FIGHTING POLLUTION IN P2P LIVE STREAMING SYSTEMS

Alex Borges, Jussara Almeida, Sergio Campos

DCC – Federal University of Minas Gerais (UFMG)
Belo Horizonte – MG – Brazil

ABSTRACT

Peer-to-peer live streaming media systems are becoming more popular each day. As in file sharing P2P system, they are susceptible to content pollution attack. In this kind of attack, a peer alters the media content decreasing the perceived quality of the streaming. In this paper we evaluate the impact of pollution attack in P2P live streaming and we present two reputation system to avoid content polluted dissemination and isolate malicious peers. Our results show that a few number of polluters is capable to compromise all the application and the 2 proposed reputation systems can quickly identify and isolate polluters and also be resistant to peers collusion.

Index Terms— Live Streaming, P2P, Pollution, Attack

1. INTRODUCTION

Live video streaming is one of the most important application nowadays in Internet. Essentials TV corporations, as NBC[1], broadcast its tv shows over Internet. Moreover, lots of sites as FreeeTV[1] offers a great diversity of live streaming media channels. These live content distribution systems suffers with client scalability as they are based on the traditional client-server model.

Peer-to-Peer (P2P) Live Streaming Systems are becoming more popular each day. They have been used as a alternative to distributed live media over Internet and in contrast with traditional client-server system, a P2P system overcomes limitations such bandwidth and client scalability. In this system, an special node encodes the live media, break it in small pieces (*chunks*) and distributes to its partners. Others clients, also know as peers, forward content to others peers, eliminating the need of powerful servers or high bandwidth in a single point.

A variety number of techniques have been proposed to structure the system and transmit the live video content over all clients. Most of popular systems, such PPLive [1] , PPStream [2] and GridMedia [3, 4] use a data-driven mesh-pull overlay technique. It works similar to Bittorrent [5] file sharing system. The live video content is divided in small chunks by one source. When a peer joins the system, it makes partnership with a subset of peers in the live content network. Peers exchange chunk maps announcing available and disarable chunks and then, they upload or download chunks to/from their neighbors, according to its interests.

These systems generally assume an altruist and non malicious behavior of the peers. As far as we know, none of the most popular p2p streaming systems assume that the video can be changed or forged. Moreover, control and data message are transmitted in pure text, without any kind of encryption [6].

A malicious node may alters or forges data, making the streaming media useless. Peers may naively request and forward polluted data, consuming resources such bandwidth with undesirable data. Polluters may also make a collusion attack, which is an agreement of malicious nodes to deceive or mislead system and get a more accurated attack. These malicious behavior is common in P2P file sharing, and, the same way that some entities try to stop the file sharing systems, they can try to stop P2P live streaming. Moreover, this behavior can be encourage due concurrency among P2P streaming systems or just for fun.

There are some research in P2P file sharing to avoid polluted content and others malicious behaviors. Practical systems have been proposed in [7, 8, 9]. In the mean time, there is no appraisals on the subject of to apply these systems to a live streaming context and, directly apply them may not be feasible because of the restrictions of live applications.

In this paper, we present through simulation an evaluation of the damage caused by a pollution attack in a mesh based P2P live streaming system. We also present modifications to reputation systems used in P2P file sharing applications and evaluated its efficacy to stop content pollution dissemination. Our results show that we can rapidly identify and isolate polluters in a P2P live streaming system. We can also avoid collusion behavior of polluters with a relative low overhead, compared with the cost of retransmission of data due polluted content.

The remainder of this paper is organized as follow: in section 2 we present related work in the area. Section 3 discuss pollution attacks in P2P live streaming media system and how harmful are them. Simulated model and methodology are presented in sections 4 and 5. Section 6 show our main conclusions and possible future works.

2. RELATED WORK

Recent work on P2P live streaming systems has focused on the selfishness behavior of the peers and in DoS attacks. Reputation systems and incentive mechanisms to avoid egoists and liars have been proposed in [10, 11, 12], improving fairness among peers and incentiving altruist behavior. Content pollution has been treated in P2P file sharing systems. Reputation systems such Credence [8] presents a decentralized distributed system, where users assign reputations to the objects they download. Systems like Scrubber [7], identify and isolate malicious peers that disseminate polluted content and allows the rehabilitation of passive polluters.

For P2P streaming, we are only aware of 2 studies investigating content pollution. However, the authors do not provide a reputation mechanism to defend against such attacks. To the best of our knowledge [13, 14] are the first works to treat other malicious behavior than DoS, such as pollution. In these papers, the authors present a comparison among 4 alternatives to check data integrity in a P2P live streaming media system. In [6], is presented an experiment in a real system and the result shows that a simple pollution attack may abase

[1]http://www.nbc.com/ ; http://www.freeetv.com/

severely all the perceived media quality. Besides this, the authors suggests some techniques to check the integrity of data stream. All 3 works indicates that is possible to mark in the source and check it in the peers with a low overhead cost. Using the proposed schemes, processing cost to mark data is $O(n)$, where n is the number of chunks in a data block. To check data stream, the additional cost is $O(1)$ per block of chunks. To transmit data is necessary about 5% more network bandwidth. The authors do not deeply evaluate pollution attack impact to all peers in a P2P live streaming network and also do not present reputation systems with intention to stop content pollution combined with others attacks, such as collusion of polluters.

3. POLLUTION DEFENSE MECHANISMS

This section describes the two simulated systems to fight pollution in P2P live streaming. These approaches are inspired in techniques used in P2P file sharing applications. An important aspect which needs to be considered is how to identify polluted data. We consider as polluted any data that is corrupted, even it has not been intentionally modified by a peer. We also consider that any of the techniques to mark and check chunks, proposed in [13, 14, 6], can be used.

3.1. Blacklist

Blacklist approach is a simple manner to repute and get information about partners in a P2P system. Peers actively monitor the behavior of theirs neighbors and periodically report information to a centralized server. The goal is to identify nodes that originate polluted content and rapidilly isolate them, avoiding loss of perceived quality by system users. When a user reports or consults the centralized server, it sends/gets a score about neighbors and it stops to interact if the neighbors have score less than a given threshold.

The reported score by a node is weighted proportionally by its own score as [10, 15], and the final score reputation of a node is a weighted mean of all previous reports as shown in Equation 1. Equation 2 shows how a peer assigns a score reputation to a neighbor, and it will be detailed further.

$$R_j = \left\{ \frac{\sum_{k \in N} I_{k(j)} * R_k}{\sum_{k \in N} R_k} \right. \quad (1)$$

In Equation 1, N is the set of nodes the repute a node j , R_j is the reputation of j in the P2P live streaming system and $I_{k(j)}$ is the reputation score reported by k about j's behavior.

3.2. StRepS

StRepS, *Streaming Reputation System*, is a distributed reputation system whose objective is to identify and isolate malicious peers that disseminate polluted chunks over the P2P live streaming system. It also promote rehabilitation in case a peer naively forwards bad chunks or it have transmission problems that corrupt their data, incentiving them to stop forward polluted content or get better resources. Its development is based on a P2P file sharing reputation system [7], extended to capture characteristics of a live streaming media system.

Node's reputation is built on two components: *Individual Experience* and *Peers Testimonials*. Each peer assign reputation for each other based on these 2 components. The individual experience is updated every time a peer gets any set of chunks from another peer. A peer also periodically asks from his neighbors their testimonials from all peers they have contacted.

Every time a peer i download chunks from peer j, it updates individual experience according to j's behavior, increasing it if j acts fairly and decreasing otherwise. Peer i individual experience from j, $I_{i(j)}$, is computed as fallows:

$$I^t_{i(j)} = \left\{ \begin{array}{ll} max(0, {I^{t-1}}_{i(j)} - \alpha_p * (1 + n/r)^y) & \text{if } n/r > \text{thr.} \\ min(1, {I^{t-1}}_{i(j)} + \alpha_g * n/r & \text{otherwise} \end{array} \right. \quad (2)$$

In Equation 2, r is the amount of chunks i requests to j and n is the amount of polluted chunks j provides to i, α_p and α_g are the reward and penalty for the iteration when it is classified as clean or polluted respectively. A polluter is identified by the relation of clean and polluted it transfers to a neighbor. To quickly identify and penalize polluters, α_p is inflated by $(1 + n/r)^y$, where $1 \leq y \leq 2$. Moreover, $\alpha_p > \alpha_g$, thus, individual experience decreases faster than it increases.

The peer testimonial captures a community consensus opinion about a peer. Each peer periodically asks neighbors their individual experiences with respect to all other peers they have received chunks. This information is used before each new iteration. To update their peer testimonials about each other peer j, node i does as follows:

$$T_{i(j)} = \left\{ \frac{\sum_{k \in N} I_{k(j)} * R_{i(k)}}{\sum_{k \in N} R_{i(k)}} \right. \quad (3)$$

In Equation 3, N is the set of neighbors of i and R_j is the current reputation that i has about a node k as defined in Equation 4. If i does not have testimonial about j, $T_{i(j)} = R_{init}$, where R_{init} is a given initial reputation score. The testimonial of each node k is weighted by its own final reputation. Consequently, opinions of nodes with higher reputations have greater impact than those with low reputations.

Node i builds j reputation based on 2 components and it must weight the importance of each of them. Thus $(0 \leq \beta \leq 1)$ controls weights given to *Individual Experience* and *Peer Testimonial*. Low values for β put emphasis on individual experience, and high values, emphasizes peer testimonials (or neighbor's opinions). The final reputation is periodically calculated in a node i as follow:

$$R_{i(j)} = \beta * T_{i(j)} + (1 - \beta) * I_{i(j)} \quad (4)$$

In Equation 4, $R_{min(i)}$, $(0 \leq R_{min(i)} \leq R_{init})$, is the minimum reputation a peer must have to be considered trustworthy by i. A peer neither uploads or downloads chunks from any other peer that it considers untrustworthy.

4. EVALUATION METHODOLOGY

We perform network simulation using NS-2 [16]. We built a set of new agents that follow a simple mesh-based protocol described in [3, 4]. This section presents the simulation model adopted. Main results are presented in section 4.

In our simulation, live content is divided in chunks, which are small pieces of streaming content. These chunks are stored in a buffer in the peer application. Peers exchange chunk maps announcing available and disarable chunks and then, they upload or download chunks according to its interests. Peers must receive a large number of chucks errors to achieve a good video quality.

System Model: we model a mesh-based data drive P2P live streaming media. An special peer, called server, originates streaming media to be delivered to the P2P system nodes. When a peer joins system, it contacts a subset of nodes that already watches the streaming session. This subset of nodes is randomly obtained by an independent bootstrap mechanism. New peer contacts each other peer in the subset and add itself in the mesh based P2P streaming system. Each peer only knows the subset it is connected, and thus it only exchanges content and control messages with its neighbors. A peer must download data in a rate equal to the server generation rate. For instance, we assume that peers have enough storage capacity to visualization buffer and they share download data for about 2 minutes.

Network Model: we developed our simulation with a BRITE[17] generated topology. Each topology used in our simulations have 1000 nodes, all nodes links always have enough bandwidth to support the application. The link distribution and links delays are typical from a Wan structure.

Peer Model: each peer in the system is categorized as either *good* or *polluter*. Polluters forward only polluted chunks and good peers do not generate polluted chunks intentionally. A polluter never leaves the system and it acts announcing a full forged chunk map. We also assume that good peers never leaves the system and they do not forge chunk maps. Nodes are limited by some resource, for example bandwidth or a maximum number of connections thus, download and upload capacity are limited by its capacity. When a peer loses a neighbor or wishes to increase its connectivity, it can request additional neighbors and again, it randomly selects its partners. We also have a *server* peer, that is a good peer that only produces chunks (do not consume them). All peers collect information about its partners and report them to a simulation log every 30 seconds.

Pollution Dissemination Model: in our simulation, polluter only disseminates polluted chunks and always answer its neighbors requests. Good peers may, with probability p_{error}, give a corrupted chunk which is also interpreted as pollution (i.e. a link error). Moreover, if does not exist a data check mechanism, all good peers naively forward polluted content.

Colusion Model: each attacker knows each other in the system. Thus, each interaction with BlackList centralized server or with partners in P2P live streaming system, polluters repute each other with a score normally distributed between threshold and the maximum score allowed by reputation system.

Experiment Setup and Scenario: we used a topology with 1000 nodes in our simulations, including server and polluters. Server produces chunks with a playback rate about 7 chunks/s, which is a common rate in this kind of application [6]. Each peer connects to maximum number of neighbors, normally distributed between 30-40. Initially, a peer tries to connect to 60% of its maximum neighbors and, it can make new partnerships during the simulation.

5. RESULTS

We ran a series of experiments varying system configuration and peer behavior parameters and we obtained qualitative similar results. In our simulation, polluters try to keep its maximum number of neighbors and they stay in the system since it joins the simulation (between minute 2 and 5) until the end of simulation (minute 60). Good peers join the system between time 5 seconds and 300 seconds, normally distributed. Presented results are mean values of 5 executions, with a coefficient of variation less then 1.5%.

Initially we discuss results for a system without any kind of reputation or data checking. Figure 1 is the inverse of cumulative distribution function of polluted data ratio during an attack. We show situation for 1, 10 and 100 polluters (0.1%, 1% and 10% respectively) and our results indicates that with just 1 polluter in the system, about 80% of the nodes downloaded at least 60% of chunks polluted. With 100 polluters, 80% of the nodes received at least 90% of pollution. The data sent by a node follow close the behavior of download once peers naively forward polluted data.

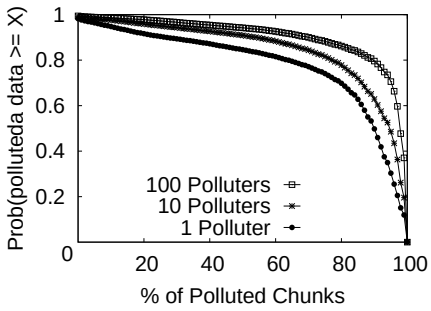
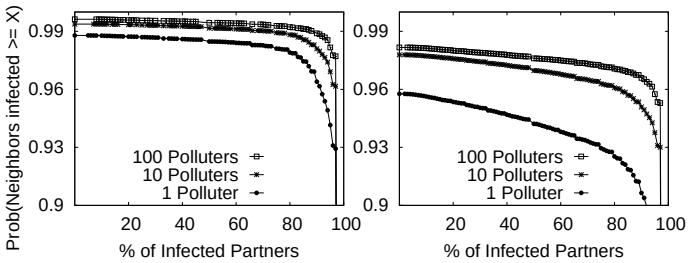


Fig. 1. Downloaded Data in System with 1000 Nodes.

Figures 2-a and 2-b show how many peers are infected by pollution during an attack. In 93% of cases, peers receive polluted chunks from at least 80% of its neighbors and in 96% of cases, peers forward polluted chunks to at least 90% of its neighbors.



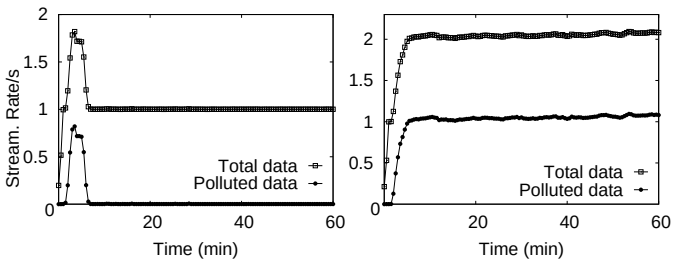
(a) Receivers (b) Emitters

Fig. 2. Infected Partners in The System.

Figures 3-a show the result for system under attack, Blacklist as reputation system and 100 polluters and no collusion. In a Blacklist approach, a high number of peers send information to a centralized server and so, a node in the system can quickly identify malicious peers. When a peer is reputed as a bad node, all others peers avoid its partnership. The Blacklist system presents a quick actuation, and the P2P system delayed about 3 minutes to isolate all 100 polluters.

We also simulated a collusion attack combined with pollution dissemination. Figure 3-b shows a scenario with 10 polluters. A Blacklist may not be resistant to collusion because a centralized server may receive a lot of false reports. Even if a node is categorized as polluter, malicious nodes may repute it well and later, it can interact with others peers, causing more damage to the P2P system.

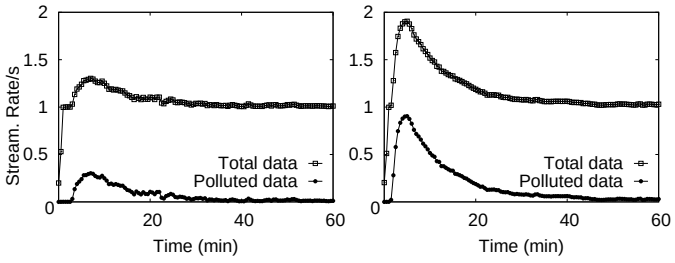
Finally, Figures 4-a and 4-b present results to simulated P2P live streaming with StRepS reputation system. Compared to Blacklist in the same situation (Figures 3-a and 4-a), StRepS takes a longer time to achieve its best performance. Although, StRepS avoids retransmission during an attack period better than a centralized Blacklist. Using Blacklist, peers may have to download almost 2 times media rate, while using StRepS, they only have to download about 1.3 times media rate. Moreover, StRepS is able to deal with a combined pollution and collusion attacks. During a combined attack, time to



(a) Without Collusion (b) With Collusion

Fig. 3. Streaming Rate With Blacklist.

get the best performance of the system is similar without collusion attack, but the required bandwidth is almost 2 times the media rate.



(a) Without Collusion (c) With Collusion

Fig. 4. Streaming Rate With StRepS.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we present an evaluation of damage caused by a pollution attack in a mesh based P2P live streaming system. We also propose two peer reputation systems as strategies to reduce content pollution dissemination. We also performed an extensive evaluation for various system and peer behavior configurations for both reputation systems, comparing their effectiveness against each other. Our main conclusions are threefold. First of all, just mark, check and reask data is not sufficient in a P2P live streaming because a polluter continues causing damages in the system, taking to a crescent necessity of resources to keep the quality of service. Second, both centralized and distributed approaches (Blacklist and StRepS) can quickly identify and isolate polluters with a low overhead compared to retransmissions caused by damage data. Finally, a distributed approach also avoids collusion behavior of malicious participants of the system.

Future work includes further evaluating the reputation systems, especially under mixed attacks, prototyping the StRepS as well as including it in a real P2P live streaming media application.

7. REFERENCES

[1] "Pplive website," `http://www.pplive.com`, 2007.

[2] "Ppstreaming website," `http://www.ppstreaming.com`, 2007.

[3] Meng Zhang, Jian-Guang Luo, Li Zhao, and Shi-Qiang Yang, "A peer-to-peer network for live media streaming using a push-pull approach," in *MULTIMEDIA '05: Proceedings of the 13th annual ACM international conference on Multimedia*, New York, NY, USA, 2005, pp. 287–290, ACM Press.

[4] X. Hei, C. Liang, J. Liang, Y. Liu, and K. W. Ross, "Insights into pplive: A measurement study of a large-scale p2p iptv system," in *In Proc. of IPTV Workshop, International World Wide Web Conference*, 2006.

[5] "Bittorrent website," `http://www.bittorrent.com`, 2007.

[6] Prithula Dhungel, Xiaojun Hei, K. Ross, and N. Saxena, "The pollution attack in p2p live video streaming: Measurement results and defenses," in *Proc. SIGCOMM Peer-to-Peer Streaming and IP-TV Workshop*, 2007.

[7] Cristiano Costa, Vanessa Soares, Jussara Almeida, and Virgilio Almeida, "Fighting pollution dissemination in peer-to-peer networks," in *SAC '07: Proceedings of the 2007 ACM symposium on Applied computing*, New York, NY, USA, 2007, pp. 1586–1590, ACM.

[8] Kevin Walsh and Emin Gün Sirer, "Fighting peer-to-peer spam and decoys with object reputation," in *P2PECON '05: Proceeding of the 2005 ACM SIGCOMM workshop on Economics of peer-to-peer systems*, New York, NY, USA, 2005, pp. 138–143, ACM.

[9] Ernesto Damiani, De C. di Vimercati, Stefano Paraboschi, Pierangela Samarati, and Fabio Violante, "A reputation-based approach for choosing reliable resources in peer-to-peer networks," in *CCS '02: Proceedings of the 9th ACM conference on Computer and communications security*, New York, NY, USA, 2002, pp. 207–216, ACM Press.

[10] Xing Jin, S.-H.G. Chan, W.-P.K. Yiu, Yongqiang Xiong, and Qian Zhang, "Detecting malicious hosts in the presence of lying hosts in peer-to-peer streaming," in *IEEE ICME 2006*. 2006, IEEE.

[11] William Conner, Klara Nahrstedt, and I. Gupta, "Preventing dos attacks in peer-to-peer media streaming systems," in *13th Annual Multimedia Computing and Networking Conference (MMCN'06)*, San Jose, CA, Jan 2006.

[12] Yun Tang, Lifeng Sun, Meng Zhang, Shiqiang Yang, and Yuzhuo Zhong, "A novel distributed and practical incentive mechanism for peer to peer live video streaming," in *ICME 2006, IEEE International Conference on Multimedia & Expo, Toronto, Canada*, Jul 2006.

[13] Maya Haridasan and Robbert van Renesse, "Defense against intrusion in a live streaming multicast system," in *Peer-to-Peer Computing*, Alberto Montresor, Adam Wierzbicki, and Nahid Shahmehri, Eds. 2006, pp. 185–192, IEEE Computer Society.

[14] Robbert van Renesse. Maya Haridasan, "Securestream: An intrusion-tolerant protocol for live-streaming dissemination.," in *Journal of Computer Communications. Special issue on Foundation of Peer-to-Peer Computing.* Elsevier, 2007.

[15] Wenjie Wang, Yongqiang Xiong, Qian Zhang, and Sugih Jamin, "Ripple-stream: Safeguarding p2p streaming against dos attacks.," in *ICME*. 2006, IEEE.

[16] S. Mccanne, S. Floyd, and K. Fall, "Network simulator," `http://www-nrg.ee.lbl.gov/ns/`.

[17] Alberto Medina, Anukool Lakhina, Ibrahim Matta, and John Byers, "BRITE: Universal topology generation," Tech. Rep. 2001-003, 1 2001.