A Case for P2P Delivery of Paid Content

Alex Sherman, Angelos Stavrou, Jason Nieh, Cliff Stein and Angelos D. Keromytis
Department of Computer Science, Columbia University, New York, NY
{*asherman, angel, nieh, cliff, angelos*}@*cs.columbia.edu*

Abstract

P2P file sharing provides a powerful content distribution model by leveraging users' computing and bandwidth resources. However, companies have been reluctant to rely on P2P systems for paid content distribution due to their inability to limit the exploitation of these systems for free file sharing. We present TP2, a system that combines the more cost-effective and scalable distribution capabilities of P2P systems with a level of trust and control over content distribution similar to direct download content delivery networks. TP2 uses two key mechanisms that can be layered on top of existing P2P systems. First, it provides strong authentication to prevent free file sharing in the system. Second, it introduces a new notion of trusted auditors to detect and limit malicious attempts to gain information about participants in the system to facilitate additional out-of-band free file sharing. We analyze TP2 by modeling it as a novel game between malicious users who try to form free file sharing clusters and trusted auditors who curb the growth of such clusters. Our analysis shows that a small fraction of trusted auditors is sufficient to protect the P2P system against unauthorized file sharing. Using a simple economic model, we further show that TP2 provides a more cost-effective content distribution solution, resulting in higher profits for a content provider even in the presence of a large percentage of malicious users. Finally, we implemented TP2 on top of BitTorrent and use PlanetLab to show that our system can provide trusted P2P file sharing with negligible performance overhead.

1. INTRODUCTION

Peer-to-Peer (P2P) file sharing is a powerful and cost-effective content distribution model due to its ability to leverage the participating users' uplink bandwidth. Popular examples include BitTorrent [4], Napster [17] and Kazaa [13]). However, content providers that offer copyrighted files online typically rely on direct download to distribute the paid content. Companies such as Apple iTunes [12], Sony, and Time Warner among others, distribute content either from their website directly or via contracted content delivery networks (CDNs) such as Akamai [2]. These companies are hesitant to rely on currently available P2P systems for paid content distribution as free-of-charge file sharing is inherent in current P2P models, as evidenced by the proliferation of file sharing communities (*e.g.,* Xbox-sky [23] and Red Skunk Tracker [20]) and of search engines that facilitate the identification of P2P networks which potentially share such content (*e.g.,* TorrentSpy [21] and PirateBay [18]). However, the use of P2P techniques has the potential to significantly reduce the distribution costs for the content providers, because of the lower bandwidth capacity/use costs, and/or lower Content Delivery Network (CDN) fees. Thus, P2P represents a profitable opportunity for distributing paid content. For P2P techniques to be adopted for authorized paid content distribution, the system must provide a cost/benefit trade-off that is better (*i.e.,* profit is higher) than in the direct download case.

We present TP2, a system that provides the cost-effective distribution capabilities of a decentralized P2P approach while maintaining sufficient trust and security parameters for paid content distribution. TP2is a layer that can be added to existing P2P file sharing systems to support paid-content distribution by imposing significant and configurable barriers against malicious exploitation of the system for free content sharing. First, it adds strong authentication and authorization to a P2P system such that users can only download content after payment. Second, it introduces the notion of *trusted auditors* (TAs) into the system. TAs are P2P nodes controlled by the content provider. Their purpose is to detect any type of malicious activity, including deviations from the normal protocol implementation, port probing, malformed messages, attempts by unauthorized nodes to access content without proper credentials, and any effort to create an out-of-band communication with another node that is participating in the system. TAs are our sentinels: they can assume the role of a source, a destination or even of a malicious user luring other malicious nodes to either probe them or to respond to their probes. When they detect malicious behavior, a variety of measures may be taken. In the simplest case, offending peers are banned from the P2P system and moved to another, isolated system for all future downloading, where they must pay for the bandwidth they consume.

TP2 provides the following key properties:

•*Content Protection.* Our system enforces strong authentication and encryption for all P2P communications, which inhibits unauthorized nodes from connecting to other P2P participants. Since a malicious node cannot obtain free content from another, non-malicious participant, its only hope is to discover other malicious nodes who are willing to trade with her out-of-band.

•*Inherent Trust.* TAs bound the ability of malicious nodes to discover other malicious nodes for illegal file sharing and trading. As a result, the content provider has *trust* guarantees that the P2P system will not be exploited by malicious nodes. Just like in a direct-download system, we cannot prevent a user who legally downloads content from sharing it out of band. However, with the use of strong authentication and TAs, we can protect the P2P system itself from being exploited for free content downloads or information scavenging for future illegal trading.

•*Bandwidth Savings.* Our system leverages P2P bandwidth for file downloads and thus provides tremendous scalable bandwidth and infrastructure savings. While the content provider needs to provision bandwidth for the TAs, we show that TAs are only a small fraction of the overall P2P network and thus the total bandwidth conservation is still significant.

We analyze TP2 by modeling it as a novel game between malicious users who try to form free file-sharing clusters and trusted auditors who curb the growth of such clusters. Our analysis shows that even a small fraction of trusted auditors is sufficient to protect the P2P system against unauthorized file sharing. Even assuming that 50% of the P2P users in the system are maliciously attempting to form clusters with other users, and 10 times as many malicious users as TAs, TP2 can detect 99% of the malicious users in steady state operation, can limit 80% of the malicious users from forming clusters, and only 10% of the malicious users are able to achieve a cluster with more than one other user. If we increase the number of TAs such that there are only 5 times as many malicious users as TAs, TP2 can limit 90% of the malicious users from forming clusters, and less than 3% of the malicious users are able to form a cluster with more than one other user.

Using a simple economic model, we further show that TP2 provides a more cost-effective content distribution solution, resulting in higher profits for a content provider even in the presence of a large percentage of malicious users. Even assuming that 50% of the P2P users in the system are maliciously attempting to form clusters with other users, TP2 achieves more than 80% higher profit per download than a direct download system assuming conservative profit numbers and bandwidth costs. Furthermore, accounting for the bandwidth costs of TAs and the lost revenue due to malicious users, TP2 achieves more than 90% of the potential profit of a pure P2P system with zero malicious users.

We implemented TP2 on top of BitTorrent to demonstrate that our system can provide its functionality in an existing, widely-used P2P system with only modest modifications. We deployed our TP2 enhanced BitTorrent prototype on PlanetLab and present some experimental data comparing its performance to an unmodified BitTorrent system running on the same PlanetLab nodes. Our results show that TP2 can provide its trusted paid content distribution functionality while imposing negligible performance overhead.

2. RELATED WORK

As broadband Internet access becomes more prevalent, digital content stores such as Apple Itunes and Amazon have begun to distribute richer digital content over the Internet, such as TV series episodes and full-length movies. Since each download requires significant bandwidth, these stores typically contract Content Delivery Networks (CDNs) to distribute their content. Commercial CDNs include Akamai [2], Limelight [15] and VitalStream [22]. CDNs typically operate thousands of distributed servers deployed in various networks and ISPs. In addition to offering vast amount of scalable bandwidth, CDNs can enforce appropriate security measures on behalf of a digital store, such as authorization of customers and encryption of served content. However, the price paid to CDNs for their services is quite high. Market research [1] suggests that digital media vendors spend 20% of their revenue on infrastructure costs for serving content. While free academic alternative CDNs such as Coral [10] and CoDeeN [11] exist, these systems are typically limited in their deployment and the amount of bandwidth they are allowed to use.

An alternative powerful distribution model is Peer-to-Peer (P2P) systems such as BitTorrent [4], Napster [17] and Kazaa [13]) among others. No extra contracted bandwidth is required as users leverage one another's upload links to "share" content. BitTorrent is perhaps the most popular of these systems, and many analytical works [25, 9, 14, 8] have shown the high efficiency and scalability characteristics of BitTorrent. Unfortunately, it is very difficult to implement proper authentication and authorization in such P2P systems to distribute copyrighted or paid content only to intended recipients. In fact, such systems typically implement a searchable directory of available content including copyrighted material. For this very reason, distributors of paid content shy away from P2P distribution models.

Various companies have attempted to address this problem with P2P systems. MoveDigital [16] implements a gateway in front of a P2P system to allow only authorized users access. However, once inside, users can leverage the system for further illegal sharing without limitations. For example, if a user can learn the IP addresses of other users inside the system, she can start sharing content with those users directly for free, bypassing the up-front payment. Moreover, users might choose to participate in the P2P system and pay to download files to gain knowledge about other participants that have similar interests. Then, they can easily form another, private P2P community, a darknet [3], for free future exchange of similar content.

Another approach is Avalanche [5], which uses network coding to encode exchanged blocks and relies on a proprietary protocol to attempt to prevent malicious use through security by obfuscation. However, if the system is hacked such that malicious nodes can participate in the system, there are no effective mechanisms to prevent its exploitation for free file sharing. In contrast, TP2 is designed explicitly to guard against such free file sharing using an open system architecture that is resistant to exploitation even in the presence of malicious nodes. Note that the trusted auditors used in TP2 are owned and managed by the content provider, and are unlike reputation-based systems [24] where users simply rate each other such that the resulting ratings may not be trustworthy.

3. ARCHITECTURE

We designed the TP2 architecture as an extra layer added to a common P2P system. This layer primarily consists of

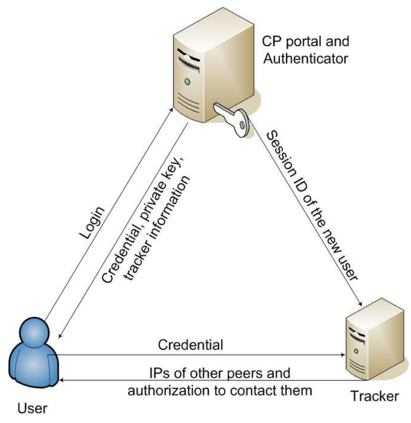

CP portal and Authenticator
Login
Credential, private key, tracker information
Session ID of the new user
Credential
IPs of other peers and authorization to contact them
User
Tracker

Figure 1: To purchase a file, the user logs in on the portal, pays, obtains a signed credential and contacts the tracker for the purchased file.

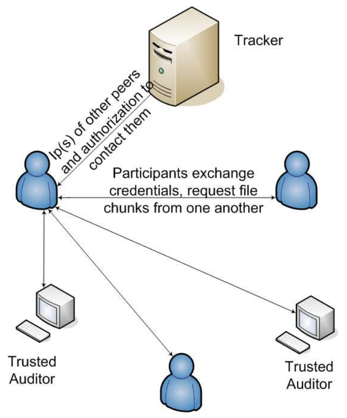

Tracker
Ip(s) of other peers and authorization to contact them
Participants exchange credentials, request file chunks from one another
Trusted Auditor
Trusted Auditor

Figure 2: Users authenticate one another and request file pieces. A fraction of trusted auditors is mixed in among the file-sharing peers.

components that enforce stronger security and trust parameters in the system: the authenticator service and trusted auditors. While TP2 layer that can be added to virtually any common P2P system, we use BitTorrent as the underlying P2P system for the purpose of the discussion in this paper and our implementation. We chose BitTorrent given its popularity, open implementation, and its very efficient file-swarming technique where users share individual blocks of a given file.

A brief overview of BitTorrent: the goal is to distribute a file as fast as possible to all connected peers. To achieve this, BitTorrent splits the file (such as a digital movie) into a number of chunks. Participating *peers* exchange individual chunks of the file using a *file swarming* approach. The swarming algorithm is fully distributed and nodes use it to decide from which peers they are going to request their missing chunks. In addition, in each file-sharing instance there are one or more *Seeds* present. Seeds are peers that have all the chunks of the given file. The party that advertises the content typically initializes one or more *Seeds* with the full content of the file. A file-sharing instance also contains a *Tracker* that tracks all participating peers. A peer joins the system by contacting the Tracker. It receives a set of usually up to 50 IP addresses of other participating Peers. The Peer then exchanges chunks of the file with the other Peers and periodically updates its progress to the Tracker via *announce* messages.

3.1 System Overview and Usage

When the user decides to purchase a content file for the first time, she registers at the content provider's portal. She picks a username and a password and enters her payment information (*e.g.,* credit card number). She then downloads a software client through the portal that allows her to browse for files, purchase content and perform P2P downloads.

Each time a user makes a purchase from the content provider (CP) at the CP's portal, she is authorized the user to perform the download by giving the user a verifiable token (signed credential). The authenticator also generates security parameters for the user to be used for secure communication during its download *session.* (We sometimes refer to the file-sharing instance as a download session.) These parameters are described in Section 3.3.

The user is then directed to a tracker that manages a file-sharing instance for purchased file. The tracker validates that the user is authorized to perform the download by verifying her token. The user's interaction with the authenticator and the tracker is depicted in Figure 1.

As in BitTorrent, the tracker gives out or *assigns* a set of other clients or *peers* to the new client. The client shares pieces of the purchased file with her assigned peers using BitTorrent's *file-swarming* approach. TP2 differs significantly from BitTorrent in the assignment of the peers. The TP2 tracker ensures that a certain fraction of the peers that it assigns are *trusted peers*, as shown in Figure 2. Trusted peers or *trusted auditors* (TAs) are special peers who, in addition to participating in a download session, detect malicious communication. The detected malicious peers are identified and "banished" from the system as described in Section 3.4.

To summarize, the main distinctions between the TP2-augmented system and vanilla BitTorrent are the following:

•Whereas in BitTorrent a user contacts a tracker directly, in TP2 a user enters the system through the authenticator service. The authenticator hands out verifiable tokens and parameters for secure communication.

•The client and the tracker software is modified to enforce strong authentication, authorization checking and encryption in all point-to-point communications.

•In its assignments of peers in each download session, the TP2 tracker includes a fraction of TAs. The trusted auditors, which are owned and managed by the content provider, and their identity is known to the tracker but not to the regular clients. The exact number of TAs is a parameter in the system that we discuss as part of our analysis in Section 4.

•TAs carry out the same protocols as regular P2P downloading clients, but also detect any malicious communication and "ban" malicious users from the P2P system as will be described further in the paper.

We now turn to the description of the new components added by TP2 and the threats they help address.

3.2 Threat Model

The TP2 architecture is designed to deal with threats that come from within the P2P system. We do not address external threats, as these are common to all content-delivery systems, be they direct download or P2P. For example, in the absence of any digital rights management (DRM) mechanisms, a user can always share content indiscriminately (*e.g.,* using a different P2P or web portal system, making copies on portable recording media, *etc.*). Our goal is to develop a P2P-based system that is *no worse* in terms of security than current-generation direct-download systems.

The first threat comes from the unauthorized users who try to obtain free content from the system. In the regular BitTorrent protocol, for example, if the IP addresses of the file-sharing peers becomes known (as it must, for communication to occur), an unauthorized user can simply bypass the tracker, connect to BitTorrent clients running on those IPs and begin sharing pieces of the content with them. TP2 helps prevent against such easy exploitation by requiring strong authentication and authorization to be used in the system by all users. The Authenticator is the new module responsible for granting authorization parameters to users upon their entry into the system. In addition, using public-key cryptography, TP2 individually enforces secure communication between each pair of authorized users. Naturally, not all users are expected to follow the protocol (and the security requirements) faithfully.

Thus, the second threat addressed by TP2 comes from malicious users who purchase content and thus obtain the proper authorization to join a file-sharing instance. Such malicious users may then attempt to discover other malicious users among the file-sharing clients and form a collaborative network for future unauthorized sharing. For instance, if five malicious users with similar interests discover one another in a file-sharing instance, then in the future only one out of five will need to purchase new content, that will then be shared with the group. TP2 deals with this threat by including TAs in the file-sharing instance. If a TA detects a malicious user attempting to scavenge information for future sharing (*e.g.,* because the malicious user contacted the TA and attempted to share unauthorized content, or because she allowed a TA impersonating a malicious user to contact her and share content without proper authorization), then that user is "banished" to an isolated direct download system as described in Section 3.4. As we show in our analysis in Section 4, the mere knowledge that TAs are present in the network causes rational malicious nodes to behave more cautiously and thus less dangerously towards the content provider.

The security parameters generated by the authenticator prevent unauthorized clients from participating in the system. Because a malicious client has to purchase content to gain a one-time system path, this creates a barrier for casual malicious users to enter the system just to scavenge for IPs of other malicious nodes. TAs help to set the bar of malicious exploitation even higher by detecting malicious users who have purchased content and thus have gained authorized entry into the system. Furthermore, TAs detect users that do not honor (enforce) the security parameters generated by the authenticator.

We now describe the functionality of the authenticator and the TAs that help cope with these threats.

3.3 Authorization

We now describe the authenticator and other modules which enforce strong authorization and authentication. When a user purchases the content at the CP's portal, their credit card is charged the cost of the content. An entry is also entered in the CP's database that authorizes the user with the given username to download the content. At that point the authenticator that runs on the CP's portal generates security parameters for the user and sends them to the user over a secure connection (using SSL).

3.3.1 Security Parameters

The security parameters given to the user include a temporary public/private key pair and a signed credential (akin to a public-key certificate) signed by the authenticator, whose public key is implicitly trusted by all participating users (*i.e.,* it is distributed along with the software, or is otherwise well known). More specifically, we use public-key-signed policy statements (similar in form to public-key certificates [6]) issued by the content provider as the basis for authorization in our system. These credentials are supplied to authorized users after a purchase is made, and can be used as proof to both the Tracker and the other participants in a P2P download session. In total, the credential includes the following:

•**Session ID**. This unique 120-byte identifier is given to the user for her download session.

•**Expiration timestamp**. Expiration time is an over-estimate of the time needed by the user to download the file given her broadband connection and the size of the content file.

•**User IP**. The IP address of the user's machine. The user's client is allowed to run only on one physical machine during the session.

•**Public Key**. The public key from the pair generated for the user.

•**Instance Id**. Unique identifier of the file-sharing instance managed by a tracker.

The authenticator assigns the user to a file-sharing instance (session) with the given Instance Id and sends the IP address, port number and the public key of the tracker that manages that instance to the user. The authenticator stores the Session ID and the Instance ID issued to the user.

3.3.2 Verification by Tracker

The authenticator also sends the Session ID of the new user to the tracker. The user establishes a TCP connection to the tracker and sends the certificate to the tracker for verification. The tracker verifies the authorization of the user by checking the following parameters:

•**Signature**. The tracker verifies the validity of the certificate by checking the signature using the authenticator's public key.

•**User IP**. The tracker checks that the IP address of the user matches the one in the credential, thereby preventing a malicious user from joining the system from multiple instances in the hope of discovering other malicious users. The user is prevented from spoofing the IP through a simple challenge-response exchange.

•**Private Key**. The tracker checks that the user is the holder of the private key corresponding to the certificate by testing that the user can decrypt a random string encrypted with the user's public key (the challenge-response exchange from the previous item).

•**Session ID, Instance ID, Expiration** The tracker checks that the session ID is a valid ID assigned to the given instance by the authenticator, and that the certificate has not expired.

If all the parameters match the tracker, the clients use their public/private keys to negotiate an RC4 session key for encrypting all future communication over their TCP connection. The tracker then assigns and sends a list of other peers to the new user, along with a new credential that lets the new user contact other nodes of the same session.

3.3.3 Peer Verification

When one node contacts another, she sends her tracker-issued credential. The receiving node performs checks similar to the ones performed by a tracker: it verifies the signature, the IP address, the public/private key binding, expiration, and instance ID. She also checks that the correct session ID is included in the credential she received from the tracker. She sends her own credential back to the new node, who performs the same checks. Then they negotiate the session key for their encrypted TCP connection.

3.4 Trusted Auditors

In each download session, the system trackers include some TAs. These mimic other P2P peers in their download and upload strategies: through both *passive participation* and *active probing,* TAs detect malicious nodes that attempt to discover other malicious nodes for future illegal file-sharing. The discovered nodes are then banished to an isolated direct-download system where they can no longer discover other malicious nodes, and where they have to pay a slightly higher fee for downloading content, reflecting the higher cost of bandwidth and management for the CP. For the remainder of this paper, we assume this simple banishment-based punishment. Other reaction strategies are certainly possible, and we leave them as the subject of future work.

Furthermore, we assume that there is a cost inherent in using TAs. Thus, while TAs will reduce the loss (and thus increase profit) of the CP, they also incur a cost (*e.g.,* in terms of the bandwidth they consume, which the CP has to pay for).

3.4.1 Malicious Users

In addition to performing the regular file downloads, malicious users perform queries of other P2P peers and discover malicious ones among them. Their software both probes and responds to probes from other malicious users. Strictly speaking, we define malicious probing as any communication that deviates from accepted TP2 protocols. In particular, the following forms of communication from other P2P nodes are deemed malicious:

1. A connection attempt by a node on the protocol's TCP port but with an improperly formatted request.

2. A connection attempt on the correct protocol port with an invalid or expired certificate.

3. A connection with a valid certificate but an unknown Session ID.

4. A connection attempt with a valid certificate, and Session ID but sending messages that are not according to the given file-sharing protocol specification. In the case of our implementation, this would be messages that are not formatted to fit one of the defined types of BitTorrent messages (*i.e.,* anything other than REQUEST, PIECE, BITFIELD, HAVE, CANCEL, CHOKE, UNCHOKE, KEEPALIVE, INTERESTED and NOT_INTERESTED messages). (See the BitTorrent specification [4] for further documentation).

5. Finally any communication by another P2P peer to a non-protocol port.

3.4.2 The Probing Game

Malicious users may probe and reply to such probes from the P2P peers assigned by the tracker. If the malicious user is lucky, she will discover a few other malicious users by such probing and can form a file-sharing *cluster* with them. However, if she is unlucky, she may probe a TA instead and get detected and banished from the P2P system. A malicious node may also get detected if it replies to a fake probe from a TA pretending to be malicious. The malicious node tries to maximize the number of other malicious nodes that it can discover, while avoiding being detected by a TA. We call this probing behavior by malicious nodes and trusted auditors a "probing game".

In Section 4 we describe and analyze the probing game in more detail. Briefly, a malicious node decides on a *strategy* in terms of how many of its neighbors it will attempt to probe or how many neighbor's probes it will reply to. A riskier strategy would involve probing more neighbors and carry a higher probability of getting caught. A riskier strategy may also carry a bigger payoff, as it may discover more malicious nodes to form a file-sharing cluster with. The full analysis is presented in Section 4. For most of our analysis in Section 4 we assume that, once a cluster is formed, it is not broken up. Thus, malicious users that are banished to the direct-download system can still share content that they pay for (or that they already have).

3.4.3 Behavior of Trusted Auditors

TAs act as hidden "sentinels" in the system to prevent excessive malicious probing, and reduce the rate of malicious cluster formation. To stay hidden, TAs mimic different roles: regular or "neutral" nodes, malicious nodes, and *seed* servers.

In their "neutral" role, TAs mimic the behavior of P2P peers by implementing the same discovery and download algorithms, and exhibit similar download speeds, arrival and departure rates as some of the regular clients. (This behavior is mimicked from historic observation of other P2P clients). They also change IP addresses between download sessions, or periodically.

In their "malicious" role, TAs mimic the behavior of malicious nodes by sending out probes to their neighbors at the same rate as other malicious nodes. To mimic the behavior of other malicious nodes, we employ two strategies. One strategy involves actively searching, studying and running the software that malicious users use on TAs. (We believe this to be a reasonable strategy, as the content provider can invest significantly more resources than individual malicious users to obtain such software.) The second strategy is learning the malicious probing format and pattern on the fly. This approach is based on recent work done at UC Berkeley on the RolePlayer system [7]. RolePlayer can quickly learn and replay various network communication patterns.

Once a TA *detects* an unauthorized probe or receives a positive reply to a fake probe of its own, it identifies the corresponding node as malicious. The system "punishes" such a user by forcing all of her future file downloads to be performed in an isolated direct-download system. The user pays a higher price for her future purchases. In our analysis, we use the very conservative approach by setting the difference in the cost the be exactly the cost of serving content to such a user in a direct-download system. This penalty is used as a deterrent to prevent any user from misbehaving. In addition, the user will not interact with P2P peers in future downloads and will not be able to grow her cluster any further. It is difficult for a user to create new identities after she has been detected, because the identity is tied to a name, and a credit card billing address of the user. Ideally, TP2 detects and places all malicious users in an isolated system and uses P2P bandwidth to serve the rest of the users.

Finally, TAs serve in the capacity of BitTorrent *Seed* servers (*i.e.,* they contain all the pieces of the shared file). Since each P2P node is assigned to some TA, this guarantees that it will be able to find all of the missing pieces among its neighbors. We mentioned earlier that, to mimic users' behavior, TAs pretend that they join the system with no file pieces. However, over time P2P peers let their neighbors know that their collection of pieces has grown. In BitTorrent, we implement this mechanism with "HAVE" messages that clients use to announce to their neighbors that they have downloaded a new piece.

3.4.4 Out-of-System Probing

Another type of unauthorized probing comes from outside of the file-sharing instance. In this type of misbehavior, a malicious node joins the system just to gather IP addresses of its neighbors. Some time after its participation, it assumes a new IP address for itself and sends out malicious probes to the previously collected IP addresses.

Even though such a probe can be detected by a TA, it is hard to determine which user in the system it corresponds to. Since the user is not logged into the system and is possibly running from a different IP, the identity of such a user cannot be determined.

We deal with this "out-of-system" probing also by performing out-of-system active probing. TAs learn the rate at which such out-of-system probes arrive and they mimic this type of active probing at a slightly higher rate. Similar to the detected probes, the fake probing is performed against IP addresses of current or prior participants. If a malicious node replies to such a probe, then the last registered user known to have logged in from that address is banished.

This fake out-of-system active probing is a deterrent against malicious nodes replying to the probes. Since the TAs probe at a slightly higher rate, malicious nodes know that if they reply to an out-of-system probe they will more likely be communicating with a TA. Thus, malicious nodes have little incentive to reply to such probes. This deterrent-based policy mitigates the threat from out-of-system probing and allows us to focus our efforts on the "in-system" probing.

3.5 Scalability of TAs

The number of trackers and TAs should scale with the growth of participants in the system. As the number of participants grows, so will the number of CP-owned machines. The cost of maintaining such machines should then scale with the growing revenue. The system does not require many physical TAs, as each such machine can participate in a number of simultaneous download sessions under virtual IPs. For instance, if the fraction of TAs in each file-sharing instance is 5%, and each machine participates under 10 virtual IP addresses, then for a 100,000 simultaneous participants only 500 physical TA-dedicated machines are necessary.

4. ANALYSIS

In Section 3 we use strong authentication and authorization to prevent unauthorized users from downloading content. Here, we are going to analyze the second threat we discussed in Section 3.2 - the authorized participants' attacks. In these attacks, malicious nodes become part of our P2P network by pretending to be regular users and registering and paying for movie downloads. If left unchecked, such malicious nodes may exploit the regular P2P system operation to extract information from other participants. The malicious nodes can then form large malicious *clusters* for future illegal file-sharing and trading. These clusters then can game the system by purchasing one copy of a file and sharing it among its members. As we will show with a simple economic analysis, such illegal sharing could significantly decrease the profits of the content provider and be a strong deterrent against the adoption of a P2P approach for content distribution. As mentioned in Section 3, we introduce trusted auditors which limit the formation of large malicious clusters. In this section, we analyze the behavior of the malicious nodes, and show, via both an analytical model and simulations, that even a small number of trusted auditors can effectively curb the growth of clusters and successfully protect content provider's profits.

4.1 Economic Impact

We propose a simple economic model to quantify the impact that malicious nodes have on the CP's profit. We assume that the average price of digital content sold by the CP is S dollars. The CP pays a large part of that price as royalties $\$R$ to the content owner (a movie studio for example), and retains $\$D$. ($D = S - R$). In a direct download system the CP also pays $\$B$ for the bandwidth required to serve a file of average size to the end user. Thus the CP's profit per movie purchase is, on average, $\$(D - B)$. The market research in [1] shows that digital movie and audio stores pay roughly $60 - 70\%$ of end price (S) in royalties and the cost of bandwidth amounts to about 20%. Using a store similar to Apple Itunes as an example, one can purchase standard length (1GB) digital movies for \$10, we assume that D, the store's profit before bandwidth cost is \$3 to \$4 and B, the cost of bandwidth is roughly \$2 per download. We fix these assumptions in this section, but our results hold for wider ranges of values (see Figure 9 in the Appendix).

Using a P2P download approach the CP saves on most of the bandwidth cost and claims a full $\$D$ as profit. Unfortunately, in the presence of malicious users the CP collects smaller amount of revenue, and thus smaller profit since the malicious nodes form downloading clusters to avoid content payment. For example, if two malicious users manage to discover each other in the P2P system they will form a cluster of size 2. Then, these users will take turns purchasing files and sharing them with each other for free instead of buying them through the CP. For simplicity, we assume that malicious and non-malicious (or *neutral*) users desire to accumulate files at the same rate (e.g. say they download one

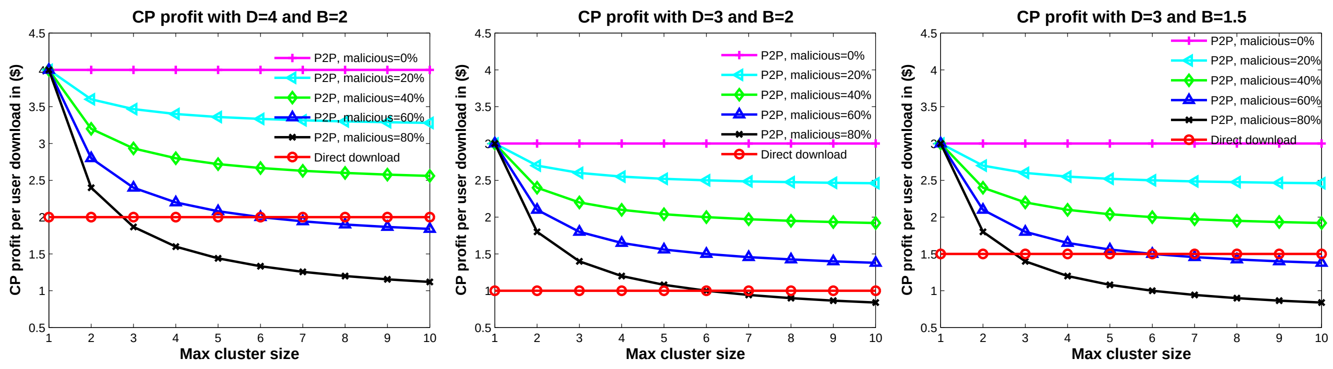


Figure 3: CP profit per user download. Distinct combinations of D (profit before bandwidth) and B (bandwidth cost) capture variations in possible royalties and bandwidth agreements

movie per week), and that their interests are similar and thus they only need to purchase files at a fraction of the rate of the neutral users. For instance, in a cluster of two malicious nodes they each purchase movies at half the rate of the neutral. More generally, users who belong to a cluster of size K need to purchase content at a $\frac{1}{K}$ fraction of the rate of the neutral users to get the same number of files in a given time interval. This scenario is pessimistic, since we assume that we lose from all malicious clusters whereas in practice, only some of the users in the cluster will want any particular file.

In our model, a single download session consists of up to N_s nodes that are all assigned to one another by a tracker. For a popular file, the system runs multiple download sessions of up to N_s nodes each. We assume that a single session contains at most M malicious nodes, T trusted auditors and Q neutral nodes with $N_s = Q + M + T$. In a bittorrent network a typical value of N_s is around $50 - 60$ nodes, thus in our system we will assume a maximum bound of $N_s = 100$. Let M_i be the number of users in the system who are malicious and who belong to clusters of size i. Then $M = \sum_{i=1} M_i$. We define $m_i = M_i/(M+Q)$ and $m = M/(M+Q)$ to denote the ratio of malicious users to the total number of malicious and neutral nodes. We can now derive an amortized profit received by the CP each time a user accumulates a file as

$$\text{Profit} = D \cdot (1 - m) + D \cdot \sum_{i \geq 1} \frac{m_i}{i}. \tag{1}$$

The first term in Equation 1 is the CP profit from neutral users who pay a full price and the second term is from malicious users who pay only a fraction $\frac{1}{i}$ of the price based on their cluster size i (assuming multiple downloads). On the other hand, the profit of the CP in a direct-download system per download is $D - B$. We remind the reader once again that we do not attempt to solve *out-of-band* sharing that can exist with both direct and P2P systems. Rather, we are interested in curbing file-sharing from clusters formed by malicious exploitation of the P2P distribution system itself.

Using Equation 1, we can produce the CP profit plots for various values of D and B. Figure 3 depicts CP profit curves for the P2P and the direct download systems for various values of m ranging from 0 to 80%. Each plot picks a different combination of values for D and B in a reasonable range as describe above to allow for variations in the cost of royalties and bandwidth. The x-axis shows the maximum size of a malicious cluster, K. The y-axis shows the average profit claimed by the CP user download. Each plot contains two horizontal lines: the top one representing a profit of a P2P system assuming no malicious nodes and the bottom one representing profit of a direct download system. The difference between the two plots is exactly B, the cost of bandwidth per download. The non-linear curves plot Equation 1 and represent the profit of a P2P system with various fractions m of malicious users. The plots show that as the fraction of malicious nodes and the file-sharing clusters that they form grow the profits for the P2P system dwindle. In fact, as shown in the Figure 4, as the malicious nodes' fraction approaches 80% and for malicious clusters of 50 nodes, the CP collects less than half the profits of a direct download approach. Even for less aggressive collections of malicious users, we see that most of the economic advantage of P2P rapidly evaporates. If the malware that implements malicious code becomes readily available on the Internet, even the non-savvy users could easily become malicious. As the fraction of malicious users proliferates and they form larger clusters with time the profits of the CP quickly erode. For a P2P system to succeed, it is thus imperative to limit the effect of the malicious nodes. In the rest of the section, we show how even a small fraction of trusted auditors could maintain the near-optimal P2P level profits for the CP.

4.2 Probing Game Revisited

To model the interaction between the malicious and trusted auditors we revisit the *probing game* presented in Section 3.4.2. To form clusters, malicious nodes try to probe or reply to probes from other malicious nodes to form and grow a malicious cluster. To detect malicious nodes, trusted auditors also pretend to be malicious. They actively send probes and reply to probes that they receive from others. Since the goal of the attacker is to grow without being detected, she has to select a growth factor GF which reflects the minimum cluster size that she aims to belong at the end of a single session (download). Therefore, the malicious nodes probe and reply to probes until they either discover at least $GF - 1$ other malicious nodes or are detected by a trusted auditor. Observe that for $GF = 1$ malicious nodes will not do any probing and will act exactly as a neutral node. On the other hand, if $GF > M$ the malicious nodes are certain to hit a trusted auditor and thus become detected before they can grow into

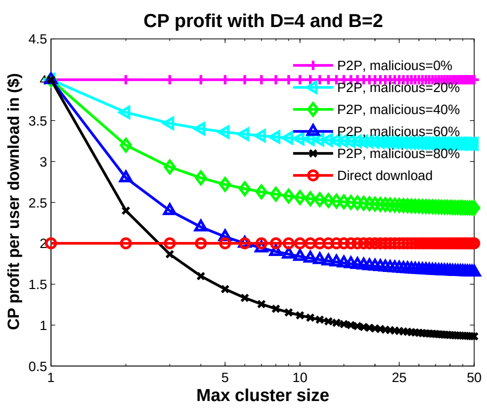


Figure 4: CP profit per user download for D=4, B=2 and clusters of size up to 50

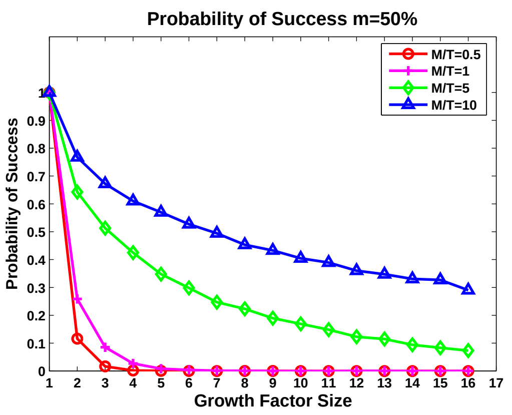


Figure 5: For a single game, probability that a malicious node succeeds in forming a cluster of at least its growth factor for m=50% **(i.e.** 50% **of users are malicious)**

a cluster of size GF. Thus, GF will take on some value in the range from 1 to M. In general, we make the following set of assumptions about a download session.

- Malicious nodes remain "active" (i.e. they send probes and reply to probes) until they reach their growth factor of GF.
- Each malicious node knows both M and T in a download session, and based on that picks the most profitable value of GF. We suggest a good value for GF later in the section based on a simulation of multiple games.
- If a malicious node e is detected by probing or replying to a probe from a trusted auditor t it stops probing. We assume that without revealing that he is a trusted auditor, t can convince e that together they have formed a cluster of size GF (*i.e. if* e *is already part of a cluster of size* $GF - k$, t *tells* e *that is part of a cluster of size* $\geq k$).
- Both malicious nodes and trusted auditors send probes to randomly chosen neighbors at the same probing rate per node. Trusted auditors send probes at the same rate to be indistinguishable from malicious nodes. Otherwise, collaborating malicious nodes could easily pick out trusted auditors in the system replying only to low frequency requests.
- Upon receiving probes, neutral nodes simply ignore them. (Having neutral nodes play a role in detecting malicious nodes can potentially help the detection of malicious but we leave it as an item for future work).

Our primary focus is to show that over time, as malicious nodes play multiple *games*, (i.e. they participate in many download sessions) most of them become *detected* and large clusters are unlikely to form. Small clusters may form, but these have limited economic impact. However, first we show some simulation-based plots to give some intuition about what happens in a single download session. We note that these plots use averaged probabilities collected from a thousand runs of a simulated game.

Figure 5 shows the probability that a malicious node *succeeds* in forming the desired cluster size. Here the number of malicious nodes in the download session m is fixed at 50% and the number of trusted T is varied over different ratios of M/T. The x-axis shows the strategy (i.e. growth factor) chosen by the malicious nodes in the game. The y-axis gives the probability that a node succeeds in achieving reaching its selected growth factor. As an example, the scenario of $M/T = 1$ (the number of malicious nodes and trusted auditors is the same) and a target $GF = 2$, shows that the probability of a node succeeding in forming a cluster of size 2 is about 25%. Thus there is a 3/4 chance that a node gets detected in such a game. An important observation about this plot is that all curves are decreasing monotonically. That means that as the malicious nodes become more aggressive by picking larger growth factors, they are also more likely to be *detected*. Interestingly, even for the top curve (the most favorable of these scenarios for malicious nodes) and the least aggressive target of $GF = 2$, there is only a 77% chance that such a node succeeds (i.e. there is a 23% chance that it becomes detected). So we see that even in a very favorable scenario the probability that the node does not become detected in k independent games is roughly $.77^k$ which decreases exponentially.

Figure 6 shows the cumulative probability of a node growing into a cluster of size $> X$ (where X is the value on the x-axis). Here m is again fixed at 50%. The three distinct plots correspond to three values of M/T, and the curves in each plot correspond to different growth factors from 2 to 5. (We picked these values for GF because for higher values malicious nodes experience a very high detection rate and thus are unlikely to choose such values). These plots show the same data as Figure 5, but in more detail by including the cumulative distribution of all cluster sizes formed. Observe that clusters of sizes both smaller and larger than the growth factor can form. Clusters smaller than the growth factor are those that are detected before they could grow to the target size. These clusters cannot continue to grow in the future games. Also observe that for a growth factor GF clusters as big as $(2 \cdot GF - 2)$ can form, when two clusters of size $GF - 1$ merge. However, the probability of forming larger clusters drops off quickly. The actual value of a strategy can only be understood by looking at the gains of

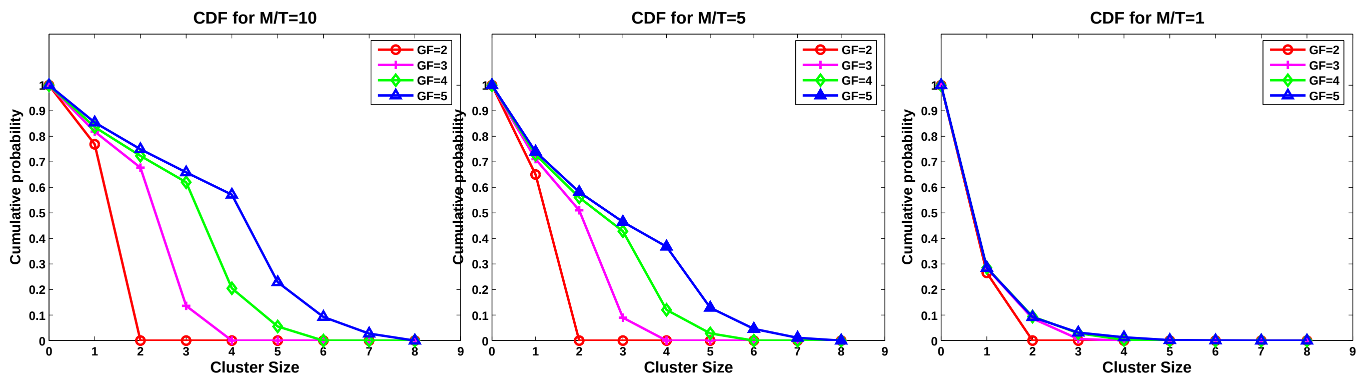


Figure 6: Cumulative probability of a malicious node forming a cluster of size $> X$ (value on the x-axis) in a single game. Curves correspond to given target growth factor GF. Again, all plots use m=50% (50% fraction of malicious nodes to neutral nodes). The ratios of malicious nodes to trusted auditors are 10, 5, and 1, respectively.

a node over multiple games. We perform this evaluation in Section 4.4.

4.3 Analytical Model

We start by analyzing the single session download model and describing the growth of clusters both in the first session and after multiple sessions (downloads). To this end, we present a Markovian model with memoryless states and well-defined transition probabilities. Using this Markovian model we can fully compute the stationary probabilities of malicious cluster formulation (*i.e. the probability that a node ends up in a cluster of size K)*. Moreover, our model can compute the stationary probabilities of the absorbing(final) states starting from any initial discrete cluster distribution. This enables us to compute the stationary distribution across multiple downloads by recursively applying our analysis starting with singleton malicious clusters and using the resulting cluster distribution as a starting distribution for the next session.

To simplify our presentation, we start by modeling a download session with a M and T malicious nodes and trusted auditors respectively. Moreover, we set the value of the growth factor to be 2, ($GF = 2$). A state in such a download session is fully captured by the tuple $< A, F >$ where A is the number of active malicious nodes (that are still probing) and F the number of detected (found) nodes. The number of clusters of size two at a given state is simply the remaining nodes: $\frac{M-F-A}{2}$. The valid transitions from $< A, F >$ are to $< A-2, F >$ (two active malicious nodes form a cluster) and to $< A-1, F+1 >$ (one active node becomes detected). Recall that all active malicious nodes and trusted auditors probe at the same rate, and thus the probability that the next probe is from a malicious node is $\frac{A}{A+T}$ and is $\frac{T}{A+T}$ that it is from a trusted auditor. The probability of a cluster being formed is simply the probability that the next probe is sent by an active malicious node or that it is sent to one other active malicious node. (Recall that the probabilities do not need to account for probes being sent to inactive malicious or neutral nodes as such nodes simply ignore the probes.) We thus have the following transition probabilities for $F \geq 1$ and $A \geq 2$:

$$P_{<A,F>,<A-2,F>} = \frac{A}{T+A} \cdot \frac{A-1}{T+A-1}.$$

We also consider the case when a node is detected, which happens when a malicious node sends a probe to a trusted auditor or trusted auditor sends a probe to a malicious node. For $A \geq 1$,

$$P_{<A,F>,<A-1,F+1>} = \frac{A}{T+A} \cdot \frac{T}{T+A-1} + \frac{T}{T+A}.$$

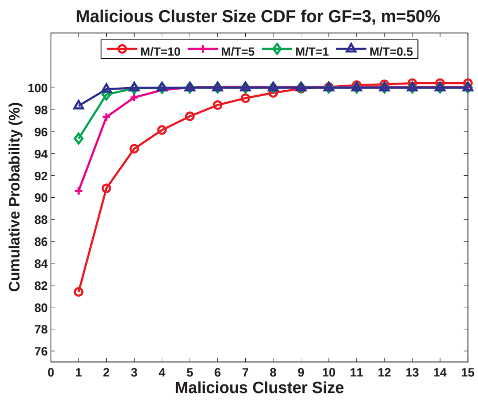
The starting state is $< M, 0 >$ and the terminal states are of the form $< 0, F >$ where no active nodes are left. The process is clearly Markovian as transition probabilities are uniquely determined by the current state, and the probabilities add up to 1.

Similarly, we can define the states and transition probabilities for games with $GF > 2$. Here the state space becomes $< A_1, \ldots, A_{2\cdot GF-2}, F_1, \ldots, F_{GF-1} >$, where A_i is the number of clusters of size i and F_j is the number of detected clusters of size j. Note, that for a target size GF it is possible to have clusters of size up to $2 \cdot GF - 2$, because two active clusters of size $GF - 1$ could join together. Also, note that the state reflects the numbers of detected clusters of different sizes, rather than lumping all the detected nodes together in one number F. We maintain this distinction because nodes that form a cluster of size $< GF$ and then become detected may still file-share files that they downloaded in the future, even though all of these nodes will be banned from the P2P system. Let $L = \sum_{k=1}^{GF-1} k \cdot A_k$ be the number of active nodes (i.e. all malicious nodes in clusters of size $< GF$). We can then proceed similarly to the case $GF = 2$ and compute the non-zero transition probabilities in three cases.

The first case is that a cluster of size i becomes detected, which occurs when a node belonging to a cluster of size i probes a trusted auditor or a trusted node probes a node in a cluster of size i.

$$P_{<\ldots,A_i,\ldots,F_i,\ldots>,<\ldots,A_i-1,\ldots,F_i+1,\ldots>}$$
$$= \frac{i \cdot A_i}{T+L} \cdot \frac{T}{T+L-i} + \frac{T}{T+L} \cdot \frac{i \cdot A_i}{L}$$

The second case is when two clusters of size i and j, with $i < GF$ and $j < GF$ join to form a cluster of size $i+j$. The



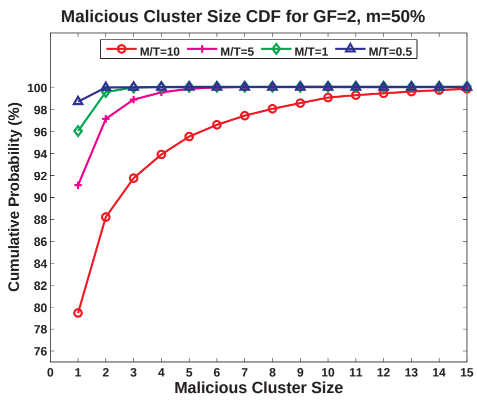


Figure 7: Cumulative probability of forming clusters for growth factor $GF = 2$ and $GF = 3$ for multiple games. Notice that both plots look similar and that for $M/T = 10$, $GF = 2$ results in slightly larger cluster sizes.

third case is like the second but with $i = j$.

$$P_{<\ldots,A_i,\ldots,A_j,\ldots,A_{i+j},\ldots>,<\ldots,A_i-1,\ldots,A_j-1,\ldots,A_{i+j}+1,\ldots>}$$
$$= \frac{i \cdot A_i}{T+L} \cdot \frac{j \cdot A_j}{T+L-i} + \frac{j \cdot A_j}{T+L} \cdot \frac{i \cdot A_i}{T+L-j}$$
$$P_{<\ldots,A_i,\ldots,A_{2i},\ldots>,<\ldots,A_i-2,\ldots,A_{2i}+1,\ldots>}$$
$$= \frac{i \cdot A_i}{T+L} \cdot \frac{i \cdot (A_i-1)}{T+L-i}$$

These are the only allowable and valid transitions. The starting state is $S =< M, 0, \ldots, 0 >$ means that all nodes are singletons (*i.e belong to clusters of size* 1). Moreover, all terminal states are of the form:

$$< 0, \ldots, 0, A_{GF} \ldots, A_{(2\cdot GF-2)}, F_1, \ldots, F_{GF-1} >$$

There are no clusters of size less that GF as all malicious nodes have either been detected or have formed clusters of size equal or more than GF.

Once we construct the transition probabilities matrix P we can easily compute the probability of specific terminal states by exponentiation $P^{(n)}$ for some $n > 1$ at which the terminal probabilities converge. $P_{S,j}{}^{(n)}$ gives the probability that a particular terminal state j occurs. (Note also that n is bounded by $M*(GF-1)$ as each malicious node transitions at most $GF - 1$ times).

4.4 Multiple Games (Downloads)

For multiple downloads, we have to consider that after the first download session, there are some undetected clusters formed. Furthermore, we assume that it is to the benefit of the malicious nodes to have only one representative from each cluster participate in the next download session since they share content out-of-band.(this way only one of them pays) For example, assuming a closed system (no new arrivals), if in the first download we had M malicious participants and all of them managed to form clusters of size 2 , in the second download only $M/2$ of them will participate. Therefore, after the first game, we will only have a fraction of the malicious nodes that participated in the first download either because they managed to form clusters or because they got detected by trusted auditors. In addition, we assume that these malicious participants are acting on behalf of the formed clusters and thus are still aiming for the same growth factor GF.

The Markovian approach we presented in the previous section can still be applied but with some modifications: we have to recompute the transition probabilities because for each games we start with a different number of malicious nodes. We can compute the number of malicious nodes between games by taking into consideration the following parameters: formed clusters from previous game (both detected and undetected), arrival of new malicious singletons and departures of malicious nodes. Formed undetected clusters are the clusters who still contain nodes that have not been detected in previous downloads. In addition, we assume that we have Ar arrivals of malicious singletons and De departures of malicious nodes that can be either previously detected or undetected. For simplicity, we chose $Ar = De$ and we call the new quantity "renewal rate". Throughout our analysis, we used a renewal rate of 5% but similar results hold for renewal rats of 1% to 10%.

Thus, based on the information on the clusters formed and detected from the previous game and the renewal rate, we compute the new number of malicious nodes that will participate in the next download. We then generate the transition probability matrix and we start from state $S =< M_n, 0, \ldots, 0 >$ where M_n is the computed malicious nodes. Using the resulting steady state probabilities and the previous cluster distribution we had prior to the download, we compute the resulting joint distribution assuming that only one node from the undetected clusters participated in the download. For example, assuming that we have a representative of cluster of size K with F detected participating in a download with growth factor $GF = 2$ and initial cluster distribution P_i then the resulting transitions are to a cluster of size $K + i$ with F detected with probability $P_s \cdot P_i$ and a cluster of size K with 1 detected with probability P_d, where P_s and P_d are the probabilities of actually forming the cluster or getting detected respectively.

Although, maintaining state for all formed clusters across multiple downloads appears to be large, in practice it is not. We start from a small initial set of malicious singletons that form clusters which grow by merging with each other, leading to rapidly declining number of non detected clusters. Moreover, the malicious to trusted auditors M/T is also decreasing, leading to more detected malicious nodes. Therefore, although there are in principal a large number of

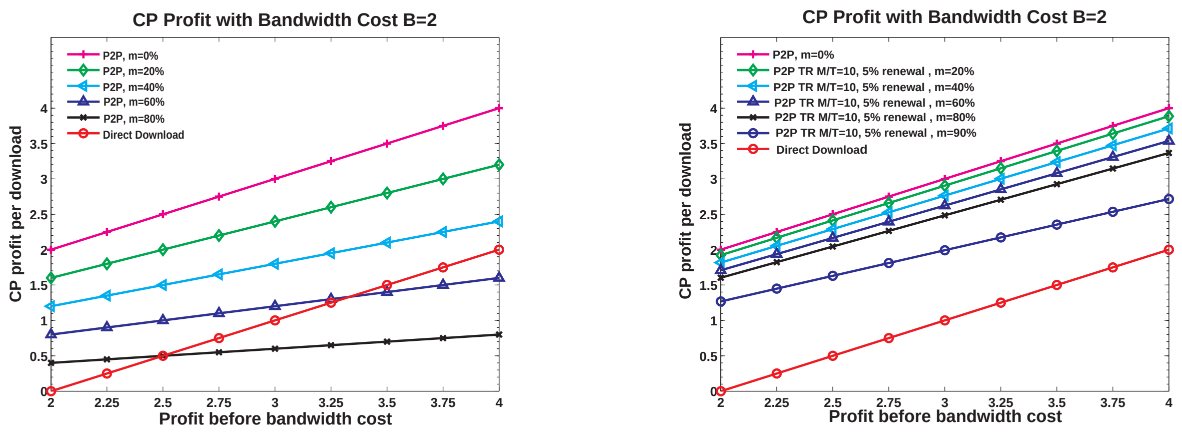


Figure 8: Comparison of CP profits between a protected and an unprotected system for bandwidth cost of 2. On the left we have a system without trusted auditors and on the right a system with a ratio of trusted auditors to malicious users being 10. Clearly, the protected system yields more profits than the unprotected system for the same fraction of malicious nodes which are very close to the ones produced by a P2P system with 0% of malicious nodes.

possible cluster sizes that can be formed, in practice the actual number of formed clusters decreases fast over multiple downloads.

4.4.1 Simulations

To verify our analytical model and to avoid the computational complexity involved in computing joint probabilities cluster sizes formed over multiple downloads of a large user population, we generated simulations of single and multiple download sessions. The results of our simulations fully agree with the ones obtained using the analytical model for growth factors $GF = 2$ and $GF = 3$. For larger values of GF, and to show that such choice of GF is not beneficial for the malicious nodes over multiple downloads, we relied on the simulation results.

We used matlab to simulate the overall behavior of a BitTorrent-like P2P system with all types of nodes: neutral, malicious and trusted. We varied the overall system size ranging from 10^5 to 10^7 participants and our results remain consistent for all of them. The plots presented in the paper are obtained using a population of $2 \cdot 10^6$ nodes. Our aim was to examine the performance limits of our system under diverse operating conditions by varying both the fraction of the malicious nodes M and their relative ratio to the trusted auditors M/T. In addition, we wanted to find which growth factor is more beneficial for the malicious nodes across multiple downloads. We picked 30 downloads as the number we use for the multiple plots because at 30 downloads we have detected the overwhelming majority of the formed clusters for all M/T ratios we consider. In addition, after 30 downloads, we notice that new clusters are formed almost exclusively by the new malicious arrivals and thus we consider the distribution to be stable.

4.4.2 Results

We now describe our concluding results about the system. We first study the affect of the parameter GF. In Figure 7, we present results from multiple downloads and for growth factors $GF = 2$ and $GF = 3$. The depicted results indicate that there is very little difference in the malicious cluster size distribution (CDF) when comparing $GF = 2$ and $GF = 3$ with the first having slightly better results. Therefore, the malicious users should select $GF = 2$ as their growth factor if the want to optimize their probability of being in a larger cluster over multiple downloads.

We now compute the actual CP profit for our system. First, we have to extend the economic model formula in Equation 1 for the amortized profit per movie to include the bandwidth that the CP pays for the trusted auditors:

$$\text{(Profit with trust)} = D \cdot (1-m) + (D \cdot \sum_{i \geq 1} \frac{m_i}{i}) - B \cdot \frac{T}{M+Q} \tag{2}$$

The new factor in Equation 2 is $B \cdot \frac{T}{M+Q}$, which accounts for the bandwidth cost used by the trusted auditors. We use the ratio $\frac{T}{M+Q}$ because m is also normalized by $M + Q$, the total number of malicious and neutral nodes in the system. In this formula, we assume that after being detected, malicious nodes are moved to a direct download system pay the bandwdith cost for future downloads. Alternatively, we can leave these users in the P2P system, but notify them that they are on probation and force them to re-download a proper software client. They are warned that if detected again they can be referred to the courts or charged with a large fine on their credit card since they have already breached the contract with illegal activity (and follow through on such threats). Observe, that assuming that such users will continue to act as neutral our Equation 2 does not change as it treats neutral users and “banned” users equally. But keeping such users in the system is more likely to retain their business as they don’t have to pay the penalty bandwidth cost for future downloads.

The CP profits from the resulting formation of clusters for multiple games are shown in Figure 8. Here we see that for all fractions of malicious nodes, the system with the trusted auditors yields significantly more profit when compared to the unprotected system with the same fraction of malicious. Furthermore, the TP2 system profits are very close to the ones we get for a pure P2P system without malicious nodes, especially for small fractions of malicious nodes.

5. IMPLEMENTATION & PERFORMANCE

We implemented an TP2 prototype by adding modifications to the existing BitTorrent client and Tracker (version 3.9.1) written in Python. Our modest modifications included adding secure channel communication using RC4 encryption, assignment of trusted auditors by the Tracker, and the distribution of credentials by the tracker to the peers.

Due to space constraints, we only briefly discuss two simple experiments we conducted using PlanetLab [19] to compare the download speed of TP2 clients compared to BitTorrent clients on a set of geographically distributed machines given the overhead of secure communication and credentials distribution and verification in TP2. Most machines used were equipped with 3GHz processors and ran the Linux 2.6.12 kernel.

For our first test, we deployed 41 BitTorrent clients on randomly picked machines in the US. One of these machines we designated as the Seed client and initialized it with a 512MB movie file. To test the absolute worst case in terms of download time, we stored no parts of the file on the rest of the clients before the test. We ran the Tracker process on a machine outside of PlanetLab, a blade server with 3.06GHz processors, running a Linux 2.6.11 kernel, and a 10Mbit/sec upload bandwidth link. We ran the test once with the unmodified BitTorrent code and once with TP2. The BitTorrent download times were only 0.8% faster on average, showing that TP2 adds negligible performance overhead.

For our second test, we performed a similar experiment as the first test but using a more dynamic scenario where peers join the download system at staggered times. We began with one Seed and 76 clients. The 76 clients joined the system at 2 minute intervals. By the time the later peers start, more clients in the system already have partial data sets. Therefore, newer clients have more sources to download the data from and thus their download times are generally faster. For this test, TP2 clients on average slightly outperformed BitTorrent by about 0.5%. We have tracked this down to the fact that the TP2 nodes contact the Tracker more frequently and receive new connection assignments at a faster rate at startup. As a result, in the beginning of the download they have slightly more choices for selecting faster sources.

The CPU overhead on the TP2 clients was also minimal as RC4 encryption is a very fast stream cipher. Average CPU utilization on the TP2 and BitTorrent clients was almost identical at roughly 1.3% and 1.23% respectively.

6. CONCLUSIONS

TP2 is the first system of its kind that can be layered on top of existing P2P systems to enable content providers to leverage the download capabilities of a P2P system, and yet elevate their trust and content control to levels similar to that provided by a direct download system. Our approach provides strong authentication and introduces a novel notion of trusted auditors into a P2P system. Strong authentication ensures that the P2P system itself cannot be directly used for free file sharing. Trusted auditors appear as regular P2P nodes, but actively and passively detect malicious participants in the system to prevent scavenging of information that could be used to identify other participants for forming out-of-band free file sharing clusters. Just like in a direct-download system, TP2 does not prevent a user who legally downloads content from sharing it out-of-band, but it does prevent the system itself from being exploited in any way to facilitate out-of-band free file sharing.

We have analyzed TP2 by modeling it as a game between malicious users who try to form free file sharing clusters and trusted auditors who curb the growth of such clusters. We have combined this analysis with a simple economic model to quantify the cost-effectiveness of our approach in the presence of malicious users. Our analysis shows that even when half of the participants in a system are malicious users, our system can detect 99% of malicious users and prevent them from forming large clusters, thereby providing strong protection of the P2P system against unauthorized file sharing. For most configurations, our analysis shows that TP2 yields profits that are significantly higher, more than 80% higher profit, than a direct download system based on conservative profit and bandwidth cost models. We demonstrate that TP2 can be implemented on top of BitTorrent with modest modifications, and provides its content protection and economic benefits with negligible performance overhead compared to vanilla BitTorrent. We believe that our analysis and system provides a strong economic motivation for content providers to adopt regular P2P system enhanced with security guarantees for their content delivery. We hope that TP2 can serve as a strong foundation for creating practical P2P systems that can be used for paid content distribution.

7. REFERENCES

[1] J. G. Aguilar. personal communication, February 2006.
[2] Akamai. http://www.akamai.com/.
[3] P. Biddle, P. England, M. Peinado, and B. Willman. The Darknet and the Future of Content Distribution. In *Proceedings of the 2nd ACM Workshop on Digital Rights Management*, November 2002.
[4] Bittorrent. http://www.bittorrent.com.
[5] P. R. C. Gkantsidis, J. Miller. Anatomy of a p2p content distribution system with network coding. In *IPTPS*, February 2006.
[6] CCITT. *X.509: The Directory Authentication Framework*. International Telecommunications Union, Geneva, 1989.
[7] W. Cui, V. Paxson, N. Weaver, and R. H. Katz. Protocol-independent adaptive replay of application dialog. In *Proceedings of the 13th Annual Network and Distributed System Security Symposium (NDSS)*, February 2006.
[8] R. D. Qiu. Modeling and performance analysis of bittorrent-like peet-to-peer networks. In *SIGCOMM*, 2004.
[9] R. D.Arthur. Analyzing the efficiency of bit-torrent and related peer-to-peer networks. In *SODA*, January 2006.
[10] M. J. F. et al. Coral. http://www.coralcdn.org/.
[11] V. P. et al. Codeen. http://codeen.cs.princeton.edu/.
[12] Apple itunes. http://www.apple.com/itunes.
[13] Kazaa. http://www.kazaa.com.
[14] M. V. L. Massoulie. Coupon replication systems. In *SIGMETRICS*, 2005.
[15] Limelight. http://www.limelightnetworks.com/.
[16] Movedigital. http://www.movedigital.com/.
[17] Napsterm. http://www.napster.com.
[18] Piratebay. http://thepiratebay.org/.
[19] Planetlab. http://www.planetlab.org/.
[20] Red Skunk Tracker. http://www.inkrecharge.com/ttrc2/.
[21] Torrentspy. http://www.torrentspy.com/.
[22] Vitalstream. http://www.vitalstream.com/.
[23] Xbox-sky. http://bt.xbox-sky.com/.
[24] L. Xiong and L. Liu. Peertrust: Supporting reputation-based trust in peer-to-peer communities. In *IEEE TKDE, Special Issue on Peer-to-Peer Based Data Management, 2004. 6*, 2004.
[25] G. d. V. X.Yang. Service capacity of peer to peer networks. In *INFOCOM*, 2004.

8. APPENDIX

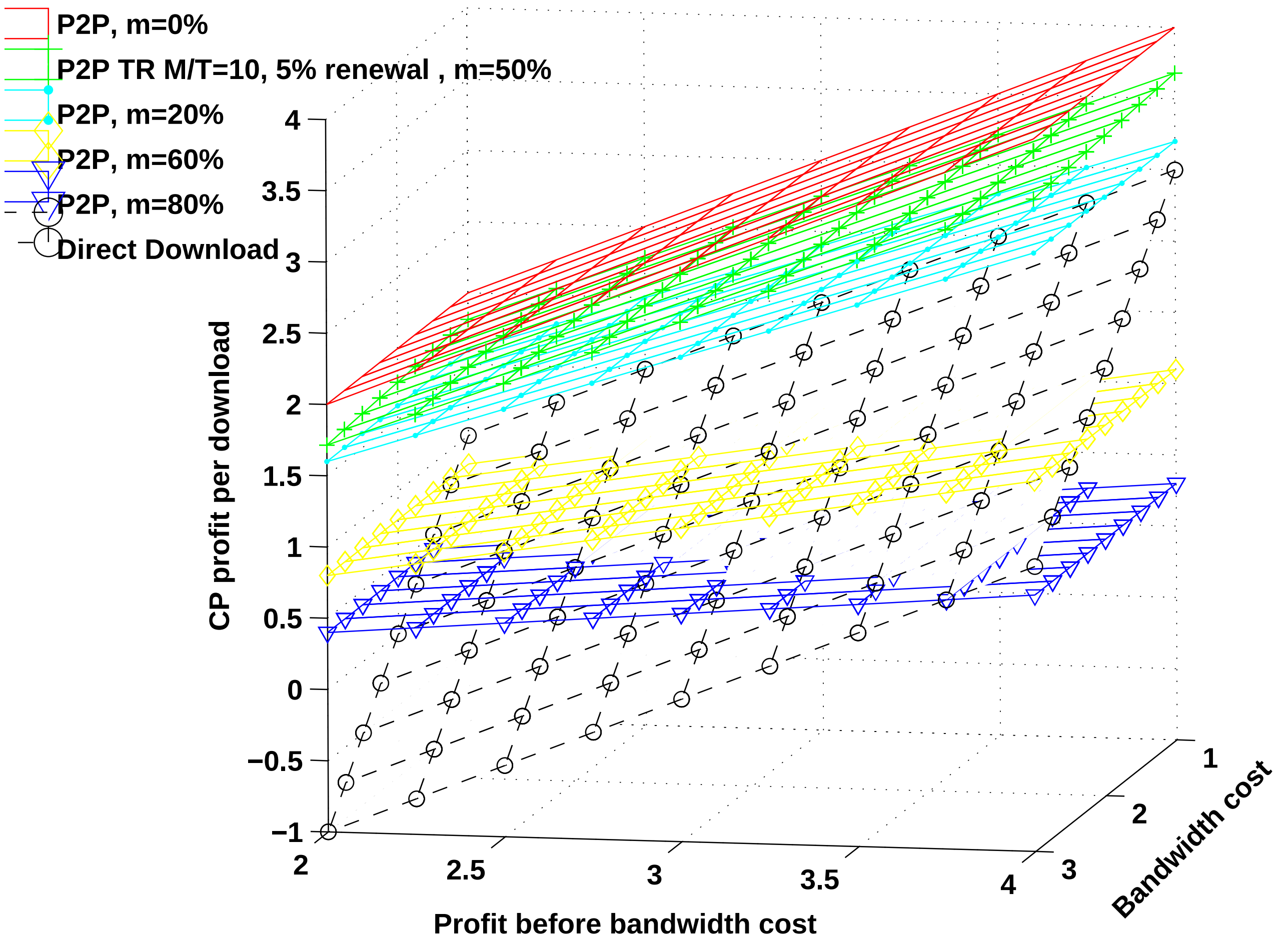


Figure 9: 3D profit plots with no trust and trust for variable profit before bandwidth D and bandwidth cost C