Adopting Traffic Locality on Secure P2P Streaming System

Masafumi SHINTANI*, Takayuki FUJINO**

Synopsis

Exchanging data on the Internet has been increasing as widespread adoption of broadband residential access and the advances in personal computer performances. Especially, video streaming services such as YouTube generate enormous amount of traffic. Peer-to-Peer streaming has become one of the most promising alternatives for distributing this type of media contents. P2P streaming offers content owner traffic load balancing mechanism and rapid deployment at low cost. However, some malicious attacks to disrupt P2P system have reported. And P2P itself has an inefficient traffic control problem. In this paper, we propose the P2P streaming system which can solve both two problems. Our approach is adopting traffic locality on existing secure P2P streaming system. We also show scalability analysis of our approach by simulations.

Keywords: Peer-to-Peer, Overlay Network, Traffic Locality

1. Introduction

Exchanging data on the Internet has been rapidly increasing as widespread adoption of broadband residential access and the advances in personal computer performances. Broadband network enables media rich contents. Among them, video streaming services such as YouTube generate enormous amount of network traffic. In traditional client-server environment, deploying the media content distribution service requires many server machines. The server machines tend to be expensive, and additional load distribution mechanism is also required to avoid user traffic flooding to limited servers. As a result, content owner may have qualms to start the new services. Peer-to-Peer (P2P) streaming system has become one of the most promising alternatives for distributing this type of media contents. P2P streaming offers content owner traffic load balancing mechanism and rapid deployment at low cost.

However, some malicious attacks to disrupt P2P system have reported. And P2P itself has an inefficient traffic control problem. In this paper, we propose the P2P streaming system which can solve both two problems. Our approach is adopting traffic locality on existing secure P2P streaming system. We also show scalability analysis of our approach by simulations.

Hereafter, section 2 gives brief explanation about P2P streaming scheme and also show the problems to be considered. In section 3, earlier studies for dealing with some security problems are introduced. In particular, SecureStream and fireflies, introduced in this section, play an important role of our method. Section 4 describes our proposed P2P architecture. We show our system can achieve to keep P2P streaming secure and also provides traffic locality function. Simulation based performance analysis is given in section 5 and followed by conclusion.

* Graduate school of Systems Engineering, Kinki University
** Department of Electronic Engineering and Computer Science, Faculty of Engineering, Kinki University
2. P2P streaming system

P2P streaming system can be classified into two categories based on the media distribution scheme. The first one is Tree-Push. In this type, P2P nodes construct tree-like network topology where root node is media distributor. The second one is Mesh-Pull. In this type, each P2P node in the network autonomously communicates with each other and collect media data from neighboring node. While the latter one provides many commercial P2P streaming services, CoolStreaming\(^2\), PPLive\(^3\), etc, the former one does not achieve any practical product. Therefore we focus on latter type P2P streaming scheme.

2.1. Mesh-Pull P2P behavior

In typical case, Mesh-Pull P2P streaming system consists of an origin video server, a channel server which manages video contents in distribution, a tracker server which is located for each video distribution and manages currently viewing users, and many distribution nodes which is the user computer and also contributes media data distribution.

Figure 1 shows basic Mesh-Pull P2P streaming architecture. Video streaming is divided into small data which is called “chunk”. When new comer node demands to join P2P network, the node at first contacts to channel server. The node (user) can get the information on the video streaming program now in distribution. Then the node connects to tracker server of certain video distribution and gets the information on P2P node list which contribute the demanded video distribution. Finally the node starts communication with the P2P nodes via TCP or UDP. When a node has a communication channel with another node, this partner node is called “neighbor”. By exchanging chunks among neighbors, video streaming is distributed. A node maintains a buffer-map, which is prepared for individual neighbors and stores what chunks the neighbor has. A node can identify what chunks the node does not have by updating and exchanging the buffer-map information. If neighbor has the chunk which the node does not have, then the node can download the chunk from the neighbor. A node buffers chunk, therefore contiguous video play is enabled.

2.2. Problems in P2P streaming

When considering the P2P streaming system deployment in the future, there are several problems to be solved.

1) Eclipse attack

In Eclipse attack, malicious nodes in collusion cause “Eclipse” in P2P networks\(^6\). In order to keep the P2P networks, each node has to exchange some administrative information. In the attack malicious nodes in collusion capture the administrative information and never redistribute it. As a result sound nodes will be in the trouble due to the lack of the information they need.

2) Pollution attack

Dhungel et al. reported the pollution attack where malicious users insert “bogus” media data to P2P streaming system\(^6\). This attack can degrade the streaming quality drastically. In \(^6\), they demonstrated the impact of this attack. Before the attack certain channel had more than 3,300 viewers, but during the attack the number of viewers dropped to less than 500 viewers in 30 minutes.

3) Inefficient traffic control

The Internet consists of a number of Autonomous Systems (AS) connecting with each other. Since the traditional network applications are based on the client-server model, the Internet traffic generally reflects the AS topology. However, P2P applications establish their own overlay network. This overlay network does not reflect the AS topology because they use other measures such as RTT or packet loss.
Adopting Traffic Locality on Secure P2P Streaming System

ratio when they decide their neighbors. The P2P traffic tends to be inefficient and they consume the ISP resources. This issue often motivates the ISP to filter or restrict the P2P traffic.

3. Prior work for the problems

Many researchers have dealt with the problems described in Section 2. In this section, we briefly introduce some earlier study results which relate to our proposed method.

3.1. Fireflies

Fireflies is a probabilistic approach to prevent the Eclipse attack°. In Fireflies each node constructs $2t+1$ structured P2P rings managed by distributed hash table (DHT) such as Chord7). The value of $t$ is derived from the assumed rate how much the malicious nodes the P2P network has. Every node is assigned the unique node ID and every DHT ring is assigned the unique ring ID. The node location of individual ring is decided autonomously by using the hash value of the ring ID and the node ID. Specifically nodes are sorted in ascending order by hash value. A node connects to the node whose hash value is located the next of the node in order. By connecting from the node which has the maximum hash value to the node which has the minimum hash value, ring has constructed. Figure 2 shows the Fireflies ring example which consists of 6 nodes and 3 rings. In the individual ring, a predecessor node monitors the nearest successor's behavior. In Figure 2 case, node 3 monitors node 1, 2 and 4 while node 3 is monitored by node 2, 1. It is quite difficult to launch the Eclipse attack for the P2P network with Fireflies because a node cannot decide its neighbor by itself.

3.2. SecureStream

SecureStream is a Mesh-Pull P2P streaming architecture which adopts Fireflies8). As described in Section 3.2, each Fireflies node can communicate with its predecessor and successor. These communication channels are called "gossip channel". The SecureStream utilizes this gossip channel as chunk distribution channel. By adoption the Fireflies the SecureStream can defense the Eclipse attack. Additionally, the SecureStream introduces the Liner Digest, which can authenticate each chunk. Liner Digest is a hash value which is added by origin video server. By using the hash value each node can verify the origin and integrity of received chunks. The SecureStream can behave well even if some malicious attempts to disrupt the service have made, however, problem is still left. Since they adopts DHT based Fireflies, chunk distribution path must ignore the AS topology.

3.3. Traffic localization attempts

Various P2P traffic localization attempts have made regardless of the P2P application treats video streaming. The attempt of Bindal et.al can be classified bottom up approach°. In the Bittorrent-like P2P network, a node tries to choose the node in the same AS whenever possible during the neighbor selection process. In contrast, P4P10 is top down approach. The P4P tries to establish the shared database scheme for given P2P applications. The database includes intra-AS topology, inter-AS topology, bandwidth, etc. P2P applications can optimize their own traffic based on database information. Currently this research and development is in progress in IETF ALTO-WG.

4. Our proposed method

Despite of various research attempts described in Section 3. There are no attempts to integrate individual research result. In this section we propose the method which can achieve both security and traffic locality. Basic idea is to adopt the P4P-based traffic locality function to existing SecureStream.

4.1. AS proximity

Our method use “AS proximity” concept. The AS proximity is defined as the distance from a node to an AS. The simplest measure of the distance is to use AS hop, but bandwidth or throughput
information can be also taken into account.

Figure 3 shows the AS proximity example where AS hop is used as distance measure. The AS certain node belongs to is defined AS proximity 1. The ASes which can reach via AS hop 1 is defined AS proximity 2, in turn, AS proximity 3, 4 are defined. If there are several AS path to certain AS, minimum AS hop is used as the AS proximity.

4.2. Assumptions and architecture overview

Our proposed method consists of SecureStream node and so-called “Oracle server”. We define the Oracle server that the server which can provide AS number resolution based on IP address, neighboring AS enumeration based on AS number. It can be said that the Oracle server assumption is reasonable because the server can be implemented easily after the P4P technology will be standardized.

Since our research attempt is to adopt the P4P-based traffic locality function to existing SecureStream, it is assumed all of the functions of SecureStream are available. In particular, each node can share the all of the SecureStream node information including their IP addresses, past behaviors (whether make sound behavior).

Each new node is assumed to know at least one other SecureStream node (via Web, etc), and also have the information on Oracle server location.

Based on the assumptions above, our approach is to establish additional overlay network over traditional SecureStream network. This new overlay network can be used as media chunk distribution channel. However, the new overlay network reflects the AS proximity, overall chunk distribution will reflect the AS topology. As a result, traffic localization is achieved while traditional security features of SecureStream is still inherited. From the next section we give the detail of the protocol.

4.4. Interaction between a node and Oracle server

In our method each node interacts with the Oracle server in two ways. We define this two type communications.

1) AS lookup query and response

A node issues the AS lookup query message which includes one IP address. The Oracle server responds the AS lookup response message which includes the AS number that IP address belongs to.

2) AS Map query and response

A node issues the AS map query message which includes one AS number and one AS distance value. The Oracle server responds the AS map response message which contains the AS information within the specified AS distance. Message format is following: [AS distance][#of ASes][AS list] ..... Hereafter, we simply describe these interaction as to perform “AS lookup” or “AS Map”.

4.5. Constructing additional overlay network

In this section we explain the protocol process flow for constructing additional overlay network over SecureStream network. Figure 4 shows the example network. Oracle server is assumed to be located at somewhere else. Each AS has it's AS number. In the following, the protocol progresses from the viewpoint of AS 1 nodes. Black circle indicates individual node. Black square indicates the new node.

**Step 1: Initiating fireflies gossip channel**

As in traditional SecureStream, each node at first establishes fireflies ring and open the gossip channel to its ring predecessor and successor. Then start exchanging chunks with its neighbors.

**Step 2: Getting local AS information**

Each node issues AS lookup using its IP address, then they get the AS number they belong to. Since any node can share all of the nodes in SecureStream
Adopting Traffic Locality on Secure P2P Streaming System

environment, individual node has the knowledge of other nodes, which includes their IP address, belonging AS number.

Step 3: Joining the local AS group

Each node joins the local AS group using other nodes information. In practical, a node communicate with other node in the same AS and establish the original fireflies ring (the number of ring is 1). This communication channel is used for mainly exchanging administrative information.

Step 4: Designated node election

By using the local AS communication channel, the designated node and backup designated node are elected autonomously. The designated node plays a role of get the AS map information from the Oracle server on behalf of other nodes in the same AS.

The designated node is desired to be the node which has enough computer resource and network resource. The designated node is also preferable to be stable node. If the designated node is disappeared, the backup designated node will be the designated node and provide its function.

Step 5: Getting the AS map information

The designated node gets the AS map information from the Oracle server by issuing the AS map message using local AS number. The AS map information is shared by the nodes in the same AS.

Step 6: Constructing additional overlay network

Upon receiving the AS map information, each node newly choose its neighbors based on AS map and AS proximity. The number of neighbors is restricted by AS proximity, which is designated by the equations below.

\[ N_{\text{max}} = \frac{N_0}{m} \]

is the total number of new overlay neighbors, \( N_0 \) is the number of acceptable neighbors in the same AS. Parameter \( m \) characterizes the proximity preference. For example, if it is assumed that maximum overlay neighbors is 20 and \( m \) is set to 0.5, then the number of neighbors are permitted as following: 10 for same AS, 5 for AS proximity 1, 2 for AS proximity 2, 1 for AS proximity 3. Figure 5 shows this selection example.

Step 7: Local AS group maintenance

In P2P network every node can add or leave in any time. Therefore local AS group member is likely changed. To maintain the local AS member, each local AS node periodically check whether their fireflies ring predecessor or successor exist.
6. Conclusion

In this paper, we proposed to adopt traffic locality function to the P2P streaming which is designed for keeping security. Assuming the existence of Oracle server based on P4P technology, and introducing AS proximity concept, we have succeeded to reduce the number of transit AS 55% compare to traditional SecureStream.

However, currently both SecureStream and our method have same scalability problem. This is the scalability restriction stemming from fireflies. Both schemes require to establish a number of fireflies ring and the number of rings increase along with the total number of P2P nodes. A lot of computer resource is required for maintaining the fireflies ring, SecureStream and our method will be collapsed in ultra-large scale P2P environment. Possible solution may to divide the P2P network and autonomously confederate each other.

Reference
3) PPLive: http://www.pplive.com/