not be propagated by the bootstrap node. To only do rate-limiting and nothing else might bring about scenarios where there are very few honest nodes in the bootstrap list. This could be due to few nodes joining the overlay for a period of time. To ensure that the bootstrap list still can not be filled

with malicious nodes, we have each node register periodically. We refer to this scheme as *Rate-limiting Bootstrap*.

V. Security Analysis

In this section, we analyze how robust the *Local Reputation* scheme is in defending against common classes of attacks. Recall that the final reputation score is derived by combining the data score, which is a positive score, and the graph connectivity score, which is a negative score. The node uses the final reputation score to decide who should remain as neighbors and who to admit as neighbors. Possible attacks that can be conducted on these reputation calculations and uses include [23]:

Self-promoting: Malicious nodes falsely inflate their own reputation. This attack is only effective in positive feedback based systems.

Slandering: Malicious nodes attack the reputation of other nodes by reporting untrue information about them. This attack is only effective in negative feedback based systems.

Orchestrated: Colluding nodes combine several strategies to game the system.

Whitewashing: Malicious nodes take advantage of a system vulnerability to restore a damaged reputation. One possible way to do this is by assuming new identities.

A. Attacks on Data Score Calculation

The reputation system is designed so that a node cannot get a high data score and thus a high reputation without doing useful work. Therefore, the data score cannot be influenced by slandering or self-promoting attacks, as the only way to change it is for a node to deliver more data. We present the following lemma which quantifies the amount of useful work done by a node given a particular data score, which can be derived from Equation 1.

Lemma 1: *For a node j to obtain a data score of L_{ij} at a neighboring node i, j must deliver data to i at a minimum rate of $E * L_{ij}$, where E is the expected amount of data a node should deliver to a neighbor in a time window (Section IV-B).*

This lemma guarantees that benign nodes will receive good performance even when surrounded by a significant number of malicious neighbors, for example, when under an orchestrated attack. This is because each malicious neighbor is forced to deliver a minimum amount of data in order not to be dropped. More specifically, if we assume a node with a fraction f of its neighbors is malicious, and assume benign neighbors always deliver the expected amount of data, then the node will receive at least $(1-f)+T_d f = 1-f(1-T_d)$ of the streaming rate (T_d is the drop threshold). For example, with $T_d = 0.5$ and $f = 0.3$, the node will receive at least 85% of the stream rate.

Furthermore, Lemma 1 imposes a high bandwidth cost on malicious nodes who seek to be a neighbor of a large number of nodes. To highlight this, consider a streaming system with 150K nodes [6], and that a malicious node desires to maintain a reputation score of T_d at every node. According to Lemma 1, with a streaming rate of 1Mbps, a neighbor-set size of 15, and assuming a T_d value of 0.5, the node must deliver data at a minimum total rate of 5Gbps.

B. Attacks on Graph Connectivity Score Calculation

When a node calculates its neighbors' graph connectivity score, it takes into account neighbor set information provided by all of its non-suspicious neighbors. This scheme is subject to slandering attacks where a malicious neighbor can provide fake neighbor set information. Slandering can be seen from two different perspectives, the ability of a node to slander others and the resistance a node has from slandering attempts. We first present the following lemma that shows the limitations a node has in its ability to slander others, which can be derived from Equation 2.

Lemma 2: *A node j can only influence the graph connectivity scores of the neighbors of node i if j has a data score of T_s with i.*

Lemma 2 shows that malicious nodes must themselves do a substantial amount of work to remain non-suspicious, which means having a data score above T_s. According to Lemma 1, this means they must deliver data to i at a minimum rate of $E * T_s$. Given that $T_s > T_d$, this imposes an even greater bandwidth constraint on attackers that want to slander others over attackers that want to simply not be dropped.

We next present a lemma that demonstrates that a node can resist slandering attacks from others. The key insight behind the lemma is that if a node transmits data at a high enough rate, its final reputation as computed by the neighbor depends on the data score alone, and is not impacted by the graph connectivity score.

Lemma 3: *Any node that delivers data to a neighbor at a rate greater than $E * T_s$ is assured of a reputation greater than T_s with the neighbor.*

We expect that most benign nodes will be cooperative and deliver data at rates close to the expected rate, which is well above $E * T_s$. Therefore benign nodes will not be subject to slandering attacks as their graph connectivity score will not even be considered.

C. Other Attacks

We discuss other attacks on the schemes, and why our approach is resilient to them:

Whitewashing attacks: In these attacks, malicious nodes who received a bad reputation may choose to rejoin the network with a different identity. We believe this attack is not a concern because of the following reasons. First, the reputation is initialized to T_s, and all new nodes will be marked suspicious initially. Therefore a new node cannot refer other nodes or report connectivity information about other nodes until it has done work and improved its reputation. Further, the newly added node will be quickly dropped unless it transmits data at a sufficient rate. Second, in our model, nodes are identified by their IP address. To cause damage, a malicious node cannot acquire a new identity by simply spoofing an IP address, but must be able to receive packets targeted to the IP address. By our attacker model in Section III-A, we assume a bound on the total number of IP addresses that the malicious node controls.

Attacks on Reputation-Based Neighbor Selection: A node adds new neighbors by taking referrals from existing non-

suspicious neighbors. This process is subject to attacks where a malicious neighbor (m) could refer a node (i) to other malicious nodes. However, to conduct this attack, the malicious neighbor m must be considered non-suspicious, and hence must deliver data at a minimum rate of $T_s * E$, where T_s is the suspicion threshold. Further, each newly inserted malicious node referred by m must also do a substantial amount of work to obtain a minimum data score of T_d (the drop threshold), or it will be dropped quickly by i.

VI. Experimental Methodology

In this section, we describe how we evaluate and compare our protection schemes with several others. We implemented the unidirectional mesh described in Section II in a mesh streaming codebase [9]. We also implemented all of our own protection schemes plus some alternatives which we will describe next, which are summarized in Table I.

A. Schemes Considered

TABLE I
MECHANISMS FOR EACH PART OF SYSTEM

Peers	Source	Bootstrap
Least Performing Peer (LP)	Pretrusted Peers (PP)	
Local Reputation (LR)	Drop Periodically (DP)	Rate-limiting (RB)

No Protection (NP): This is our baseline scheme which has no protection for any of the system components.

Local Reputation (LR), Drop Periodically (DP) and Rate-limiting Bootstrap (RB): These are the schemes we proposed in Section IV.

Least Performing Peer (LP): This is a peer level scheme, similar to the one used in CoolStreaming [11], that drops the in-neighbor that is currently contributing the least amount of data. We chose this alternative to LR because of its simplicity and to show that while simple schemes such as this prove to be effective in a setting where all nodes are benign, more robust methods are needed when malicious nodes are present.

Pretrusted Peers (PP): This is a source protection scheme that assumes the existence of pretrusted peers in the system. The source only accepts as neighbors those whom the pretrusted peers validate as nodes that are willing to send data to them. We selected this scheme as an alternative to DP, to demonstrate that when no pretrusted nodes are available, DP provides similar results while making less assumptions.

B. Experiment Configuration

TABLE II
NOTATION

D_r	Deadline at which a peer considers a request for data dropped
T_s	The suspicion threshold
T_d	The drop threshold
α	When calculating $R'_{ij}(t)$ gives a weight to $C_{ij}(t)$
λ	When calculating $R_{ij}(t)$ gives a weight to the previous value of $R_{ij}(t-1)$ and the current value of $R'_{ij}(t)$

The experiments were run on the PlanetLab overlay testbed. The source was located on a host at our lab. We set D_r, to be 1 second. We determined this value experimentally as we observed that in a non-malicious scenario 96% of nodes receive 99% of chunks within 1 second. Each node is configured to obtain up to 15 in-neighbors and the maximum number of out-neighbors is proportional to its bandwidth. The source will obtain 30 out-neighbors.

Malicious nodes always conduct the data dropping, neighbor selection and source attacks described in Section III. In addition, when we explicitly state, malicious nodes also conduct a bootstrap list pollution attack. To facilitate these attacks malicious nodes advertise the chunks of data that they have, but do not fulfill any requests unless stated otherwise. They also accept up to 100 out-neighbors.

We used overlay deployments of 300 nodes. Each experiment lasted for 10 minutes. For each experiment we varied the percentage of malicious nodes from 0 to 30% and fixed the source's streaming rate at 1 Mbps. Each experiment was run for 10 times and the results were averaged. Standard deviations are plotted where appropriate. The malicious nodes joined at the beginning of the experiment and stayed for the entire duration. Benign nodes both join at the beginning of the experiment and also during the experiment. We modeled the join times by using a Poisson process and the participation time by a Pareto distribution. The mean of the Poisson process was 3 and the Pareto distribution is used with an α of 1.42 giving a mean participation time of 300 seconds and we also assume a minimum participation time of 90 seconds. The parameters have been used previously by Bharambe *et al.* [29] and were motivated by traces of real multicast systems [3] and Mbone sessions [30].

Choosing Parameters: For *Local Reputation* we by reason set its parameters to appropriate values and validated them experimentally. We set T_s to be 0.7 to tolerate transient network conditions. We note that T_s can be set by the user, to the minimum quality threshold that he is willing to tolerate. We set α to be 0.5 since we consider data plane feedback to be more useful than control plane feedback. We also conduct a sensitivity study of α in Section VII. For nodes to evict malicious nodes that are both suspicious and highly connected, the equation $T_d \geq T_s - \alpha$ must hold. Therefore we set T_d to be 0.2. We set λ to be 0.4 to give a greater weight to the history of the reputation but also be able to change quickly if nodes consistently behave badly. We set the time period for the recalculation of the scores to be every 3 seconds.

VII. Experimental Evaluation

In this section we experimentally show the schemes we proposed in Section IV are able to effectively mitigate attackers. We evaluate the effectiveness of the attacks and solutions with the goodput ratio. The goodput ratio represents the percentage of useful data a node received while in the overlay, averaged across all nodes. We use it to measure the effects of churn on the quality of the goodput. The higher the goodput ratio, the higher the quality of the stream received. We exclude overhead

metrics as we have designed our schemes to have minimal computational overhead and any added network traffic piggybacks on existing traffic.

A. Robust Neighbor Selection

To give motivation to our *Local Reputation (LR)*, we first compare it to *Least Performing Peer (LP)* and *No Protection (NP)*. As can be seen in Figure 2(a) both *NP* and *LP* perform worse than *LR*. This difference becomes more pronounced as the percentage of malicious nodes increases. *NP* is ineffective simply because nodes never change who their neighbors are, regardless of their poor performance.

LP is not as effective as *LR* since a node never drops all of the malicious nodes from its neighbor set. Further investigation shows that for a node running *LP* the number of malicious nodes in its in-neighbor set decreases as some of the malicious nodes will be dropped. However, there are still malicious nodes present in the in-neighbor set because *LP* does not prevent the node from reconnecting multiple times to the same malicious nodes. When the node is running *LR*, it does not reconnect anymore to malicious nodes since malicious behavior is captured in the reputation score for those nodes.

Importance of considering graph connectivity: We examine the contribution of the graph connectivity score on *LR* and identify regimes in which its use is beneficial. We compare the case when the reputation score computation is based only on the data feedback (i.e. $\alpha = 0$) to the case when both data and graph connectivity are considered (i.e. $\alpha = 1$).

As we can see in Figure 2(b) when attackers drop 25% or 75% of the data they were expected to deliver, the performance does not change no matter the value of α. For the case of 25% dropping, recall that a node i will only calculate the graph connectivity score for a neighbor j if it marks j as suspicious (i.e. $L_{ij} < T_s$). When j drops 25% of the data it will not be marked as suspicious since we use a T_s value of 0.7, thus the graph connectivity score will not be considered. In the case of 75% dropping, enough data is dropped that the neighbor will be perceived as malicious by its data score alone. Hence graph connectivity is most useful in regimes where the amount of data dropped by a malicious node is large enough to be marked as suspicious, but not large enough to be interpreted as malicious by their data scores alone. This is the case for 50% dropping. In Figure 2(b), when attackers drop 50% of the data, *LR* combining the two scores performs better than *LR* using only data score. The information from the control plane about the existence of a neighbor selection attacks helps effectively identify malicious nodes. We varied α even more to find values that give better performance but we found that a value of 1 is sufficient across all percentages of attackers.

B. Protecting Other Components

While *LR* performs much better than other schemes, the goodput ratio achieved is still far from satisfactory. We believe this is because *LR* does not protect the streaming source, as we explained in Section IV. Further investigation into the source's performance confirms our hypothesis. While peers using *LR* expel all malicious nodes from their in-neighbor set, the source's out-neighbor set is almost full of malicious nodes. This illustrates the importance of having additional mechanisms to protect the source.

We next evaluate mechanisms that can be used to protect the source. When using *Drop Periodically (DP)* the source will drop a node after it has been a neighbor for 1 minute. For *Pretrusted Peers (PP)* we use 4 peers that the source trusts. Figure 3(a) shows the results. The goodput ratio is significantly raised for *DP* combined with *LR* (i.e. the curve titled *LR+DP*). This is because *DP* effectively reduces the percentage of malicious neighbors at the source to a value that is very close to the percentage of malicious nodes in the overlay at all times, for all settings.

PP when combined with *LR* (i.e. *LR+PP*) raised the goodput ratio even higher, though the improvement over *LR+DP* seems marginal. This is because *PP* reduced the percentage of malicious neighbors at the source even further. However, this scheme requires deployment of pretrusted peers, and preserving the anonymity of these peers, both of which are difficult to guarantee. We conclude that *DP* is preferable because it makes less assumptions while providing good performance.

Figure 3(a) also shows that *DP* alone is not sufficient. This is not surprising, because *DP* protects only the source, not the peers. This again highlights that solutions must be employed at both the source and peers to achieve satisfactory performance.

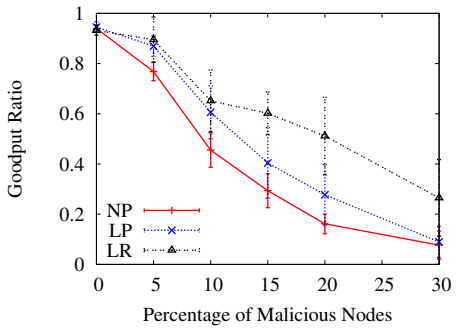
We now consider attacks where malicious nodes also pollute the bootstrap node by registering often. We evaluate the effectiveness of *Rate-limiting Bootstrap (RB)* in mitigating such attacks. We show the results in Figure 3(b). The attack causes a goodput ratio drop of 15% in the scenario where there are 30% malicious attackers. However, RB is able to mitigate the attack and bring the goodput ratio up to 92%.

VIII. Related Work

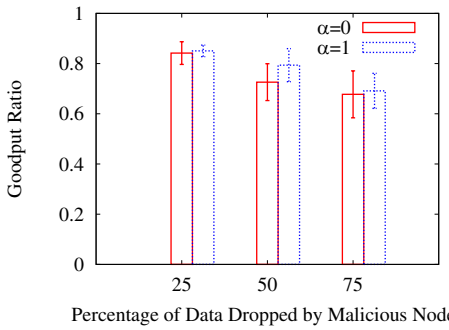
The security challenges in designing mesh-based streaming protocols has received little attention. The only prior work we are aware of focuses on attacks where malicious nodes pollute data sent to other nodes [20]. In contrast, our focus is on data availability and prevention of neighbor selection attacks.

Attacks on data availability have been considered in the context of tree-based multicast [31]. The proposed solution takes advantage of the tree structure, knowing that if a child did not receive a message then an ancestor can be traced back to that is at fault for dropping it. Meshes do not have parent-child relationships but rather nodes get data from many neighbors, so this approach cannot be applied to them. Attacks against measurement-based neighbor selection were studied in the context of tree-based streaming [32]. The proposed solution uses outlier detection to identify malicious nodes that report wrong measurement results. This approach only works with systems that employ such measurement-based adaptation.

Dealing with selfish and Byzantine behavior using game theoretic principles has been investigated in several previous works [33], [34]. Most similar to our work is Flightpath [33], a P2P streaming system that is designed to give selfish

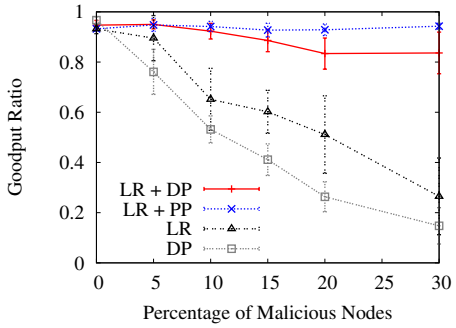


(a) *No Protection*, *Least Performing Peer* and *Local Reputation* schemes running at the peers with *No Protection* at the source.

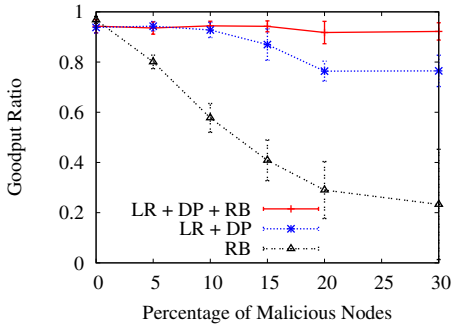


(b) Goodput ratio when 10% of participants are malicious and they drop varying amounts of the stream. *Local Reputation* is protecting the peers and α is varied.

Fig. 2. Importance of the *Local Reputation* scheme.



(a) Peers running the *Local Reputation* scheme with *Drop Periodically*, *Pretrusted Peers* and *No Protection* at the source. For comparison we also plot the *Drop Periodically* only scheme.



(b) *Rate-limiting Bootstrap* and *No Protection* are combined with *Local Reputation* and *Drop Periodically*. *Rate-limiting Bootstrap* with *No Protection* is also shown.

Fig. 3. Importance of *Drop Periodically* and *Rate-limiting Bootstrap* schemes.

peers incentive to obey protocols and can tolerate Byzantine behavior. Unlike their work, we do not assume synchronized clocks or synchronous communication channels.

Several schemes have been proposed to mitigate neighbor selection attacks (referred to as eclipse attacks) in the context of distributed hash tables (DHTs) [35], [36]. The solutions are DHT-specific and do not apply to streaming protocols. A key aspect that distinguishes streaming protocols is the potential for feedback from the data-plane. In particular, it is possible to infer malicious behavior based on lack of data received from a neighbor. Our solutions leverage this observation resulting in significantly simpler designs.

Reputation systems have been a subject of wide interest, especially for P2P file-sharing systems. File-sharing reputation systems generally fall into two categories of purpose, incentivizing users to share files [27], [37], or thwarting file pollution [38]. Piatek et al. [37] show the feasability of using one-hop reputations to incentivize interactions between users in BitTorrent. They take advantage of the fact that there are some users who are in many BitTorrent overlays and thus can be used as intermediaries, keeping track of long-term reputation values for others and facilitating data exchanges. While our work also uses local reputations, we differ in that our goal is mitigating malicious adversaries and not creating incentives. Also, as users usually only watch one video stream at a time, this precludes them from being in many overlays at once, making it impossible for some users to be intermediaries. Thus, streaming presents new challenges for reputation systems and has unique features that create opportunities, such as the continual downloading of data and stringent data deadlines, that we take advantage of.

IX. Conclusion

In this paper, we present one of the first efforts aimed at systematically analyzing and addressing the vulnerabilities of mesh-based P2P streaming systems to malicious insider attacks. We consider both direct attacks on the data plane, as well as attacks on the control plane which could in turn lead to further disruption of data delivery. We present a design for securing data delivery, of which a key component is a reputation scheme that helps nodes identify malicious peers and build a robust neighbor set. Through detailed security analysis, we show that our scheme is resistant to a variety of attacks commonly associated with reputation schemes such

as self-promotion, slandering, and white-washing [23].

We present an extensive evaluation of our design through experiments on PlanetLab. Our results show that (i) without our solution, the data delivery can be seriously disrupted by attacks exploiting the vulnerabilities we identified. For example, 15% malicious nodes caused the average goodput ratio to decrease to less than 30%. (ii) Our solution is effective in mitigating the attacks; it achieves an average goodput ratio of more than 90% even when there are 30% malicious nodes. (iii) While each of the mechanisms we introduce can individually benefit the system, the solution is most effective when all the mechanisms are combined. Overall, these results are promising, and indicate the feasibility of ensuring effective P2P streaming even under the presence of malicious participants.

REFERENCES

[1] S. Deering and D. Cheriton, "Multicast routing in datagram internetworks and extended lans," *ACM Transaction on Computer Systems*, vol. 8, pp. 85–110, 1990.

[2] Y. Chu, S. G. Rao, and H. Zhang, "A case for end system multicast," in *ACM SIGMETRICS*, 2000.

[3] Y.-H. Chu, A. Ganjam, T. S. E. Ng, S. Rao, K. Sripanidkulchai, J. Zhan, and H. Zhang, "Early experience with an internet broadcast system based on overlay multicast," in *USENIX*, 2004.

[4] PPLive, http://www.pplive.com.

[5] PPStream, http://www.ppstream.com.

[6] Y. Huang, T. Z. Fu, D.-M. Chiu, J. C. Lui, and C. Huang, "Challenges, design and analysis of a large-scale p2p-vod system," in *SIGCOMM*, 2008.

[7] X. Hei, C. Liang, J. Liang, Y. Liu, and K. W. Ross, "A measurement study of a large-scale p2p iptv system," *IEEE Trans. on Multimedia*, vol. 9, pp. 1672 – 1687, 2007.

[8] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "Splitstream: High-bandwidth multicast in cooperative environments," in *SOSP*, 2003.

[9] V. Pai, K. Kumar, K. Tamilmani, V. Sambamurthy, and A. Mohr, "Chainsaw: Eliminating trees from overlay multicast," in *IPTPS*, 2005.

[10] V. Venkataraman, K. Yoshida, and P. Francis, "Chunkyspread: Heterogeneous unstructured tree-based peer-to-peer multicast," in *ICNP*, 2006.

[11] X. Zhang, J. Liu, B. Li, and T.-S. P. Yum, "CoolStreaming/DONet: A data-driven overlay network for peer-to-peer live media streaming," in *IEEE INFOCOM*, 2005.

[12] D. Kostic, A. Rodriguez, J. Albrecht, , and A. Vahdat, "Bullet: High bandwidth data dissemination using an overlay mesh," in *SOSP*, 2003.

[13] UUSee, http://www.uusee.com.

[14] SOPCast, http://www.sopcast.com/.

[15] K. Cho, K. Fukuda, H. Esaki, and A. Kato, "Observing slow crustal movement in residential user traffic," in *CONEXT*, 2008.

[16] M. Meeker and D. Joseph, "The state of the internet, part 3," in *Web 2.0*, 2006.

[17] J. Liu, S. G. Rao, B. Li, and H. Zhang, "Opportunities and challenges of peer-to-peer internet video broadcast," in *Proceedings of the IEEE, Special Issue on Recent Advances in Distributed Multimedia Communications*, 2007.

[18] N. Magharei and R. Rejaie, "PRIME: Peer-to-peer receiver driven mesh-based streaming," in *IEEE INFOCOM*, 2007.

[19] J. Seibert, D. Zage, S. Fahmy, and C. Nita-Rotaru, "Experimental comparison of peer-to-peer streaming overlays: An application perspective," in *IEEE LCN*, 2008.

[20] P. Dhungel, X. Hei, K. Ross, and N. Saxena, "The pollution attack in p2p live video streaming: Measurement results and defenses," in *ACM SIGCOMM P2P-TV Workshop*, 2007.

[21] Z. Liu, Y. Shen, K. W. Ross, S. S. Panwar, and Y. Wang, "Substream trading: Towards an open p2p live streaming system," in *ICNP*, 2008.

[22] B. Cohen, "Incentives build robustness in BitTorrent," in *P2P Economics*, 2003.

[23] K. Hoffman, D. Zage, and C. Nita-Rotaru, "A survey of attack and defense techniques for reputation systems," *ACM Computing Surveys*, 2008.

[24] B. Biskupski, R. Cunningham, J. Dowling, and R. Meier, "High-bandwidth mesh-based overlay multicast in heterogeneous environments," in *AAA-IDEA*, 2006.

[25] J. Douceur, "The Sybil Attack," in *IPTPS*, 2002.

[26] H. Yu, P. B. Gibbons, M. Kaminsky, and F. Xiao, "Sybillimit: A near-optimal social network defense against sybil attacks," in *Symposium on Security and Privacy*, 2008.

[27] S. Kamvar, M. Schlosser, and H. Garcia-Molina, "The EigenTrust Algorithm for Reputation Management in P2P Networks," in *Proceedings of WWW2003*. ACM, 2003.

[28] T. Condie, V. Kacholia, S. Sankararaman, J. M. Hellerstein, and P. Maniatis, "Induced churn as shelter from routingtable poisoning," in *NDSS*, 2006.

[29] A. Bharambe, S. Rao, V. Padmanabhan, S. Seshan, and H. Zhang, "The impact of heterogeneous bandwidth constraints on DHT-based multicast protocols," in *IPTPS*, 2005.

[30] K. Almeroth and M. Ammar, "Characterization of mbone session dynamics: Developing and applying a measurement tool," Georgia Institute of Technology, Tech. Rep. GIT-CC-95-22, 1995.

[31] L. Xie and S. Zhu, "Message dropping attacks in overlay networks: Attack detection and attacker identification," *ACM Trans. Inf. Syst. Secur.*, vol. 11, no. 3, pp. 1–30, 2008.

[32] A. Walters, D. Zage, and C. Nita-Rotaru, "Mitigating attacks against measurement-based adaptation mechanisms in unstructured multicast overlay networks," in *ICNP*, 2006.

[33] H. C. Li, A. Clement, M. Marchetti, M. Kapritsos, L. Robison, L. Alvisi, and M. Dahlin, "Flightpath: Obedience vs choice in cooperative services," in *OSDI*, 2008.

[34] I. Keidar, R. Melamed, and A. Orda, "Equicast: Scalable multicast with selfish users," in *PODC*, 2006.

[35] M. Castro, P. Drushel, A. Ganesh, A. Rowstron, and D. Wallach, "Secure routing for structured peer-to-peer overlay networks," in *OSDI*, 2002.

[36] A. Singh, T.-W. J. Ngan, P. Druschel, and D. S. Wallach, "Eclipse attacks on overlay networks: Threats and defenses," in *IEEE INFOCOM*, 2006.

[37] M. Piatek, T. Isdal, A. Krishnamurthy, and T. Anderson, "One hop reputations for peer to peer file sharing workloads," in *NSDI*, 2008.

[38] K. Walsh and E. G. Sirer, "Experience with an object reputation system for peer-to-peer filesharing," in *NSDI*, 2006.