A Mesh Check Scheme against P2P Live Streaming Attacks *

Jinkang Jia
School of Electronics and Information Engineering
Beijing Jiaotong University
Beijing,100044,P.R.China
jinkangjia@yahoo.com.cn

Changjia Chen
School of Electronics and Information Engineering
Beijing Jiaotong University
Beijing,100044,P.R.China
changjiachen@sina.com.cn

ABSTRACT

With the wide spread of P2P streaming systems, there appear some hackers who try to pollute the system by inserting fake data chunks into the system. These "dirty" chunks will then be propagated to many normal peers, which will result in the failure of the whole system. Considering the playback quality and the burden of the client, most systems nowadays don't adopt any measures to protect the video content, such as encryption, CRC, or other methods.

We think it's more important to eliminate the polluters actively than to prevent the peers from polluted passively. In the paper, we propose a new scheme which tries to leverage the P2P nature of the system. We piggyback some "check" bytes in the data chunks exchanged between different peers. On one hand, these "bytes" can verify the authenticity of the data chunks, which prevent the diffusion of the polluted chunks. On the other hand, they can also help to detect and identify the "polluters" of the system. By simulation, the effects of our schemes are evaluated and the results are very inspiring. We think this light-weight solution against the streaming attacks is promising and can be deployed in real world.

Categories and Subject Descriptors

C.2.4 [**Computer Communication Networks**]: Distributed Systems; C.4 [**Performance Of Systems**]: Raliability, availability, and serviceability

General Terms

Performance, Security, Measurement

*This work was supported in part by the Chinese NSFC under Grant 60672069 and 60772043,China 973 2007CB307101 Chinese Ministry of Education under grant 20050004033 and Beijing Jiaotong University under grant 2005SM006.

Qshine08 July 28-31, 2008, Hong Kong, China.
 ISBN 978-963-9799-26-4
DOI 10.4108/ICST.QSHINE2008.3853

Keywords

P2P, live streaming, pollution

1. INTRODUCTION

P2P live streaming systems [1, 2, 3, 4, 5], such as CoolStreaming, PPLive, PPStream, have become one of the most popular applications across the Internet which has attracted large number of users in rather short period. Abundant program resources, convenient access method, low configuration requirements, good playback quality, free service, they all contribute to the great success of these systems.

Like some P2P file-sharing applications, most commercial live streaming systems adopt the mesh-based data-driven mechanisms [3] for video diffusion. On one hand, the P2P nature of the system helps to increase the spread speed of video data chunks and ease the server load at the same time. But on the other hand, the direct exchange between peers also incurs many security problems. Some hackers reverse-engineered the private protocols and try to pollute the whole system, and some researchers crawl many peers to analyze the characteristics of users and evaluate the overall performance. These behaviors all do no good to the system, or even ruin the whole system. However, due to the limitation of bandwidth and processing cost, most deployed streaming systems nowadays don't apply any encryption algorithms to protect the protocols as well as the video chunk data.

As far as we know, there is little work which has paid attention on the pollution of these P2P streaming systems. In [6] the authors have revealed how serious the number of peers will decline after the pollution is inserted into the streaming system. But the solutions proposed in the paper are either hard to be deployed or heavy-weight and complex. We will discuss these schemes in section 3.

In the paper we propose a simple scheme which is easy to be put into real deployment. A mesh-based check network is formed by all neighbors of one peer. Each peer is monitored by others and the reputation between each pair of connection is accumulated by serving good content to each other. Several fake chunk exchanges will be discovered by the downloader and the connection between the downloader and the polluter is cut immediately. This lightweight and self-rescue mechanism will eliminate the fake chunk intruders and effectively prevent the system from being polluted.

Besides, we also developed the codes for simulation. We try to reveal the relationship between the vital parameters of system design and pollution effect. And our scheme is proved to be effective to find out the polluters at early stage.

The rest of the paper is structured as follows. In Section 2, we will introduce some present work concerning pollution in various P2P systems. We will go on to present the infrastructure of these mesh-based P2P streaming systems as well as some solutions on anti-pollution in Section 3. In Section 4, our light-weight scheme will be proposed for the identification of fake chunks and polluters as well. The simulation on pollution diffusion and validation of our scheme are given in Section 5. We will conclude our work and raise some future work in Section 6

2. MOTIVATION AND RELATED WORK

With inborn nature of free sharing of various P2P systems, abundant resources and great convenience have been brought to users. However, as side-effects, the security problems have become more and more prominent and serious, which will decide the future development of P2P technology.

For file-sharing systems, because the illegal sharing of these copyrighted files decreases the benefits of the audio and video companies, the fight between the copyright protectors and free users becomes more and more fierce. Some protectors even hire some pollution companies to help them prevent the files from being propagated. With the violent controversies in commercial and industry fields, there are some measurement-based or theoretic studies which pay great attention to the pollution of P2P networks [7, 8, 9, 10]. As far as we know, [8] on FastTrack network is the first which tries to reveal the situations of pollution. It's observed that the pollution is wide spread and is pervasive especially for recent popular songs. The following studies [7, 9, 10] have done much in modeling of the pollution propagation process through theoretic analysis. In [9] the authors adopted the well-known model for disease spread and proposed the modified model and some counter-measures for the diffusion of polluted files. In [10], some non-linear differential equations are listed and the authors developed a suite of fluid models to characterize the pollution proliferation of P2P systems.

For streaming systems, things are a little different, because it's the companies themselves instead of free users that publish the streaming content. Usually, companies who are operating these P2P systems have paid content providers for publishing the video or audio programs. The fight lies mainly on the hackers and these companies. However, the pollution consequences are much more serious for streaming systems compared with file sharing systems, and some fake chunks will make the service of the whole system unavailable. By experiments[6], it has been proved that the number of users will decrease rapidly after the insertion of polluted chunks by one high-bandwidth polluter. Therefore, the pollution attack is more devastating for streaming systems.

Our motivation for the research is two-fold. On one hand, from the commercial perspective, these streaming systems are becoming more and more widely used by millions of audiences from every corner of the world. Therefore, the failure of the service makes the service providers of these commercial systems begin considering the pollution problems. On the other hand, from the technological view, these P2P streaming systems have their own characteristics which make the old methods for file sharing systems cannot be adopted any more. New techniques should be proposed to meet the requirements of the live P2P systems. Firstly, the rigid time demands for chunk diffusion makes the protocols must be light-weight and easy to be implemented. Secondly, the heterogeneous processing capabilities of different peers further increase the difficulty and make old method unavailable such as encryption, hashing, and so on.

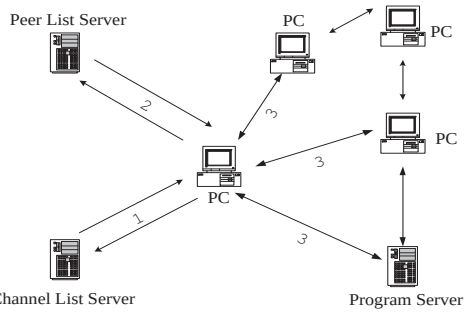

Figure 1: Communication procedures for typical P2P streaming system

Based on the motivations, in the paper we propose a light-weight scheme to protect the system from polluted. Our solution tries to isolate the polluters actively instead of check each chunk passively. In addition, the modification of the protocol is trivial. The key scheduling algorithms and the connection management policies needn't be changed at all. The only modification of the client is some codes on check algorithms and the format of queries(responses).

3. SYSTEM INFRASTRUCTURE AND DEFENSE SOLUTIONS

It is shown the whole procedure how a P2P streaming user joins the system in Fig. 1. After launching the client, the software will automatically connect to the channel list server to update channel lists. After the user selects the channel to watch, he has to wait several or tens of seconds for video playing. During this period, the client will firstly request the peer list server for some other users who are watching the same channel. Each user is identified by its IP address and the listening port. After harvesting enough <IP, port> pairs, the client will try to connect these peers to download data chunks. In most systems, each chunk is identified by the playtime offset which is encoded by the program source. After the user successfully connects to other peer, they will exchange the playtime offset and the buffer maps (BMs) which denote the availability of the chunks periodically. Based on these BMs, the client will schedule to download which chunks from which peer. After receiving enough chunks in the buffer, the media player will pop up immediately and the program begins playing.

Therefore, the whole topology of the system looks like a mesh instead of other structured topology, such as binary tree, forests, and so on. Thus, if one polluter pretends to be a normal peer and gets registered successfully into the system. In procedure 3 of Fig. 1, he can easily insert fake chunks, which will be further propagated by normal peers. The diffusion of the pollution will not only decrease the playback quality but also even destroy the whole system.

There are some solutions proposed by [6], we will analyze their advantages and shortcomings in detail next.

Traffic Encryption: This scheme works well only if the system under consideration isn't open source. However, by

reverse engineering of the codes of client, the system cannot be kept secure any longer. If the reverse engineering process is thorough enough, and considerable fraction of the system protocol and messages can be revealed, thus facilitating the polluter to inject pollution into one or more streams. Furthermore, the decryption of the encrypted data chunk is still time-consuming and may deteriorate the playback quality for some old PCs.

Blacklisting: One method is to find the "unusual" peer who behaves different from others. For example, polluters may declare they have all data chunks in their BMs to attract more benign peers to download from them. Another method is to identify if one chunk is polluted based on mining characteristics or some audio/video techniques. In addition, by identifying these polluters, peers need to broadcast the blacklist to other peers. The disadvantages of the scheme are obvious. Firstly, one attacker may behave like an ordinary peer and create fake chunks similar to ones received from neighbors. Even worse, the polluter can even send duplicate valid chunks to different chunk queries, which can successfully avoid being detected by all above techniques adopted. Secondly, the broadcast of the blacklist takes time and may incur the so-called "broadcast storm". Furthermore, it should be noted that if the protocols of the streaming system is open source, the polluter may initiate a "blacklist announcing attack" by just slandering others instead of spreading fake chunks.

Hash verification: Similar to BitTorrent, some researchers proposed to introduce some hash information to verify the integrity of the chunks. In this way, each peer needs to get the hash of each chunk from the source. Therefore, the source server's load will be much higher if the system adopts the centralized fashion. To relieve the load, to distribute the hashes through the P2P system itself has been proposed. However, it's easy for the polluters to insert the "fake chunks" and corresponding "fake hashes" simultaneously, which will result in peers being fooled into believing the integrity of the chunk. In brief, hashing method is not a viable solution for P2P live streaming.

Chunk signing: In this scheme, some "authentication information" needs to be transmitted to the receivers along with the chunks. The authentication information can either be provided by the source (in which case the load on the source might be high) or could be distributed through the P2P system itself, in the form of a separate stream or be piggybacked onto chunks. There are two reasons which prevent the scheme from being deployed in real systems. First, the high computational requirements are needed for all peers. Secondly, some polluters can still insert fake "authentication information" to confuse the check of each chunk.

In all, the schemes mentioned above can be easily made ineffective or are hard to be put into real deployment due to their obvious disadvantages. As far as we know, our scheme is the first which try to utilize the correlation among different chunks and randomize the check information to prevent peers from pollution. The formation of the mesh check topology can assure the polluter can be easily identified and eliminated. Furthermore, our scheme is distributed and can be deployed easily.

4. MESH CHECK SCHEME FOR ANTI-POLLUTION

As far as we know, no current deployed systems take policies to examine the authenticity of data chunks, therefore, it's easy for the polluters to insert fake chunks and pollute other benign peers. The consequences are destructive. Furthermore, some benign peers who receive fake chunks will act as "middle" polluters unintentionally and accelerated the diffusion of the pollution. Even worse, the frequent freezes of the images will make some peers leave the system forever. The degradation of the playback quality and the decrease of the amounts of peers will finally result in the unavailability of the streaming service. Based on the P2P nature of these live streaming systems, we devised a simple scheme to assure the integrity of data chunks and eliminate attackers as well. In addition, our scheme is light-weight and easy to be deployed. Next we will introduce our scheme in detail.

4.1 Two Observations

The scheme we propose is based on two characteristics of the real deployed systems from our measurements[11, 12].

1. The frequent exchange of the BMs which indicate the (un)availability of data chunks in the buffer gives the peer the perspective and ability to schedule the downloads of data chunks from different peers. Meanwhile, the understanding of the chunk distribution in neighbors' buffers may help the peer verify the chunk received from peer A by peer B which has the chunk;

2. From the real measurement [11], it's revealed that the buffer update step of the peer is block-based instead of chunk-based; the block must be a playable segment which contains tens or even hundreds of data chunks. This relative stability of the buffer can help us to utilize the correlation among adjacent chunks.

4.2 Mesh-based Inter-chunk Check Scheme

The idea is very simple. All current deployed real systems are pull-based instead of push-based. That is, each peer will independently determine which chunk he will download from which neighbor based on the BMs he received. Based on these characteristics, one peer can piggyback some ultra "check" bytes query of another chunk as he sends the request for one data chunk. For example, when peer A requests data chunk I_1 from peer B, he can also piggyback another question such as: what are the first two bytes of data chunk I_2 given peer B has I_2 in its buffer. At the same time, peer A can download data chunk I_2 from peer C and piggyback the query as: what are the last two bytes of data chunk I_1? Thus, a verification network is formed. Each data chunk received from one peer can be validated by answers from other peers and the monitoring mechanism among each other is formed consequently. The neighbors of each peer form the mesh or even all-connected graph. In addition, some reputation system can be adopted. Each neighbor of one peer need to accumulate reputation by serving good chunks, and a single fake chunk will ruin the reputation greatly. If the reputation decreases below some threshold, the peer will cut off the connection actively to the neighbor who serves fake chunk. It should be noted that even the chunk is encrypted, it makes no difference for our scheme to work effectively. Piggybacking encrypted "check" bytes also works well.

In addition, our scheme is distributed and each peer need only make decisions on its own. To make it clear, we will give a simple scenario to illustrate how our scheme works.

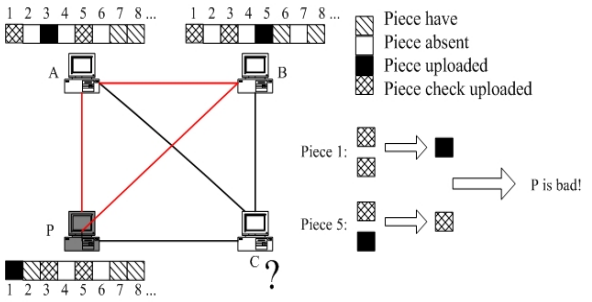

Figure 2: Two typical scenarios of our scheme

As illustrated in Fig. 2, suppose that peer C has downloaded from and simultaneously uploaded chunks to two benign peers for some time: peer A and B, Which are shown in black line. After polluter P joins the system and tries to spread fake chunks by announcing its BM to other peers, peer C may request to download chunks according to present scheduling algorithms. As shown, suppose that peer C will send queries to pull chunk 1, 3, 5 respectively from peer P, A, B, the scheduler of peer C should also piggyback some "check" bytes requests for the chunks. For example, When requesting chunk 3 from peer A, the scheduler of peer C will randomly pick another chunk which has at least two duplicate copies in all its neighbors' BMs, including peer A. After determining the chunk for verification, it will go on to choose two or more random bytes of the chunk as the "check" bytes of the chunk. Consequently, the chunk 3 as well as two random-selected "check" bytes of chunk 1 are downloaded. It should be noted that once the "check" bytes of certain chunk are chosen, they will kept unchanged for later requests. Therefore, as each peer of the system takes the responsibility of serving certain chunks, it is also looked on as the "monitor" of other chunks. Thus, though AB, AP and PB aren't connected physically, in fact, they have been connect logically by the check mesh which are represented by red lines in Fig. 2.

After accumulating some chunks and "check" bytes in the buffer, peer C will check the authenticity of the chunks. Two "cutting rules" are proposed to help to judge if the chunk is fake or not.

Rule 1: If none of "check" bytes of certain chunk can be matched to the chunk and there exists two or more "bytes" providers which have responded with the same "bytes". Then we will reward these neighbors with same "bytes" with higher reputation and punish the uploader and all the other "bytes" providers. See chunk 1 as Fig. 2 for illustration.

Rule 2: If the chunk has been matched to one of the "check" bytes, then the reputation of the uploader and the matching "check" bytes providers should be increased. Meanwhile, those who provided the fake "check" bytes should be punished severely. See chunk 5 in Fig. 2 for illustration.

Adopting these rules, the connection will be cut off if the reputation of the connection drops below some predefined or dynamic threshold. When devising the reputation system, we prefer to take the "Additive Increase/Multiplicative Decrease (AIMD)" which has been adopted in controlling the window size of TCP. For each peer, it can keep record of behaviors of each neighbor. The reputation of each connection can be accumulated correspondingly by serving good chunks to this peer, if any one of the neighbors reports bad "check" bytes or bad chunks, the reputation will drop greatly and the connection will be cut off if the reputation is below some threshold. Our scheme is totally distributed and the reputation doesn't need to be diffused in the system. This design further eliminates the possibility of broadcasting fake reputation by attackers in the system. Suppose that most peers are benign peers, our scheme works very well and a few polluters cannot attack the system at all.

4.3 Discussions

The advantage of our scheme is obvious. Each polluter can be quickly identified by the adoption of the "check" bytes. If the polluter tries to transfer the fake chunk, it can be easily found by the two identical "check" bytes of other benign peers and is cut off consequently. If the polluter tries to respond with the bad "check" bytes, it can also be recognized by the matching of chunk which another benign peer uploaded and other correct "check" bytes. Therefore, the polluter has to do good no matter for the chunk itself or the "check" bytes so as not to be disconnected by other peers.

In addition, we can assume an extreme scenario. It's possible for two or more malicious polluters to collude together to give a false comment on other good node. But in reality, it's hard to be the case. Firstly, the possibility that one peer connects to two or more attackers simultaneously is rather trivial. The inborn connection scheduling scheme for each peer will tend to keep neighbors selectively instead of randomly. That is, even the malicious peers try to pollute the specific benign peer, this peer may not keep both of these connections simultaneously. Secondly, even two or more attackers have connected to one target peer, the pollution can also be detected for there are more benign connections than malicious ones. The cooperation of the attackers may succeed in fooling the scheme one or two times, they are determined to be found by other benign peers consequently. Finally, the random selection of peers who provide "check" bytes and real chunks will make the cooperation among malicious peers hard to be deployed.

Given all protocols are reverse-engineered, the random selection of chunks and corresponding bytes for verification further increases the difficulty for polluters to diffuse fake chunks. And The large number of normal peers and connections per client makes it harder to conspire among all polluters. Besides, our scheme is easy to be deployed. The client developers need only add some simple protocols to query and receive "check" bytes while keeping all the other important policies untouched, such as the scheduling algorithms. The bandwidth needed to transfer these bytes and the processing capacity for peers are trivial.

5. SIMULATION AND EVALUATION

In our simulation, we adopt discrete event-driven mechanism which tries to catch main characteristics of the system.

5.1 Assumptions

To pay more attention on the anti-pollution effects, some basic assumptions are adopted and listed below.

1. All peers join the system at the beginning of the program, and to evaluate the function of our scheme, we assume that the polluted peer won't leave the system;

2. To imitate the mesh topology of the peers, we adopt

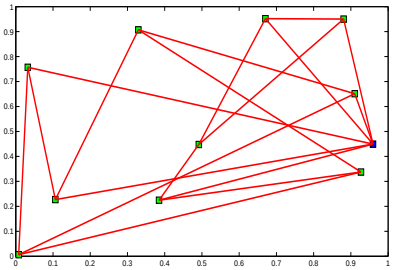

Figure 3: Sample topology for simulation

the random connected graph as the topology of the system. The degree of each peer is generated according to the degree distribution which is revealed by the real measurement. In Fig. 3 we plotted a sample topology in which all peers except the seed have the degree of 3. To some extent, this mesh-based topology can accelerate the spread rate of chunks as well as pollution;

3. To simplify the system, we assume that all connections have the same bandwidth. We think this simplification will not influence the pollution diffusion process much. In addition, each peer has the same buffer size as well as the update size of block.

5.2 Parameters and Metrics

For the convenience of statement, we'd like to define some key parameters of the streaming system.

- Buffer size (S chunks): the buffer needed to store the video data chunks. In our simulation, we assume that the unit of S is chunks instead of bytes;
- Seed bandwidth (B_s): the maximum number of chunks which can be served by the seed for one peer in one unit of time;
- Peer bandwidth (B): the maximum number of chunks one peer can serve for another peer in one unit of time;
- Seed degree (D_s): the maximum degree for the seed;
- User number (N): the number of normal peers;
- Playback rate (r chunk/s): the playback rate for the program. We still adopt chunk/s as the unit instead of byte/s;
- Update pace (U s): the pace for the buffer to update. Because the update of the buffer is block-based instead of chunk-based, in the simulation, we assume that each peer will update its buffer when it has downloaded the whole chunk number of $r * U$ in size. That is, it has downloaded the video chunks which can be played for U seconds. Intuitionally, the longer the update pace is, the more "check" bytes can be used to verify the integrity of the data chunks;
- Placing factor (f): from our work[11], we have proved that each peer will determine its own playback start point based on the first several BMs he received. That is, it will choose the start point (offset) according its neighbors' buffer. The peer will choose the offset as formula $(max(P_i) - min(P_i)) * f + min(P_i)$,where P_i denotes all pieces that are in peer i's neighbors' buffers;

Table 1: Parameter Settings

N	S	f	N_p	U	r	P_s
500	100	0.34	1	1	10	$Defend$
B	D_s	B_s	D_p	B_p	T_j	deg
10	5	20	5	20	15	$uniform$

- Polluter number (N_p): the number of the polluters;
- Polluter degree (D_p): the maximum degree of the polluter;
- Polluter bandwidth (B_p): the maximum number of chunks that each polluter can upload to each peer in one unit time;
- Polluter join time (T_j s): the time unit when the polluters join the system;
- Polluter strategies (P_s): there are two ways for the polluters to adopt to spread fake chunks;
  - Positive attack ($Attack$): In order to spread the pollution more quickly and widely, one polluter will declare that it has all chunks which its neighbor needs by filling BMs with all "1"s. Its neighbor will tend to download the chunks from this polluter if it's the sole owner of some chunks.
  - Passive pollution ($Defend$): To hide the identity, some polluters will act as ordinary peers and obey all the policies the protocol regulated. By imitating the behavior of normal peers, the polluters are hard to be identified by current systems.
- Degree distribution (deg): the degree distribution of normal peers. In our simulation, we generate the degree for each peer which lies between 5 and 10 uniformly. That is, the minimum degree for one peer is 5, and the maximum is 10.

In our simulation, unless pointed out specifically, the basic parameters we adopted are shown in table 1. We assume that all polluters take the $Defend$ scheme by default.

Because the source plays the video at r chunks per second, it should be noted that there are $r * t$ chunks which are made available in total at time slot t. However, some early chunks will disappear due to the slide of peers' buffers. In the following graphs, we will take the time slot as well as chunk sequence to show the pollution or anti-pollution processes.

5.3 Pollution Diffusion

In the subsection, we will discuss the relationship between some key parameters and the pollution diffusion process. To smooth the random error, we repeated the simulation of each scenario for 10 round. Then we calculate the average of these metrics and the results are plotted consequently.

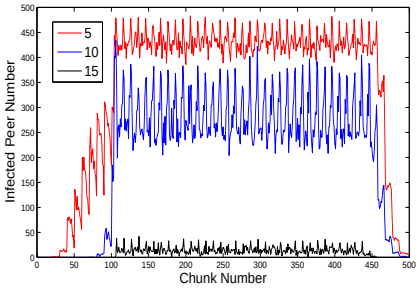

Figure 4: Polluter join time

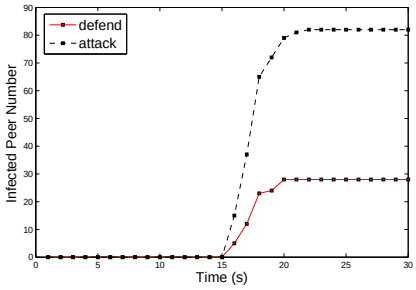

Figure 5: Polluter strategies

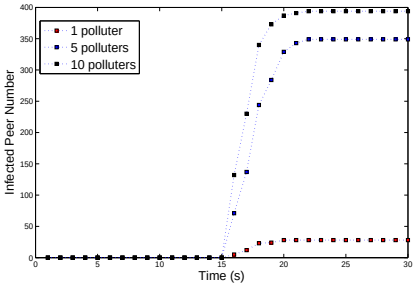

Figure 6: Polluter Number

5.3.1 *Polluter Join Time*

When the program source initiates to play, all buffers of the connected peers are empty. It's more effective for polluters to insert fake chunks as earlier as possible.

The simulation results are plotted in Fig. 4. The polluter adopts the *Attack* strategy and we fine-tune T_j to be 5, 10, 15 respectively. It's obvious that the earlier the polluter joins the system, the higher percentage of certain chunk will be infected. The polluter occupies all benign peers' buffers ahead of the source. Considering the red curve for the join time at 5, of all 500 peers, on average there are more than 400 peers who receive fake chunks during all the time for simulation. On such condition the whole system will be destroyed and it's very hard to recover. The final decline is due to the end of our simulation, which can be neglected.

Another hint for us is that the pollution effect will be stable when the polluter joins the system after most buffers of peer has been filled with good chunks. It means that it's same for the polluter to join the system at 15 or 150. Because each peer has more candidate uploaders for each chunk, it's a little difficult for the pollution to spread widely.

To prevent the system from being polluted at the early stage, we propose two heuristics here. Firstly, for program source, it can do the BM check at the beginning of one play. That is, it can recognize the polluter by querying BMs to each peer who tries to join the system. If the BM contains the chunk indicators which the program source has not published, the polluter can be identified and blocked by the source. By this way the positive attackers can be eliminated from the system. Secondly, for client, the scheduler can send more chunk queries to the program source instead of other peers if the peer list returned by the source is below some threshold. Besides, the passive pollution can be eliminated by our scheme which will be discussed later.

5.3.2 *Polluter Strategies*

We will compare the pollution effects brought by different strategies in the subsection. As shown in Fig. 5, it's obvious that the adoption of positive *Attack* strategy can pollute more peers than the passive conservative $Defend$ scheme do. When there is one normal peer which received at least one fake chunk in one update pace (time unit), the infected peer number will be added by one. Therefore, the y-axis denotes the number of peers who have once received bad chunks in the time slot. After the polluter joins the system at time slot 15, the pollution spreads like the disease and reaches stable after some time. Under same conditions, the aggressive scheme can infect more than 80 peers while the passive one can only affect less than 30 peers.

5.3.3 *Single Polluter vs. Multiple Polluters*

In Fig. 6 it's plotted how the polluter number will influence the pollution spread process. It's revealed that more polluters will infect more benign peers. However, it doesn't mean that the pollution effect can be good enough by solely increasing the polluter number. From the graph, it's revealed that the infected number of peers increases much when there are 5 polluters in the system comparing with only 1 polluter. When the polluters increase to 10, the increase rate is not high enough as before.

In fact, the phenomenon can be explained by the classical model of disease spread. After there is more than half number of peers infected, the possibility that an infected peer encounters another benign peer becomes less than that of the encounters between two infected ones. Thus the infection rate will decline correspondingly. In addition, the relative stable topology and the continuous shift of chunks in buffer will hinder the further diffusion of pollution. In brief, there exist an optimal number of polluters if peer number N is fixed. It's rather difficult to infect each peer in the system.

5.4 Our Scheme

In the subsection, we will evaluate the performance of our scheme on hindering the pollution diffusion. We will firstly consider the one polluter scenario, and then go on with the discussion of multiple polluters. To eliminate the influence of other parameters, we fixed the topology of the simulation for each scenario. Thus the simulation will be run for only once instead of 10 rounds in last subsection.

5.4.1 *One Polluter Scenario*

It's plotted in Fig. 7 the anti-pollution effect of our scheme, and it's shown how the infected number of benign peers evolves without or with our scheme. It's obvious our scheme works well in cutting the source of the pollution. From our simulation, all 5 connection of the polluter are cut after the polluter stays in the system for 4 time units, that is, before 20 second in our simulation. However, the pollution still exists for about 11 time units due to the differentiation of buffers of other benign peers. Furthermore, there are 23 of 1871 connections among benign peers which are cut off, for they have relayed some fake chunks. These side-effects are trivial compared with the large number of connections. Because the pollution sources are eliminated successfully, the buffer slide of all peers helps the system clean again.

From the chunk view, we replot the polluted number for

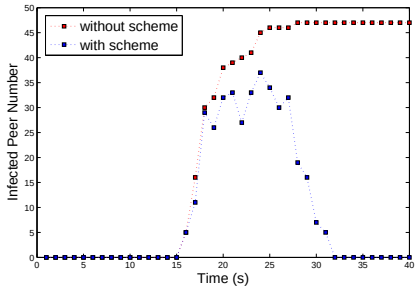

Figure 7: Infected peer number

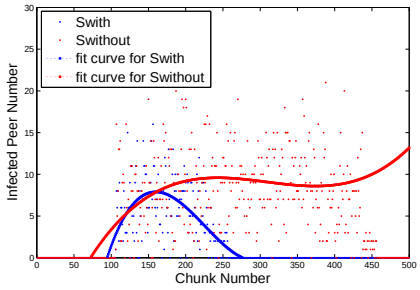

Figure 8: Infected chunk

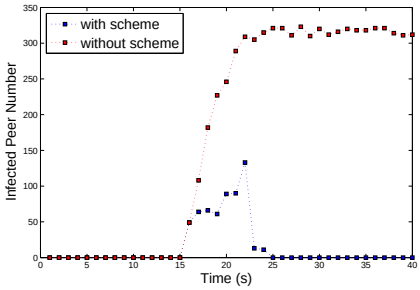

Figure 9: Multiple polluters

each infected chunk in Fig. 8. In the figure, the x-axis denotes the chunk sequence number while the y-axis represents the polluted number of peers who received this chunk. We adopt S_{with} and $S_{without}$ to denote the pollution diffusion results with and without our scheme. To be more clearly, we fitted these points with polynomial of degree 3. It's revealed that the pollution will decrease quickly and diminish at last.

From above discussions, we find the quickest way to remove pollution is to identify the pollution source as soon as possible. After the source is found out and cut off, the slide of the playback window will self-clean the system. Furthermore, our scheme doesn't need the help of the servers at all, and the self-secure mechanism with the help of all the trusted neighbors works very well.

5.4.2 *Multiple Polluters Scenario*

When there are multiple polluters who join the system simultaneously, our scheme does well, too. In Fig. 9 we plotted the anti-pollution effect of our scheme when there are 10 polluters joined at time unit 15. For each peer, if there are more normal neighbors than the polluted ones, these polluters will be correctly identified and cut off. Again we noticed the fluctuation of the infected number of peers during the convergence process. After careful observation of their buffers, we confirmed that it is caused by the "middle" relay peers who lag behind the source, and the distribution of buffer offset. The peers who have downloaded fake chunk from them may have cut off the connection, but these dirty chunks will diminish finally because of the ongoing of the playback process.

6. CONCLUSION AND FUTURE WORK

In the paper we propose a simple scheme to prevent the normal peers from being polluted by polluters in various P2P streaming systems. These polluters try to insert fake chunks into the system and destroy the playback of other normal peers. The consequence is devastative, which has been proved by the measurements. Compared with other solutions which have been proposed, our solution is more effective and easy to be deployed. The key idea is simple. Adopting the P2P nature of these streaming systems, the check topology among all neighbors of one peer will be a mesh consequently. Each chunk served by one peer will be monitored and checked by others, and the peer is responsible for the validation of chunks served by others simultaneously. From our simulation, the connection can be cut off at the beginning of the pollution diffusion.

7. REFERENCES

[1] http://www.pplive.com.
[2] http://www.ppstream.com.
[3] X. Zhang, J. Liu, and B. Li. Coolstreaming/donet: A data-driven overlay network for peer-to-peer live media streaming. In *Proc.IEEE Infocom'05*, Miami, USA, Mar. 2005.
[4] http://www.tvants.com.
[5] http://www.vvsky.com.
[6] Prithula Dhungel, Xiaojun Hei, Keith W. Ross, and Nitesh Saxena. The pollution attack in p2p live video streaming systems: Measurement results and defenses. In *Proc.ACM SIGCOMM'07(P2P-TV'07)*, Kyoto, Japan, Aug. 2007.
[7] Uichin Lee, Min Choi, Junghoo Cho, M. Y. Sanadidi, and Mario Gerla. Understanding pollution dynamics in p2p file sharing. In *Proc.IPTPS'06*, Santa Barbara, CA, USA, Feb. 2006.
[8] Jian Liang, Rakesh Kumar, Yongjian Xi, and Keith W. Ross. Pollution in p2p file sharing systems. In *Proc.IEEE Infocom'05*, Miami, USA, Mar. 2005.
[9] Richard Thommes and Mark Coates. Epidemiological modelling of peer-to-peer viruses and pollution. In *Proc.IEEE Infocom'06*, Barcelona, Spain, Apr. 2006.
[10] Rakesh Kumar, David D.Yao, Amitabha Bagchi, keith W. Ross, and Dan Rubenstein. Fluid modeling of pollution proliferation in p2p networks. In *Proc.ACM SIGMETRICS'06*, Saint-Malo, France, Jun. 2006.
[11] Jinkang Jia and Changjia Chen. Linear placement scheme based study on offset delay distribution in p2p streaming system. *Submitted to IEEE/ACM Trans. on Networking*.
[12] Jinkang Jia, Chunxi Li, and Changjia Chen. Characterizing ppstream across internet. In *Proc.IEEE NPC'07*, Dalian, China, Sep. 2007.