SWAP: protecting pull-based P2P video streaming systems from inference attacks

Giang Nguyen†, Stefanie Roos, Benjamin Schiller, and Thorsten Strufe†
TU Dresden † and SFB 912 - HAEC
{giang.nguyen, stefanie.roos, benjamin.schiller1, thorsten.strufe} <at> tu-dresden.de

***Abstract*—In pull-based Peer-to-Peer video streaming systems, peers exchange buffer maps to reveal the availability of video chunks in their buffer. When collecting these buffer maps, a malicious party can infer the system's overlay structure and even identify head nodes, the direct communication partners of the stream's source. Attacking these head nodes can isolate peers from the source resulting in a disruption of the video dissemination for most peers in the system. We introduce a lightweight SWAP scheme, which allows peers to proactively change their partners, to reduce the chance of head nodes to be identified by such an inference attacker. Extensive simulation studies demonstrate that our scheme effectively undermines the attack's accuracy in identifying head nodes. So, SWAP lowers the chunk miss ratio while causing only a slight increase in signaling overhead.**

I. Introduction

The constantly growing demand for the delivery of video streams requires a solution to disseminate them over the Internet in a cost-effective and resource-efficient manner. Peer-to-Peer (P2P) video streaming systems have become a viable solution. In these systems, the participating peers form an overlay network and provide resources, including upload bandwidth, to disseminate video streams. The main challenge is to construct overlays which are resilient to both churn and sabotage. Most deployed P2P video streaming systems use a pull-based architecture [1]. In these systems, peers establish and maintain a mesh overlay and explicitly request video packets (or chunks) from their partners. This requires peers to know which chunks are available at which partner. Hence, peers periodically exchange buffer maps, small packets containing information about the availability of chunks. Hei et al. [2] make use of these buffer maps to infer the overlay structure of PPLive [8], a pull-based system. Peers form distinguishable tiers that indicate the flow of the video dissemination from the source via its partners (or head nodes) to other peers.

The overlay structure revealed via this exchange of buffer maps can be abused by an adversary, called inference attacker, to perform Denial-of-Service (DoS) attacks to the system. If an attacker targets head nodes, the resulting damage can be severe and leads to the isolation of peers and the disruption of video dissemination. Therefore, it is critical for pull-based systems to reduce the chance of inferring head nodes from buffer maps.

Toward this end, we propose *SWAP*, a lightweight partner swapping scheme as a countermeasure against an inference attacker. Our approach is to increase the organized dynamics of the system, meaning that peers constantly change their partners in a proactive manner. Consequently, the attacker is unable to accurately infer the overlay structure and identify head nodes. Specifically, each peer nominates one of its partners as a candidate to replace itself. The nomination can be forwarded several times to ensure a more diverse swapping, which further hinders the inference attacker. We evaluate the accuracy in identifying head nodes using both theoretical bounds and concrete simulations. Our comparison of *SWAP* with state-of-the-art solutions indicates that *SWAP* greatly reduces the severity of an inference attack.

Our contributions in this paper are two-fold: First, we show that inference attacks are highly damaging when based on an accurate identification of head nodes. Second, we present *SWAP*, a partner swapping scheme as the countermeasure against the inference attacker. Our simulation studies indicate that SWAP drastically reduces the damage caused by the inference attacks.

The remainder of this paper is structured as follows: After discussing the related work in Section II, we describe the inference of the overlay structure in Section III. Afterwards, Section IV and Section V present the attacker model and our SWAP scheme respectively. Next, we introduce the analysis of the inference attacker and the countermeasure in Section VI. After discussing the results in Section VII, we conclude the paper in Section VIII.

II. Related work

The construction of resilient overlays, which are robust to churn and resistant to attacks, for video dissemination has been an area of active research for more than a decade. In the following, we describe the most important approaches, differentiated in push-based and pull-based systems.

Push-based systems like Probabilistic Resilient Multicast (PRM) [3] and FatNemo [4] organize peers into a single multicast tree, i.e., each peer receives the complete video stream from a single parent peer. PRM strengthens the overlay by establishing redundant connections to increase connectivity. FatNemo reduces the height of the tree by moving peers with more children closer to the source. Both approaches are robust to random failures but cannot protect the system against attacks, since these systems depend on a few peers close to the source. When those nodes are shut down, the whole system is disrupted.

978-1-5090-2185-7/16/$31.00

To address the fragility of these single-tree topologies, multi-tree overlays have been proposed. Each peer has multiple parents, which forward parts of the video stream, so-called stripes, to their children. Systems like [5], [6] tackle both problems of churn and sabotage by minimizing the direct dependency between any two peers throughout all trees. Peers are organized into inner-node disjoint spanning trees, which delivers the chunks in separate stripes. The resilience of those systems was proven theoretically, but they have not been deployed in real-world environments.

The majority of deployed systems like DONet [7] and PPLive [8] are pull-based: Peers construct a mesh overlay and periodically request video chunks from their partners. The mesh overlay allows peers to maintain multiple source-to-peer paths, which improve pull-based systems' robustness to churn. However, a recent measurement study on PPLive shows that the information from buffer maps reveals the overlay structure of the system, which is clustered into tiers. The knowledge of this structure enables the attacker to shut down the partners of the source, the so-called head nodes, which heavily disrupts the video dissemination. The striping scheme [9] addresses these issues by enforcing peers to request chunks from a diverse subset of partners, which significantly increases the number of partners. This makes it more costly for attackers to remove all head nodes and thereby increases the system's resilience. However, the striping scheme cannot avoid the inference attack itself.

In summary, the problem of constructing resilient overlays for P2P streaming systems has been tackled often to achieve robustness to churn and resistance to DoS attacks. However, how to protect the overlay of pull-based streaming systems against inference attacks is still an open question.

III. Inferring the Overlay Structure

The information from buffer maps in a pull-based P2P video streaming system can provide knowledge about the overlay structure of the system [2]. Fig. 1 illustrates the mapping from the availability of chunks in a video buffer into a buffer map consisting of an array of "0" and "1" bits. The buffer head's sequence number (or *buffer head*, for short) indicates the latest chunk in the buffer. Peers' buffer heads differ from one another in general depending on network conditions (such as jitter) and who their partners are. For example, a peer connecting directly to the source can receive video chunks significantly earlier than others, who do not have direct connections to the source. Comparing the buffer heads of peers buffer maps, therefore provides their relative position in the overlay structure.

Let's consider two peers u and v at time t_i with their buffer maps arrived at time $t_{u,i}$ and $t_{v,i}$ containing buffer heads $h_{u,i}$ and $h_{v,i}$, respectively. Let L denote the chunk size and r the bit rate of the video stream. The relative offset of u with respect to v is defined as follows:

$$\delta_{u,i} := (h_{u,i} - h_{v,i}) - (t_{u,i} - t_{v,i}) \cdot r/L \qquad (1)$$

Now, we compute $\delta_{u,i}$ for all peers u over time using a common reference point. Peers' relative offsets exhibit a clear and constant ordering. From those offsets, peers closer to the source are clearly distinguished from those farther away.

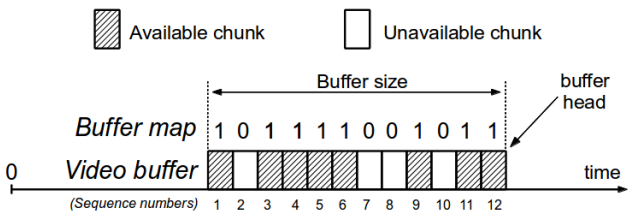

Fig. 1. A buffer map, partially filled with downloaded video chunks and available for playing back. Buffer's head is the newest available chunk.

IV. The Inference Attacker

In this section, we describe the attacker model used in the remainder of this paper. We derive the model from the inference technique presented in the previous section. That technique has been used by Hei et al. [2] to infer the overlay structure of a small setup of PPLive. We extend the technique in the inference attacker model to identify head nodes and experiment it on DONet, a conventional yet more generic pull-based system.

The attacker's goal is to prevent peers from receiving a continuous video stream for a certain period of time. To achieve this, it attacks a set of peers to break the overlay structure of the system. We assume that an attacker *A* is able to shut down arbitrary nodes except the source within the system. However, *A* is limited with regard to the number of nodes *x* it can shut down at any given time, referred to as the *attacker's budget*. Given this limitation, the attacker must attempt to shut down the most relevant peers, the head nodes, to cause the most damage for some period after the attack.

In the following, we outline the approach followed by the inference attacker. It consists of three steps: *probing*, *inference*, and *attack*.

1) Probing buffer maps: The attacker collects buffer maps from peers to aggregate the contained information and consequently infer the overlay structure of the system. This accumulation of buffer maps can be performed either by actively requesting them or by storing them from regular buffer map exchange operations. While the regular buffer map exchange is a valid operation in all pull-based systems, active requests are only allowed in some systems, e.g., in [2]. Regardless of the way this collection is implemented, buffer maps are periodically gathered in batches, called *probes*. In an ideal attack scenario, the attacker obtains the buffer maps from all active peers in the system for each probe. However, to be more realistic, we assume that the attacker obtains a buffer map from a peer with a probability q $(0 \leq q \leq 1)$ which depends on network latency, congestion, etc. Subsequently, we assume that an attacker performs a total of m probes $p_1, p_2, \ldots p_m$ starting every T_p seconds. These probes should be performed shortly before the attack to ensure an up-to-date information.

2) Inferring the overlay structure: First, the attacker selects a reference peer v, potentially under its control, whose buffer

map appears in all probes. Using Eq. 1, the attacker calculates the offset $\delta_{u,i}$ for each buffer map from a peer $u \neq v$ collected during the probe p_i. After m probes, the attacker calculates the average offset $\bar{\delta}_u$ of each probed peers u as follows:

$$\bar{\delta}_u := \frac{\sum_{i=1}^{m} \delta_{u,i}}{m} \tag{2}$$

3) Attacking the system: Head nodes receive video chunks before other peers, so that their averaged offset should be higher. Therefore, the attacker sorts the list of all probed peers u in descending order of their averaged offsets $\bar{\delta}_u$. From this sorted list, the attacker selects the top x nodes to attack and thereby shut down. Assuming that these nodes are closest to the source, attacking them should cause the greatest damage possible with the attacker's resources.

In summary, this section presents the inference attacker, a practical attack model that works in three steps: (*i*) collecting information about peers by probing their buffer maps; (*ii*) inferring the system's overlay structure; and (*iii*) executing attacks by targeting the most promising peers. Note that we use a simplified version of this strategy for our theoretical analysis to limit the complexity.

V. The *SWAP* scheme

In this section, we describe the *SWAP* scheme to counter the inference attacker. Such a countermeasure needs to satisfy the following three requirements:

1) The countermeasure should reduce the attack's accuracy significantly in order to effectively mitigate the negative impact of the attacker.
2) The performance penalty should be reasonably low.
3) The additional overhead should be low.

To fulfill the first requirement, there are two possibilities to undermine the attack's accuracy to infer the overlay structure, especially in identifying head nodes. First, we could insert bogus information into the exchanged buffer maps so that the attacker cannot reliably infer the overlay structure. Second, we could enforce the overlay structure to constantly change, so that the overlay structure inferred by an attacker is quickly outdated. Adding bogus information can confuse the attacker, but it confuses benign peers as well, causing inefficient requests for video chunks. Consequently, this reduces the overall system's performance. Hence, this approach directly violates the second requirement and is therefore eliminated.

The second approach requires peers to proactively change their partners. Consequently, the system introduces more dynamics and overhead. However, we have also learned that pull-based P2P streaming systems are highly robust to the dynamics of peers. So, there potentially exists a certain level of dynamics that minimizes negative impacts to the system while allowing the system to mitigate damages caused by an inference attacker.

A. Basic idea behind the SWAP scheme

The basic idea of our *SWAP* scheme is to enforce peers and especially the source to proactively change their partners. This means to replace an existing partner with another peer (or so-called *replacement partner*). At this point, there are two follow-up questions for the swap operation:

1) Which partners should be dropped?
2) Which replacement partner should be selected?

The answer to the above two questions should minimize negative impacts on the system. It is however hard to fulfill the requirement if peers have to decide from their local knowledge only. Our strategy is to leverage the information in buffer maps received from a peer's partners. They provide peers a little more information about their neighboring peers and consequently the local structure between themselves.

To simplify the explanation in this section, we introduce the concepts of upstream and downstream partners of a peer v, meaning whether they tent to receive video chunks earlier or later than v does. To quantify the timing of received chunks, we use Eq. 2 to calculate the average offsets of v's partners with regard to v itself from recently received buffer maps. Let $\bar{\delta}_u$ denote the average offset of v's partner u. Since $\bar{\delta}_v = 0$, u is called an upstream partner of v if $\bar{\delta}_u > 0$ and a downstream partner otherwise. The separation of partners basing on their average offsets is illustrated in Fig. 2.

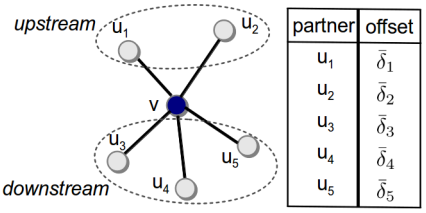

Fig. 2. Peer v's partners: upstream (u_1, u_2) and downstream (u_3, u_4, u_5), given their average offsets satisfy $\bar{\delta}_1 \geq \bar{\delta}_2 \geq 0 \geq \bar{\delta}_3 \geq \bar{\delta}_4 \geq \bar{\delta}_5$ ($\bar{\delta}_v = 0$ since v is the reference point for itself)

Our answer to the first questions is that a peer selects a downstream partner to drop, since dropping an upstream partner reduces the chance of the peer itself in obtaining video chunks. Regarding the second question, a downstream peer should also be selected to replace the dropped partner because it can introduce a potential security issue to do otherwise. Allowing peers to connect to upstream peers enables malicious peers to swap frequently, hoping to mount up the overlay structure to block the source node and disrupt the video delivery to benign peers.

To summarize, our idea to defend a pull-based P2P streaming system against the inference attack is to allow peers as well as the source to proactively change their partners. Both the dropped partner and its replacement should be selected from downstream peers. Following the above high-level sketch, the design of *SWAP* will be detailed next.

B. Design of the SWAP scheme

The main concern here is the selection of a replacement partner. A straightforward solution is to request a random peer from the membership service, such as a tracker keeping a list of active peers in the system. It is, however, very unlikely

that peers can obtain downstream replacement partners as they need. Therefore, preparing a downstream replacement partner for swapping operations requires a collaboration between peers. SWAP realizes the collaboration by the following two primitives, namely *partner nomination* and *nomination forwarding*. After describing those two primitives shortly, we present the swap operation.

1) Partner nomination: To prepare for swapping operations, every peer should suggest its replacement candidate for itself once the peer is dropped by one of its upstream partners. For that, a partner nomination is required, in which a downstream peer is suggested as the replacement partner for the swapping operation. The follow-up question is how a peer should nominate its partners. In SWAP, a peer nominates its downstream partners to its upstream ones in a Round-Robin manner. For example, in the Fig. 2, two among the three downstream partners $\{u_3, u_4, u_5\}$ can be randomly nominated to the upstream ones $\{u_1, u_2\}$. In this way, SWAP nominates partners evenly to minimize the chance of a partner to be selected by many peers. Nomination should be performed periodically to ensure a constant availability of replacement partners.

2) Nomination forwarding: Accepting nominations from partners immediately only allows peers to swap with nearby peers in the overlay. That cannot help the system from an inference attacker with a large budget. SWAP solves that problem by requiring peers to forward each nomination several times towards upstream peers. This way, peers can connect to more diverse peers whose positions are farther away in the downstream, further hindering the attacker from inferring the overlay structure. Therefore, a counter in each nomination message is introduced. The counter is preset to n_f representing the total number of forwarding until the nomination is eventually accepted. After each forwarding, the counter is decremented. When the counter of a nomination reaches zero, peers should accept that nomination. Decrementing the counter also prevent nomination messages from circulating forever in the system, causing overhead. Fig. 3 illustrates two examples for swapping, where n_f equals one and two respectively. In the former case, A accepts the nomination from C and connects to this node before dropping its current partner B. In the latter case, the nomination from D is accepted by A after being forwarded via C and B. Afterwards, A connects to D before dropping B.

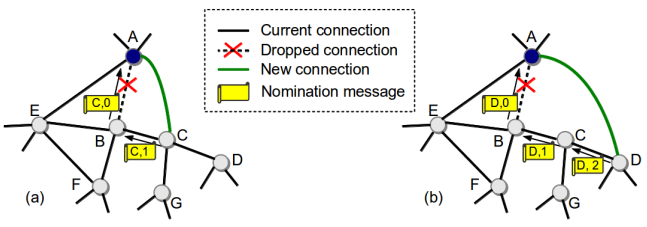

Fig. 3. Swap operation in which peer A: a) accepts C's nomination forwarded via B, connects to C and drops B ($n_f = 1$); or b) accepts D's nomination forwarded via C and B, connects to D and drops B ($n_f = 2$). The nomination message contains the name of the nominated peer and a counter, which is decremented after each forwarding at an intermediate node.

3) Swap operation: Since an attack can happen at any time, the system should be ready against those attacks all the time. The swap operation is therefore executed regularly by each peer. In each swap cycle, a peer selects a few partners and swap them with their respective replacement partners obtained via nominations.

4) Parameters: Our SWAP scheme is characterized by three parameters for the nomination and swap operations. The number of nomination forwarding n_f needs to be specified. The minimum value of n_f is one, meaning to swap with nearby peers in the overlay. The greater n_f is, the further downstream peers can be selected for swapping. A too large n_f, however, can lead to circulated nominations and increases the overhead. The swap operation uses two parameters, namely the swap interval T_s and the number of swapped partners m_s in each swap operation. Their combination can influence significantly the resistance to attacks as well as the overall performance of the systems. Swapping too frequently or swapping too many partners at once might help the system to change the overlay structure more swiftly and drastically to avoid inference attacks. However, this also increases dynamics to the system that it might be unable to sustain.

C. Specification and Implementation

To integrate our *SWAP* scheme into an existing pull-based P2P streaming system, we describe essential adaptations as follows. To allow for periodic swapping and partner nomination operations, two timers are required for each peer. The first timer is triggered every T_n seconds to update the peer's partners about its nominated peer's address. The second timer is triggered every T_s seconds to execute the swap operation and its parameters.

VI. Theoretical Analysis

The purpose of this section is twofold. First, we aim to show that an adversary can easily identify head nodes in a stable system by frequently requesting buffer map probes. In particular, we determine a lower bound on the attack's accuracy, i.e., the fraction of sabotaged head nodes, for m requested buffer maps. Based on this result, one can also determine an upper bound on the number of probes required to achieve a certain accuracy.

Second, we aim to prevent the attacker from determining the head nodes. For this purpose, we proposed to regularly swap downstream nodes. In this section, we determine how to choose the swapping interval T_s such that the attack's accuracy is guaranteed to remain below a certain threshold. In contrast to the first bound, we now determine an upper bound on the attack's accuracy if the attacker can request up to m buffer map probes in one swapping interval. If we require the attack's accuracy to be below a certain threshold, our results enable us to determine a number m of buffer map probes the attacker may be allowed to request in order to confirm with the required threshold. The length of the proposed swapping interval is determined by multiplication of m with the time between two subsequent requests for buffer map probes.

A. Identifying Head Nodes

We aim to derive a lower bound on the attack's accuracy without topology changes. Such a lower bound indicates that the attacker can identify critical nodes with high accuracy at a low cost. First, we formalize the problem of identifying the head nodes in a stable system. Based on the formalization, we then simplify the mathematical model to use common notions from the area of probability theory. Last, we obtain the desired bound for the simplified problem.

In order to obtain lower bound on the accuracy an attacker, we propose one attacker strategy and then compute a lower bound for the proposed strategy. Note that we do not claim that the proposed attack strategy inevitably offers the maximum accuracy. However, as we are interested in a lower bound on the achievable accuracy, analyzing one strategy is sufficient.

Recall that N is the total number of nodes in the system, N_h the number of head nodes, and x the attacker's budget. Furthermore, the attacker requests m probes of buffer maps. A node's buffer map is contained in a probe with probability q. Let $Y_{m,p}$ be the random variable denoting the attack accuracy, i.e., the fraction N_x/N_h with N_x denoting the number of head nodes the attacker sabotages. Formally, we thus aim to determine a lower bound on $\mathbb{E}(Y_{m,p})$, with the expectation being defined over all possible buffer map probes.

In order to solve the problem, we first introduce some additional notation. Let $ord(i,j)$ be the j-th node to receive the i-th chunk. We enumerate the buffer maps probes with $\alpha = 1 \dots m$, and let $C(v) \subset \{1,\dots,m\}$ indicate the probes containing v's buffer map. Note that C corresponds to a function from the set of nodes V into the power set $\mathcal{P}(\{1,\dots m\})$, i.e., the set of all subsets, of possible buffer map probes indices $\{1,\dots,m\}$. Furthermore, for the α-th probe, let $lci(\alpha, v)$ denote the index of the latest chunk in the buffer map of v if $\alpha \in C(v)$. We can now formalize our assumptions and results in an adequate manner.

We make the following assumptions to obtain in our mathematical model for deriving the attack's accuracy. First, we assume that the order in which nodes receive chunks is the same for all chunks, i.e., $ord(j) = ord(i_1, j) = ord(i_2, j)$ for all i_1, i_2. As a consequence, we can enumerate nodes $v_1,\dots,v_N$ such that $ord(j) = v_j$. Furthermore, we assume that a probe of buffer maps only contains buffer maps from one specific time t_0, rather than buffer maps from varying points in time. In this manner, v_j has received at least as many chunks as v_l for $j < l$, i.e., if $\alpha \in C(v_j) \cap C(v_l)$, then $lci(\alpha, v_j) \geq lci(\alpha, v_l)$. Last, we assume that all head nodes receive chunks before non-head nodes, i.e., the head nodes correspond to the nodes $v_1,\dots,v_h$. In practice, network dynamics, inhomogenous latencies, and network jitter preclude the above assumptions. Our model could account for such an instability, as long as the variation is small, by including probabilistic changes in the order. However, the drastically increased model complexity is disproportional to the expected gain. We assume that the attacker identifies head nodes based upon the 'intuition' that their latest chunk is of a higher index. The above assumptions merely formalize this intuition in a straightforward manner.

Now, we describe the attacker strategy for which we derive the lower bounds. First, the attacker only includes nodes for which it received any buffer maps in its list of potential head nodes. So, let $V_C = \{v \in V : C(v) \neq \emptyset\}$ be the set of such nodes. Furthermore, let $v \prec_a u$ denote the fact that the adversary perceives v to receive chunks earlier than u. Let $M_C(v) = \{u : v \prec_a u\}$ be the set of nodes the adversary perceives to receive chunks later than v for the buffer map probes defined by the sets $C(v)$. We suggest the attacker to sabotage x nodes $v \in V_C$ for which $|M_C(v)|$ is maximized, i.e., the attacker sabotages nodes such the number of nodes which are guaranteed to receive chunks later than these nodes is maximized. The motivation for the strategy is the increased probability that such nodes are likely to be close to the source as many nodes receive chunks later. It remains to detail how the adversary derives $M_C(v)$, or more specifically the partial order relations, given the buffer map probes. Our key observations for determining partial order relations between v and u for one probe are

1) $v \prec_a u$ if $\alpha \in C(v) \cap C(u)$ and $lci(\alpha, v) > lci(\alpha, u)$, and
2) $v \prec_a u$ if there exists a node w with $v \prec_a w$ and $w \prec_a u$.

The first condition holds because clearly v_j received one chunk before u and thus by consistency of the order always receives chunks first. The second condition follows by the transitivity of a partial order. Thus, by successively considering each probe and applying the above rules, the attacker can determine $M_C(v)$ for all $v \in V_C$ and thus select its x nodes, breaking ties randomly. With regard to the proposed attack strategy, we can now derive a lower bound on the expected attack accuracy. Note that this attack strategy is different to the one suggested in Section IV. The reason lies in the fact that the order of receiving chunks is not necessarily strictly constant in practice due to network jitter. Nodes with similar delays to the source receive chunks in a different order. Thus, the buffer map probes in practice do not result in a partial order, so that a more complex strategy is required to overcome such differences in the retrieval order. However, in our simplified model, the above attack strategy is sufficient and reflects the main idea of both attack strategy: identify head nodes due to their low delay to the source. By using a simplified, possibly less accurate attack strategy, we obtain a good lower bound and focus on the main ideas for the attack's effectiveness.

For simplification, we assume that u and v's buffer maps always allow us to determine the partial order relation between the nodes, i.e., we have either $lci(\alpha, u) > lci(\alpha, v)$ or $lci(\alpha, v) < lci(\alpha, u)$. In practice, the above assumption clearly does not hold. As the number of pairs such that $lci(\alpha, v) = lci(\alpha, u)$ depends on the streaming rate and the latencies, allowing for this possibility increases the model complexity and reduces the generality of the approach by incorporating system-dependent parameters. Thus, we first

consider the simplified problem.

Proposition VI.1. *Let*

$$\forall j \neq l, \alpha \in C(v_j) \cap C(v_l) \implies lci(\alpha, v_j) > lci(\alpha, v_l). \quad (3)$$

Then a lower bound on the expected attack's accuracy is

$$\mathbb{E}(Y_{m,q}) \geq \sum_{C:V \to \mathcal{P}(\{1,\dots,m\})} \frac{\{j \leq N_h : |V_C| - |M_C(v_j)| < x\}}{N_h} \prod_{i=1}^{m} q^{|C(v_i)|} (1-q)^{m-|C(v_i)|} \quad (4)$$

with $M_C(v_j) = \{v : v_j \prec_a v\}$ denoting the set of nodes the attacker perceives to receive chunk later than v_j.

Proof. Let BM denote buffer map probes known to the attacker and consider one realization of BM uniquely defined by a function C. We derive a lower bound $acc(C)$ on the accuracy of the attacker for the buffer map probes C assuming that the attacker utilizes the above strategy. Afterwards, we determine the probability $P(BM = C)$ for this buffer map probe. Formally, we hence obtain the bound by

$$\mathbb{E}(Y_{m,q}) = \sum_{C:V \to \mathcal{P}(\{1,\dots,m\})} acc(C) P(BM = C). \quad (5)$$

In order to determine $acc(C)$, recall that the attacker sabotages x nodes v for which the sets $M_C(v)$ is of maximal size. Consider that if less than x nodes are contained in $V_C \setminus M_C(v_j)$, there are also less than x nodes v with $|M_C(v)| \geq |M_C(v_j)|$. We shortly prove the above statement. Let $v \in M_C(v_j)$, i.e., we have $v_j \prec_a v$. Furthermore, for all nodes $w \in M_C(v)$, $v \prec_a w$ holds. Hence, by the definition of $M_C(v)$, we have $M_C(v) \subset M_C(v_j)$. Due to the anti-symmetry of $\prec_a$, $v_j \notin M_C(v)$ and thus the subset $M_C(v)$ is a real subset of $M_C(v_j)$. So, $|M_C(v)| < |M_C(v_j)|$ for all $v \in M_C(v_j)$. Thus, a head node v_j is guaranteed to be sabotaged if all but x nodes are contained in $M_C(v_j)$, resulting in the lower bound

$$acc(C) \geq \frac{\{j \leq N_h : |V_C| - |M_C(v_j)| < x\}}{N_h} \quad (6)$$

on the attack's accuracy.

Now, we compute the probability $P(BM = C)$ of certain buffer map probes. Note that each probe contains the buffer map of a node v with probability q. Thus, the probability of v's buffer map to be contained in the probes specified by $C(v)$ is $q^{|C(v)|}(1-q)^{m-|C(v)|}$. As buffer maps are probed independently, we indeed get

$$P(BM = C) = \prod_{i=1}^{N} q^{|C(v_i)|} (1-q)^{m-|C(v_i)|}. \quad (7)$$

The claim follows by inserting Eq. 6 and Eq. 7 in Eq. 5. □

Note that Eq. 4 require computation cost exponential in the number of requested buffer maps. Thus, we apply Monte Carlo sampling when computing the desired bounds for our evaluation.

B. Swapping Interval

In this section, we determine an upper bound on the expected attack accuracy if all nodes switch m_s downstream partners each T_s seconds. From such a bound, we can obtain the swapping interval T_s and the number of swapped nodes f per interval in order to maintain an attack's accuracy below a certain threshold. We assume that the attacker continuously requests buffer maps and then decides to sabotage x nodes at some point in time. We bound the attack's accuracy of this attack. Note that our bound holds regardless of the attacker's strategy.

Proposition VI.2. *Let T_p be the interval at which the attacker requests buffer map probes, q be the probability of node's buffer map to be probed, let $T_s = m \cdot T_p - T_{init}$ be the swapping interval with T_{init} denoting the time a new head node requires to catch up to the old head nodes with regard to the number of received chunks and m_s be the number of swapped nodes. Let A_x denote the event that newly selected head nodes are not chosen from the x nodes currently marked for sabotage. An upper bound on the expected attack accuracy Z_{m,q,m_s} at any point in time is given by*

$$\mathbb{E}(Z_{m,q,m_s}|A_x) \leq \frac{1}{m} \sum_{j=1}^{\infty} \left(1 - \frac{m_s}{N_h}\right)^{j-1} \frac{m_s}{N_h} \sum_{i=0}^{m-1} \left(1 - (1 - q + q(1-q)^x)^{mj+i}\right). \quad (8)$$

Proof. The main idea of the proof is that the attacker has to be able to compare the buffer maps of new head nodes with its current x candidates for sabotage before it considers the node as a new head node. This need for comparison is independent of the actual attacker strategy. We first derive an upper bound on the probability that a head is detected using γ buffer map probes and secondly derive the probability that the attack requested γ buffer maps since the head node caught up with the other head nodes. Formally, let I be the event that a random head node is correctly identified and R the number of buffer map probes requested by the attacker after the new head node caught up.

$$\mathbb{E}(Z_{m,q,m_s}|A_x) = \sum_{\gamma=1}^{\infty} P(I|R=\gamma) P(R=\gamma). \quad (9)$$

In the following, we determine $P(I|R=\gamma)$ and $P(R=\gamma)$.

When considering $P(I|R=\gamma)$, note that for one probe, i.e., $\gamma = 1$, the probability that a comparison is not possible is given by $1 - q + q(1-q)$. The probability follows because either v's buffer map is not contained in the probe or v's buffer map is contained but none of the buffer maps of the x nodes. As the probes are chosen independently, we obtain

$$P(I|R=\gamma) = 1 - (1 - q + q(1-q)^x)^{\gamma}, \quad (10)$$

the complementary probability of the event that a comparison is not possible for all γ probes.

It remains to determine $P(R = \gamma)$. Note that $R = m \cdot R_1 + R_2$ with R_1 denoting the number of swaps and R_2 denoting the number of buffer map probes considered since the last swap. R_2 is uniformly distributed in $0, \ldots, m-1$. For determining R_1, note that the probability to swap a certain head node is $\frac{m_s}{N_h}$, the fraction of swapped head nodes per swap. Thus, the number of swaps since a certain node was last swapped is given by a hypergeometric distribution with parameter $\frac{m_S}{N_h}$. Rewriting $\gamma = j \cdot m + i$, we obtain

$$P(R = \gamma) = P(R_1 = j, R_2 = i) = \left(1 - \frac{m_S}{N_h}\right)^{j-1} \frac{m_S}{N_h} \frac{1}{m}. \tag{11}$$

Thus, we have derived the missing terms in Eq. 9. The claim in Eq. 8 follows by inserting Eq. 10 and Eq. 11 in Eq. 9 and elementary mathematical operations. □

We have derived the desired bounds on the attack's accuracy for our (simplified) model. In the following, we compare these bounds to the attack's accuracy in a simulation study and relate the attack's accuracy to the quality of service, measured by the chunk miss ratio.

VII. Evaluation

In this section we investigate the impact of the inference attacker on pull-based systems with and without *SWAP*. Specifically, we would like to answer the following three questions: *i*) How accurate does the inference attacker identify head nodes? *ii*) To which extent does *SWAP* increase the resilience of pull-based systems against the inference attacker? *iii*) How large is the additional overhead of *SWAP*? We start by describing the metrics and our simulation first.

A. Metrics

To measure the resilience, we define the *chunk miss ratio* as the fraction of chunks that missed their play-out deadlines divided by all chunks that should be played out. We use that metric, instead of the commonly used continuity index, since the two metrics are complementary to each other and the miss ratio is more intuitive to quantify the system's damages. Furthermore, we introduce two subsequent metrics to look at the chunk miss ratio from two perspectives: ***average miss ratio*** which is the average chunk miss ratio over a significant period of time after the attack and ***maximum miss ratio***, which is the maxima of per-second chunk miss ratios during that period. The latter estimates the upper limit of damages that the attacks can cause to the system. Additionally, to understand the direct consequence of the inference attack, we introduce the ***attack's accuracy***, which is the fraction of accurately attacked head nodes over all available head nodes. Finally, we measure the ***signaling overhead***, which is the ratio of the signaling volume over the total exchanged volume.

B. Simulation model

Our evaluation is conducted using OSSim [10], which allows for packet-level simulations of different classes of P2P video streaming systems. Using OSSim, we implement DONet [7], a conventional pull-based system in the literature. We use DONet as the representative system for two reasons. First, DONet's design and protocol descriptions are documented in detail, which supports a verifiable implementation of the system in simulation. Second, DONet's performance is comparable to the state-of-the-art [11].

To emulate the realistic characteristics of the underlying Internet, we use the GT-ITM [12] topology generator to generate a transit-stub core network consisting of 20 core and 400 edge routers, interconnected by 1212 links. 1000 peers are randomly attached to the edge routers at the beginning of each simulation. To simulate peers' underlying churn model we use the Pareto and Lognormal distributions for inter-arrival times and session durations, respectively. The distribution and their corresponding parameters follow the measurement study by Veloso et al. [13]. In addition, we allow leaving peers to rejoin the system after a random period, to maintain a rather stable system size.

The stream source is constantly fed by a stream of 2500-Byte video chunks. The streaming bit rate is set to 400 kbps, which is a typical average rate reported in the literature [14]. Peers with video buffers storing up to 30 seconds of video chunks start playing out when roughly 20 percent of their buffers are filled. The upload bandwidth of the source and peers are 8 Mbps and 1 Mbps respectively. Even though unrealistic, the assumption of homogeneous peers are reasonable since it eliminates the impact of the peers' upload bandwidth in the results and focus only on the resilience of pull-based systems against attacks. The simulation duration is 1200 seconds and the attacks are performed at the $800th$ second, when the system already reaches its steady state. We repeat each simulation setting 30 times.

In order to evaluate the strength of the attack and our defense mechanism, we consider the impact of the probability q and the number of probes m to the accuracy of the the attack. The probing interval T_p equals 0.5 seconds in all settings. Next, we vary q between 0.1 and 1.0, while the value of m was varied between 1 and 5. The attacker's budget x is chosen between 5 and 50 in steps of 5. This completes our set-up for the evaluation. Due to space constraints, we only show selected parameter settings. However, the remaining results are similar and entail the same conclusions.

C. Results

In the following, we first summarize our main findings on the characteristics of the inference attacker. Afterwards, we evaluate our SWAP scheme in mitigating the negative impact of the inference attacker. This also includes the tradeoffs of our scheme.

1) On the accuracy of the inference attacker: In this section, we seek the understanding about the dependence of the attack's accuracy to the different parameters of the inference attacker model. Specifically, we would like to answer the question: How the probability q and the number of probes m influence the attack's accuracy. We compare our simulation results with the theoretical bound given by Eq. 4.

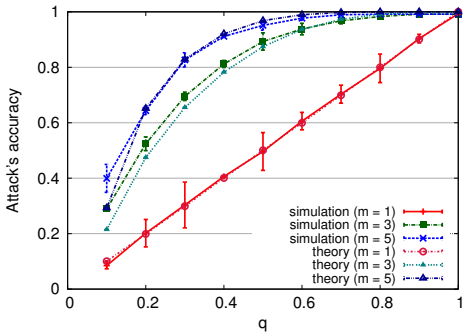

Fig. 4. The dependence of the attack's accuracy to the probing interval and the number of considered probes.

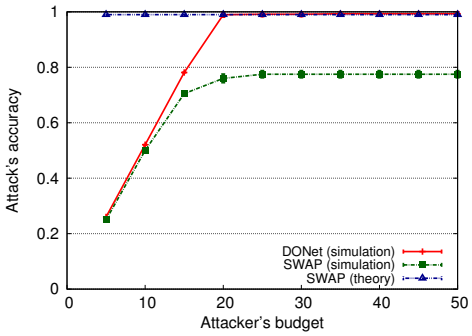

Fig. 5. Comparing the attack accuracy when attacking DONet and *SWAP*, with respect to the attacker's budget.

We expect that an increased value of q can significantly improve the attack's accuracy. Moreover, the higher the number of probes is used to infer the overlay structure of the system, the better the attack's accuracy is.

Fig. 4 displays the dependency of the attack's accuracy on the probability q and the number of probes m. The results show the following: (*i*) The lower bounds from the model predictions and the simulation results are very close to each other. Thus, the impact of network jitter on the delay order, which is ignored in our model, does not seem to have a significant impact, as the model's results are validated by the simulations. Indeed, the simplified model presents a lower bound on the actual attack's accuracy. The reason for the difference between model and simulation is given by the model's simplified attack strategy. The attacker only relates two nodes if it can establish a partial order. A head node v_h is only considered identified if the number of nodes without a clear partial order relation to v_h is less than the attacker's budget. In contrast, the more complex attack strategy introduced in Section IV can compare any pair of nodes. A head node might thus be compared against nodes for which no clear order can be determined. Even so the head node v_h might thus not be detected as receiving chunks earlier for certain, the attacker might still sabotage v_h as it is an equally likely targets as other nodes. Thus, the theoretical bound is slightly lower than the actual result for $m > 1$. (*ii*) The attack's accuracy is improved with an increased q and m. Specifically, when $m = 1$, the attack's accuracy increases almost linearly with an increase of q. However, with $m > 1$ the increase in the attack's accuracy is larger for a smaller q, while the increase is less for a larger q. As we can see from the figure, the results agree with our expectation. There are, however, some interesting observations. First, if the attacker can probe buffer maps of all peers in the system, it can identify all head nodes accurately with only one probe. Second, when q is small, probing several times can significantly improve the accuracy of the inference attacker.

In the coming section, we investigate how the SWAP scheme helps improve the system's resilience by undermining the inference attacker.

2) Comparing the resilience of SWAP *and DONet:* In this experiment, we would like to answer the question: To which extent does *SWAP* increase the resilience of pull-based systems against the inference attacker? We compare our simulation results with the theoretical bound given by Eq. 8.

As plotted in Fig. 5, the attack's accuracy increases drastically from around 0.25 to reach almost 0.8 and 0.7 in cases of DONet and *SWAP* respectively when x increases from 5 to 15. However, when $x \geqslant 20$, *SWAP* limits a stable attack's accuracy upto 0.8 while DONet loses most of its head nodes. Here, the theoretical upper bound is only of limited usefulness as it overestimates the attack's accuracy. For all parameter settings, the theoretical bound is close to 99 %. The reason for such a high difference between model and simulation lies in the generality of the model. The upper bound assumes that a new head node is detected as soon as it is first observed together with any of the previous candidate nodes. However, due to the new node's need to catch up to the other head nodes, the head nodes are generally identified much later than predicted. Nevertheless, the theoretical model is of interest as it gives a guaranteed upper bound, independent of the attacker's strategy. Even though a large fraction of head nodes of *SWAP* are correctly identified, the inference attacker still misses a significant fraction of them.

To explore the consequences of the attack's accuracy on system's resilience, we compare the chunk miss ratios caused by the attacks in both cases of DONet and SWAP. Subsequently, we collect the maximum and average chunk miss ratios. Results of both the *SWAP* scheme and DONet are plotted in Fig. 6(a) and Fig. 6(b), respectively. When $x \geqslant 20$, the inference attacks cause extremely high damages on DONet with the maximum and average miss ratios of almost 60% and around 15% respectively. Whereas, under similar attacker's budget, SWAP's maximum and average miss ratios remain below 3% and 2% respectively. Those results are consistent with the attack's accuracy and can be explained as follows. When some of the head nodes are not attacked, either by lacking attacker's budget or by incorrectly being identified, they still connect the source and the remaining peers. The stream delivery flow is affected but remains connected. Therefore, the chunk miss ratios are very small. However, when all the head nodes are correctly attacked, the remaining peers are unable to obtain video chunks from the source, causing extremely high chunk miss ratios.

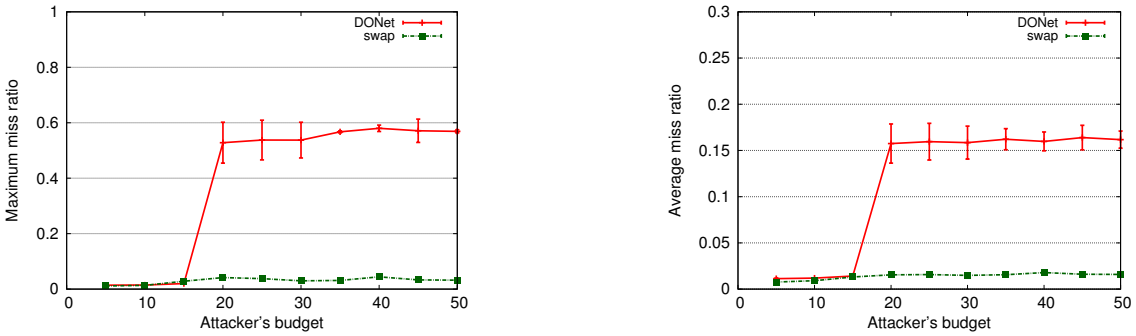

Fig. 6. Comparing the resilience of *SWAP* to DONet against the inference attacks in terms of the maximum (a) and average (b) miss ratios.

3) Signaling overhead of SWAP*:* Finally, we would like to answer the question: At which cost does *SWAP* perform in benign situations? Consequently, we discuss our finding on the cost of *SWAP* in terms of the signaling overhead in dependence with the peers' number of partners. The results are plotted on Fig. 7. When the number of partners increases from 6 to 16, the signaling overhead of DONet ranges from 2.0% to almost 4.5%. On top of that, *SWAP* generates less than 1% extra overhead in each configuration. Considering the *SWAP* alone, a larger number of forwardings n_f reduces the overhead. This is due to the fact that, the smaller the value of n_f the more peers are able to trigger swap operations, resulting in more dynamics and signaling overhead in the system.

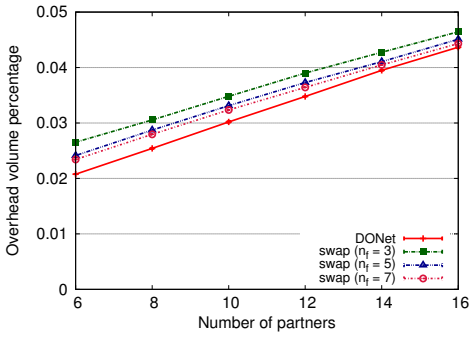

Fig. 7. Signaling overhead of DONet and *SWAP* (with different values of n_f) in dependence with the number of partners, in benign scenarios.

VIII. Conclusion

This paper addresses the problem of buffer map exchange in pull-based P2P video streaming systems, in which an inference attacker collects buffer maps to infer the overlay structure of the system, especially to identify head nodes for attacks. Using both theoretical model and simulation studies we showed that the inference attacker can identify and attack head nodes with a high accuracy, resulting in high chunk miss ratios. Subsequently, we introduced *SWAP*, a lightweight partner swapping scheme as a countermeasure against the inference attack. SWAP enforces peers to frequently change their partners to lower the chance of the inference attacker in identifying head nodes. Extensive simulation studies demonstrate that SWAP effectively undermines the attack accuracy, significantly lowering both the maximum and average chunk miss ratios at the cost of a slight increase in the system's signaling overhead.

IX. Acknowledgements

This work is supported (in part) by the German Research Foundation (DFG) within the Collaborative Research Center SFB 912 - HAEC.

References

[1] Y. Gu, N. Zong, Y. Zhang, F. Piccol, and S. Duan, "Survey of p2p streaming applications," October 2014. Available: https://tools.ietf.org/html/draft-ietf-ppsp-survey-09

[2] X. Hei, Y. Liu, and K. Ross, "Inferring network-wide quality in p2p live streaming systems," *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 9, pp. 1640–1654, December 2007.

[3] S. Banerjee, S. Lee, B. Bhattacharjee, and A. Srinivasan, "Resilient multicast using overlays," *IEEE/ACM Transactions on Networking*, vol. 14, pp. 237–248, April 2006.

[4] S. Birrer, D. Lu, F. Bustamante, Y. Qiao, and P. Dinda, "Fatnemo: Building a resilient multi-source multicast fat-tree," in *Web Content Caching and Distribution*, ser. Lecture Notes in Computer Science, C.-H. Chi, M. Steen, and C. Wills, Eds. Springer Berlin Heidelberg, 2004, vol. 3293, pp. 182–196.

[5] M. Brinkmeier, G. Schafer, and T. Strufe, "Optimally dos resistant p2p topologies for live multimedia streaming," *IEEE Transactions on Parallel and Distributed Systems*, vol. 20, no. 6, pp. 831–844, 2009.

[6] M. Fischer, S. Grau, G. Nguyen, and G. Schaefer, "Resilient and underlay-aware P2P live-streaming," *Computer Networks*, vol. 59, pp. 122–136, 2014.

[7] X. Zhang, J. Liu, B. Li, and Y.-S. Yum, "Coolstreaming/donet: a data-driven overlay network for peer-to-peer live media streaming," in *IEEE INFOCOM 2005*, vol. 3, March 2005, pp. 2102–2111.

[8] PPLive, 2015. Available: http://www.pptv.com

[9] G. Nguyen, M. Fischer, and T. Strufe, "On the resilience of pull-based p2p streaming systems against dos attacks," in *Stabilization, Safety, and Security of Distributed Systems*, ser. Lecture Notes in Computer Science, P. Felber and V. Garg, Eds. Springer International Publishing, 2014, vol. 8756, pp. 33–47.

[10] G. Nguyen, M. Fischer, and T. Strufe, "Ossim: A generic simulation framework for overlay streaming," in *Summer Computer Simulation Conference*, July 2013.

[11] Y. Zhou, D.-M. Chiu, and J. C. S. Lui, "A simple model for chunk-scheduling strategies in p2p streaming," *IEEE/ACM Transactions on Networking*, vol. 19, pp. 42–54, February 2011.

[12] E. Zegura, "Gt-itm: Georgia tech internetwork topology models," 1996. Available: http://www.cc.gatech.edu/fac/Ellen.Zegura/graphs.html

[13] E. Veloso, V. Almeida, W. Meira, A. Bestavros, and S. Jin, "A hierarchical characterization of a live streaming media workload," in *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, ser. IMW '02. New York, NY, USA: ACM, 2002, pp. 117–130.

[14] S. Xie, B. Li, G. Y. Keung, and X. Zhang, "Coolstreaming: Design, theory, and practice," *IEEE Transactions on Multimedia*, vol. 9, pp. 1661–1671, 2007.