Ripple-Stream: Safeguarding P2P Streaming Against DoS Attacks

Wenjie Wang, Yongqiang Xiong, Qian Zhang

Abstract— **Compared with file-sharing and distributed hash table (DHT) network, P2P streaming is more vulnerable to denial of service (DoS) attacks because of its high bandwidth demand and stringent time requirement. This paper studies the design of DoS resilient streaming networks using credit systems. We propose a novel framework—ripple-stream—to improve DoS resilience of P2P streaming. Ripple-stream leverages existing credit systems to introduce credit constraints in overlay construction such that malicious nodes are pushed to the fringe of overlay networks. Combining credit constraints with overlay optimization techniques, ripple-stream can achieve both DoS resilience and overlay efficiency. Our evaluations show that ripple-stream can effectively reduce the convergence time of overlays under attacks and significantly improve the average receiving data rate of the participating peers.**

I. Introduction

Peer-to-peer networks, especially P2P file sharing networks, have been quickly adopted by large Internet communities in the past few years. Recently, the emergence of P2P streaming service, such as conference broadcasting [1] and Internet TV [2], demonstrates its potential to deliver high quality media streams to a large audience. The thriving of P2P networks starts to attract denial of services (DoS) attacks. It is well known that P2P file sharing networks are under intense attacks from the music industry with the intention to reduce illegal music swapping [3]. Recent researches begin to study the effect of such DoS attacks in P2P systems. Various defense schemes have been proposed, including fair resource allocation [4], randomized peer selection [5], and secure routing updates [6]. However, the existing defense schemes share the following drawbacks. On one hand, existing approaches study the DoS attacks case by case instead of providing a unified solution. On the other hand, in existing approaches, peers don't share DoS detection results so each has to detect malicious behaviors on its own. Moreover, some attacks cannot be efficiently detected without cooperation among peers.

Compared with the widely used file-sharing networks, P2P streaming networks are more vulnerable to DoS attacks for the following reasons. 1) Streaming, especially video streaming, usually requires high bandwidth. A certain amount of data loss could make the whole stream useless. 2) Streaming applications require their data to be delivered in a timely fashion. Data with a missed deadline are useless. 3) A streaming network usually consists of a limited number (sometimes only one) of data sources. The failure of the data source could bring down the whole streaming system. Currently we are not aware of any systematic study on DoS attacks and defenses specifically targeting P2P streaming networks.

In this paper we first identify the possible attacks on P2P streaming networks. We then propose a generic DoS resilience framework named ripple-stream. To identify DoS attackers and prevent the system from being corrupted by malicious nodes, the ripple-stream framework employs a credit system to allow peers to evaluate other peers' behaviors and introduces a credit-constrained peer selection mechanism to organize the overlay. In a ripple-stream based overlay, peers share the credit information with each other, peers with high credibility are kept in the "core" of the overlay structure. Malicious nodes, with low credibility, are pushed to the fringe of the network. While enforcing the credit constraints, ripple-stream customizes techniques such as triangular optimization and random node recovery to guarantee overlay efficiency. Moreover, the ripple-stream framework is designed to be flexible so it can be applied to different overlay networks and P2P streaming schemes. It is also designed to be extensible to incorporate existing and future DoS defense mechanisms.

Our evaluation results show that under typical attack scenarios, ripple-stream can effectively shorten the convergence time of overlay networks and significantly improve the data rate for overlay peers. When there are no malicious nodes, ripple-stream can achieve good overlay quality even with the credit constraints it imposes

This work was done while Wenjie Wang was a visiting student at Microsoft Research Asia.

Wenjie Wang is with the University of Michigan at Ann Arbor, E-mail: wenjiew@eecs.umich.edu. Yongqiang Xiong is with Microsoft Research Asia, E-mail: yqx@microsoft.com. Qian Zhang is with Hong Kong University of Science and Technology, E-mail: qianzh@cs.ust.hk.

on the connectivity among overlay peers.

Note that in this paper, first of all, we are not designing a DoS-resilient credit system. Ripple-stream can work with future or more advanced credit system to improve its efficiency. Secondly, currently we only focus on *internal* DoS attacks that are launched from peers that run malicious streaming clients. These attacks are more severe than external attacks that are launched on the network level. We discuss external attacks in Section VI.

The rest of this paper is organized as follows. We first analyze the possible DoS attacks targeted on P2P streaming. The ripple-stream scheme is presented in Section III. The performance evaluation is shown in Section IV. We discuss the related work in Section V and conclude in Section VI.

II. DOS ATTACKS IN P2P STREAMING NETWORKS

We categorize the possible DoS attacks on P2P streaming networks into two categories: attacks on the control plane and attacks on the data plane.

A. DoS Attacks on the control plane

We first identify four unique attack schemes targeting P2P streaming networks: *RTT cheating, accepting too many downstream peers, connecting to multiple upstream peers,* and *advertising fake data availability.* The first three attacks can cause severe damage in single-tree overlays, where a single tree is maintained to delivery the streams, while the last one only works in multiple-tree overlays where a peer may receive data from multiple peers in parallel [2].

For *RTT cheating*, a malicious node can either inflate its RTT distances to other peers to discourage them from connecting back or deflate the RTT distances to attract connections. RTT cheating can degrade the quality of the constructed overlay [7]. This type of attacks can also be applied to other network metrics, such as bandwidth.

The purpose of *accepting too many downstream peers* is to corrupt the quality of their data stream. Such attacks can be used together with RTT cheating to affect more peers. A malicious node can disconnect its downstream peers frequently to create instability in the overlay.

The objective of *connecting to multiple upstream peers* is different with accepting too many downstream peers. Peers in streaming overlays limit the number of connections they can accept due to the high data rate of streaming. By connecting to a large number of upstream peers, a malicious node can deplete the degree resources in the overlay.

Advertising fake data availability is targeted to multiple-tree overlays. For example, in CoolStreaming, a peer connects to another that has the portion of data it needs (indicated by a bitmap that marks the availability of data parts). A malicious peer can advertise fake bitmaps to attract peers in order to prevent them from locating the desired data parts in time.

The above attacks are unique to P2P streaming. There are also attack schemes derived from file-sharing and DHT networks that can be used against P2P streaming. These attacks include the query flooding attacks and routing-based attacks discussed in Section V. In addition, a malicious node can corrupt overlay operations by dropping valid control messages and generating invalid ones. The invalid control messages can confuse other peers about the status of overlay nodes or overlay links.

B. DoS Attacks on the data plane

Attacks launched on the data plane are rather simple to explain: a malicious node can drop, corrupt, delay, duplicate, and forge media data. Dropping or corrupting data is similar to content attacks in DHT networks. By delaying data, the malicious node can intentionally make all or part of the data miss their playback deadline so they will become useless to its downstream peers. By duplicating or forging data, a malicious peer can flood its downstream peers with duplicated or bogus data packets to waste the bandwidth of the network.

We notice that most attacks launched solely on the data plane can be detected and avoided if downstream peers disconnect from peers with bad data quality. Such technique is originally designed to guarantee the quality of streaming. Meanwhile, it is difficult for DoS attacks on data plane to corrupt the perceived QoS of a large number of peers. However, an attacker can combine data plane attacks with control plane attacks to attract more peers, such as combining data dropping with RTT cheating, which would severely degrade the streaming performance.

There exist some mechanisms to detect the above attacks, such as [4][8][6]. In this proposal, instead of providing case-by-case detection schemes, we design a unified framework to provide an overall solution that can incorporate existing detection schemes to defend against various DoS attacks.

III. RIPPLE-STREAM DESIGN

In this section we first present the architecture of ripple-stream and its two key components, and then we analyze how ripple-stream can be used to achieve DoS resilience.

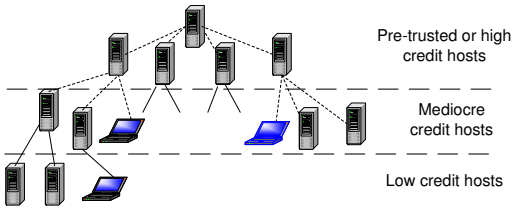

Fig. 1. Example of a ripple-stream based overlay.

A. Architecture

The essential idea of ripple-stream is to leverage an existing credit system to keep malicious peers or untrustworthy peers on the fringe of the overlay. Basically, in a ripple-stream based overlay, peers are organized around the data source based on their credit. The higher the credit, the closer a peer can be to the data source. Figure 1 shows a simple example. The ripple-stream framework enables peers to share their knowledge such that further attacks from discovered malicious nodes can be prevented. It can work with existing or future defense schemes to construct DoS resilience overlay networks.

We can explain the ripple-stream framework by describing the procedures a new node goes through in a ripple-stream overlay. When a new peer A joins the overlay, it will first obtain a list of peers with mediocre credit from a bootstrap mechanism. The join procedure can vary among different overlays. After joining the overlay, A accumulates credit by fulfilling its duties on the control plane and on the data plane. These credit-related operations are handled by the credit component included in ripple-stream. Meanwhile, A also tries to find upstream peers that can provide better service based on some overlay optimization principles. These upstream peers should have similar credit values as A does. If A discovers malicious behaviors of other peers, it disconnects from these peers and reports its discovery to the credit system.

Thus, we can see that the ripple-stream framework consists of two key components, credit management and overlay management. Credit management defines the interface between overlay events or transactions and the underlying credit system used by ripple-stream [9]. It regulates how credits are accumulated and how penalties are applied in streaming overlays. Overlay management defines credit-constrained peer selection and overlay optimization techniques. It regulates how overlay peers should interconnect with each other to construct DoS resilient and efficient overlays. DoS resilience is achieved by enforcing credit constraints based on the credit system, and overlay efficiency is guaranteed by overlay optimization based on network proximity metrics. In all, credit management identifies malicious nodes with low credit and overlay management pushes them to the fringe of the overlay.

B. Credit Management

Ripple-stream uses a credit system to identify ill-behaving peers. To achieve this goal, its credit management component needs to translate a user's behavior to its credit value. The credit management component defines the following principles for this credit translation. Since ripple-stream relies on existing credit systems to maintain peers' credit, only the credit systems that can implement these principles are qualified for ripple-stream.

- *Transaction importance*. In ripple-stream, events and transactions have different importance levels. Data or control tampering, once detected, is marked as an important event.
- *Transaction rating*. Nodes gain credit faster by serving more data. This will help nodes with high bandwidth accumulate credits fast and move up the overlay structure where they can serve more peers.
- *Credit aging*. Ripple-stream requires a credit aging algorithm that takes transaction time into consideration. The idea is to prevent a node from maintaining high credit after it stops serving data.
- *Long-term credibility and short-term trust*. The credit system should be able to respond quickly to the changes in a peer's personality.

Besides credit maintenance and calculation, ripple-stream also relies on the credit system to deal with collusive behaviors in the overlays. We find PeerTrust [9] to be a close match to the credit system required by ripple-stream.

PeerTrust defines the following trust metric,

$$T(u) = \alpha * \sum_{i=1}^{I(u)} S(u,i) * Cr(p(u,i)) * TF(u,i) + \beta * CF(u)$$

In this equation, $T(u)$ stands for the credit of peer u, $I(u)$ is the total number of transactions performed by u, $S(u,i)$ is the normalized amount of satisfaction u gets from transaction i, $p(u,i)$ represents the other peer in transaction i, and $Cr(p)$ stands for the credibility of feedbacks from peer p. $TF(u,i)$ is the transaction context factor. $CF(u)$ is the community factor of peer u, and α and β are normalized weight factors.

To use the above trust metric in our framework, ripple-stream has new semantics for variable $S(u,i)$, $TF(u,i)$ and $I(u)$. In ripple-stream, $S(u,i)$ stands for transaction rating that can differentiate credit gained from data plane and control plane, $TF(u,i)$ is the transaction importance

TABLE I
A TYPICAL CREDIT SYSTEM SETTING FOR RIPPLE-STREAM

Credit Type	Trans. Rating: S(u,i)	Trans. Importance: TF(u,i)
Control	c	t
Data	s*c (s = [5-10])	t
Tampering	-c	W*t (W>100)
Disconnecting	-s*c	w*t (w>1)

that enforces high penalty for control and data tampering. $I(u)$ has to take transaction time into consideration during transaction accounting. Ripple-stream can use the latest subset of $I(u)$ to calculate the short-term trust of a peer. The short-term trust value is used to quickly identify the malicious peers that start their attacks after accumulating a certain amount of credit.

Based on the credit management principles and the trust metric used, we currently define four credit types in ripple-stream, *control, data, tampering*, and *disconnecting*. These credit types correspond respectively to the credit a peer will gain or lose if it helps with control, serves data, tampers control or data messages, and disconnects its downstream peers. Among these credit types, *tampering* has the highest transaction importance, and *disconnecting* has a higher importance value than *data* and *control*. Table I shows a typical credit setting for ripple-stream, in which c is the unit for normalized satisfaction values, and t is the unit for transaction importance. The credit gained from serving data is s times of the credit gained from serving control.

To penalize serious attacking behaviors such as data tampering, a high credit penalty, controlled by W, is used. *Disconnecting* penalty is used to punish the behavior of frequent peer disconnecting. The amount of the penalty is equivalent to the credit a peer can accumulate by serving data for a certain time period (represented by "w" in Table I). In ripple-stream, a downstream B disconnects from peer A if it observes low data quality from A. This disconnecting event costs both A and B the disconnecting penalty. For a malicious node that attracts other peers then disconnects them, such penalty can quickly degrade its credit.

The credit management provides the foundation for the defense mechanisms of ripple-stream. It requires peers, regardless of the type of DoS detection schemes, to report their experience to the credit system. The credit management works with the underlying credit system to aggregate peers' experience and calculates their credibility. The credibility information will be used by the overlay management component to construct DoS resilient overlays.

C. Overlay Management

Overlay management defines overlay construction principles based on peers' credibility. It adopts credit-constrained peer selection to regulate how overlay peers should interconnect with each other based on their credit values. Meanwhile, overlay management adopts overlay optimization techniques to guarantee overlay efficiency based on network proximity metrics.

Credit-constrained peer selection requires peers to select neighbors based on their credit values. A peer will only allow queries or data requests from peers whose credit is *approachable*. A peer considers another peer's credit level to be *approachable* if the normalized credit difference between them is below a predefined threshold θ, Periodically, a peer checks whether its credit difference with connected neighbors exceeds such a threshold. It will disconnect the peers whose credits become unapproachable. Note that one peer only needs to estimate the credit difference to enforce the credit constraints. There is no strict requirement on the accuracy or freshness of peers' credit value.

To assist *approachable* peers in locating each other, ripple-stream employs a random node discovery mechanism. A node maintains several peering neighbors with approachable credits. These peering neighbors serve as alternatives upstream peers. When a new peer joins the overlay, it will ask for the peering neighbors from its upstream peers from which the new peer discovers its own peering neighbors.

To provide opportunities for later-joined nodes to serve data, ripple-stream introduces credit-constrained triangular optimization algorithm to avoid links with high cost (similar to [10]) among peers with *approachable* credits. Triangular optimization can reorganize the overlay in an efficient way without the dependency on the sequence of node arrival.

D. DoS resilience analysis

In this section we demonstrate how ripple-stream can be used to defend against DoS attacks. Ripple-stream can use existing DoS detection schemes to catch the ill behaviors of malicious peers. Once detected, ripple-stream uses a unique way to adjust the overlay to defend.

We use RTT cheating as an example. A malicious node can deflate its distance to other peers to attract their connections. Such RTT cheating can be detected by additional measurements. That is, before a peer A switches to another peer B, it will verify the low RTT claim from B with its own measurement. A control tampering report will be sent to the credit system if peer

A's measurement result is significantly different from B's claim.

The negative feedbacks for the malicious nodes can quickly degrade its credit value, especially its short-term trust value. Its upstream peer will notice the low credibility of the malicious node and disconnect it. In this way, the malicious node can only stay at the lower part of the overlay structure due to its low credit. It can still misrepresent its RTT distance to other peers, but it cannot attract other peers because the credit-constrained peer selection will exclude this malicious node from being selected.

For other forms of attack on the control plane, similar procedure can be applied. In this paper, we will not analyze all DoS attacks listed in Section II case by case. The key idea remains the same: peers report their observations into the credit systems. Ripple-stream overlay management can then adjust the overlay accordingly to push the malicious nodes to the fringe of the overlays where their attack behaviors cause the least or no damage.

We notice that sometimes it is difficult to differentiate attacks from network congestion. High loss rate in a data flow can either cause by intentionally packet dropping or real congestion in the network. However, such misunderstanding won't affect the quality of the constructed overlays. First, congestion happens occasionally won't affect a peer's credibility. Moreover, if congestion indeed happens frequently, this peer should be placed on the fringe of the network because of its bad network condition. This reflects the design objective of ripple-stream: honest peers with good network condition will accumulate credit fast and be placed on the top of the overlay structure.

IV. Performance Evaluation

In this section, we evaluate the performance of ripple-stream in two aspects: how it performs if there are no malicious nodes in the P2P streaming system, and how well ripple-stream responds to DoS attacks. We base our evaluation on a root-path tree protocol with the ripple-stream framework. We name it *credit RP protocol*. Similar to [11], the optimization goal of the RP protocol is to minimize peers' latencies to the root of the tree (root distance). In a RP tree, a node maintains its path to the root of the tree along with the root distance and tries to minimize this distance by randomly querying other peers.

A. Performance Metrics and Simulation Setup

1) *Average root RDP* (Relative Delay Penalty [12]), evaluates how close peers are to the root.
2) *Initial stabilization time*, evaluates how long it takes an overlay to stabilize initially after a peer joins.
3) *Number of disconnected peers*.
4) *Average receiving rate*.

The first two metrics are used to evaluate how ripple-stream performs with no malicious nodes. Our expectation is that ripple-stream takes a longer time to stabilize initially and it can achieve similar RDP performance as the original tree. The last two metrics are used to evaluate the performance of ripple-stream under DoS attacks.

We conduct our simulations on a transit-stub topology of 4,000 nodes generated by the GT-ITM topology generator [13]. We vary the overlay group size from 50 to 1000. For each overlay size, we randomly select the overlay nodes from the network topology.

B. Performance under DoS attacks

To test how credit RP trees react under DoS attacks, we setup the following attack scenario. It advertises low root distance after it join the overlay. Meanwhile, it forwards only part of media data to its downstream peers. As analyzed in Section II, this is a typical combinational attack scenario launched from the data plane and the control plane. In this paper, we only present this typical attack scenario in this paper to show the effectiveness of ripple-stream.

Fig. 2 shows the number of disconnected nodes after the malicious nodes start the DoS attacks in a group of 600 nodes. The credit RP tree can quickly stabilize after a number of peers identify the malicious nodes. While in the original RP tree, the malicious nodes keep attracting innocent peers. Since peers do not share their knowledge with each other, these innocent peers may connect to another malicious node even though that malicious node has been discovered by other peers. This explains the seesaw in the number of disconnected nodes for the original RP tree.

In Fig. 3, we present the average receiving rate of the overlay peers. We assume there is 1% data loss on the link between two good peers. The malicious nodes report half of their actual receiving rate. Fig. 3 shows that the average receiving rate of the credit RP tree stabilizes at a high level quickly even with a large number of malicious nodes.

C. RP Tree without Malicious Node

Fig. 4 shows the average root RDP of the original RP tree and the credit RP tree. The credit RP tree has similar average root RDP as the original RP tree for small groups. For large groups, the credit RP tree shows

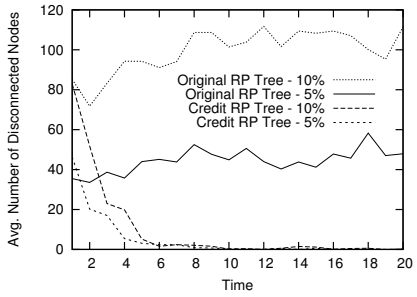

Fig. 2. Number of disconnected peers with 5% and 10% malicious nodes.

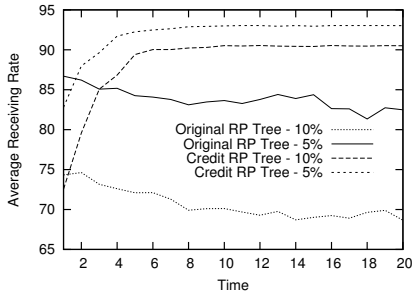

Fig. 3. Average receiving rate of peers with 5% and 10% malicious nodes.

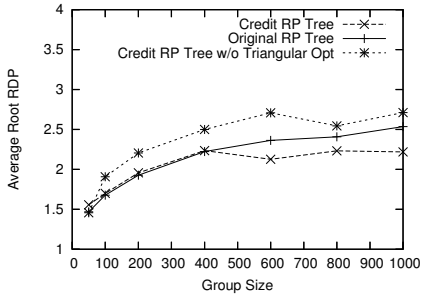

Fig. 4. Average root RDP of RP tree for various group sizes.

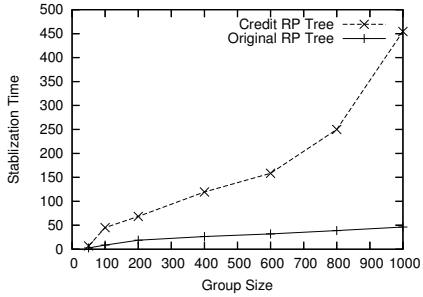

Fig. 5. Initial stabilization time of credit RP tree.

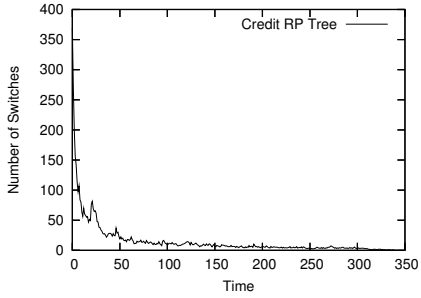

Fig. 6. Number of switches over time in credit RP tree for group size 600..

better root RDP. For fair comparison, we also disable the credit-constrained triangular optimization in the credit RP tree and include the result in Fig. 4. The root RDP then becomes about 9.8% higher than that of the original RP tree.

We compare the initial stabilization time of the original RP tree and the credit RP tree in Fig. 5. In this simulation, from an empty tree, we add ten peers into the overlay in each time unit. After all peers are added into the overlay, we start counting the stabilization time. From Fig. 5, we can see that on average, it takes the credit RP tree about 10 times longer to stabilize than the original RP tree. The credit constraints in the credit RP tree generate additional switches that prolong the initial stabilization process as the credits of peers evolve over time. This is the price we pay to achieve DoS resilience shown earlier. However, from Fig. 6 which shows the average number of switches in the whole overlay of 600 nodes at each time unit, we see that the number of switches in the overlay is in fact very low after time 100, which means the overlay becomes relatively stable very quickly.

V. Related works

Existing DoS studies for P2P system focus on file sharing networks and DHT networks. In P2P file sharing networks, copyright owners intentionally spread a large number of bogus files in order to discourage the sharing of copyright protected contents. This type of content-based attacks is named *file pollution* [3]. Dumitriu *et al.* studied the effects of two type of semantic attacks, false reply attack (reply with fake query results) and slow node attack (direct users to slow or overloaded peers) [5].

DoS studies in DHT networks focus on content attacks and routing attacks. To launch a content-based attack, a malicious node denies the existence of the content it is responsible for. For routing-based attacks, a malicious node can mis-route, corrupt, forge, or drop routing messages [8]. Castro *et al.* conclude that three requirements should be met to secure routing in DHT: secure assignment of node IDs, secure routing table maintenance and secure message forwarding [6]. Some streaming overlays that use routing to construct data forwarding path, are subject to routing-based attacks. However, unlike the discrete unicast messages in DHT, the data in P2P streaming networks are continuous streams and in most cases are broadcasted. Thus, the defense schemes proposed in DHT may not be suitable or efficient for streaming.

Some attacks target all overlay networks, including overlays used by DHT or streaming. Query flooding is one example. A malicious node floods another peer with queries to deplete its process power. Daswani has studied this type of attacks in file sharing and DHT networks [4]. A malicious node can initiate an eclipse attack [14] to an overlay network by manipulating a large number of nodes to isolate part of the network. Singh *et al.* show that eclipse attacks can be prevented by limiting the degrees of individual nodes. It is relatively difficult to launch query flooding and eclipse attacks in ripple-stream based overlays because the credit constraints already limit the number of peers the malicious node can contact.

For P2P streaming, we are only aware of one study investigating RTT cheating in the context of application-level multicast [7]. However, the authors do not provide a mechanism to defend against such cheating attacks.

VI. Conclusion and Further Discussion

In this paper we investigate the possibility to use credit system to improve the resilience of P2P streaming networks to DoS attacks. We study various DoS attacks on P2P streaming and then propose an open framework named ripple-stream to construct efficient overlays with high DoS resilience. Our evaluations show that, when under attacks, ripple-stream can effectively stabilize the overlay and significantly improve the streaming quality. Moreover, ripple-stream has the flexibility to incorporate existing or future DoS defense schemes as long as they plug their feedbacks into the credit system. Ripple-stream can also work seemlessly with more advanced credit system to increase its efficiency.

To the best of our knowledge, this is the first proposal to systematically address the DoS issues in P2P streaming system. In this paper we only focus on internal DoS attacks while P2P streaming is also vulnerable to external attacks which are launched on the network level targeting the limited number of streaming sources. We plan to do more investigation to guarantee efficient data delivery without revealing the source of the P2P streaming network.

References

[1] Y. Chu, S. Rao, S. Seshan, and H. Zhang. Enabling Conferencing Applications on the Internet using an Overlay Multicast Architecture. In *Proc. of ACM SIGCOMM '01*, San Diego, CA, USA, Aug. 2001.

[2] X. Zhang, J. Liu, B. Li, and T.-S. P. Yum. CoolStreaming/DONet: A Data-driven Overlay Network for Live Media Streaming. In *Proc. of IEEE INFOCOM 05*, Miami, FL, USA, Mar. 2005.

[3] J. Liang, R. Kumar, Y. Xi, and K. Ross. Pollution in P2P File Sharing Systems. In *Proc. of IEEE INFOCOM 05*, Miami, FL, USA, Mar. 2005.

[4] N. Daswani and H. Garcia-Molina. Query-Flood DoS Attacks in Gnutella. In *ACM Computer and Communications Security*, Washingtion, DC, USA, Nov 2002.

[5] D. Dumitriu, E. Knightly, A. Kuzmanovic, I. Stoica, and W. Zwaenepoel. Denial-of-Service Resilience in Peer-to-Peer File-Sharing Systems. In *Proc. of ACM SIGMETRICS 05*, Banff, Canada, Jun. 2005.

[6] M. Castro, P. Drushel, A. Ganesh, A. Rowstron, and D. Wallach. Secure Routing for Structured Peer-to-Peer Overlay Networks. In *Proc. of OSDI*, Boston, MA, USA, Dec. 2002.

[7] L. Mathy, N. Blundell, V. Roca, and A. El-Sayed. Impact of Simple Cheating in Application-Level Multicast. In *Proc. of IEEE INFOCOM 04*, Hong Kong, China, Mar. 2004.

[8] E. Sit and R. Morris. Security Considerations for Peer-to-Peer Distributed Hash Tables. In *Lecture Notes in Computer Science*, volume 2429, pages 261–269, 2002.

[9] L. Xiong and L. Liu. PeerTrust: Supporting Reputation-Based Trust for Peer-to-Peer Electronic Communities. In *IEEE Transactions on Knowledge and Data Engineering*, volume 16, pages 843–857, Jul. 2004.

[10] B. Zhang, S. Jamin, and L. Zhang. Host Multicast: A Framework Delivering Multicast To End Users. In *Proc. of IEEE INFOCOM '02*, New York, NY, USA, Jun. 2002.

[11] P. Francis. Yoid: Extending the Internet Multicast Architecture. Apr. 2000. http://www.aciri.org/yoid.

[12] Y. Chu, S. Rao, and H. Zhang. A Case For End System Multicast. In *Proc. of ACM SIGMETRICS 00*, Santa Clara, CA, USA, Jun. 2000.

[13] K. Calvert, M. Doar, and E. Zegura. Modelling Internet Topology. In *IEEE Communications Magazine*, Jun. 1997.

[14] A. Singh, M. Castro, P. Druschel, and A. Rowstron. Defending Against Eclipse Attacks on Overlay Networks. In *Proc. of ACM SIGOPS European Workshop*, Leuven, Belgium, Sep. 2004.