On Pollution Attacks in Fully Connected P2P Networks Using Trusted Peers

Cristóbal Medina-López[1(✉)], Ilshat Shakirov[2], L.G. Casado[1], and Vicente González-Ruiz[1]

[1] University of Almería (ceiA3), Almería, Spain
{cristobalmedina,leo,vruiz}@ual.es
[2] Perm State University, Perm, Russia
im.shakirov@gmail.com

Abstract. This work studies the impact of the *pollution attack* in a fully connected live video streaming P2P push based overlay network, as well as the effectiveness of using trusted peers as the only defense mechanism. Determining the best strategies for both: (i) trusted peers (TPs) with the aim to detect and expel malicious peers (MPs) and (ii) malicious peers in order to pollute without being detected, is a challenge. Preliminary experimental results for the studied strategies show that the use of TPs is not an enough solution to defeat a large number of MPs because the number of TPs should be at least equal to the number of MPs. Additionally, this study shows the ingredients of a possible game theory problem formulation where the proposed attacker strategy could be improved to detect TPs, which in turn could improve the defense strategy, and so on.

Keywords: P2PSP · Content integrity · Live streaming · Pollution attacks · Peer-to-peer networks · Network performance evaluation

1 Introduction

The scalability problem in live video streaming systems based on a client-server model is well known: the server is forced to upload a copy of the content per client. In a peer-to-peer (P2P) system, the number of copies uploaded by the server is usually orders of magnitude lower than the number of peers, which are in charge of sharing the content among them, using their upload bandwidth excess. For that reason, the P2P model is an attractive alternative to the client-server model, especially since the available upload bandwidth grows progressively at the client side.

In this context, the Peer-To-Peer Straightforward Protocol[1] (P2PSP) was developed as an application-layer protocol designed for the real-time broadcasting of media over a P2P overlay. P2PSP establishes a push-based fully-connected

[1] http://p2psp.org/en.

A.M. Madureira et al. (eds.), *Intelligent Systems Design and Applications*,
Advances in Intelligent Systems and Computing 557,
DOI 10.1007/978-3-319-53480-0_15

mesh scheme where every peer sends data to each other. A basic P2PSP network (see Fig. 1) consists of a *splitter* (S) and a collection of *peers* (P_i) named *team*. The splitter receives a media stream from a *source* (O), divides the stream into chunks of data with the same size and sends them to the team following a Round-Robin scheme. Next, each peer forwards those chunks received from the splitter to the rest of peers (chunks received from other peers are not forwarded). Finally, each peer sends the reconstructed stream to a *listener* (L), which usually is a media player. A round consists in the splitter sends one chunk to every member of the team.

Because peers obtain most of the content from other peers rather than from the splitter, it is easy for a malicious peer (MP) to change the delivered content in some way, i.e. to produce a *pollution attack* [4]. Pollution attacks can be classified as *persistent attacks* [4], when the attacker always sends polluted chunks to their victims or as *on-off attacks*, when the attacker only poisons some chunks (for instance, 10% of the total number of chunks sent to their victims), in order to overcome reputation systems [6]. Free-riders or selfish peers (those that do not send data) affect in a similar way as polluters in our system and are treated in the same way.

Previous attacks can be modified by including *selective (non-selective)* and *collaborative (non-collaborative)* characteristics. In the selective attack, the chunks are poisoned for just one peer or a small subset of peers. In the collaborative attack, several attackers collaborate to perform the attack in order to increase the damage and to avoid their detection. In the presence of Trusted Peers (TPs), attackers could also collaborate to discover them.

An attacker can use a *hand-wash attack*: it leaves the team in case it thinks it was discovered and returns with another identity. This can be seen as a type of *sybil attack*, where the same attacker uses different identifications to deploy several peers in the team.[2] Additionally, an attacker can use a *bad-mouth attack* [7] under reputation systems by complaining about well-intended peers (WIPs), but we do not consider reputation systems in this study.

In this research, the defense Strategy is based on Trusted Peers and Digital Signatures (STrPe-DS) [10]. In STrPe-DS, each transmitted chunk is signed with the private key of the splitter to allow their verification by peers. In this study, peers do not handle other cryptographic material, and the content is not ciphered.

The main contributions in this study are:

- To define policies for TPs and MPs in a full connected overlay P2P network using a push model.
- To experimentally determine the necessary number of TPs to expel MPs using previous policies.
- To show future lines of study for policy improvement.

The rest of this paper is organized as follows. Related work is presented in Sect. 2. Section 3.1 shows a description of the studied strategies and the impact

[2] In the context of P2PSP, peers are identified by their IP addresses, so, attacker should use botnets to perform a sybil attack.

of different attacks. Results from the simulation are shown in Sect. 4. Finally, Sect. 5 shows the conclusions and the future work.

2 Related Work

Pollution attacks produced by MPs are some of the vulnerabilities of P2P overlays. Consequently, they have been studied extensively, specifically in the live streaming scenario [2,4,6,9]. Most of the defenses are based on trust management by gathering the reputation information from subsets of neighbors in the network. The main drawback of reputation systems are the false positives and false negatives, mainly due to bad-mouth attacks, resulting in MPs remaining in the team and WIPs being expelled.

The use of a set of TPs will reduce the number of MPs in the team. A framework with a set of TPs to monitor the bandwidth usage of untrusted peers in the system to prevent free-riding was studied in [3]. A detection algorithm for a set of TPs was considered in [5], where peers inform to the assigned TPs about the correct reception of a chunk. Then, the set of TPs can identify MPs by solving an inference problem. In P2PSP, TPs have their own information about the team because it is a complete connected overlay network. An attacker is expelled by the splitter based on the information received from TPs. Splitter expels a peer by removing it from its list of peers and therefore it will receive chunks anymore.

3 Description of the Model and Attacks

3.1 STrPe-DS Model

STrPe-DS has been designed to mitigate the selective attack and to identify poisoned chunks by using digital signatures. Any peer can know if a chunk was poisoned and/or relayed by a different peer than the one in charge of it, as shown below. The following rules define the STrPe-DS [10]:

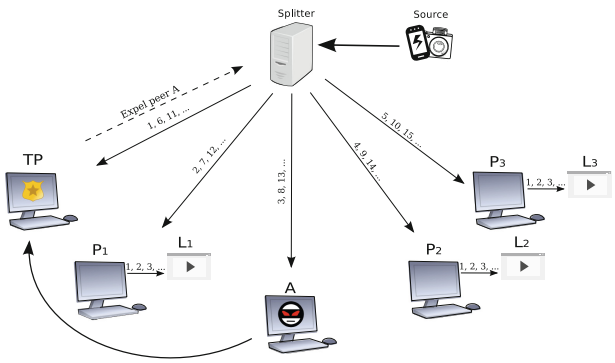


Fig. 1. A P2PSP team using STrPe-DS.

1. A peer receives the splitter's public key (S_{pub}) from the splitter, plus other necessary information, when a peer joins the team.
2. The splitter sends a different chunk and a message (m) to different peers. The message includes the chunk number (`cNumber`), the destination address (`dst`) and a digital signature $m = \{\texttt{cNumber}, \texttt{dst}, S_{priv}(H(\texttt{chunk}+\texttt{cNumber}+\texttt{dst}))\}$, where H is a hash function, and S_{priv} is the private key of the splitter.
3. When a peer receives a message from another peer, it performs the following steps:
   (a) Check whether `dst` matches the address of the sender. Notice that this action is vulnerable to the well-known *spoofing attack* [1].
   (b) If 3a was satisfied, it checks the correctness of the hash value in the message.

   The sender is removed from the list of peers of the current peer if 3a or 3b fails.
4. The splitter periodically requests, and the TPs serve, the list of removed peers since the previous request.
5. Peers removed by any TP will be expelled by the splitter after a random time, in order to disassociate the expulsion with the action taken by the attacker.

3.2 The Impact of Pollution Attacks

When an MP attacks a WIP or a TP, both know that they have been attacked (see Rule 3 on Sect. 3.1). For the sake of simplicity, the worse scenario is supposed: all MPs collaborate and attack anytime they want to, and the splitter does not use a reputation system. Some notation is defined in Table 1.

Table 1. Notation used for measuring the impact of attacks.

Symbol	Meaning
p	Probability of a TP detects an MP in a round
A_C	Total number of polluted chunks per round independently of their cNumber
C_M	Number polluted chunks with different cNumber per round and non-MPs
M	Number of MPs per round
T	Number of TPs per round
W	Number of WIPs per round
N	Number of peers in the team (T+W+M)

Non-selective Attacks. In a non-selective attack, MPs poison[3] chunks to the non-MPs. Non-selective attacks can be classified into persistent and on-off attacks. In the first case, an MP poisons all the chunks received from the splitter.

[3] Or refuse-to-send, which is equivalent for our analysis.

In the second case, MPs poison a chunk with a probability q. Because STrPe-DS is memory-less, in a persistent attack, $q = 1$, $A_C = N - M$ and $C_M = M$. For on-off attack, $p = q$, $A_C = q(N - M)M$ (each MP performs $q(N - M)$ attacks) and $C_M = qM$.

Non-selective attacks are not appealing for MPs in the presence of TPs because there exist a trade-off between the probability of being detected (qT) and the number of attacks per round (A_C).

Selective Attacks. $M = 1$ is not interesting to overthrowing the network because it can pollute a maximum of $N-1$ peers and an attacked peer will have just one polluted chunk out of N, per round. Using selective attack, trying to escape from TPs, will reduce the impact of the non-selective attack.

The probability to be discovered by a TP increases with the number of attacked peers. Therefore, a collaborative-selective attack is more interesting for attackers. Several collaborative-selective attacks can be performed:

Each one-to-one: Each MP attacks one different peer. Its impact is low because $C_M = 1$ and $A_C = M$. The only motivation of this type of attack is to scan the team looking for TPs. The expelled MP can determine who could be the TP(s).

Each one-to-many: Each MP attacks a set of n peers. The motivation is to increase the impact of each one-to-one attack. Different objectives arise:

Increase C_M: MPs attack the same set of non-MPs. So, $A_C = Mn, C_M = M$ for n peers and $C_M = 0$ for the rest. The aim of this attack is to hinder the reproduction of the stream in the set of attacked peers because degrading the streaming quality will cause that a WIP leaves the team.

Increase the scouting of TPs: MPs attack a different set of n non-MPs, minimizing the overlap among them. The altered chunks/round $A_C = Mn$. When the sets do not overlap, $C_M = 1$ for Mn peers. The aim is to narrow the TPs location when $M << N$. Otherwise, *each one-to-one* attack is preferable.

Many-to-one: All MPs attack just one peer. In this case, $A_C = M$ and $C_M = M$ for the attacked peer and $C_M = 0$ for the rest. Its aim is to increase the probability that the attacked peer leaves the team due to the low quality of the stream. MPs may reduce their exposure if a new prey is not selected while the current one is on the team.

Many-to-many: Each MP performs a *one-to-many* attack, but MPs collaborate to determine the targets.

MPs have a difficult multi-objective optimization problem: to (i) maximize the number of detected TPs, (ii) minimize the number of MPs (cost), (iii) minimize the number of expelled MPs, (iv) maximize C_M in WIPs and (v) perform previous tasks in a minimum number of rounds.

TPs have also a multi-objective optimization problem: to (i) minimize the number of MPs (this minimize C_M in WIPs), (ii) minimize the number of TPs (cost) and (iii) performs previous tasks in a minimum number of rounds.

The difficulty of both problems increases with the existence of churn. Several solutions can be devised, and each one will present a possible countermeasure. Following, we show the strategies studies here as possible non-optimal solutions.

Discovering and Hiding TPs. MPs must devise an algorithm to detect TPs. In a simple approach, each MP increases the set of attacked peers by just one per round. MPs collaborate to attack a peer that has not been attacked before. Under the assumption that a TP informs to the splitter as soon as it is attacked, the MPs team can collaborate to make out who is a TP when an MP is expelled.

For $T = 1$, the TP can be discovered in $\lceil \frac{N}{M} \rceil$ rounds. The expelled MP will inform to others MPs in the team and MPs will attack the discovered TP anymore. Then, the impact of the attack would be $p = 0$, $A_C = N - T - M$ (all WIPs), $C_M = M$ in the worst case.

To overcome the previous algorithm, the splitter will expel an MP in one of the next rounds (at random). Moreover, a TP should leave the team in next rounds (at random) when it informs to the splitter about an MP, and another TP with different identity should be included in the team. In this way, MPs have more difficulties to figure out who a TP is.

4 Experimentation

The experimentation is performed to measure the number of affected peers during the attack, the number of affected chunks (per peer and round), the average expulsion time for MPs, the average number of WIPs and MPs and the state of the buffer of WIPs. Time is expressed in rounds.

4.1 Test Bed

For all the experiments we used a simulator specifically coded for evaluating different aspects of the P2PSP protocol. Both, the protocol and the simulator are open source and they are available at [12,13]. They were run on a Unix machine (Ubuntu server) with 2.3 TB of RAM and 8 Intel Xeon E7 8860v3 (16 cores) processors.

Churn:

1. The initial team consists of a splitter and an initial number of WIPs (#IWIPs).
2. To avoid any tracking of TPs, they will be incorporated into the team at random, mixed with the arrival of the rest of peers (MPs and WIPs) up to reach a $\#\text{TP}_{max}$.
3. As $\#\text{TP}_{max}$, the values of the maximum number of WIPs ($\#\text{WIP}_{max}$) and MPs ($\#\text{MP}_{max}$) limit the number of entries of WIPs and MPs in the team along the simulation.
4. A new peer (MP or WIP) arrives with probability P_{in}. The peer will be WIP with probability P_{WIP} and MP with probability P_{MP}. Additionally, when a new TP is needed, it joins the team also with probability P_{in}.

5. Peers (all types) stay in the team a period of time modeled by a Weibull distribution with the shape parameter $a = 1$ [11].
6. Events in peers are triggered when a new chunk arrives at them. A peer that spends its Weibull time will leave the team with probability P_{out} in next event.

MPs Behavior:

1. MPs always attack (persistent, selective and collaborative attack).
2. An MP should not attack over 50% of the team to reduce the probability of being detected.
3. MPs should scout for TPs. We define NAY as the set of peers Not Attacked Yet. Only one MP attacks to a NAY peer. NAYs peers that were not attacked by expelled MPs are attacked before others.
4. Additionally, MPs should attack peers as much as possible. Then, every MP also attacks peers that have been attacked more than MPTR (Malicious Peer Test Round) by other MPs.
5. MPs share the list of attacked peers and how many rounds an MP was attacking each of them. Thus, they know the NAY and peers attacked more than MPTR rounds.

WIP Behavior Under Attacks:

1. They will leave the team if the Exponential Smoothing of Chunks Loss Rate (ESCLR) is greater than ESCLR_{max}. ESCLR defined as:

$$\text{ESCLR}_i = \alpha \text{CLR}_i + (1 - \alpha)\text{ESCLR}_{i-1} \tag{1}$$

where CLR_i is the chunks loss rate measurement in last round.
CLR is defined as:

$$\text{CLR} = \frac{\text{polluted/lost chunks in B}}{|\text{B}|} \tag{2}$$

where B is the buffer of the peer and |B| is the size of B.

TPs Behavior Under Attacks:

1. TPs inform to the splitter about attacks as soon as they are attacked.
2. TPs, as WIPs do, will leave the team if the ESCRL>ESCLR_{max}.
3. A TP will leave the team after it complains about an MP with probability P_{TPL} in order to avoid its detection by MPs.

Splitter Behavior Against Attacks:

1. A reported MP will be expelled in current round with probability P_{MPL}.

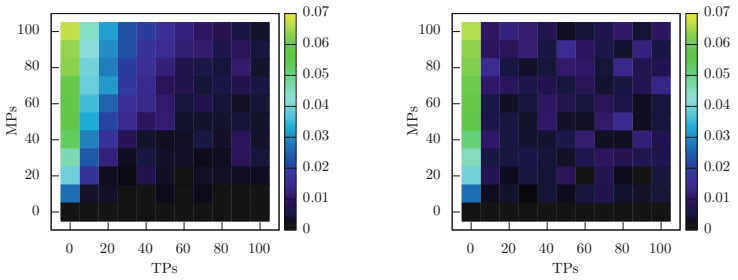
4.2 Experimental Results

The main aim of this study is to know if it is feasible to mitigate pollution attacks with previous strategies. We focus on the defense side and the tune of P_{MPL} and P_{TPL} values. For all experiments P_{in}=1, P_{out}=0.25, P_{MP}=P_{WIP}=1, MPTR=2 and ESCLR_{max}=1 (peers stay on the team even when the stream has a low quality). $\#\text{TP}_{max}$ and $\#\text{MP}_{max}$ values were taken from the 100 combinations of values in {0, 10, 20, ..., 90, 100}. The simulation stops 100 rounds

after the requested number of peers (#WIP$_{max}$+#TP$_{max}$+#MP$_{max}$) got in the team. Additionally, P_{MPL} and P_{TPL} values used in each of the previous combination are shown in Table 2. Therefore, the number of experiments is 1100. Five executions have been carried out for each experiment, and after discarding the outliers, the average of the remaining three values was returned.

Table 2. Parameter values for the different experiments.

Exp	1	2	3	4	5	6	7	8	9	10	11
P_{MPL}	100	100	100	100	100	100	80	60	40	20	0
P_{TPL}	100	80	60	40	20	0	100	100	100	100	100



(a) Exp 1. (b) Exp 6.

Fig. 2. Numerical results. Each small square is a combination of #TP$_{max}$ and MP$_{max}$ values. Right column shows ESCLR values. The darker the better.

All the instances in Table 2, except Exp. 6, show similar results to Exp. 1 instance, shown in Fig. 2a. Therefore, variation in P_{MPL} and P_{TPL} has not a great impact on the results. In general, when #TPs≥#MPs the ESCLR is low.

Figure 2b shows the results for Exp. 6 with P_{TPL}=0. Now, TPs do not leave the team during the entire simulation. The value of MPTR=2 and MPs sharing their "safe to attack" list including TPs produce that all MPs attack TPs and consequently they are expelled. A solution for MPs is to increase the number of attacked peers just by one in each round and a different one by each MP. In this way, the number of MPs attacking a TPs due to the "safe to attack" report of other MPs is much smaller than the entire MP set.

Figure 3 depicts the state of the team and the chunk-loss ratio for the Exp. 1 and two different combinations of #TP$_{max}$ and #MP$_{max}$. In order to have a stable state in the team before the attack, the initial rounds, where only WIPs are allowed, are not plotted). These results show that TPs can expel almost all MPs when their number is similar (all MPs if TPs are a majority), but in general TPs cannot do it when they are a minority. In such case, the remaining number of MPs is approximately #MP$_{max}$ − #TP_{max}. Therefore, other MPs' control

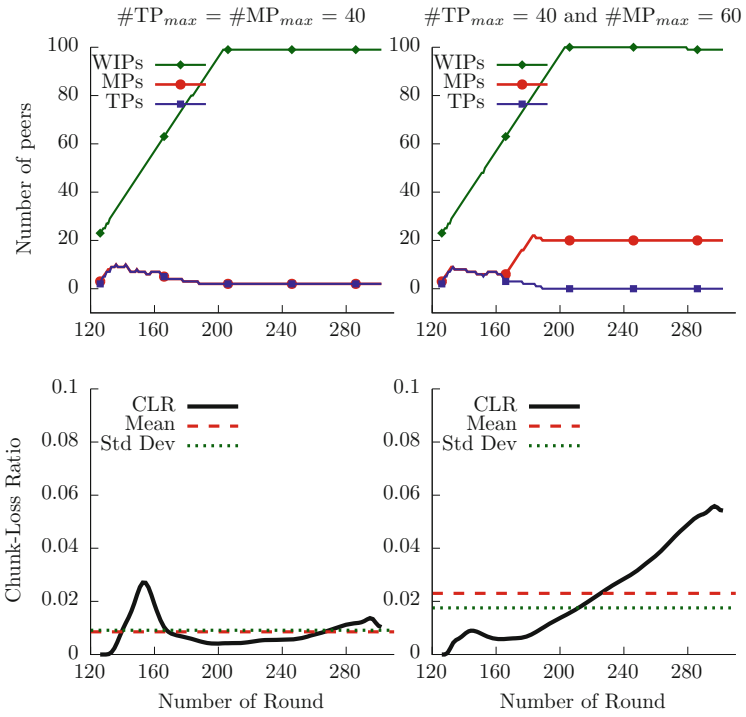


Fig. 3. Examples of changes in the composition of the team for Exp. 1 (up) and their consequences in the chunk-loss (down).

mechanisms, such as reputation systems, well behavior rewards, etc. should be added to the use of TPs. Notice that most of these systems have been studied for pull based P2P systems, but there exist few studies for push based systems, like the P2PSP [8].

5 Conclusions

Attackers and defenders have to solve a multi-objective optimization problem and each decision made by one part may have a countermeasure by the other part. A possible collaborative-selective algorithm performed by attackers is devised with two main objectives: (i) to discover trusted peers and (ii) to pollute as much as possible the buffer of well-intended peers without been expelled. At the defense side, the proposed algorithm tries to hide the trusted condition of a peer to attackers. Both algorithms are interesting because the following question could arise: Who are the good/bad guys? The answer depends on the actual scenario. In a normal one, the stream belongs to defenders. But good guys are the attackers when bad guys perform an illegal public free rebroadcast in a P2P network of their private stream, i.e., the stream belongs to attackers.

The experimental results show that using only trusted peers as a defense strategy is not appealing when the number of malicious peers can be large. Therefore, the improvement of the policies and additional techniques should be addressed. It will be done as a future work.

Acknowledgments. This paper has been supported by the Spanish Ministry (TIN2015-66680), in part financed by the European Regional Development Fund (ERDF). Cristóbal is supported by an FPU Fellowship (FPU14/00635) from the Spanish Ministry of Education (MECD). The work done by Ilshat was funded by Google Summer of Code (GSoC) 2015 initiative.

References

1. Bellovin, S.M.: Security problems in the TCP/IP protocol suite. SIGCOMM Comput. Commun. Rev. **19**(2), 32–48 (1989)
2. Borges, A., Almeida, J., Campos, S.: Fighting pollution in P2P live streaming systems. In: 2008 IEEE International Conference on Multimedia and Expo, pp. 481–484. IEEE (2008)
3. Conner, W., Nahrstedt, K.: Securing peer-to-peer media streaming systems from selfish and malicious behavior. In: Proceedings of the 4th on Middleware Doctoral Symposium, MDS 2007, pp. 13:1–13:6. ACM, New York (2007)
4. Dhungel, P., Hei, X., Ross, K.W., Saxena, N.: The pollution attack in P2P live video streaming: measurement results and defenses. In: Proceedings of the 2007 Workshop on Peer-to-Peer Streaming and IP-TV, pp. 323–328. ACM (2007)
5. Gaeta, R., Grangetto, M.: Identification of malicious nodes in peer-to-peer streaming: A belief propagation-based technique. IEEE Trans. Parallel Distrib. Syst. **24**(10), 1994–2003 (2013)
6. Hu, B., Zhao, H.V.: Pollution-resistant peer-to-peer live streaming using trust management. In: 2009 16th IEEE International Conference on Image Processing (ICIP), pp. 3057–3060, November 2009
7. Kang, X., Yongdong, W.: A trust-based pollution attack prevention scheme in peer-to-peer streaming networks. Comput. Netw. **72**, 62–73 (2014)
8. Lin, E., de Castro, D.M.N., Wang, M., Aycock, J.: SPoIm: A close look at pollution attacks in P2P live streaming. In: 2010 18th International Workshop on Quality of Service (IWQoS), pp. 1–9 (2010)
9. Marti, S., Garcia-Molina, H.: Identity crisis: anonymity vs reputation in P2P systems. In: 2003 Proceedings of the Third International Conference on Peer-to-Peer Computing (P2P 2003), pp. 134–141. IEEE (2003)
10. Medina-López, C., Casado, L.G., González-Ruiz, V.: Pollution attacks detection in the P2PSP live streaming system. In: Herrero, Á., Baruque, B., Sedano, J., Quintian, H., Corchado, E. (eds.) International Joint Conference. AISC, pp. 401–410. Springer, Heidelberg (2015)
11. Stutzbach, D., Rejaie, R.: Understanding churn in peer-to-peer networks. In: Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement, IMC 2006, pp. 189–202. ACM, New York (2006)
12. The P2PSP Team: P2PSP war-games repository. https://github.com/P2PSP/war-games
13. The P2PSP Team: Peer to Peer Straightforward Protocol Source Code. http://code.p2psp.org