RIPPLE-STREAM: SAFEGUARDING P2P STREAMING AGAINST DOS ATTACKS

Wenjie Wang, Yongqiang Xiong, Qian Zhang, Sugih Jamin

wenjiew@eecs.umich.edu, yqx@microsoft.com, qianzh@cs.ust.hk, jamin@eecs.umich.edu

ABSTRACT

Compared with file-sharing and distributed hash table (DHT) network, P2P video 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 overlays. Combining credit constraints with overlay optimization techniques, ripple-stream can achieve both DoS resilience and overlay efficiency.

1. 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 of reducing illegal music swapping [3]. Recent research has begun 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 is 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 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 evaluations 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 on the connectivity among overlay peers.

We note however that we are not designing a DoS-resilient credit system. Ripple-stream can work with future or more advanced credit system to improve its efficiency. We currently 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. External attacks will be our future work.

2. RIPPLE-STREAM DESIGN

In this section we first introduce the categorization of DoS attacks on P2P streaming network. Then we present the architecture of ripple-stream and its two key components: credit management and overlay management.

We categorize the DoS attacks on P2P streaming networks into attacks on the control plane and attacks on the data plane. Attacks on the control plane include *RTT cheating, accepting too many downstream peers, connecting to too many upstream peers,* and *advertising fake data availability.* Attacks

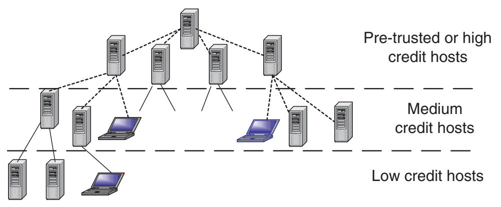

Fig. 1. Example of a ripple-stream based overlay.

on the data plane include dropping, corrupting, delaying, duplicating, and forging media data. Details about these attacks are available in [7].

2.1. 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. 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. Fig. 1 shows a simple example. The ripple-stream framework enables peers to share their knowledge such that further attacks from malicious nodes can be prevented. It can work with existing or future defense schemes to construct DoS resilience overlay networks.

We explain the ripple-stream framework by describing the join procedure a new node goes through in a ripple-stream overlay. When a new peer A joins the overlay, it first obtains a list of peers with medium 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 exhibited by 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 [8]. 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. Working together, the credit management system identifies malicious nodes with low credit and the overlay management system pushes them to the fringe of the overlay.

2.2. 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: *Transaction importance*, *Transaction rating*, *Credit aging*, and *Long-term credibility and short-term trust*. In ripple-stream, events and transactions have different importance levels. Data or control tampering, once detected, is marked as an important event. Transaction rating controls the rate nodes gain credit. Nodes gain credit faster by serving more data. The credit aging system prevents a node from maintaining high credit after it stops serving data. With short-term trust, the credit system is able to respond quickly to the changes in a peer's personality.

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. We find PeerTrust [8] to be a close match to the credit system required by ripple-stream. PeerTrust employs DHT to store and lookup the credit of peers. It adopts trust-based peer selection to deal with collusive behavior.

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 variables $S(u,i)$, $TF(u,i)$ 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 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 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 with 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*. To penalize serious attacking behaviors such as data tampering, a high credit penalty is used.

Disconnecting penalty is used to punish the behavior of frequent peer disconnection. The amount of the penalty is equivalent to the credit a peer can accumulate by serving data for a certain time period. In ripple-stream, a downstream B disconnects from peer A if it observes low data quality from A. This disconnection 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.

Credit management is the fundamental defense mechanism of ripple-stream. It requires peers, regardless of the type of DoS detection schemes, to report their experience to the credit system. 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.

2.3. Overlay Management

Overlay management defines overlay construction principles based on peers' credibility. It adopts credit-constrained peer selection to regulate the way overlay peers interconnect with each other based on their credit values. In addition, 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 pre-defined 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 alternative 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 [9]) among peers with *approachable* credits. Triangular optimization can reorganize the overlay in an efficient way independent of the sequence of node arrivals.

3. 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 (or RP tree protocol) with the ripple-stream framework. We name it *credit RP protocol*. Similar to [10], 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.

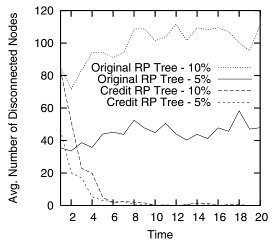

Fig. 2. Number of disconnected peers with 5% and 10% malicious nodes.

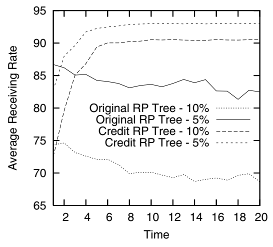

Fig. 3. Average receiving rate of peers with 5% and 10% malicious nodes.

3.1. Performance Metrics and Simulation Setup

1. *Average root RDP* (Relative Delay Penalty [11]), 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 expectations are that ripple-stream takes a longer time to stabilize initially and 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 topology of 4,000 nodes. We vary the overlay group size from 50 to 1000. For each overlay size, we randomly select the overlay nodes from the network topology.

3.2. Performance under DoS attacks

To test how credit RP trees react under DoS attacks, we set up the following attack scenario. A malicious node advertises low root distance after it join the overlay. Meanwhile, it forwards only part of media data to its downstream peers. This is a typical attack scenario launched from both the data plane and the control plane. Due to space limitation, we only present this attack scenario in this paper to show the effectiveness of ripple-stream.

Fig. 2 shows the number of disconnected nodes after 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 unsuspecting peers.

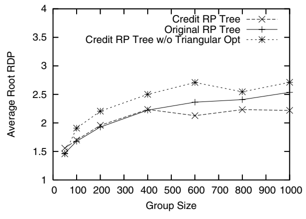

Fig. 4. Average root RDP of RP tree for various group sizes.

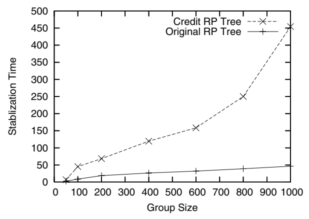

Fig. 5. Initial stabilization time of credit RP tree.

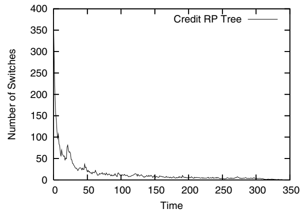

Fig. 6. Number of switches over time in credit RP tree for group size 600..

Since peers do not share their knowledge with each other, these unsuspecting 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 receive rate of the overlay peers. We assume there is a 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.

3.3. 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 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 10% 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 resiliency 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. The stabilization time can be shortened if we assign a peer's initial credit based on its bandwidth and distances to other peers. This approach assumes that every peer is good by default, and it requires initial network measurements from each peer.

The overhead of ripple-stream depends on the credit system it employs. For credit system using DHT, the overhead of a ripple-stream based overlay is $O(logN)$ times of the original protocol, because each credit report and lookup will be forwarded by $O(logN)$ peers on average. N is the group size. We skip the analysis here due to space limit.

4. CONCLUSION AND FURTHER DISCUSSION

In this paper we propose an open framework named ripple-stream to construct efficient overlays with high DoS resilience. Our evaluations show that, under attack, ripple-stream can effectively stabilize the overlay and significantly improve the streaming quality. Ripple-stream has the flexibility to incorporate existing or future DoS defense schemes as long as they send feedback into the credit system. Ripple-stream can also work seamlessly with more advanced credit system to increase its efficiency.

5. 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*, Washington, 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] W. Wang, Y. Xiong, and Q. Zhang, "Ripple-Stream: Safeguarding P2P Streaming Against DoS Attacks," Microsoft Research, Tech. Rep. MSR-TR-2006-08, Jan. 2006.

[8] L. Xiong and L. Liu, "PeerTrust: Supporting Reputation-Based Trust for Peer-to-Peer Electronic Communities," in *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 7, Jul. 2004, pp. 843–857.

[9] 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.

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

[11] 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.