SECURITY AND PRIVACY ISSUES IN P2P STREAMING SYSTEMS: A SURVEY

Gabriela Gheorghe, Renato Lo Cigno, Alberto Montresor

August 2009

Technical Report # DISI-09-001
Version 1.0

Also: this work is supported by the European Commission through the NAPA-WINE Project (Network-Aware P2P-TV Application over Wise Network – www.napa-wine.eu), ICT Call 1 FP7-ICT-2007-1, 1.5 Networked Media, grant No. 214412

Security and Privacy Issues in P2P Streaming Systems: A Survey[*]

Gabriela Gheorghe, Renato Lo Cigno, Alberto Montresor
DISI – University of Trento, Italy
e-mail: name.surname@disi.unitn.it

August 20, 2009

Abstract

Streaming applications over Peer-To-Peer (P2P) systems have gained an enormous popularity. Success always implies increased concerns about security, protection, privacy and all the other 'side' properties that transform an experimental application into a service. Research on security for P2P streaming started to flourish, but no comprehensive security analysis over the current P2P solutions has yet been attempted. There are no best practices in system design, no (widely) accepted attack models, no measurement-based studies on security threats to P2P streaming, nor even general surveys investigating specific security aspects for these systems. This paper addresses this last aspect.

Starting from existing analyses and security models in the literature, we give an overview on security and privacy considerations for P2P streaming systems. Our analysis emphasizes two major facts: i) the Byzantine – Altruistic – Rational (BAR) model offers stronger security guarantees compared to other approaches, at the cost of higher complexity and overhead; and ii) the general perception (not necessarily the truth, but a commonplace belief) that it is necessary to sacrifice accuracy or performance in order to tolerate faults or misbehaviors, is not always true.

— **Keywords:** P2P streaming, IPTV, security, privacy

1 Overview

Peer-to-peer systems have gained more and more momentum over the last years as a means to access multimedia contents, albeit initially in form of file downloads. The evolution to streaming and multicast (e.g., TV) was just a consequence. Their power to accommodate large amounts of users, together with their resilience to churn, reliability, and low cost are some of the reasons why they are preferred over dedicated servers or content distribution solutions. In spite of these advantages, or maybe because of them, some P2P features make these systems more difficult to defend against some classes of attacks.

Security-wise, P2P streaming systems are more challenging than other P2P applications because they are more vulnerable to QoS fluctuations. Live streaming protocols, and TV in particular, are most sensitive to delay and delay jitter: it is enough for a host to be prevented from receiving some packets in time, and the user may grow dissatisfied with the quality of the delivery and leave the system altogether. If some other peers are connected to that machine, they will be damaged as well. From the watcher's viewpoint, even slight quality fluctuations, or choppiness, cause the viewing experience to loose appeal and the user to drop the service (or switch channels if others offer better quality).

Apart from their time-sensitive nature and bandwidth dependency, P2P streaming are susceptible to manipulation and threats at the transport and network layers. Clever attacks can compromise selectively the guarantees that a streaming session should provide, rendering some channels unusable, or making the broadcast unavailable in particular locations. Both events can be classified as targeted censorship violating the freedom of speech and expression. Analyzing the threat models in all these cases gives relevant indicators over possible risks and vulnerabilities in the transmission.

In what follows, we provide a brief security analysis of some of the existing P2P live streaming solutions, and show a classification of both attack points and

[*]This work is supported by the European Commission through the NAPA-WINE Project (Network-Aware P2P-TV Application over Wise Network – www.napa-wine.eu), ICT Call 1 FP7-ICT-2007-1, 1.5 Networked Media, grant No. 214412

Aspect	Influences what	Results into
Peer nodes	P2P protocol QoS	partitioning,censorship delays,isolation
Supernodes	P2P protocol QoS	partitioning,censorship delays,isolation
Overlay routing	data privacy QoS routing	data leaks delays partitioning,censorship
Application code	P2P protocol data privacy	censorship data leaks
Distributed data	data integrity	partitioning

Table 1: Common sources of vulnerabilities in P2P streaming. The aspect in the first column indicates the vulnerability, the second column refers to the system feature the vulnerability influences, while the third column shows a possible result of the vulnerability being exploited.

solutions to common vulnerabilities. We provide general considerations and features that novel P2P streaming proposals should consider in order to minimize the chances of attack.

The remaining of this paper is organized as follows. Section 2 discusses the threats and security models for P2P streaming applications, with specific attention to P2P-TV systems. Section 3 describes the easier, and hence readily realized, attacks in P2P streaming systems. The study continues with discussing security practices (Sect. 4) particularly for the tree and mesh overlays (these latter combined with data-driven dissemination). We conclude discussing the trade-offs of this type of mechanisms, open issues and future work.

2 Threat and Security Models for P2P Streaming

2.1 Threat Model

In any security analysis, it is important to consider all possible sources of risk against any elements assumed to be trusted. Discussing the *threat model* in general P2P systems involves considering the security of peer nodes, supernodes and overlay routing. An overview of the threat sources in P2P streaming applications is given in Table 1 and is detailed hereafter.

There are five major sources of vulnerabilities in P2P streaming:

Peer nodes Malicious or malfunctioning nodes can always alter protocol behavior. For instance, they may not reply to requests, or may reply or generate wrong messages, or they can collude. This can result into biasing the neighbor selection process of another node, thus into network partitioning or even censorship. Censorship has grave consequences, as the less number of users in the protocol, the poorer the quality of the transmission [16]. From the point of view of QoS, peers can also do delayed forwarding and hence jeopardize once more live streaming and TV systems.

Supernodes Supernodes do not always exist in P2P applications, but it is envisioned that they can greatly benefit applications requiring large bandwidth and low, constant delays. Supernodes bring similar vulnerabilities to streaming systems as common peer nodes; the emphasis however is on their higher responsibility in data dispersion: if superpeers do not behave fairly and honestly with all peers, they can bias the service toward preferred users. As a consequence, partitioning and censorship are more stringent at the supernode level. Supernodes become even more critical as some projects explore the possibility that they are controlled by ISPs in an effort to make P2P overlays and IP networks cooperate [14].

Overlay Routing Routing messages among peers aims at reliability and quality. Secure routing deals with both maintaining secure routing tables, and securely transmitting messages [25]. The data in transit can be sniffed and if the channel is not secure, it can even be leaked or modified. The dispatching of tampered data to fair peers depends on the security of the routing tables; not only routing can undergo malicious delaying, but also partitioning (sending tainted data to the same peers) and/or censorship (not sending anything to a group of peers).

Application Code Wallach [25] notices that the P2P code runs with numerous privileges on peer machines: it normally uses the network connection and the local hard drive. When unrestricted, local access and external communication lead to information leaks or malicious code installed on the local machine that could alter the overall P2P protocol. The remedy is twofold: sandboxing the P2P application to use just an isolated location on the local drive, and denying operations that are not coherent with its purposes to the P2P application. The application code poses a particular threat to user *privacy*, because embedded malware could leak sensitive information to non-authorized recipients.

Distributed Data Data integrity is essential in streaming and TV systems, because the purpose of the application is liveness. If a TV-channel is redistributed on the P2P system but part of the news/programs are altered with some users treated differently from others, this can lead to partitioning or in the worse case, loss of users.

Category	System Feature
System operation	Reliability Availability Dependability Node Autonomy Access Control
Content management	Authenticity Integrity Non-repudiation Confidentiality Anonymity

Table 2: Desirable security and privacy features for P2P streaming systems.

2.2 Security Model

If the threat model identifies the sources of potential jeopardy to the system, the *security model* identifies the aspects of the system that are jeopardized by the threat. In Table 2, we split these aspects into system operation and content management concerns.

For what system operation is concerned, we identify the following list of main properties to be granted to streaming systems in face of threats and attacks:

Reliability The up-time of the system in steady state is modeled by reliability. Reliability can be a global property of the entire system or it can refer to the vision of the system conditioned to one specific peer or a subset of peers. A single failure, even if recovered, implies loss of reliability.

Availability Is the ability of the system to be up and running. A system can be unreliable, yet highly available, simply because recovery from failures is faster than the user/application of the system can detect. In P2P streaming, for instance, churn can be a source of unreliability, since peers leaving implies that from the point of view of some other peers, a portion of the system has failed. However, topology reconfiguration can be fast enough to avoid the loss of any information so that the system remains available even from the perspective of peers that are affected by churn.

Dependability A system which is reliable and available may still suffer from correlated failures that make it non-dependable. Dependability is a subtler property of the system: it reflects the ability of a system to work and derogate services in critical moments. An example will clarify the point. Cellular telephone systems are in general reliable and available, however they are not dependable with respect to emergencies and civil protection: during accidents the cells covering the area of the accident become congested because people call with higher rate than normal and resources are locally insufficient; during natural disasters, besides the above phenomenon, normally the electricity fails, and the base stations do not have adequate power backup. In the context of P2P streaming and TV applications, the system may turn to be undependable because simple attacks can ruin specific event streaming (e.g., popular broadcasts) which causes a higher-than-average amount of traffic; in these cases, simple traffic-volume based attacks can jeopardize the most useful (or prized) events.

Node Autonomy This is a requirement specific to P2P systems. Each node is peer with all the others and its autonomous functioning should be guaranteed at all times: at any point during service, each node should be empowered to perform the actions that it is specified to perform at that step, without the need of external intervention. Any external call that the node makes at the protocol runtime is an indication that the node may not be autonomous.

Access Control Some applications (e.g., public TV) require no access control for service provision, but others may be limited to groups of authorized users: membership is controlled, and the system should provide means to protect membership in face of attacks, both for breaking the control and for denying service to authorized members.

The properties most delicate in a P2P TV or streaming system are availability and dependability; however, node autonomy may be a requirement to prevent censorship attacks, and access control can be fundamental for commercial services.

Considering now content management, we have the usual aspects and properties threatened, but also some that are specific to P2P streaming:

Authenticity and Integrity The data transmitted at protocol runtime must be guaranteed and not tampered with, and it must be guaranteed that it was emitted by the intended transmission entity.

Non-repudiation Refers to the situation when the nodes that received a certain piece of data cannot deny that they received it.

Confidentiality The content that is transmitted during the streaming process can only be used or retransmitted to other nodes involved in the protocol. This property interlaces with access control. In fact an access control system that prevents unauthorized participation to a streaming, but is not supported by a content management system that can prevent recording and later replication of the content becomes useless. Recent studies on commercial TV streaming solutions have shown that they do not perform encryption [4], which makes the protocol lose not only confidentiality but also authenticity and integrity.

Anonymity This is one of the most controversial properties, since in many contexts he property of a user to remain anonymous is associated to potentially unlawful activities. However, specifically in TV systems, the right of a user to watch a program without disclosing his identity is key to privacy protection and is guaranteed by broadcasting systems. This property should be guaranteed also by P2P streaming systems, not only in face of external observers, but also with respect to the other users of the same system.

Haridasan and van Renesse argue that not all applications need anonymity and confidentiality, but the features that matter most in frequent cases, are authenticity, integrity and non-repudiation [10]. Still, we have seen that anonymity becomes a key issue of privacy protection in TV systems. Non-repudiation, in the same systems, may be of secondary concern, unless a node can build claims on the fact that some information has not been delivered.

2.3 Assets to be protected

For P2P systems, there are two important values to be protected so that the protocol functions fairly: i) the data exchanged between peers, and ii) the hardware and software resources of each node. Exchanged data is essential. In streaming systems the data being shared has a time window associated with the playout time: the data turns stale when the playout. This adds a new dimension to the problem of data protection: delay makes data useless. Bandwidth is another important resource. As P2P systems are decentralized, it is usually easy for malicious peers to flood the system with their requests in such a way that they would exhaust the bandwidth of the system [5].

2.4 Auditing

Auditing is a detective means by which violations of predefined courses of actions can be identified. Auditing is an 'after the fact' measure and the outcome of its analysis influences future course of actions. Auditing requires the existence of logs with recordings of certain activity, the mechanism that is periodically triggered to write to these logs, and an auditor —the entity verifying the logs. As far as the checking mechanism is involved, auditing can be continuous —at certain time intervals or on all records— or probabilistic —at random moments of time or on random recordings.

In P2P systems, audit can function as a means to check whether a peer node functions according to a predefined contract or protocol. The idea of distributed audit in the sense that nodes trade local storage with storage on other nodes, is hinted in [25]. Of course, in order to perform it, the auditing method must be secured; this involves making sure that any nodes cannot influence what is being written in the logs, nor hide the logs themselves. Full access to query these logs must be entrusted to the requesting entity; moreover, the mechanism evaluating the events logged in the file must not misinterpret or ignore anything that was recorded. A simple way to ensure that most of these conditions are satisfied, is to impose a reward/punishment/incentive mechanism that makes the entities involved in the audit process cheat as little as possible.

Target	Attacks
Node autonomy	Membership/Eclipse attack Neighbor selection attack Collusion attack
Confidentiality and integrity	Forgery attack Pollution attack
Node authentication	Sybil attack
Dependability and availability	DoS omission attack

Table 3: Common attacks in P2P streaming systems.

3 Common Attacks in P2P Streaming Systems

Most serious attacks in P2P systems comes from the inside of the system and involve more than just a single node. This happens because only an internal node runs the internal protocols used between hosts, and can thus exploit them. In the BAR gossip model [15], for instance, nodes are known once they join the system and moreover, an unknown node has a very restricted set of actions that it can perform. Therefore, the security of P2P application should look to protect internal nodes from other (malicious) internal nodes. In what follows we will focus on some possible situations of vulnerability and describe the favorable conditions in which they take place.

Collusion attacks As observed in [10], these attacks occur when one malicious node compromises a set of nodes to conduct correlated attacks onto the whole system. This type of attack breaks the node autonomy requirement stated in the previous section. As expected, these are the most dangerous attacks since it may be extremely difficult to track down the attacker if nodes function correctly at each step or on short-term, while overall misbehaving or deviating the protocol on the long run.

Forgery attacks Forgery attacks break the condition of confidentiality and integrity of data mentioned in the previous section as a requirement of P2P streaming systems. The data being transmitted is tampered with, therefore a cryptographic technique as e.g., the use of message signatures, can easily solve the vulnerability.

Membership and Eclipse attacks With this type of attacks, the membership protocol or the way nodes are admitted into the overlay are compromised. A special type of membership attack is the Eclipse attack, where, as noticed in [24], an attacker which controls a portion of the overlay neighbor scheme, *eclipses* fair nodes by dropping or rerouting any messages meant for those nodes. In other words, in Eclipse attacks, the attacker can gain some control over the routing mechanisms in the P2P system.

Unstructured overlays are more susceptible to this type of attacks than the structured overlays; the latter do impose some constraints over the neighbors of one node, while the former do not. For this reason, the unstructured overlays use floods of random walks to gain knowledge of the network topology; the more they use these mechanisms, the higher the probability that an attacker will control more nodes in the system. One possible solution described in [24] is to use a mechanism that bounds the in-degree and out-degree of the nodes in the P2P overlay. In this way, an attacker is prevented from communicating with more nodes than what normally should.

Neighbor selection attacks These attacks refer to the situations in which an attacker controls the neighbor selection mechanism of some nodes, and makes them choose it as information provider. Malicious nodes can thus infiltrate and dominate sets of neighbors. The attacker will influence the way the overlay communicates and the neighbor selection process happens, so that it can control the traffic and subvert the whole system. These attacks are referred to as *epidemic* by [21], as fair nodes will “unknowingly reference compromised peers in their neighbor set”. Of course, the problem is even worse if the membership server is itself attacked in this way. One idea of solving this problem with Distributed Hash Tables is to identify the invariants in the placement of peers in the overlay, and detect attacks in the form of deviations from these invariants. A solution adapted to mesh-based systems is shown in [21].

Sybil attacks These attacks happen when the reputation mechanism established within the P2P system is compromised. Specifically, an attacker creates a large number of entities which bear the same disguised identity in order to become more powerful. Depending on how the id-s of nodes and reputation constraints are generated, the reputation system may be more or less vulnerable to such at-

tacks. The idea is that once disguised, the attacker profits from the trust that is given to the real entity it impersonates.
Guarding against such attack may involve a trusted third entity which certifies that a name or a reputation id is attached to the exact entity it is supposed to carry it. Therefore, certified node identifiers is one of the most straightforward techniques to repel masquerading. In addition to this method, auditing is another way to prevent the Sybil attack. An interesting solution employing auditing is provided in [24], where a node periodically challenges one of its neighbors to provide it with a list of that node's inbound contacts; if that list appears unfair or tampered with, then the requester node can act upon this discovery.

DoS attacks Denial of Service happens when malicious nodes send excessive amounts of requests or duplicate packets intended for their peers. The ability to bring a contribution to the streaming session is thus compromised, because a fair node would be flooded with useless messages or too many requests for it to handle. In this way, the resources of the system are exhausted with a relatively small effort on the attacker side. When the resource on which the attack focuses is bandwidth, the attack has been also termed as *request spreading attack* [5]. These problems were previously studied in the case of distributed systems as well as P2P streaming scenarios and there are several approaches in counteracting this type of attacks [6, 5, 26].

Omission attacks are at the other extreme than DoS attacks, implying that all the packet of data or just a part of it is not sent further according to the protocol specification. Again, just like for the DoS attacks, this behavior can compromise the whole P2P system even if a small number of peers collude. As noted by [10], the problem with this attack is that the guilt of a node cannot be proved easily.

Pollution attacks in P2P streaming occur when the attacker mixes junk pieces of data into the P2P distribution. In this way, the quality of the transmission decreases considerably: polluted chunks which arrive at fair peers degrade the protocol efficiency, and in the same time these peers will forward the *junk* to other peers and the whole effect will exponentially span over the network. Proof that the effects of this type of attack can be devastating in a streaming scenario are given by Dhungel et al., along with proposing four possible defenses: blacklisting, traffic encryption, hash verification and chunk signing [7].

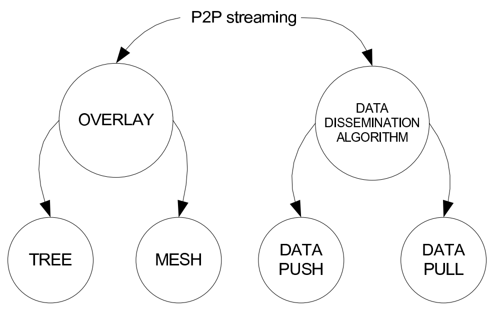

Figure 1: Important structural aspects in securing P2P streaming.

4 Security practices

In this section we provide a close-up on the existing security solutions in P2P streaming. We expose and discuss the vulnerabilities of each approach and then derive a few patters and conclusions that would help in protecting against attacks in P2P streaming systems.

As shown in Fig. 1, there are two building blocks of P2P systems to be considered: overlay topology and data dissemination mechanisms. The topology of the overlay defines how to connect each node in the network with the right neighbors; in other words, in a situation in which nodes are constantly joining and leaving the system, to find a solution in which each node sees as its neighbors only the nodes it is most interested (and is fair) to communicate with. The criteria to choose neighbor range from locality to certain QoS values.

In itself a non-trivial problem, the topology of the overlay is in tight connection with the application: the application domain determines the topology of the network, while in its turn the overlay topology influences runtime application aspects that can be either functional or non-functional: searching, routing, performance, efficiency, robustness [11, 2, 3]. In complex applications, where the topology changes dynamically, the mechanisms involved in the construction of the overlay have increased importance because they are invoked continuously; consequently, keeping these mechanisms protected against attacks becomes essential in order to maintain their compliance with the protocol schemes.

According to several other classification studies [28, 20, 18, 16] that there are two typical overlay topologies in P2P streaming applications:

1. Tree-shaped overlays: in which the overlay is usually built in the shape of a tree. This means that the way in which overlay nodes send and receive messages is structured and embedded in the overlay topology: The source is the root of the tree and leaf nodes receive but not redistribute the data;

2. Mesh-shaped overlays: where the overlay has the structure of a generic mesh. That is, every peer has one or more parent peers and a set of child peers and there is not a definite structure. The media is distributed among different peers and then each of them transmits the media further.

Apart from the overlay construction, the other defining aspect of P2P systems is the *data dissemination mechanism* among peers. That is, while the overlay deals with connecting a node with the right neighbors, the data dissemination algorithm is concerned with how to pick member pairs of nodes that communicate. Again, there are two or sometimes three basic ways of disseminating data in P2P streaming systems [16, 23]:

1. The *push*, or source-driven approach means that a peer transmits a chunk to its neighbors, assuming they do not have it yet; the directions in which the data is sent are determined by the parent-child relationship among nodes, be it a tree or mesh overlays. It is easy to see this way of performing data dissemination is prone to redundant pushes and thus to DoS attacks (e.g., flooding neighbors with data they already have), to neighbor selection and omission attacks (bias in where to push data);

2. The *pull*, or receiver-driven approach is an alternative to the previous scheme, by which a peer uses buffer maps to create pull schedule with the peers it decides to communicate with. A peer requests the information it is missing. This approach is more robust than the previous, but vulnerable to collusion: peers that already have data may not advertise it to others;

3. the *data-driven* approach does not differentiate between push and pull interactions, but allows each entity to *randomly* choose the neighbors it wants to exchange information with. Epidemic algorithms are instantiating this approach, an gossiping ones in particular. Gossip in P2P is a data dissemination mechanism that does not employ the support of the overlay but can manage itself overlay patterns. It is also useful in data aggregation and resource allocation [13]. The reason for its popularity is that gossiping mechanisms are simple and more robust than others. Security-wise, we believe they are interesting to study because they are more general than the push and pull mechanisms. The biasing vulnerabilities suffered by the other approaches are easily solved with gossip, since it is not easily predictable in which way data flows. In addition, as previously noted [23], gossip-based mechanisms are less sensitive to peer dynamics, thus to churn.

For the reasons above, in what follows we will analyze the two overlay approaches in conjunction with the gossip protocol from a security standpoint. We will bring into light what are the vulnerabilities and strengths induced to the systems that adopt these approaches.

4.1 Tree-based Approaches

Generally, streaming in tree-based overlays imposes that the source of the media is the root of the tree, and that the rest of the peers are children of the source and children/parents among themselves. The path that the data must follow in this case is fixed: first from the source to the first-order parents, then from those to their children, and so on. A visible functional problem that occurs with this kind of overlay structure is simple: the efficiency of the hierarchy is overcome by the large imbalance between parent nodes and leaf nodes (parents forward data while leaves do not, so everybody wants to be a leaf). Historically, the solution to this issue took the form of *multi-tree* overlays, as [21, 20] notice, in other words: more leaves, more trees. This approach leads to distributing the data in multiple distinct trees.

There are other problems related to the topology of this overlay [28, 16], and they are summarized in Table 4. Since in tree overlays each node receives data from only one source node, bandwidth fluctuations can be highly damaging, and paths that are closer to the root are more likely to turn into bottlenecks. Security-wise, minor protocol deviations of single nodes can affect easily entire subtrees. More, when nodes closer

Envisaged solutions	Problem /attacks
Using multi-trees, gossip	Imbalance root vs. leafs Bandwidth fluctuation, bottleneck Protocol deviations on parent node
Monitor,acknowledgment	Identifying malicious nodes DoS, omission Membership attacks
Signatures	forgery, repudiation
Not yet solved	Sybil attacks

Table 4: Common fairness and security issues in tree-based P2P streaming systems.

to the root leave the system (e.g. they crash or are attacked), they leave unserviced a large percentage of the nodes.

Trying to solve the above problems, Zhou and Liu have combined the tree overlay with gossip data dissemination so that the two approaches would compensate each other's faults [28]. Because the tree model is brittle but yet time-efficient, it is used as a second option: by default all data is transmitted by gossiping, and if a node does not receive anything for a certain period of time, the tree overlay will help it obtain the data from its parent. Security-wise, because the protection level for a composite system is the protection level of its weakest link, this solution is prone to all vulnerabilities of the tree overlay.

Another solution adopted in tree-based overlays is presented by Shetty et al. in [22]. In the tree-shaped overlay, the streaming quality depends on the cooperation of the non-leaf nodes (namely the nodes in the overlay tree that are neither leaves nor the source). The possible attacks that are considered are thus DoS, omission, forgery and repudiation attacks. Shetty et al. identify that one of the problems with the current security solutions in P2P streaming is that they cannot identify the malicious nodes themselves, just the fact that there are malicious nodes. This is because in overlay multicast streaming, if a fair peer receives tampered data, it cannot determine if its parent is malicious (since its parent might have taken that data from some other peer).

A *signed acknowledgment* together with a *random monitoring* scheme were shown to be a solution to detecting the exact attacker peers. The former mechanism is used by peers to prove their fairness, while the latter helps trusted peers to monitor in a random fashion some of their peers suspected to malfunction. The problem with this solution design is that it relies on *one single session trust manager*, which imposes a scalability issue and a single point of failure. The trust manager decides whether a peer is malicious or not, by receiving 'complaints' from peers and employing a localization scheme. If it cannot detect the exact location of the omission or forgery attack, the trust manager will decrease the trust value of both peers (the reporter and the reported). Otherwise, the child and the tree that inherit from the reported node are moved to another peer-tree.

On the downside, this solution does not handle collusion attacks in the form of Sybil attacks: if a malicious node assigns itself several identities, then the only way to prevent it from gaining control is by assigning strong identities from a central identity manager, or conversely, to implement a punishment scheme, where a malicious node can be evicted, provided bad service or punished in money. Moreover, having the trust manager as one fixed peer throughout all sessions is a single point of failure, therefore passing this responsibility to different nodes with high levels of trust for each session, should be a straightforward improvement.

A common solution in tree-based approaches is given by SecureStream [10]. SecureStream is able to repel several types of attacks because of its multiple intrinsic mechanisms that help in eliminating most vulnerabilities. For example, in order to protect itself against membership attacks, SecureStream uses the *Fireflies* protocol in which members monitor each other in case of failure, by pinging each other. The pinging protocol is more complicated in that is it based on the gossip protocol, that each node is assigned certain other nodes to monitor, and there is a limitation on the number of neighbors that a certain node can accuse of failures (this comes to stop malicious nodes from accusing too many fair ones). Obviously, each peer has a predefined set of neighbors.

To guarantee the integrity of the data being streamed over the network, SecureStream avoids signing groups of packets with asymmetric keys, but computes content hashes that are signed with the sender's private key. In order to minimize the number of malicious neighbors, this solution takes a smart approach: in each round, the source of the transmission notifies its neighbors that it has available packets. These packets last as long as an availability window. Each neighbor, in its turn, requests from the source the packets that it misses (this is the interest window), but trying not to overload the source. Whenever they obtain a new packet, these neighbors in their turn will send notifications to other neighbors, and so on. Overall, this method of dissemination is resilient to attacks, especially omission attacks: if a peer does not reply with the promised packet, then another one is contacted. It should be noted here that the nodes located close to the source do not necessarily receive packets faster.

Moreover, there is a limit in the number of requests that arrive at a peer (this repels the possibility of node flooding). In order to eliminate free riders, an auditing mechanism ensures that all nodes contribute to the protocol at least as specified by a fixed limit. Auditing is distributed: local auditors are periodically elected to evaluate the contribution of each of their neighbors. The punishing of nodes that behave malicious, as well as employing the pull-approach that disables attackers to gain control deterministically over sets of nodes, these are the salient features that make SecureStream tolerant to Byzantine attacks. In comparison to the BAR-Gossip approach described in Sect.4.3, it can be noted that the source does not need to know all the members of the protocol: here the membership is dynamic, so scalability is not bounded.

4.2 Mesh-based Approaches

Mesh-shaped overlays carry less structure in comparison to the tree solutions. A membership server may keep track of the existing nodes in the system if required, and there is no fixed flow that data must follow. Recent works see these meshes as unidirectional, in the sense that nodes always have an inbound and outbound degree. The number of neighbors that a node can accept is limited only by resources.

Empirically, a comparison between multi-tree and mesh-based overlays in streaming scenarios is given in [20, 18] and the conclusion is that mesh approaches are more robust. The study shows that overlays that are mesh-shaped bear better performance when the size of the network is large, the streaming rates are high, and the nodes have high bandwidth and low round-trip times. On the downside, they introduce a large number of duplicate packets in the network. Multi-trees, in comparison, are more time-efficient in heterogeneous networks, but on large scales they perform worse than meshes. Some issues of the mesh-based overlay are shown in Table 5.

Two classical examples in mesh-based overlay solutions are Prime [17] and CoolStreaming [27]. In CoolStreaming, the approach is data-driven: the data availability drives further propagation; gossip communication is used to disseminate network membership and content availability. Building on CoolStreaming, which

Envisaged solutions	Problem
Monitor and audit schemes	Identifying malicious nodes Flooding, omission attacks Membership attacks
Not yet solved	Collusion attacks Data diffusion problems Acknowledgment / Repudiation pb.

Table 5: Common security issues in mesh-based P2P streaming systems.

does not form a typical mesh but several trees onto an initial mesh, Prime is historically one of the first mesh streaming systems. In Prime, content delivery (or swarming) has two phases: push reporting is done by parents (announcing availability of data) and pull-reporting by children (retrieving data using some packet scheduling algorithm). For advertising the new content dedicated links are in place (diffusion connections) over diffusion trees.

From a security point of view, neither Prime nor CoolStreaming protect themselves from effects of several types of attacks. For example, Prime assumes that peers are all fair and connect in a random fashion one with another. Moreover, it is assumed that the mesh formed by peers is directed – every node has an inbound and an outbound degree. Since there is no mechanism to check whether instead of randomness, some nodes can connect only to certain their nodes on purpose, so coalitions (or network partitioning) can form. Apart from the simple collusion attack, the integrity of the diffusion connections is not enforced. There is no mechanism in place to make sure that one node declares its content availability to all or none or a fraction of its neighbors; there is no guarantee that the bandwidth, outgoing and ingoing degree of each node are used properly. Even more importantly, there is the issue of acknowledgement and repudiation: there is no guarantee that peers eventually receive (some) streaming data.

4.3 Gossiping and Byzantine Faults

Gossip algorithms are mostly used for content dissemination in dynamic distributed systems. They rely on what is termed as “probabilistic exchange of information” [13]: nodes use randomness in determining to which neighbor they would forward/retrieve data. Gossip protocols in general are robust, scalable and rapidly spreading information, their only fault being that they might generate more traffic than the nodes can handle. This traffic quantity, however, is a price that gossip protocols pay for redundancy. Still, even if dangerous, they can become a very useful tool for rapid and scalable epidemic dissemination when proper attention is given to the message propagation mechanism.

Based on an analytic view [13], there are three main features of common to gossip protocols:

- Peer selection is about how a peer A selects another peer B in order to interact with it. This choice must be done randomly in a typical gossip protocol, but it can be biased if the scheme is undergoing an attack: peer A might see that there are three other available peers B, C and D, but its choice on communicating with B might not be a random choice. For this reason, securing the gossiping protocol involves adding a mechanism to enforce that peer selection is random and cannot be tampered with.

- Data exchanged is an application-dependent choice belonging to each of the two peers involved in the exchange. It is not necessary that peers exchange data —they might as well exchange references to other peers. From this point of view, it is essential that neither of the two parties cheats: one or both peers give the other junk data. This constraint can be enforced by checking the validity of the data right before the content exchange happens but before the communication between the two peers ends. In addition, the security of the channel established between the two peers must be enacted.

- Data processing refers to how each peer handles the data it has received. It involves either storing the message for the next round or handing it to an application. Neither the former nor the latter problem are of any concern in this paper.

Generally, gossip protocols are scalable and very reliable; by randomly selecting peers, gossiping avoids message losses or node failures. Still, as noticed in [9], gossip schemes cannot deal with situations in which attackers falsify the information being disseminated from one peer to another, because gossip protocols do not verify the data being exchanged. This subclass of problems goes deep into the class of Byzantine faults. For this reason, there are a number of solutions trying to address these issues, and hereafter we will briefly discuss some of them.

Compared to previous structured overlay approaches, a more realistic solution (from the point of view of Byzantine faults) to P2P streaming is given by Dolev et al. in S-Fireflies [8]. The purpose of the P2P overlay network is double: tolerate Byzantine nodes and self-stabilize (to adapt dynamically to churn). S-Fireflies builds probabilistic graphs (random graphs) with nodes of low in- and out-degrees, that is stable in terms of Byzantine presence. The result of the algorithm is the enforcement of a 'rigid' complete graph, so that nodes get to know all their neighbors (with a high probability). Onto this robust connection graph, Dolev et al. establish a monitoring mechanism that uses gossip to report node failures and propagate transmission rounds. The protocol is verifiable at the level of each node, so any peer can verify that another node communicates with correct neighbors. Moreover, there is an enforcement mechanism for nodes not to impersonate other nodes. In terms of the streaming session, there is a system-wide process with the purpose of updating all nodes in the network; even if a quarter of the whole number of nodes are temporarily faulty, the system is still able to recover.

The above solution provides some useful mechanisms for a P2P network to timely adapt to Byzantine faults: the construction of the random graph coupled with verifiable adaptiveness. However, although it controls the effect of malicious behavior in its general form, it does not deal with the causes of this behavior: nodes should be encouraged to participate in the game, because the more peers, the better the performance of the streaming session.

BAR-Gossip (Byzantine-Altruistic-Rational) [15] is the next step in making P2P streaming more secure and less treacherous. This new model increases the safety and liveness guarantees formerly offered by Byzantine fault tolerance because it features an incentive-based mechanism for non-byzantine peers that may become malicious. This solution leverages on three different peer behaviors: purely byzantine, altruistic and rational nodes. The modification to the original peer selection scheme in gossip, is that this process is *pseudo-random and verifiable*. The strength of the overall protocol is multiple-fold:

- it is extremely robust when faced with Byzantine and selfish nodes,

- it can face collusion attacks,

- it provides stable short-term throughput, while the bulk of the other approaches target maximizing bandwidth on the long term,

- it is usable for short-window transmissions/streaming,
- it does not use reputations, so Sybil attacks are already dealt with.

BAR-Gossip functions in rounds, or transmission sessions; in every round, the source transmits the correct packets, and then peer nodes propagate these packets simultaneously in two schemes: a balanced exchange, and an optimistic push (of non-expired packets) protocol. BAR-Gossip also details the explicit exchange protocol between nodes, and takes all actions to balance the amount of information that is swapped between the two sides (very much like tit-for-tat). Moreover, the protocol seeks to monitor and reward/punish individual node activity so that there is an overall equilibrium between all nodes involved. Some solutions that this protocol gives to common problems, are:

- *neighbor selection attacks*, because selection becomes now verifiable. On the downside, however, the convergence is not as fast as in the traditional gossip way, because the mechanism of selecting neighbors replaces *just* randomness with pseudo-randomness **and** verification performed by the selected node onto the selector.
- *nodes lying about their history* It is no longer desirable to lie in the short-term, because it is no longer in their interest neither to under-report nor to over-report their packet history.
- *forgery attacks* are repelled because of a clever mechanism by which data is first traded and verified, and then exchanged for good. If the data is forged, this would be noticed before the actual exchange, so the potential receiver would realize the attack and report it. Again, making sure that data is not forged prior to the exchange involves more overhead in the transmission.
- *stability on the short term* is also grounded on the notion of Nash equilibria, so that any node would consider that its peers are following the protocol. This belief is actually an incentive to act by the protocol, to the node itself.
- *no free-riders* Eliminating free-riders is achieved by allowing junk updates, to compensate for the free-updates of an altruistic node.

Nevertheless, BAR-Gossip has its limitations. First if all, it only supports a static membership system, that is —all participating nodes first subscribe to the broadcaster before any round— to this end, the system gains a centralized identity management scheme with a static list of node id-s. In the case of large amounts of nodes that come and leave, this could turn into a scalability problem. In addition, by using the comments in [1], the question of how would nodes discover themselves can arise; discovery is arguably the heart of gossip-based protocols, so a discovery solution should consider the topology of the network, and should be as decentralized as possible so that nodes can use it at all times.

Furthermore, it can be noticed that in the case of the optimistic push protocol, nodes are likely to waste bandwidth by sending junk —this again can turn into a problem if bandwidth is scarce or the quantity of junk that is being sent is large. From this points of view, an interesting idea would be that of Martin in [19], where the efforts are concentrated toward leveraging on altruistic nodes to carry the burden of rational nodes. In other words, in the BAR-Gossip solution, altruistic nodes and rational ones behave in the same way according to the specification; however, rational nodes may refuse to participate in some computation if the cost of their involvement is higher than their utility. In this case, performance of the overall system can be improved if altruistic nodes take upon themselves the work that was refused by the rational nodes. Of course, burdening altruistic nodes should be done with a reward, as much as 'anarchy' (rational nodes refusing participation) should be punished.

5 Discussion

There are a number of vulnerabilities that P2P solutions are prone to. When it comes to live streaming, problems get worse because of the bandwidth demand and timeliness of this type of systems. Seamless performance and attack-proof design are impossible to achieve at the same time. Even worse, given the large variety of attacks, countering all or most of them is even more challenging.

5.1 Tradeoffs: Security vs. Performance

BAR-Gossip is able to overcome a very large number of different attacks from the list in Section 3, but there is a number of trade-offs it has introduced in return (see

Tradeoff between...	...and
punishing innocent nodes	fairness incentives
neighbor choosing	dynamic peer membership
bandwidth utilization	allowing fake data delivery
timeliness	punishment for misbehaving nodes
performance	cryptographic schemes used

Table 6: Tradeoffs in BAR-Gossip.

Table 6). The essential idea it applies in order to repel a large range of attacks is to *encourage nodes to behave*. If nodes misbehave, then they are punished; this can be easily implemented by some form of penalty or by placing the wrong-doers further away from the source of broadcast, thus ensuring that their possibility to harm is diminished. However, it is easy to notice that punishing one node may involve punishing the nodes that the current node will communicate with; thus, the effect of the punishment is likely to occur to innocent nodes as well. This is a trade-off in its own way: the decision to punish also some nodes that do not misbehave, in order to 'set an example' for other nodes.

Neighbor choosing is another trade-off. Instead of allowing a tree-like structure in which nodes know from the beginning who to communicate with, it is wiser, from a security point of view, to sacrifice some performance in order to eliminate vulnerabilities as much as possible. Using the pseudo-random scheme together with selection verification is a far safer approach that using a centralized membership directory, which apart from bearing scalability problems, is also a single point of failure. From this point of view, there is another trade-off remarked by Jesi in [12]: BAR gossip is able to let neighbors control how random the peer selection process is for certain nodes, at the cost of ruling out dynamic peer membership. This is the reason why all peers first have to register themselves to the broadcaster, before participating to the streaming round. If it were to perform this sacrifice the other way around, Jesi's solution caters for dynamic membership where nobody can control how randomly a node selects its neighbors.

Bandwidth utilization is the reason of another compromise. Since there can be nodes that offer packets (data) at a lower cost than any other nodes, this would imply that all requesters would crowd to use these free suppliers; balance is brought into this scene by introducing the possibility that requester nodes receive junk if they turn into a burden for the altruistic ones. In this case however, bandwidth is wasted with the sole purpose of 'teaching a lesson' to the misbehaving nodes.

Timeliness of transmission is very seldom compromised in P2P streaming solutions. This is straightforward for the simple reason that users hate choppiness or low quality, and as long as they encounter any of them, they leave the system. Because this is not in the system's best interest —the more users, the higher the combined bandwidth— then timeliness is not a parameter to be touched. However, if nodes misbehave, punishment can take the form of placing these nodes farther from the source (if applicable), with the clear effect of obtaining packets which are closer to expiry.

Furthermore, protection mechanisms come in place over the transmission data to insure its integrity, confidentiality and fair-use. Overall, these mechanisms, ranging from signing, to briefcase negotiation and eventually briefcase exchange, function in the detriment of performance. Each node consumes bandwidth and processor cycles for the system's best interest, even if it is not always in its own interest. Of course, nodes should not be allowed to act by themselves, and as long as the risks that they encounter are the same for all other nodes, the same measures should be taken for them all. Needless to mention, again —countering any forms of 'anarchy' assures the well-being of the entire system; for that, an incentive/punishment technique needs to be in place to protect the streaming process.

5.2 Further Work

Currently, collusion attacks remain one of the most problematic types of attacks in P2P streaming systems. Collusion do not necessarily mean the protocol is not respected, but it can also refer to a slight deviation from the protocol, which is hard to locate and cure. In BAR-Gossip, for instance, collusion may occur to rational nodes: a group of nodes that are not satisfied with the previous exchanges, group together in order to maximize collective utility. Their uncooperative behavior toward the rest of the network can manifest in a slower propagation of messages in the exterior of their group, compared to the one within the group. Again, this does not disrupt the overall protocol, it just decreases its effectiveness.

In addition, more work is needed as to analyze the utility of nodes to deviate from the protocol; finding a bound for this utility, correlated with the application, would be useful in finding quantitative incentives for not deviating from the protocol. Moreover, churn in streaming systems remains another open problem, for which one possible solution can be that of self-adaptive networks, as described in 4.3.

6 Conclusions

A common security approach applicable in any domain is a combination of preventive and detective measures. In practice, by no means the possibility of countering attacks can be null, so as long as the system keeps some functionality, circumvention of its mechanisms remains probable. Thus, it is not only the preventive/reactive measures that require attention, but also the detective ones. Specifically, once an attack happened, it needs to be confined to as small an area of the P2P network as possible. Once the attacker cannot easily gain control over a bigger portion of the system, some mechanisms need to be in place to detect its actions and eventually its location. It is not always easy to detect who the malicious node is (it is always from within the network, assuming that no other hosts can interfere with the protocol) but in practice there are some techniques that can be used for this purpose: one of them is the one requiring a trust manager [22], with the addition that this machine needs to be replaced periodically, and should not be central to the whole network (since we want to eliminate single points of failures). A complementary approach is to use the mechanism of incentives and punishments, where nodes are stimulated to stick to the protocol; if they do not comply, then a distributed monitoring mechanism (performed via the malicious nodes' neighbors) should help in enforcing a punishment onto the bad performers. Moreover, audit is another technique not to be overlooked: it can provide essential information on the behavior of the entire protocol as well as of individual nodes.

As shown in the previous examples, tree-based streaming in P2P networks are not only vulnerable to protocol failure, but are also far faster contaminated by an attacker if the hierarchy of nodes is fixed. In the eventuality no other constraints are put onto the degree of each node (either inbound or outbound), then the structure is vulnerable and cannot contain most attacks. This happens because in a tree, if a node is contaminated, then its children will be too. In a mesh, on the other hand, the infection spreads in a one-by-one fashion rather than in a one-to-many fashion.

Compared to tree-based streaming, mesh and gossip approaches are more robust, scalable and Byzantine-tolerant. The largest amount of recent works concentrate on either of these two approaches, and attach a wide variety of additional mechanisms in order to counteract as many attack types as possible. The overall trend is to delegate many monitoring and security functions to each peer instead of keeping separate entities exclusively for these tasks. Membership and neighbor selection mechanisms are driving the flow of any different protocol, while tune-ups mostly try to leverage churn and node coalitions.

References

[1] L. Alvisi, J. Doumen, R. Guerraoui, B. Koldehofe, H. Li, R. van Renesse, and G. Tredan. How robust are gossip-based communication protocols? *SIGOPS Oper. Syst. Rev.*, 41(5):14–18, 2007.

[2] D. Carra, R. Lo Cigno, and E. W. Biersack. Graph Based Analysis of Mesh Overlay Streaming Systems. *IEEE Journal on Selected Area in Communications (JSAC)*, 25:1667–1677, Dec. 2007.

[3] D. Carra, R. Lo Cigno, and E. W. Biersack. Stochastic Graph Processes for Performance Evaluation of Content Delivery Applications in Overlay Networks. *IEEE Transactions on Parallel and Distributed Systems*, 19:247–261, Feb. 2008.

[4] D. Ciullo, M. Mellia, M. Meo, and E. Leonardi. Understanding P2P-TV Systems Through Real Measurements. 30 2008-Dec. 4 2008.

[5] W. Conner and K. Nahrstedt. Securing peer-to-peer media streaming systems from selfish and malicious behavior. In *MDS '07: Proceedings of the 4th on Middleware doctoral symposium*, pages 1–6, New York, NY, USA, 2007. ACM.

[6] W. Conner, K. Nahrstedt, and I. Gupta. Preventing DoS attacks in peer-to-peer media streaming systems. In *Proceedings of the 13th Annual Conference on Multimedia Computing and Networking (MMCN'06)*, San Jose, CA, USA, Jan. 2006.

[7] P. Dhungel, X. Hei, K. W. Ross, and N. Saxena. The pollution attack in P2P live video streaming: measurement results and defenses. In *P2P-TV '07: Proceedings of the 2007 workshop on Peer-to-peer streaming and IP-TV*, pages 323–328, New York, NY, USA, 2007. ACM.

[8] D. Dolev, E. N. Hoch, and R. van Renesse. Self-stabilizing and byzantine-tolerant overlay network. In E. Tovar, P. Tsigas, and H. Fouchal, editors, *Proceedings of the 11th International Conference on Principles of Distributed Systems (OPODIS'07)*, volume 4878 of *LNCScience*, pages 343–357, Guadeloupe, French West Indies, Dec. 2007. Springer.

[9] K. Han, G. Pei, B. Ravindran, and E. Jensen. Real-time, byzantine-tolerant information dissemination in unreliable and untrustworthy distributed systems. In *Proceedings of the IEEE International Conference on Communications (ICC'08)*, pages 1727–1731, may 2008.

[10] M. Haridasan and R. van Renesse. Secure-Stream: An intrusion-tolerant protocol for live-streaming dissemination. *Computer Communications*, 31(3):563–575, 2008.

[11] M. Jelasity, A. Montresor, and O. Babaoglu. T-Man: Gossip-based Fast Overlay Topology Construction. *Elsevier Computer Networks*, 53:2321–2339, 2009.

[12] G. Jesi. *Secure Gossiping Techniques and Components*. PhD thesis, PhD Thesis, University of Bologna, Department of Computer Science, may 2006.

[13] A.-M. Kermarrec and M. van Steen. Gossiping in distributed systems. *SIGOPS Oper. Syst. Rev.*, 41(5):2–7, 2007.

[14] E. Leonardi, M. Mellia, A. Horvath, L. Muscariello, S. Niccolini, and D. Rossi. Building a cooperative P2P-TV application over a wise network: the approach of the European FP-7 strep NAPA-WINE. *Communications Magazine, IEEE*, 46(4):20–22, 2008.

[15] H. C. Li, A. Clement, E. L. Wong, J. Napper, I. Roy, L. Alvisi, and M. Dahlin. BAR gossip. In *Proceedings of the 7th SIGOPS Symposium on Operating Systems Design and Implementation (OSDI'06)*, page 14, Seattle, WA, 2006. USENIX Association.

[16] Y. Liu, Y. Guo, and C. Liang. A survey on peer-to-peer video streaming systems. *Peer-to-Peer Networking and Applications*, vol. 1, March 2008.

[17] N. Magharei and R. Rejaie. PRIME: Peer-to-peer Receiver-drIven MEsh-based streaming. In *Proceedings of the 26th IEEE International Conference on Computer Communications (INFOCOM'07)*, pages 1415–1423. IEEE, 2007.

[18] N. Magharei, R. Rejaie, and Y. Guo. Mesh or multiple-tree: A comparative study of live P2P streaming approaches. In *Proceedings of the 26th IEEE International Conference on Computer Communications (INFOCOM'07)*, pages 1424–1432, May 2007.

[19] J.-P. Martin. Leveraging Altruism in Cooperative Services. Technical Report TR-2007-76, Microsoft Research, Cambridge, June 2007.

[20] J. Seibert, D. Zage, S. Fahmy, and C. Nita-Rotaru. Experimental comparison of peer-to-peer streaming overlays: An application perspective. Technical Report CSD TR 07-020, Purdue University, 2007.

[21] J. Seibert, D. Zage, and C. Nita-Rotaru. Won't you be my neighbor? Neighbor selection attacks in mesh-based peer-to-peer streaming. *Purdue University Technical Report*, 2008.

[22] S. Shetty, P. Galdames, W. Tavanapong, and Y. Cai. Detecting Malicious Peers in Overlay Multicast Streaming. Florida, USA, 2006.

[23] T. Silverston and O. Fourmaux. Source vs data-driven approach for live P2P streaming. In *ICNICONSMCL '06: Proceedings of the International Conference on Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies*, page 99, Washington, DC, USA, 2006. IEEE Computer Society.

[24] A. Singh, M. Castro, P. Druschel, and A. Rowstron. Defending against eclipse attacks on overlay networks. *Proceedings of the 11th workshop on ACM SIGOPS European workshop*, page 21, 2004.

[25] D. S. Wallach. A Survey of Peer-to-Peer Security Issues. In M. Okada, B. C. Pierce, A. Scedrov, H. Tokuda, and A. Yonezawa, editors, *Proceedings of the Mext-NSF-JSPS International Symposium on Software Security – Theories and Systems*

(ISSS'02), volume 2609 of *LNCS*, pages 42–57, Tokyo, Japan, Nov. 2003. Springer.

[26] J. Yang, Y. Li, B. Huang, and J. Ming. Preventing DDoS attacks based on credit model for P2P streaming system. In *ATC '08: Proceedings of the 5th international conference on Autonomic and Trusted Computing*, pages 13–20, Berlin, Heidelberg, 2008. Springer-Verlag.

[27] X. Zhang, J. Liu, B. Li, and Y.-S. Yum. CoolStreaming/DONet: a data-driven overlay network for peer-to-peer live media streaming. In *Proceedings of the 24th IEEE International Conference on Computer Communications (INFOCOM'05)*, volume 3, pages 2102–2111, Mar. 2005.

[28] M. Zhou and J. Liu. A hybrid overlay network for video-on-demand. In *Proceedings of the IEEE International Conference on Communications (ICC'08)*, pages 1309–1311, 2005.