Pollution and Whitewashing Attacks in a P2P Live Streaming System: Analysis and Counter-Attack

Rafael Barra de Almeida*, José Augusto Miranda Nacif†, Ana Paula Couto da Silva‡, Alex Borges Vieira*

*Universidade Federal de Juiz de Fora, Brazil
†Universidade Federal de Viçosa, Brazil; ‡Universidade Federal de Minas Gerais, Brazil
rafael.barra@ice.ufjf.edu.br; jnacif@ufv.br; ana.coutosilva@dcc.ufmg.br; alex.borges@ufjf.edu.br

***Abstract*—P2P live streaming are increasingly popular nowadays. Due to their popularity, these systems may be a target of attacks and opportunistic user behavior. In this paper, we address the pollution attacks in such systems. We present a pollution damage model and also analyze a reputation system as a tool to fight attacks in P2P live streaming systems. The model we propose evidences that attacks are harmful even in a system with a small number of polluters. In this case, peers must have more than 3 times network bandwidth than they should have in a system without polluters. Our experimental results on PlanetLab show that just check data integrity is not an effective protection. In this case, we observe a very high data loss rate. Finally, the reputation system is effective against pollution attack. When peers do not whitewash their identities, the reputation system quickly identifies polluters. In this case, the overhead and loss rate can be negligible. During a whitewashing, the new approach presents less than 20% overhead and 3% of loss.**

I. Introduction

Live streaming video over the Internet is one of the most important applications nowadays. A large number of TV stations around the world have already started to broadcast their live content. Moreover, world events as the 2012 Olympic Games have been transmitted over the Internet, attracting hundred thousands of users.

Traditionally, the client-server architecture has been used for video distribution. However, this approach presents problems such as a single point of failure and low scalability. In this context, the P2P architecture allows dealing with network failures and also increases the system scalability, enabling a large number of users without requiring high resources.

Almost all popular live streaming P2P application organize their peers in a mesh like overlay [1]. In these systems, a special peer encodes the streaming media and all other peers requests (or trade) streaming chunks to their partners. During chunks exchanges, peers may behave in a malicious and opportunistic way. Moreover, these malicious peers may take advantage of system's flaws to perform attacks.

We highlight 2 attacks to P2P system that can be combined together [2]: *the content pollution*: which is an attack where peers alter the content of a video before transmitting to their neighbors; *the whitewashing attack*: which occurs when a peer repeatedly leaves and re-joins the system with a new identity to avoid the penalties it may receive due to its malicious behavior.

Some defense attempts have succeeded in fighting specific attacks in P2P live streaming [2]–[4]. However, these solutions may fail when malicious peers perform whitewashing attacks. Due to the ease of getting a new identity and to the difficulty to characterize a whitewasher, this behavior becomes a challenge topic. Moreover, it is hard to implement a reputation system under a whitewashing attack [5].

In this work, we analyze the pollution and whitewashing attack impact in P2P live streaming systems. Our contributions are twofold. First we have created a simple model that captures the damage a pollution attack causes to a mesh-pull P2P live streaming system. The model we propose shows that pollution attacks are harmful even in a system with a small number of polluters. Due an attack, peers may receive polluted data. Even if peers correctly identify all polluted chunks, polluters still alive in the system and will keep polluting.

Second, we developed a P2P live streaming application prototype and a reputation mechanism that fights pollution/whitewashing attacks. The system implementation on PlaneLab, allows evaluating the proposed reputation mechanism in a real environment. Our approach quickly blocks the pollution and whitewashing attacks: the overhead on the P2P system is less than 20% and the loss rate is less than 3%.

II. Related Work

We observe 2 main approaches to deal with pollution/whitewashing attacks in P2P systems. First, we may use a centralized entity to associate identities to system peers. This approach is not cost effective and also introduces a single point of failure in the P2P system [6]. Second, we may impose penalties to all systems' newcomers. However, Feldman *et al.* [7] show that this approach could affect the system scalability, as penalties are imposed to all systems newcomers, eve if we do not observe whitewashing attacks.

Authors in [2]–[4] propose reputation systems that take into account the individual experience of each participant and the network testimonial to fight attacks to the P2P live streaming system. For instance, Seibert *et al.* [2] mark newcomers as suspects and as consequence, the mechanism reduces the available resource to these peers. Authors believe that this mechanism discourage peers to practice whitewashing.

Chen *et al.* [5] present a model that tries to capture the behavioral differences between whitewashers and the other peers based on system observation over time. The authors state that each peer must maintain a sequence of similar actions even

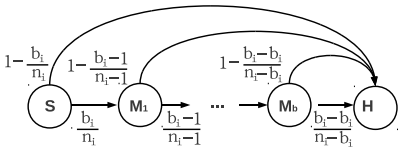


Fig. 1. Requesting data until a success.

when they leave and re-join the system. Once the malicious behavior of the peer is identified it is possible to deny any attempt to connect to another peer in the system.

Lin *et al.* [8] present simulation results that show the impact of pollution attacks do not depend on network size. They also show that the success of pollution attacks is highly correlated with the stability and bandwidth of the polluters. Hu and Zhao [9] propose a system that detects pollution and identifies attackers. Pollution is detected as early as possible and trust management is used to identify polluters. They present two schemes to address the tradeoff between pollution resistance and system overload. With these schemes, when a peer identifies an attacker, it immediately checks the buffer to prevent the spread of polluted chunks. Although this scheme helps to reduce the overload, preventing propagation of polluted chunks, data retransmission is still needed and is responsible for the most part of system overhead.

III. Pollution Attack Discussion

A. Pollution attack/damage model

We define a P2P live streaming as a system with m peers that collaborate to each other to watch a live transmission. There is a special peer, called *server*, which encodes the video and starts the transmission. The *server* also splits the video into *chunks* before it starts sharing the live content. Each video chunk has an id number, normally starting as 0.

Each system peer p_i has n_i partners. They also have a buffer B_i to store video chunks before they are played or shared. Each peer delimits its buffer size and initializes each available buffer position as empty. We index the buffer as a sliding window where $B_i[s]$ represents the less recent data we need (s is the last available buffer space). In the same way, $B_i[0]$ represents the most recent data we need. At each time interval, p_i consumes the less recent data and discharges it. It also shifts buffer index turning the new $B_i[0]$ an empty space.

Periodically, peers exchange buffer maps with their partners. Thus, they know their partners available chunks and then, they can schedule chunks requests. There are 2 main chunk schedule requests. Peers may schedule requests depending on chunks availability (Rarest First - RF), or they may schedule depending on playback deadline (Earliest Deadline First - EDF). The RF politic tries replicating a chunk as soon as possible while a EDF politic try to make playback smoother. We assume a RF approach during our work, as peers have to watch the content with a low latency. For P2P systems, the rarest chunk is the last chunk provided by the server. It means that, for the system we are dealing with, the very recently chunk (earliest one) is a rare chunk. Then, a peer has to search for this specific chunk over all its neighbors.

During a pollution attack, the P2P system presents b malicious peers, also called *polluters* ($b \leq m$). Each non-polluter peer p_i, called *good peer* has n_i partners. Among its n_i partners, peer p_i have b_i polluters, where $0 \leq b_i < n_i$. When a good peer p_i receives a chunk h, which is polluted, it has to discharge it and request again to a different partner. We assume that system peers use an effective way to determine if a chunk is polluted or not, as the ones proposed in [10]. The peer p_i will continue requesting for the damage chunk h until it can consume it. In other words, until the $B_i[h]$ has not been discharged from the buffer sliding window.

We assume, w.l.o.g, a homogeneous P2P system. We also assume that a peer p_i has enough resource to serve its partners. Moreover, the number of p_i partner does not alter significantly. Finally, we assume that the b polluters are equally distributed among all peers' partnerships. This way, we can predict how many times a peer p_i will have to request for a chunk h before it gets a non-polluted data.

Figure 1 shows the process a peer p_i does until it receives a valid data (a data "hit"). First, from the initial state S, peer p_i chooses one of its partners to request data. If all partners have equivalent resources and chunk maps, p_i may randomly select any one partner. If p_i chooses a *good peer*, it will have a data hit. Other case, a polluter will be chosen and p_i will have to ask data to another partner (discharging the polluter from its candidates). Peer p_i repeats this process until receiving a valid data. As we assume that peers have at least one good partner ($0 \leq b_i < n_i$), a peer p_i will get a hit in the worst case, if it asks for all polluters' partners.

Due to space constraints, we briefly discuss the overhead we observe during the process of requesting data. As p_i presents n_i partners whose b_i are polluters, the probability to reach a hit in the first attempt is $1 - (b_i/n_i)$. If p_i gets a bad data, it removes the polluter from its candidate partners and repeat the process. At the second attempt, p_i will have a success probability of $1 - \left({}^{b_i - 1}/_{n_i - 1}\right)$.. The worst case, p_i will have to ask data to each polluter partner it has the final probability is $1 - \left({}^{b_i - b_i}/_{n_i - b_i}\right)$; in other words, $p = 1$ (because it no longer has polluter in its candidate partners list).

$$
\begin{aligned}
l &= \left[1 - \left(\frac{b_i}{n_i}\right)\right] * 1 \\
&+ \left[1 - \left(\frac{b_i - 1}{n_i - 1}\right)\right] * \frac{b_i}{n_i} * 2 \\
&\cdots \\
l &= 1 - \left(\frac{b_i}{n_i}\right) + \sum_{s=2}^{b_i+1} 1 - \left(\frac{b_i - (s-1)}{n_i - (s-1)}\right) * s * \prod_{j=0}^{s-2} \frac{b_i - j}{n_i - j} \quad (1)
\end{aligned}
$$

Equation 1 presents the mean overhead a peer p_i has to reach a data hit. The overhead is the number of data miss it has until p_i reaches the final state in Figure 1. The overhead is proportional to the number of polluters partners a peer has. The minimum network resource a peer needs to get a chunk is 1 and the maximum is $b_i + 1$ chunk streaming rate.

B. Reputation model and chunk selection scheduling

The base model previously discussed is influenced by the chunk selection scheduling strategy. Considering the rarest first (RF) chunk scheduling, peers will observe a different chunk distribution among their partners. In the basic model (section

III-A), homogeneous chunk availability is assumed. When a peer tries to schedule a rare chunk, it will have fewer options among its partners. Since polluters can forge data, announcing a fake chunk map, they may attract peers request peers do not have any other partner with the rare chunk.

The rarest chunk in the P2P streaming system is the one the server just created. When the server announces its chunk map, only the peers that are connected to it may have the opportunity to get the rarest chunk. When the server creates a new chunk, it shifts its buffer map. At this moment, some of server partners may also have the previously rarest chunk and thus, it is no more the rarest in the system.

Figure 2 shows this chunk diffusion process. During the first moment, shown in figure 2(a), only the server has the chunk. The next moment, shown in figure 2(b), some of server partners received this chunk (shifted to the next server buffer position). Finally, figure 2(c) shows the scenario where the chunk reaches peers 3 that are hops away from the server.

(a) (b) (c)

Fig. 2. Chunk diffusion process in a P2P live streaming system.

Until the chunk reaches at least one partner of a peer p_i, it will not be able to request the chunk. But, if the P2P system is under an attack, polluters may announce fake data. Until the system reaches an homogeneous state to a given chunk, peers will be under a severe pollution attack. We also note that, even if a chunk has been well spread, only a portion of p_i partners have the desirable chunk. It increases the proportion polluters/good-partners and, as consequence, the pollution attack damage. We denote w as the proportion of good peers that can serve data to peer p_i. Equation 2 presents the new inflated pollution overhead. The new overhead is inflated by the mean network radius h.

$$l = h + 1 - \left(\frac{b_i}{w}\right) + \sum_{s=2}^{b_i+1} 1 - \left(\frac{b_i-(s-1)}{w-(s-1)}\right) * s * \prod_{j=0}^{s-2} \frac{b_i - j}{w - j} \quad (2)$$

where h is the mean distance server/peer and $w = y * n_i - b_i$

C. Model result analysis

We have analyzed the previous overhead pollution model using PRISM [11] probabilistic model checking tool. Parameters we use were collected during a SopCast live session [12].

TABLE I
PARAMETERS USED IN THE MODEL EVALUATION.

Parameter	Description	Value
m	system peer numbers	1000
n_i	mean number of partners	50;100
b_i	mean number of polluters partners	1; 10
y	Proportion of useful partners	0.5 ; 1
h	Mean distance server/peer	1.677

Table I presents mean values we use to evaluate our models. In a scenario where peers have 100 partners and 1 polluter among they, we have found an expressive pollution overhead. Even if all partners can serve a request ($y = 1$), we have found about 167% of data overhead due retransmissions imposed by pollution. When we grow the number of polluters to 10, the overhead increases to 177%. When we consider 50 partners to each peer, the overhead due pollution increases to 170% (when all partners can attend a request). If only 50% of partners are useful to attend a request, this number jumps to 330%.

In sum, our results show that even in a scenario where peers have a large number of partners, and most of all can serve chunks, pollution attacks severely impacts the P2P system. Peers must have up to 3 times more network bandwidth than it should have in a system without polluters.

IV. Defense Mechanism: Reputation

In this work, we use a simple decentralized reputation mechanism, based only on the individual experience from each peer to its partners [4]. In this case, a peer p_i periodically computes the reputation of each partner p_j, namely $R_i[p_j]$, based on the average of amount of satisfaction it receives for each transaction with p_j. According to eq. 3, we calculate the amount of satisfaction based on the total chunks *r* p_i requests to p_j. As response to the r chunks requests, partner p_j can provide *n* bad responses (where $0 \leq n \leq r$).

We define a bad response the one that forces p_i to ask data again to another partner. It can be, for instance, a response with a polluted chunk or non intentional damaged package. If the ratio n/r is above threshold T_i^{max}, p_i decreases p_j's local reputation. Otherwise, it increases its local reputation. We assume that peers have a homogeneous chunk request distribution, and as consequence, the ratio is a fair metric to measure the amount of satisfaction a peer perceives from its partners. Finally, the final reputation of p_j at p_i, $R_i[p_j]$ is:

$$R_i[p_j] = \begin{cases} max(0, R_i[p_j] - \alpha_{p_i} * (1 + n/r)^{y_i}) & \text{if } n/r > T_i^{max} \\ min(1, R_i[p_j] + \alpha_{g_i} * (1 - n/r)) & \text{otherwise} \end{cases} \quad (3)$$

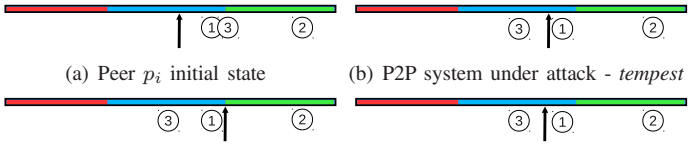
where y_i is the exponential penalty factor, α_{p_i} and α_{g_i} are penalty and reward factors, respectively. We made $\alpha_{p_i} \geq \alpha_{g_i}$ and $1 < y_i \leq 2$ to quickly identify and penalize polluters. All newcomers receive an initial reputation score. Each peer p_i has a reputation threshold R_i^{min} ($0 \leq R_i^{min} \leq 1$). If $R_i[p_j]$ drops below R_i^{min}, p_i removes p_j from its partners set.

The potential problem of relying solely on the individual experience to compute reputation is that once peer p_i considers partner p_j as a polluter, p_j will never have the opportunity to exchange data with p_i again. The peer p_j will never be able to rehabilitate itself. Therefore, in order to allow its partners rehabilitation, a peer p_i dynamically changes its minimum reputation threshold R_i^{min} reacting to the network condition.

If p_i senses that the network is under attack, it increases R_i^{min}, penalizing its bad partners quickly. Otherwise, it decreases R_i^{min}, enabling new interaction to previously punished partners. We define two system states, namely *calm* and *tempest*. A system is in *calm* state to a peer p_i if it believes there is no one polluter in the system. Otherwise, the system is in *tempest*. Periodically, p_i checks the system state and updates R_i^{min} according to equation 4. If the system is *calm*, p_i decreases its local threshold R_i^{min} by γ_{g_i}; otherwise R_i^{min} is increased by γ_{p_i}. We use $\gamma_{p_i} > \gamma_{g_i}$ so as to react faster to polluters. The limits RT_i^{min} and RT_i^{max} such that $0 \leq RT_i^{min} \leq R_i^{min} \leq RT_i^{max} \leq 1$.

$$R_i^{min} = \begin{cases} max(RT_i^{max}, R_i^{min} + \gamma_{p_i}) & \text{if system is in } \textit{tempest state} \\ min(RT_i^{min}, R_i^{min} - \gamma_{g_i}) & \text{if system is in } \textit{calm state} \end{cases} \quad (4)$$

Each peer infers the system state basing only on its experiences with its partners: if peer p_i receives *any* polluted data from one of its partners it suspects that the system is under a *tempest*. Figure 3 shows how a peer p_i reacts to pollution. In this figure, the horizontal bar represents a reputation scale from low (left) to high (right) values, the arrow represents the current value of R_i^{min}, whereas numbers 1, 2 and 3, indicate the current reputations at p_i of three partners (p_1, p_2 and p_3).



(a) Peer p_i initial state (b) P2P system under attack - *tempest*

(c) Malicious peers remain attacking (d) Malicious peers cease attack - *calm*

Fig. 3. Dynamic Minimum Reputation Threshold of Peer p_i.

Note that, in Figure 3-a, all partners are on the right side of the arrow, indicating their reputations are above the minimum threshold. As soon as p_i receives any polluted chunk, it sets the system state to tempest, increasing its minimum reputation threshold (Figure 3-b). Once the local reputation of peer p_3 falls, p_i understands that p_3 is a polluter and removes it from its partners set. Afterwards, p_i continues receiving polluted data and, once again, increases R_i^{min} (Figure 3-c). Peer p_1 is also removed from p_i's partners set, as its reputation is below the minimum threshold. After that, p_i stops receiving polluted data, thus changing the system state to *calm* and decreasing R_i^{min}. The new value of R_i^{min} allows p_i accepting p_1 as a partner once again. Note that changes in R_i^{min}, according to Equation 4, are performed independently of the changes in the local reputations of each partner, defined in Equation 3.

Polluters may often change their identities, performing a whitewashing attack combined to the pollution attack. Therefore, we create a new scheme to reduce the effects caused by whitewashing. In this new scheme, all newcomers peers receive low reputation score that is the limit that allows data exchange with other participants in the system ($R_i[p_j] \tilde{=} R_i^{min}$). Thus, any attempt to pollution will reduce $R_i[p_j]$ below R_i^{min} and then p_j will be removed from its list of neighbors. It is important to note that this low reputation still allows newcomers exchange data with other peers, so, they will not be punished if they have a good behavior.

In order to not punish good partners, we also propose that system peers keep a short history of their partnerships. Thus, as soon as a previously good partner join (or re-joins) the system, it may be reputed with a good reputation score. In our system, we make $R_i[p_j]$ equal to the last reputation p_i assigned to p_j. These simple modifications help to fight whitewashing and keep good system peers receiving a smooth streaming. The reputation mechanism is used during all the experiments in the scenario where polluters also do whitewashing.

V. Experimental Scenarios and Methodology

We have evaluated our reputation mechanism on a real setup running on PlanetLab. We have implemented a mesh-pull based P2P live streaming system, as discussed in section III-A.

We have setup a dedicated server in our campus network, transmitting a 30-minute CBR stream at 120 kbps bitrate. We have used 133 PlanetLab nodes, each of them running our live streaming system: 90% of them acting as good peers and 10% acting as polluters. We do not impose any additional processing/bandwidth constraints to the server or to the PlanetLab nodes. We have set all peers to remain active in the system throughout the transmission. In order to maintain the P2P overlay radius close to a real scenario (about 4 hops), we have limited each peer connection to at most 18 partners.

Polluters attack the P2P system since the moment they join the network. The attack lasts until the end of the live transmission. During the attack, polluters announce a complete chunk map and forges chunks to answer their partners' requests.

Any previously proposed data verification scheme (e.g., hash-based signature) [10] could be used to automatically identify polluted chunks. In this work, we disregard any overhead that may be introduced by these data verification techniques, focusing, instead, on the overhead caused by chunk retransmission due to the reception of polluted content.

We consider 2 different attack patterns. In the first one, polluters remain in the P2P system until the end of the experiments. In the second case, polluters attack the system and in addition, they re-joining the system every 215 seconds, assuming a new identity (whitewashing attack).

VI. Experimental Results

We analyze the reputation mechanism using 3 metrics: the chunks pollution rate, known as chunk miss rate; the pollution overhead; and the chunk loss rate. These 3 metrics are able to capture the damage a pollution attack causes to a P2P system. We plot mean results we found in our experiments and upper/lower bounds of a 95% confidence interval

Figure 4 shows the results in a scenario under pollution attack without whitewashing. The chunk miss rate, which is the probability to receive a polluted chunk in the first request, is almost 100% in a system that we do not try to isolate polluters. The simple reputation system decreases the miss rate from 90% to 6%. We clear note that, if peers do not have any chance to ask for the polluted data, the data verification scheme will lead peers to a complete streaming loss.

Figure 5 presents the pollution overhead in the same scenario. In a system using only the check data and retransmission approach, the overhead stabilizes at 230%. In this case, peers must have more than 3 times network bandwidth than in a system without polluters. In contrast, the simple reputation mechanism reduces the overhead to values as low as 5%.

Finally, we present system overall loss rate due to several retransmissions requests. Figure 6 shows that pollution attacks really impact a system using only the data verification approach. In this case, the chunk loss rate achieves values up to 82%. As a consequence, the peers will not be capable to watch a smooth streaming. Using the simple reputation mechanism, the loss rate drops to a negligible value, lower than 0.7%.

Figure 7 presents the chunk miss rate in a scenario where polluters whitewash their identities. Our results show that, even

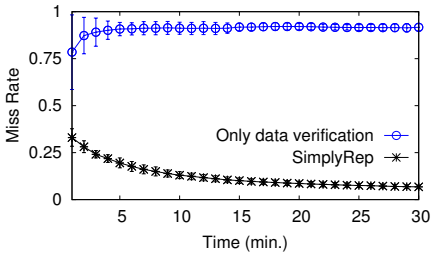


Fig. 4. Chunk miss rate.

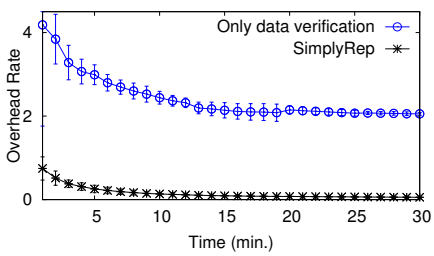


Fig. 5. Pollution overhead.

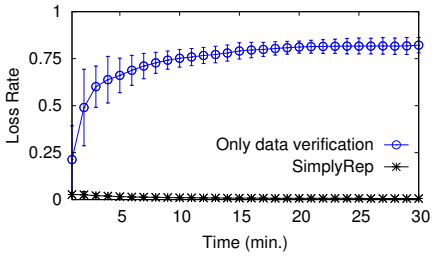


Fig. 6. Loss rate.

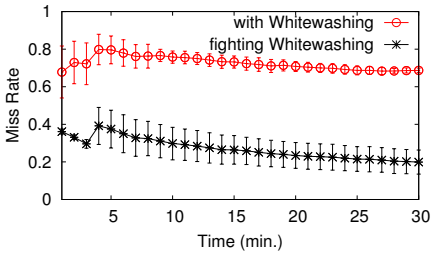


Fig. 7. Chunk miss rate.

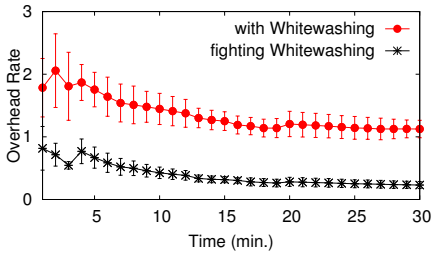


Fig. 8. Pollution overhead.

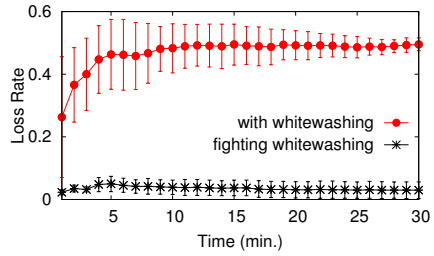


Fig. 9. Loss rate.

with the original simple reputation mechanism, whitewashing causes high miss rate (almost 70%). The simple reputation mechanism prepared to deal with whitewashing attacks reduces the chunk miss rate to 19%. Although, the probability to get a polluted chunk in the first request can not be ignored.

The pollution overhead also presents a significantly different behavior from a scenario where polluters do not whitewashing. Figure 8 shows that whitewash is an unwanted behavior. Using a non-prepared reputation mechanism, peers experience 112% overhead. In this case, peers should have more than two times the network bandwidth required in a scenario without attacks. The simple reputation mechanism prepared to deal with whitewashing could reduce this overhead to 20%.

Despite this non-negligible value to system overhead, we note the loss rate remains low. According to Figure 9, the loss rate when we use the simple reputation mechanism prepared to deal with whitewashing is lower than 3%. Moreover, we can not underestimate the viability to use the reputation mechanism in practice. First, the new scheme is simple/cheap to implement. Second, our approach does not need to maintain a costly centralized/distributed identification mechanism commonly used to avoid whitewashing.

VII. Conclusions

In this work, we study the impact o pollution content dissemination in a P2P live streaming environment. Moreover, we analyze combined attacks as polluters collusion and whitewashing. We have created a simple model that captures the damage a pollution attack imposes to a mesh-pull P2P live streaming system. We also have developed a mesh-pull based P2P live application to evaluate reputation mechanisms.

Our model evidences that pollution attacks are harmful even in a system with a small number of polluters. Due the time constraints and chunk scheduling policies, system peers are susceptible to get a non-negligible number of polluted chunks. Peers in the system must have more than 3 times network bandwidth than it must have in a system without polluters.

Our experimental results show that a simple reputation mechanism is effective against pollution attack and peers collusion, but when malicious peers do whitewashing, the mechanism becomes ineffective. Our new approach significantly reduce the effects of pollution/whitewashing presenting less than 20% overhead and 3% of chunk loss.

Acknowledgment

This work was partially supported by the Brazilian agencies: FAPEMIG, CNPq and CAPES.

References

[1] X. Hei, C. Liang, J. Liang, Y. Liu, and K. W. Ross, "A measurement study of a large-scale p2p iptv system," *IEEE TMM*, 2007.
[2] J. Seibert, X. Sun, C. Nita-Rotaru, and S. Rao, "Towards securing data delivery in peer-to-peer streaming," in *COMSNETS*, 2010.
[3] A. Vieira, J. Almeida, and S. Campos, "Fighting pollution in p2p live streaming systems," in *IEEE ICME*, 2008.
[4] A. Vieira, S. Campos, and J. Almeida, "Fighting attacks in p2p live streaming. simpler is better," in *IEEE INFOCOM Workshops*, 2009.
[5] J. Chen, H. Lu, and S. Bruda, "A solution for whitewashing in p2p systems based on observation preorder," in *IEEE NSWCTC*, 2009.
[6] N. Oualha and Y. Roudier, "A game theoretical approach in securing p2p storage against whitewashers," in *18th IEEE WETICE'09*, 2009.
[7] M. Feldman, C. Papadimitriou, J. Chuang, and I. Stoica, "Free-riding and whitewashing in peer-to-peer systems," *IEEE J-SAC*, 2006.
[8] E. Lin, D. de Castro, M. Wang, and J. Aycock, "Spoim: A close look at pollution attacks in p2p live streaming," in *IWQoS*, 2010.
[9] B. Hu and H. Zhao, "Joint pollution detection and attacker identification in peer-to-peer live streaming," in *IEEE ICASSP*, 2010.
[10] M. Haridasan and R. V. Renesse, "Securestream: An intrusion-tolerant protocol for live-streaming dissemination," *Elsevier ComCom*, 2008.
[11] M. Kwiatkowska, G. Norman, and D. Parker, "Prism: Probabilistic model checking for performance and reliability analysis," *ACM SIGMETRICS Performance Evaluation Review*, 2009.
[12] A. Vieira, P. Gomes, J. Nacif, R. Mantini, J. Almeida, and S. Campos, "Characterizing sopcast client behavior," *Elsevier ComCom*, 2012.