Adaptive Peer-To-Peer Topologies

Tyson E. Condie, Sepandar D. Kamvar, Hector Garcia-Molina
Stanford University, Stanford, CA 94306
{tcondie, sdkamvar, hector}@cs.stanford.edu

Abstract

We present a peer-level protocol for forming adaptive, self-organizing topologies for data-sharing P2P networks. This protocol is based on the idea that a peer should directly connect to those peers from which it is most likely to download satisfactory content. We show that the resulting topologies are more efficient than standard Gnutella topologies. Furthermore, we show that these adaptive topologies have the added benefits of increased resistance to certain types of attacks, intrinsic rewards for active peers and punishments for malicious peers and freeriders.

1 Introduction

While peer-to-peer networks have great potential for large-scale, robust, distributed information sharing, current P2P systems such as Gnutella are not highly efficient or scalable because peers are connected randomly to other peers in the network. Alternative P2P organization protocols that place content at nodes based on hash functions have been proposed, [12], [13]. Such schemes are efficient for queries where the exact search key is known, but behave poorly for approximate queries. Furthermore, neither of these schemes addresses the issues of malicious peers or freeriders as inherent parts of the topology design.

A useful concept for designing scalable, efficient, and robust overlay topologies is that of *interaction topologies*. The interaction topology for a P2P file-sharing network is a graph whose nodes are the peers in the network, and whose arcs are defined by downloads. For example, one may place an arc between nodes i and j if node i has downloaded content from node j more than 3 times.

We propose a peer-level protocol for forming self-organizing topologies based on the idea that efficiency and robustness can be achieved by designing an overlay topology to match the interaction topology. We call the resulting topologies Adaptive P2P Topologies and we call the protocol the APT protocol.

More specifically, the APT protocol is based on two fundamental notions. First, peers should directly connect to those peers from which they are likely to download satisfactory files. Second, peers may use past history to determine the peers from which they are likely to download satisfactory files. (For simplicity, we refer to retrieved content as “files” even though in general it could be any type of query results.) At a basic level, the practical implementation of these ideas involves each peer keeping a score of how many good files it has downloaded from each other peer in the network. Peers connect to those peers that have high scores, and disconnect from peers with low scores. After describing our network model in Section 2, we present a self-organizing, distributed protocol for forming such topologies in Section 3.

We will show several natural characteristics of a topology derived from peers connecting to download sources based on past history. One such property is *efficiency*; peers are more likely to receive responses to their queries with less query forwarding overhead. This is discussed in further detail in Section 4.5. Another property is *security*; malicious peers are moved to the fringe of the network, thereby increasing the network’s resistance to inauthentic file attacks. This is the subject of Section 4.2. The third is *incentives*; freeriders are moved to the fringe of the network, whereas peers that actively share files are moved to the center of the network. Peers that lie on the fringe of the network receive a limited number of responses to their queries. On the other hand, active peers have a wider view of the network and consequently receive a higher quality of service, even if they limit the number of connections they make. The intrinsic rewards and punishments are discussed in Sections 4.3 and 4.4.

2 Network Model

2.1 Network Topology

The P2P network is represented as an undirected graph $G = (P, E)$, where P is the set of nodes, and E is the set of edges (i, j) describing the connections between nodes $i, j \in P$. The connections in the P2P network are symmetric and describe a peer’s neighbor set. That is, peer i’s neighbor set is defined as $N(i) = \{j \mid (i, j) \in E\}$. The network is initialized to contain some number of peers whose connections form a power-law topology, as described in [3].

2.2 Joining the network

A peer joins the P2P network by contacting a designated pong server to obtain some number of live IP addresses [9]. Before adding a node to its neighbors set, the peer must be permitted to make a direct connection. A connection request message $R(i, j)$ is initiated by peer i requesting a direct connection to peer j. The request is sent directly to peer j which then decides whether to accept the connection. If peer j accepts the connection then both peers add one another to their neighbor sets. System bootstrap proceeds by having each peer attempt an initial number of connections γ that does not exceed some system wide maximum number of connections τ.

Bootstrapping and node discovery are still open issues in P2P systems. Schemes that distribute the IP addresses maintained by the pong servers to nodes in the network are evaluated in [10]. Such schemes can easily be incorporated into our model, since the APT protocol is independent of the method for establishing connections.

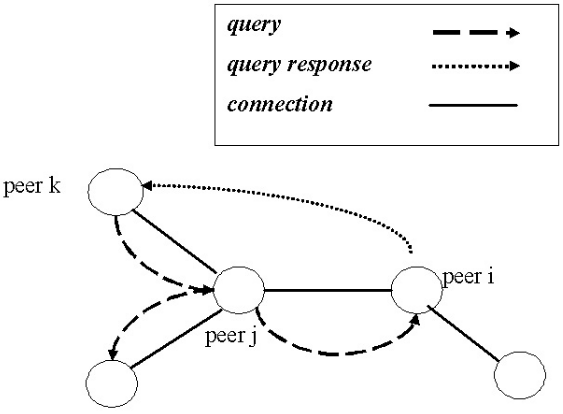

Figure 1: Query Propagation.

2.3 Query Propagation

Query messages are propagated via flooding-based broadcast. A search query can be initiated by any node in the network by first broadcasting to all peers in its neighbor set. Each node, upon receiving a propagated search query, will examine its local file system for a match. Any matches are returned directly to the query initiator. The peer may then propagate the query to all its neighbors except for the node from which it received the query. Each query maintains a time-to-live (TTL) field to limit the scope of the query flooding. At query time, the issuing peer will set the TTL field to some default value, which is then decremented by one at each propagation. A node receiving a query with $TTL = 0$ will not forward the query.

Figure 1 illustrates how a query propagates through the P2P network. In the example, peer k initiates the search query, with $TTL = 1$, by broadcasting to all of its neighbors, in this case peer j. Peer j decides not to respond to the query and forwards the query to its neighbors, except to peer k. Upon receiving the query, peer i directly responds to peer k with a match. The query flooding terminates at peer i since the TTL is now 0.

3 Adaptive P2P Topologies

3.1 Local Trust Scores

Adaptive P2P Topologies are based on two notions:

1. A peer should directly connect to those peers from which it is likely to download files in the future.

2. A peer may use its past history to estimate the likelihood of a future successful download.

In the APT protocol, a peer encodes its past history with a set of *local trust values* [1]. Peer i stores a local trust value for each peer it has interacted with. If $sat(i,j)$ is the number of satisfactory transactions peer i has had with peer j, and $unsat(i,j)$ is the number of unsatisfactory transactions peer i has had with peer j, then we define the local trust value as

$$s_{ij} = sat(i,j) - unsat(i,j).$$

Peer i may deem a transaction unsatisfactory if, for example, the file downloaded is inauthentic or tampered with, or if the download is either slow or interrupted. The *local trust vector* $\vec{s_i}$ associated with peer i contains all s_{ij} values where j varies over all peers in the network. In our implementation, each peer maintains a hash table containing the local trust values of all its acquaintances. An acquaintance is then defined as an entry in the hash table. If peer i has never interacted with peer j then there will be no entry in peer i's table for peer j, and thus $s_{ij} = 0$.

3.2 Protocol

We define the *trustworthiness* of a network to be:

$$Q = \sum_{i=1}^{v} \sum_{j=1}^{v} connection(i,j) \times s_{i,j}$$

where v is the number of nodes in P and $connection(i,j) = 1$ if $(i,j) \in E$, otherwise $connection(i,j) = 0$. Intuitively, a network where peers are connected to peers that they trust will have a high Q value.

Generally, in any P2P System, there will be some connection barriers; for example, peers may wish to set a limit τ on the number of peers to which they connect in order to conserve bandwidth. Connection barriers are common in real world P2P Systems and support fair sharing of resources [9]. We may therefore formulate the problem of creating a trustworthy network as: maximize Q subject to the constraint that each peer i has at most τ connections.

The APT Protocol (Algorithm 1) is a peer-level greedy algorithm for maximizing Q and it proceeds as follows. Peer i joins the network by attempting γ random connections as described in Section 2.2. The join process is repeated until at least 1 connection is made. After downloading an authentic file from peer $j \notin N(i)$, peer i with $n_i < \tau$ connections sends the connection request $R(i,j)$ to peer j. If $n_i = \tau$ then peer i sends a connection request if one of the following holds:

1. Peer $j \notin N(i)$ achieves a higher local trust value than one of peer i's neighbors (message $R(i,j)$ is sent).

2. Peer $j \in N(i)$ is assigned a lower local trust value than some acquaintance $k \notin N(i)$ of peer i (message $R(i,k)$ is sent).

The first scenario describes an authentic download from peer j, while the second occurs after an inauthentic download from peer j. In both cases, if peer i's connection request is granted then it will disconnect from its neighbor with the lowest local trust value. A special case occurs if a neighbor of peer i is assigned a negative local trust score and peer i is not able to connect to an acquaintance. In this situation peer i immediately disconnects from that neighbor and attempts a connection to a random peer.

Peer j will only accept peer i's connection request if peer j's local trust value s_{ji} of peer i is non-negative and one of the following conditions is true:

1. Peer j has fewer than τ connections.

2. Peer j's local trust value s_{ji} of peer i is greater than the local trust value of at least one neighbor.

In the second case peer j will replace its lowest trust valued neighbor with peer i. The careful reader will notice that peer j would have previously made a connection request to peer i because of peer i's desirable s_{ji} value. It is assumed that peer i had denied the connection request $R(j,i)$ but then later sent the connection request $R(i,j)$.

```
function JoinNetwork() {
while $| N(i) | < 1$ do
    connectToRandomPeers();
end
}

function connectToRandomPeers() {
$R$ = some set of random IP addresses from pong server;
foreach $j \in R$ do
    if $j \notin N(i)$ && $s_{ij} \geq 0$ then
        if $connect_j(i) = true$ && $| N(i) | \geq \gamma$ then
            return from function;
        end
    end
end
}

function download(peer $j$) {
if download satisfactory then
    $s_{ij} = s_{ij} + 1$;
    if $| N(i) | < \tau$ then
        connect$_j(i)$;
    end
    else
        find $k \in neighbors$ with lowest $s_{ik}$;
        if $s_{ij} > s_{ik}$ && $connect_j = true$ then
            disconnect$_k(i)$;
        end
    end
end
if download unsatisfactory then
    $s_{ij} = s_{ij} - 1$;
    if $j \in N(i)$ then
        $A = \{k \mid k \notin N(i)$ && $s_{ij} < s_{ik}$ && $s_{ik} \geq 0\}$;
        while $A \neq \emptyset$ do
            find peer $x \in A$ with max $s_{ix}$;
            if $connect_x(i) = true$ then
                disconnect$_j(i)$;
                return from function;
            end
        end
        if $s_{ij} < 0$ then
            disconnect$_j(i)$;
            connectToRandomPeers();
        end
    end
end
}

function connect(peer $j$) {
if $| N(i) | < \tau$ && $s_{ij} \geq 0$ then
    return true // accept connection from peer $j$;
end
else
    find neighbor $k$ with min $s_{ik}$;
    if $s_{ik} < s_{ij}$ then
        disconnect$_k(i)$;
        return true // accept connection from peer $j$;
    end
end
return false // connection denied;
}

function disconnect(peer $j$) {
$N(i) = N(i) \setminus j$;
if $| N(i) | < \gamma$ then
    connectToRandomPeers();
end
}
```

Algorithm 1: Basic APT Protocol

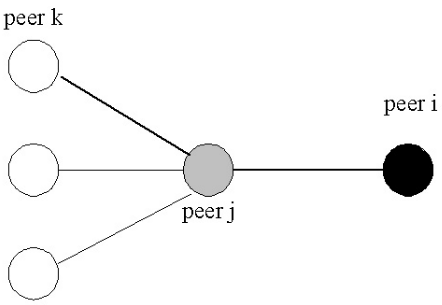

Figure 2: Hidden Malicious Attack.

3.3 Practical Issues

Hidden malicious peers and local maxima are two practical issues not addressed by the basic algorithm presented here.

Hidden Malicious Peers. A malicious peer may disseminate corrupt or inauthentic files by connecting to a peer to which it does not directly upload files. To illustrate with Figure 2, let an altruistic peer be defined as a peer that shares files but does not download files. Suppose that peer i is a malicious peer and peer j is an altruistic peer. Since peer j does not download any files, its local trust vector will be an all-zero vector. According to Algorithm 1, peer j will never disconnect from any other peer. Furthermore, other peers will connect to peer j because it provides authentic files. Therefore, the altruistic peer j unwittingly serves as a conduit through which malicious peer i receives queries used to disseminate inauthentic files. In effect, malicious peer i is ”hiding” behind altruistic peer j.

We addresses this issue by defining the notion of *connection trust*. While the local trust value s_{kj} quantifies the number of authentic files that peer k downloads from peer j, the connection trust value r_{kj} quantifies the number of authentic files that peer k *discovers* through peer j. For example, suppose that in Figure 2 peer j forwards peer k's query to peer i which then responds with a match. If peer k downloads a file from peer i and the file is authentic, then the local trust s_{ki} and the connection trust r_{kj} are incremented by 1. If the file is inauthentic, then both values are decremented by 1.

Algorithm 2 extends Algorithm 1 by including connection trust in the overall value of an acquaintance. We are then able to combat the hidden malicious peer problem by having peers disconnect from neighbors with low connection trust values. By dropping connections to peers that serve as malicious conduits, hidden malicious peers lose their ability to upload malicious content. To recover from the connection loss, the altruistic peer will break all of its connections and attempt new connections.

In order to keep track of the peer through which a file is discovered, the standard query-response protocol needs to be slightly modified. Peer i issuing a query must tag the query with an encrypted identifier of the neighbor to which it initially sends the query. Any query responder must append the same encrypted identifier to its query response. Response messages that do not contain an encrypted identifier to some outstanding query are dropped. This allows peer i to know the peer through which it discovers each of the files it downloads.

Local Maxima. A peer may find itself in a part of the network where it receives few responses to its queries. We define the notion of a void download to be the situation in which a peer receives no response to a query. The basic principle behind adaptive P2P topologies is that the peer should be able to bootstrap its way to a better-suited part of the network by establishing and breaking connections. However, if after a long period of time, a peer is still

```
function connectToRandomPeers() {
$R$ = some set of random IP addresses from pong server;
foreach $j \in R$ do
    if $j \notin N(i)$ && $s_{ij} \geq 0$ && $r_{ij} \geq 0$ then
        if $connect_j(i) = true$ && $\mid N(i) \mid \geq \gamma$ then
            return from function;
        end
    end
end
}

function download(peer $j$) {
if download satisfactory then
    $s_{ij} = s_{ij} + 1$;
    $r_{ic} = r_{ic} + 1$;
    // Rest same as Algorithm 1;
end
if download unsatisfactory then
    $s_{ij} = s_{ij} - 1$;
    $r_{ic} = r_{ic} - 1$;
    $A = \{k \mid k \notin N(i)$ && $r_{ik} > r_{ic}$ && $r_{ik} \geq 0\}$;
    while $A \neq \emptyset$ do
        find peer $x$ with max $r_{ix} \in A$;
        if $connect_x(i) = true$ then
            disconnect$_c(i)$;
            return from function;
        end
        else
            $A = A \setminus x$;
        end
    end
    if $r_{ic} < 0$ then
        disconnect$_c(i)$;
        connectToRandomPeers();
        return from function;
    end
    // Rest same as Algorithm 1;
end
}

function connect(peer $j$) {
if $\mid N(i) \mid < \tau$ then
    if $s_{ij} \geq 0$ && $r_{ij} \geq 0$ then
        return true // accept connection to peer $j$;
    end
end
else
    find neighbor $k$ with min $s_{ik}$;
    if $s_{ik} < s_{ij}$ && $r_{ij} \geq 0$ then
        disconnect$_k(i)$;
        return true // accept connection to peer $j$;
    end
end
return false // connection denied;
}

function disconnect(peer $j$) {
$N(i) = N(i) \setminus j$;
if $numLostConnections(i) > threshold$ then
    drop all current connections;
    connectToRandomPeers();
    reset numLostConnections($i$);
end
else
    drop neighbor $k$ with lowest $r_{ik}$;
    connectToRandomPeers();
    increment numLostConnections($i$);
end
}
```

Algorithm 2: APT Protocol Connection Trust Extension

local trust	peer score based on download transaction.
connection trust	neighbor score based on the outcome of a query forwarded by the neighbor.
void downloads	number of queries with no response.

Table 1: Connection Variables

seeing many void downloads, it may be stuck in a local maxima.

We handle this issue using random restarts. That is, if a peer has a consistently low number of responses per query over a long period of time, then it may choose to do the following:

1. Exchange connections that exhibit low performance [1] with new connections to random peers.

2. Break all connections and re-enter elsewhere in the network.

The first option avoids dropping current connections that may still provide future use. The second is used when all connections fail to provide responses to a large number of queries. Algorithm 2 incorporates these options, which are loosely analogous to simulated annealing techniques in traditional optimization [15].

4 Empirical Results

In this section, we assess the performance of the proposed scheme, and compare it to a P2P network with a standard power-law topology. The performance analysis is given for standard conditions as well as a variety of threat models. The success of the APT protocol is dependent on the variables a peer uses to determine its connections to other peers. Equipped with the variables listed in Table 1, a peer is able to avert malicious attacks and choose, from its acquaintances, peers that share similar interests.

4.1 Simulation

Our findings are based on simulations of a P2P network model described in [3]. The network model is described in Section 3, while the peer and content models are briefly explained here.

Node model. The network consists of good nodes (normal nodes, participating in the network to download and upload files) and malicious nodes (adversarial nodes, participating in the network to undermine its performance). We also consider cases that include freeriders, nodes that only download files and do not share any files of their own [5]. We analyze different behaviors of a malicious peer in the network using threat models. These models will be described in more detail later on.

Content distribution model. Interactions between peers i.e. which queries are issued and which queries are answered by given peers, are modeled using a probabilistic content distribution. The detailed model is presented in [3]. Peers are assumed to be interested in a subset of the total available content in the network, such that each peer initially picks a number of content categories and shares files only in these categories. It is shown in [2] that files shared in a P2P network are often clustered by content categories. When the simulator generates a query, it does not generate a search string. Instead, it generates the category and rank (or popularity) of the file that will satisfy the query. The category and rank are based on Zipf distributions. Each peer that receives the query checks if it supports the file category and if so, whether it shares the file. Files are assigned probabilistically to peers at initialization based on file popularity and the content categories the peer is interested

[1] The performance of a connection can be measured by the number of responses received through the connection.

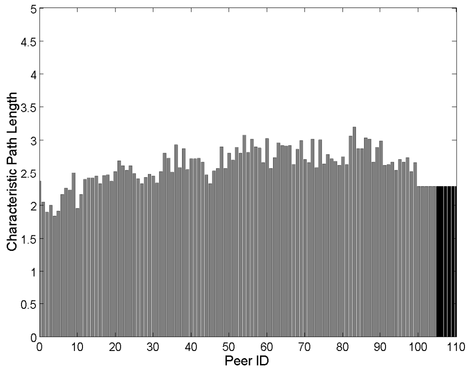

Figure 3: Average Characteristic Path (Cycle 0).

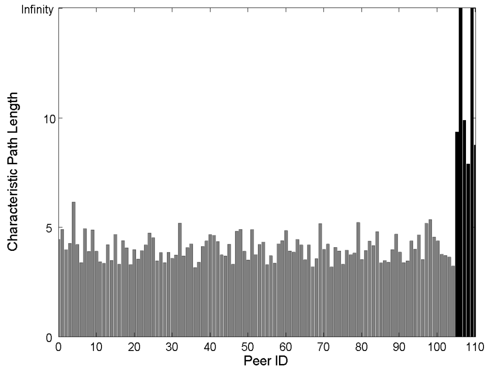

Figure 4: Average Characteristic Path (Cycle 95).

in. This implies that peers are likely to share popular files, even if they have few files. Distributions used in the model are taken from measurements in real-world P2P networks [4].

Simulation execution. The simulation of a network proceeds in query cycles. In each query cycle, peer i in the network may be actively issuing a query, be inactive, or even down and not responding to queries passing by. Upon issuing a query, a peer waits for incoming responses, selects a download source among the nodes that responded and starts downloading the file. The latter two steps are repeated until a peer has properly received a good copy of the file or runs out of responses.

The base settings that apply for most of the experiments are summarized in Table 2. The settings represent a fairly small network to make our simulations tractable. However, we have experimented with larger networks and our conclusions continue to hold. That is, schemes that do well in a small setting, proportionately do as well even when the network is scaled up.

Metrics. We are interested in measuring the efficiency, security, and incentives of adaptive P2P topologies. We use the following two metrics to measure the efficiency of the network: the number of messages passed and the number of authentic responses per query. We define the total number of messages to be the total number of queries, responses and file downloads in the network. We define the number of authentic responses as those query responses that would lead to the download of an authentic file. The number of inauthentic responses per query and the average characteristic path length to malicious peers are used to evaluate the security of our protocol. The number of inauthentic responses per query is defined as the total number of query responses that would lead to the download of an inauthentic file. The characteristic path length to a peer is the average hop-count to the peer from all other peers in the network. We use the average characteristic path length to active peers and freeriders to measure the incentives that our protocol gives for participation.

A peer's search query adds 1 to the number of messages passed at each propagation point. Each response to a search query increments the number of messages passed by 1. The number of inauthentic responses per query represents the total number of query responses originating from a malicious peer. All other response messages increase the number of authentic responses by 1. A peer's characteristic path length is the average hop-count to the peer from all other peers in the network.

4.2 Malicious peers move to fringe

4.2.1 Principle

One common attack on P2P networks today is the inauthentic file attack, where malicious peers upload corrupt, inauthentic, or misnamed content onto the network. Since the APT protocol disconnects from peers that upload unsatisfactory files, malicious peers eventually move to the fringe of the network. Moving malicious peers to the fringe of the network has been shown to be a very effective strategy in combatting certain types of attacks [8].

4.2.2 Experiments

We use the *characteristic path length* to illustrate how the malicious peers move to the fringe of the network using the APT protocol. We define the average characteristic path length to a peer i as the average of the shortest path lengths between all other peers in the network and peer i.

$$cpl_i = \frac{1}{|P\backslash i|} \sum_{j \in P\backslash i} \text{shortestPath}(i, j)$$

where $P\backslash i$ is the set of all peers in the network except peer i.

In Figure 3, the x-axis represents the peer id and the y-axis is the characteristic path length to that peer. The data was extracted from the Query Cycle Simulator [3] at Cycle 0 which describes a power-law topology. The x-axis is segmented into two regions each reflecting a type of peer. The gray region represents peers in the network that are considered good and the black corresponds to malicious peers. The average characteristic path length to any peer is about 2.5 hops. During network bootstrap, malicious peers were encoded to aggressively connect to good peers, and thus the average characteristic path length for malicious peers is 2.3 hops.

Figure 4 shows the characteristic path length after 95 simulated cycles. The overall average characteristic path length is 4.94 hops. Notice that the average characteristic path to a malicious peer (9.84 hops) is much higher than that to a good peer (4.02 hops). The separation in characteristic path length shows that as good peers drop connections based on inauthentic downloads, malicious peers are pushed into the fringe of the network.

Figure 5 plots the evolution of the average characteristic path length for good peers and malicious peers. Beyond the initial 100 cycles, no path exists from a good peer to a malicious peer. The divergence in the characteristic path lengths of good and malicious peers indicates that these two peer types become distinct over time. A good peer can take advantage of this result by limiting the scope

Network	# of good peers	100
	# of malicious peers	10
	# of initial neighbors of good peers γ	3
	# of initial neighbors of malicious peers γ	5
	# Maximum number of allowed connections τ	20
	# Time-to-live for query messages	3
Content Distribution	# of distinct files at good peer i	file distribution in [4]
	set of content categories supported by good peer i	Zipf distribution over 20 content categories
	# of distinct files at good peer i in category j	uniform random distribution over peer i's total number of distinct files
	top % of queries for most popular categories and files malicious peers respond to	20%
	% of time peer i is up and processing queries	uniform random distribution over [0%, 100%]
	% of up-time good peer i issues queries	uniform random distribution over [0%, 50%]
Peer Behavior	% of download requests in which good peer i returns inauthentic file	5%
	% of download requests in which malicious peer i returns inauthentic file	100%
	% of queries malicious peer i responds to	20%
	download source selection algorithm	random
Simulation	# of simulation cycles in one experiment	100-200
	# of experiments over which results are averaged	5

Table 2: Simulation settings

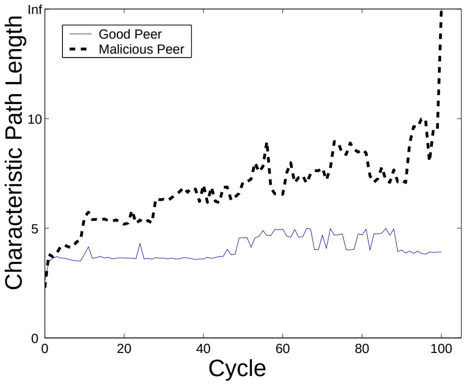

Figure 5: Characteristic Path Length (Cycle 0-100).

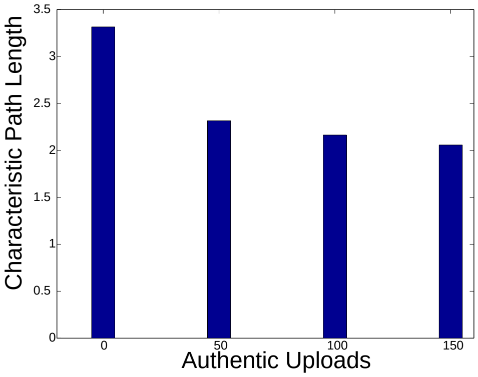

Figure 6: Characteristic Path Length based on file uploads.

of its search query so that a malicious peer never receives its query. Section 4.5 describes a strategy to avoid malicious query responses by reducing the TTL for a peer's search query.

4.3 Freeriders move to fringe

4.3.1 Principle

When peer i finds a peer that it is likely to download from, it connects to that peer and disconnects from its neighbor with the lowest local trust score. Since a freerider has a local trust score of 0, freeriders move to the fringe of the network as well.

4.3.2 Experiments

As shown in Figure 4, certain good peers have higher than average characteristic path lengths. These peers are freeriders, and their path length reflects the fact that it is not advantageous to connect to peers that do not share any files. Notice also that the path length to any freerider is still shorter than the path length to any malicious peer. Since freeriders are not actively uploading inauthentic files, other peers remain connected to them until they find a more desirable peer.

Figure 6 is an equidistant histogram of the characteristic path length for all good peers after a 100 cycle simulation. The x-axis lists four buckets representing the number of uploads a peer has provided. Peers with no uploads (freeriders) fall into the bucket labeled 0, while a peer that has uploaded N authentic files falls into bucket $B = \lceil \frac{N}{50} \rceil \times 50$. For example, peer i with $N = 60$ uploaded files is counted in bucket $B = 100$. The figure shows that freeriders take an average of 3.4 hops to reach while peers with uploads take around 2 hops. Therefore, the peers that do not share files are given a narrow view of the network.

4.4 Active peers are rewarded

4.4.1 Principle

Active peers have more opportunities to connect to other active peers, since their local trust scores will be high. For example, an active peer i with $\tau = 3$, may have connections with local trust values of 10, 6, and 4. An inactive peer will not have the opportunity to connect to peer i, while an active peer that has provided more

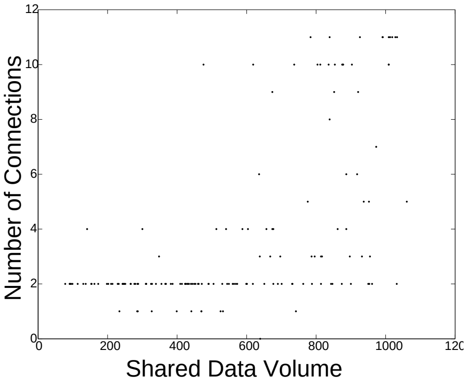

Figure 7: Connections vs. Shared Data Volume.

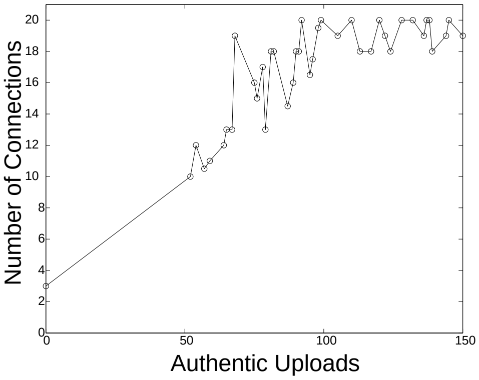

Figure 8: Connections vs. Authentic Uploads.

than 4 authentic files to that peer will. Thus, the reward for being an active peer is the opportunity to connect directly to other active peers.

4.4.2 Experiments

The number of connections a peer has relative to the number of files that the peer shares is plotted in Figure 7. To make the results more apparent, the network was set up to limit the max number of connections τ to 11 (in this experiment only). Notice the trend that as the number of shared files increases so does the number of connections. This shows that peers that share many files are rewarded with a wider view of the network.

A peer is rewarded for sharing high quality files. In Figure 7 there are peers that share large amounts of data but receive only few connections in return. These peers were set to share many unpopular files. Since connections are derived from downloads, peers are not rewarded for sharing vast amounts of unpopular data.

One indication of a peer that shares popular files is in the number of executed uploads. Figure 8 plots the number of connections a peer has relative to the number of authentic uploads it has performed. The graph clearly shows that peers that actively upload files are rewarded with a wide view of the network.

Figure 9 shows the fraction of authentic responses received by a peer compared to the total number of authentic files it uploads. For example, on average 43% of all query responses returned to a peer that executed 50 uploads originated from a good peer. Notice that peers with more than 65 authentic file uploads receive only authentic responses. This shows that an active peer is further rewarded with connections to peers with high fidelity. Section 4.5 describes clusters or neighborhoods that contain peers with similar interests and quality of service. These clusters become unreachable to a malicious peer residing on the fringe of the network.

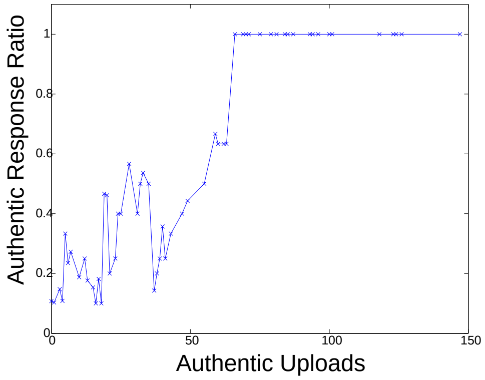

Figure 9: Authentic Response Ratio.

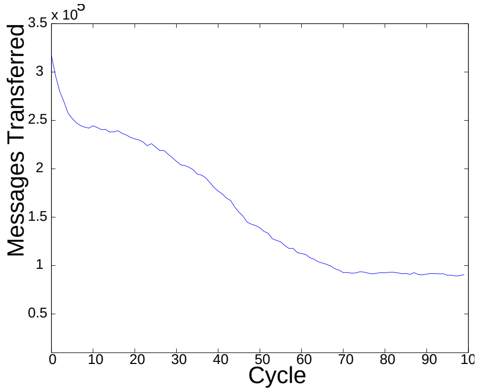

Figure 10: Network Traffic.

4.5 Efficient Topology

4.5.1 Principle

In the APT protocol, connections are made based on download history, so peers connect to peers that share their interests. Eventually, clusters of like-minded peers form and are connected to other clusters by hub peers which are active and have many interests. These characteristics describe a *small-world network* [14]. Such a network is sparsely connected, but has a short average characteristic path length and a high cluster coefficient. A small-world network thus allows a wide view of the network with low communication overhead.

4.5.2 Experiments

The data shown in Figures 10, 11, and 12 is extracted from the same simulated session. In this experiment, peers follow the APT protocol and messages are set with a 3 hop TTL. Furthermore, malicious peers were encoded to aggressively flood the network with their query responses by responding to all queries they receive. Throughout the experiment, the average overall characteristic path length to a peer was around 2.9 hops and the network diameter was between 7 and 9 hops.

The total number of query and response messages transferred during a given cycle is plotted in Figure 10. The decline in network

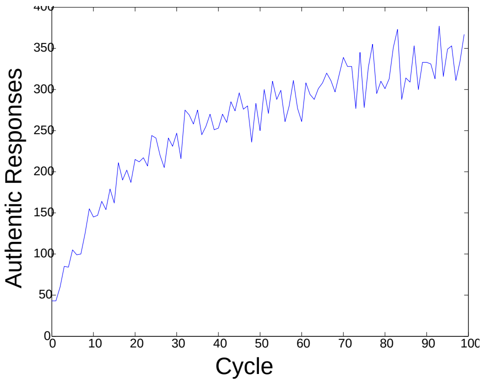

Figure 11: Authentic Responses.

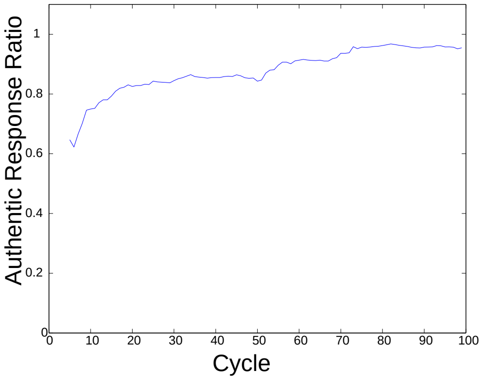

Figure 12: Authentic Response Ratio.

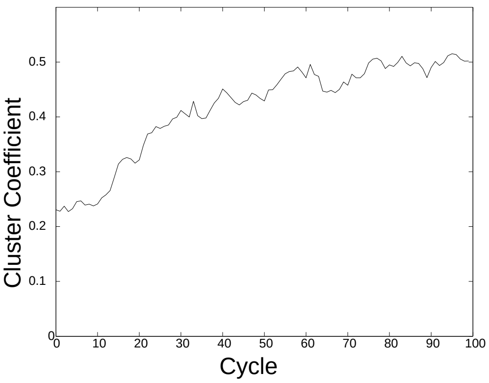

Figure 14: Cluster Coefficient.

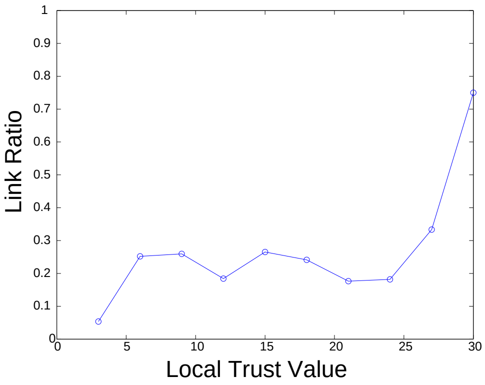

Figure 15: Link Ratio based on Local Trust.

traffic can be attributed to good peers dropping connections to malicious peers, since malicious peers are unable to respond to queries as they lose their grasp on the network. Figure 11 shows that the number of authentic responses returned to a peer increases over time. The result supports the fact that peers connect to other good peers that share similar content, and consequently more queries are being answered. At 70 cycles, the average number of messages transferred per cycle is about 100,000, which is 1/3 of the number of messages transferred during the first cycle. Furthermore, Figure 12 shows that 97% of all responses after 70 cycles are authentic.

A small TTL setting takes advantage of the fact that malicious peers move to the fringe of the network. The number of malicious responses to a query during a given cycle under three separate TTL settings, MAX_TTL = 4, MED_TTL = 3, and MIN_TTL = 2 is given in Figure 13. Since a malicious peer is on average 3 hops further away than a good peer, there is a significant decline in malicious query responses when using MED_TTL or MIN_TTL. Therefore, setting the TTL to be less than the overall average characteristic path can be effective in reducing the number of malicious query responses.

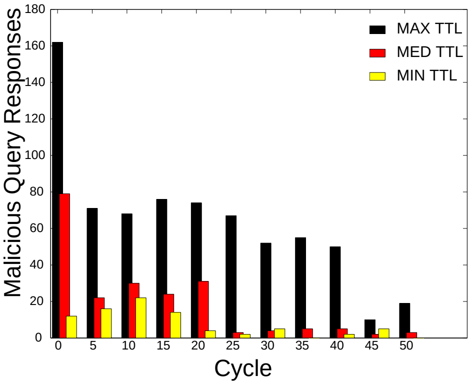

Figure 13: Malicious query responses based on TTL.

In the previous figures we have shown that even as the number of messages passed decreases, peers still receive good quality of service. One reason is that malicious peers are moved to the fringe of the network, thereby decreasing the unnecessary message overhead caused by malicious responses. Another reason is that peers are organized into clusters of peers that share similar interests, so the files that are of interest to a peer are likely located nearby.

We use the average cluster coefficient to quantify the clustering effect of the APT protocol. The local cluster coefficient C_i for peer $i \in P$ with k_i neighbors is defined as follows:

$$C_i = \frac{2E_i}{k_i(k_i - 1)}$$

where E_i is the actual number of edges that exist between the k_i neighbors. Using this definition, if peer i and all peers in $N(i)$ form a clique then $C_i = 1$. The average cluster coefficient is then defined as C_i averaged over all peers in P.

Figure 14 measures the average cluster coefficient for all peers in the network. The increase in cluster coefficient from the initial power-law topology at cycle 0 shows that clusters form using the APT protocol. These clusters consist of peers that have had many positive interactions and thus share similar content interests.

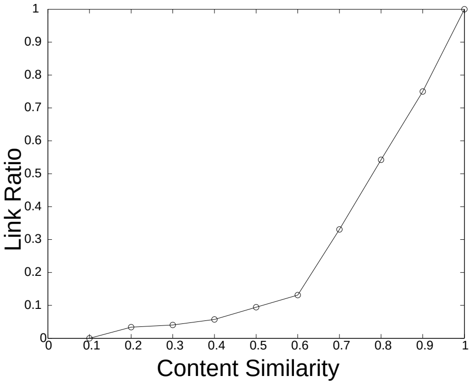

Figure 16: Link Ratio based on Content Similarity.

Figures 15 and 16 demonstrate this clustering effect by measuring the link ratio. The link ratio is defined as the percentage of peers assigned a particular value that are also neighbors. Figure 15 measures the link ratio with respect to local trust values. For example, edge (i, j) exists in 29% of all cases where peer i has a local trust value of 25 for peer j. Notice the low ratio for local trust scores below 5 and the high ratio for scores above 25. Since local trust scores define successful transaction, the plot shows that a peer's connections are determined by its transactions with other peers.

In Figure 16, the link ratio with respect to the similarity of the peers' content is plotted. The content similarity of peer i and peer j is defined as,

$$S(i, j) = \frac{1 - (\sum_{t=1}^{n} |c_{it} - c_{jt}|)}{2}$$

where n is the total number of content categories and c_{it} the number of files peer i shares in content category t normalized by the total number of files shared by peer i. Using this definition, the value $S(i, j) = 1$ means that peer i and peer j have the same content distribution vectors c. That is, $c_{it} = c_{jt}$ for each content category t. The graph shows that clusters form out of peers having the majority of their content similar.

Clustering based on content similarity increases the probability that a query is answered within a few hops. Moreover, responses to queries from closer peers can be trusted more than responses from further away. By lowering its query horizon, a peer can take advantage of the clustering that occurs under the APT protocol, and thereby increase the overall network efficiency.

5 Threat Scenarios

In previous sections it was assumed that malicious peers simply flood the network with inauthentic files in an attempt to subvert the system. We now evaluate the performance of our protocol in preventing malicious connections (connections leading to a malicious peer) and inauthentic file downloads under a variety of threat scenarios.

5.1 Threat Model A

In this model, malicious peers respond to all queries except those issued by a neighbor. If a malicious peer is chosen as a download source, it will upload an inauthentic file.

Node Model. Let $G \subset P$ be a set of good peers and m be a malicious peer directly connected to all peers in G. The peers in G also hold connections to peers in $P \setminus \{G \cup m\}$ as shown in Figure 17.

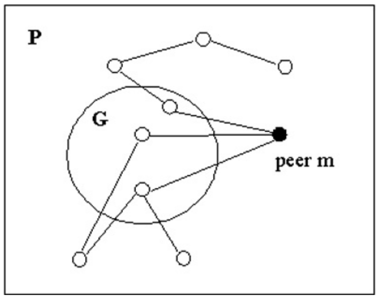

Figure 17: Node Model for Threat Models A and B.

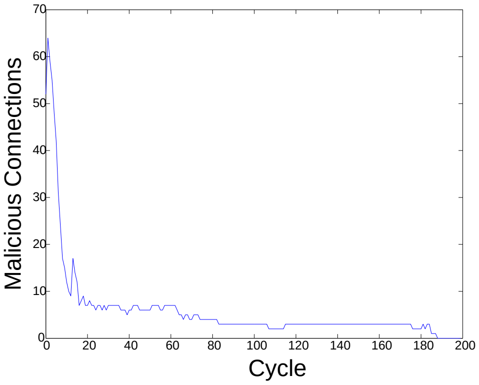

Figure 18: Threat Model A Malicious Connections.

Query Model. All queries received by peer m are handled by the following two cases.

1. If the query originated by some peer in G the peer m does not respond.

2. If the query originated by some peer in $P \setminus \{G \cup m\}$ then peer m responds according to the default settings listed in Table 2.

After receiving and potentially responding, peer m drops the query. Peer m does not generate search queries of its own.

Given this query model, peer m only uploads inauthentic files to peers in $P \setminus \{G \cup m\}$. Consequently, peer m avoids an inauthentic file detection by a neighboring peer in G. That is, peer $i \in G$ will have a zero value local trust score for peer m. The goal of the malicious peer is to prevent connection drops due to negative local trust scores.

Threat Model A is naturally combatted by the APT protocol. At some point peer $i \in G$ begins to notice numerous peers disconnecting from it, and so assumes it is relaying queries to a malicious peer. The connection loss triggers peer i to replace a low trust value connection with some random connection, as outlined in Algorithm 2. The question now is whether peer i makes the right decision by dropping the connection to peer m.

To answer this observe that a malicious peer behaves as a freerider, which makes it a less than desirable connection. According to Algorithm 2, after a connection loss, a peer drops its connections to peers with low local trust scores and reconnects to random peers. Since malicious peer m will have a local trust score of 0, it is more likely to be dropped.

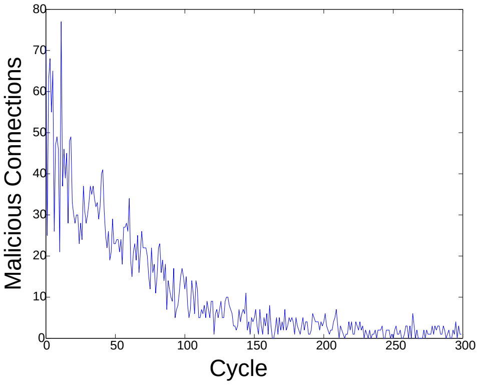

Figure 19: Threat Model B Malicious Connections.

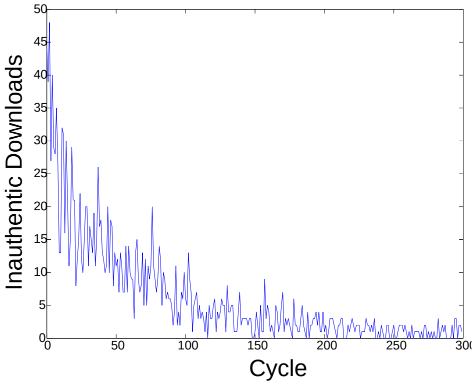

Figure 20: Threat Model B Inauthentic Downloads .

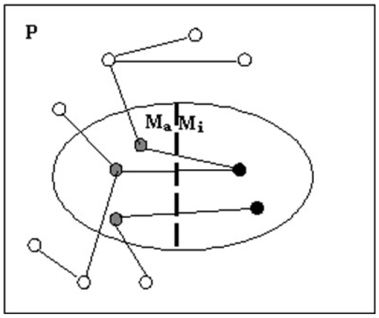

Figure 21: Node Model for Threat Model C.

Figure 18 shows the progression of malicious connections for a simulated session set up for Threat Model A. The early drop in the number of malicious connections is the result of swapping out low value connections to malicious peers. The short life of a malicious connection evenly distributes the inauthentic content placed in the network by a malicious peer. This even distribution works against the malicious peer, since more peers are made aware of its intent. As the simulation moves forward, malicious peers encounter resistance in making new connections. After cycle 190, all connection requests made by malicious peers are denied by good peers. Thus Threat Model A is handled well by the APT protocol.

5.2 Threat Model B

In Threat Model B the malicious peer entices its neighbors with a few authentic files in order to gain some local trust and evade connection loss. The malicious peer will likely want to minimize the number of authentic file uploads due to the counterproductive cost. That is, the purpose of a malicious peer is to disrupt the sharing of authentic files, not support it.

Node Model. Let $G \subset P$ be a set of good peers and m be a malicious peer directly connected to all peers in G. Peers in G are then connected to peers in $P \setminus \{G \cup m\}$ as shown in Figure 17.

Query Model. Peer m responds to queries as specified by the default settings listed in Table 2 subject to the constraint that no more than 10% of all uploads are to peers in G. Instead of propagating the query, peer m drops it. Peer m does not generate a search query of its own.

Under Threat Model B, peer m will service authentic files to peers in G and upload inauthentic files to peers in $P \setminus \{G \cup m\}$. By servicing peers in G with authentic files, peer m increases its chances of maintaining its connection to $P \setminus \{G \cup m\}$, via G.

Threat Model B is thwarted by the connection trust extension described in Section 3.3. According to Algorithm 2 the peers in G will eventually lose their connections with peers in $P \setminus \{G \cup m\}$ due to poor connection trust scores caused by relaying messages to peer m. The loss of the connections to $P \setminus \{G \cup m\}$ lessens the value of peers in G to peer m. Peer m will likely try to form connections with peers in $P \setminus \{G \cup m\}$. However, reconnecting becomes increasingly difficult since peer m will have built up negative local trust scores with peers in $P \setminus \{G \cup m\}$. Assuming that the majority of uploads by peer m are inauthentic (as in any productive malicious attack), peer m will eventually lose the ability to connect to good peers in P.

Figure 19 plots the succession of malicious connections in a simulated session under Threat Model B. The graph starts out similar to that shown in Figure 18. However, in later cycles the number of malicious connections remains around 5, while in Figure 18 all malicious peers are completely disconnected. Nevertheless, at this point, most queries never reach the malicious peers since the average path length to them is 4 hops more than to a good peer. Consequently, as shown in Figure 20, the number of inauthentic downloads is negligible.

A considerable amount of noise is present in Figures 19 and 20, which can be attributed to the volatile nature of connections to malicious peers and the authentic files uploaded by malicious peers. The noise attrition in both figures is caused by the increased resistance toward malicious connections.

Peers that remain connected to malicious peers do so because they are serviced authentic files. These peers are tagged as bad connections because of poor connection trust scores. Dropping peers with low connection trust scores closes the conduit to the malicious peers. The malicious peers may then seek other, more fruitful connections. However, poor local trust scores make it difficult for malicious peers to form new connections, and thus they are trapped in their current connections. If malicious peers are not able to continually satisfy queries, then by Algorithm 2, the connections will be severed due to void downloads. Although not as effective as in Threat Model A, the APT protocol is able to prevent most inauthentic downloads and keep malicious peers at bay.

5.3 Threat Model C

In Threat Model C, a set of malicious peers that upload inauthentic files are connected to another set of malicious peers that provide authentic files. The malicious peers serving authentic files maintain the connection to the rest of the network while the others flood the network with inauthentic files.

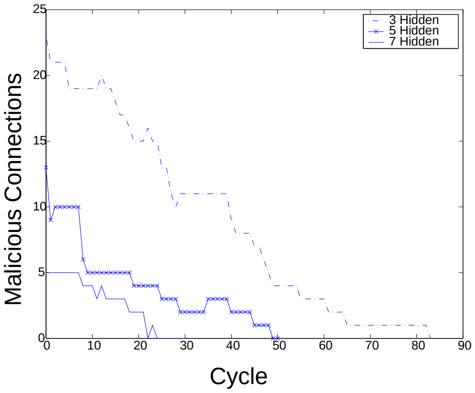

Figure 22: Threat Model C Malicious Connections.

Node Model. Figure 21 illustrates the node model for Threat Model C. Let $M \subset P$ be the set of all malicious peers in P. We partition M into two disjoint sets M_a and M_i. Peers in M_i are connected to all peers in M_a and malicious peers in M_a also maintain connections to peers in $P \setminus \{M_a \cup M_i\}$.

Query Model. Both sets of malicious peers will respond to queries as specified in Table 2. The search queries received by peers in M_a are only forwarded to peers in M_i, while peers in M_i do not propagate the queries they receive. Neither of the malicious sets will generate a search query of their own.

After being chosen as a download source, peers in M_a upload authentic files, while those in M_i upload inauthentic files. The motivation behind Threat Model C is two-fold:

1. Malicious peers no longer depend on servicing a large number of requests made by a single good peer.

2. The connection trust is inherently weaker than local trust at preventing malicious connection. This threat model exploits that weakness by using malicious peers as conduits that purely upload authentic files to the rest of the network.

The main advantage of this threat model over the previous is that peers in M_a are never assigned a negative local trust score. Therefore, good peers rely entirely on negative connection trust scores to stave off malicious connections from peers in M_a.

Figure 22 shows three simulated sessions using different partitions of malicious peers in M_i (hidden peers) and M_a, where $\mid M_i \mid \ + \ \mid M_a \mid = 10$. As expected, the malicious attacks containing more hidden peers have a shorter connection life to peers in $P \backslash \{M_a \cup M_i\}$. Notice that in all cases malicious peers are eventually discovered and disconnected from the network. Therefore, the connection trust is sufficient in eliminating malicious peers from the network.

6 Related Work

Related topologies have been proposed in [7], [6], and [11]. In [7], a peer connects to peers initially at random, and disconnects when it becomes overloaded. The peers that are disconnected will then connect to other peers. The connections in [7] can be search links (through which search information is sent) or index links (through which indexing links are sent). Lv et al. present a similar scheme in [11]; the differences here are that there is only one type of link and that each peer tracks its neighbors capacities and suggests a replacement peer after it breaks a connection. The SLIC mechanism proposed in [6] does not add or break connections, but rather allows each peer to rate its neighbors, and use these ratings to control how many queries from each neighbor to process and forward.

7 Conclusion

We have shown a simple protocol for the formation of adaptive P2P topologies. The resulting topologies are highly efficient, robust to malicious attacks, and provide built-in incentives and punishments that are consistent with positive peer contribution. As each peer chooses its neighbors, clusters of peers with similar interests and quality of service form. The creation of communities containing congenial peers have deep implications towards the personalization of P2P networks.

References

[1] S. Kamvar, M. Schlosser, and H. Garcia-Molina. The EigenTrust Algorithm for Reputation Management in P2P Networks. In *WWW 2003*, 2003.

[2] A. Crespo and H. Garcia-Molina. Semantic Overlay Networks. *Submitted for publication*, October 2002.

[3] M. Schlosser, T. Condie, and S. Kamvar. Simulating a File-Sharing P2P Network. In *SemPGRID03*, 2003.

[4] S. Saroiu, P. K. Gummadi, and S. D. Gribble. A measurement study of peer-to-peer file sharing systems. In *Proceedings of Multimedia Computing and Networking 2002 (MMCN '02)*, San Jose, CA, USA, January 2002.

[5] S. Kamvar, M. Schlosser, and H. Garcia-Molina. Incentives for Combatting Freeriding on P2P Networks. In *Euro-Par*, 2003.

[6] Q. Sun and H. Garcia-Molina. SLIC: A Selfish Link-based Incentive Mechanism for Unstructured Peer-to-Peer Networks. Technical Report, Stanford University, 2003.

[7] B. F. Cooper and H. Garcia-Molina. Ad hoc, self-supervising peer-to-peer search networks. Technical Report, Stanford University, 2003.

[8] N. Daswani and H. Garcia-Molina. Query-Flood DoS Attacks in Gnutella. In *ACM CCS*, 2002.

[9] Gnutella website. http://www.gnutella.com.

[10] B. Yang, P. Vinograd, and H. Garcia-Molina. Evaluating GUESS and Non-Forwarding Peer-to-Peer Search. In *ICDCS*, 2004.

[11] Q. Lv, S. Ratsnasamy, and S. Shenker. Can Heterogeneity Make Gnutella Scalable? In *First International Workshop on P2P Systems*, 2002.

[12] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for Internet applications. Technical Report TR-819, MIT, March 2001.

[13] B. Zhao, J. Kubiatowicz, and A. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report UCB/CSD-01-1141, Computer Science Division, U. C. Berkeley, April 2001

[14] D.J. Watts and S.H. Strogatz Collective dynamics of 'small-world ' networks. *Nature, pp. 393*, 1998.

[15] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi Optimization by Simulated Annealing. *Science, Number 4598*, 13 May 1983.