On the Resilience of Pull-Based P2P Streaming Systems against DoS Attacks

Giang Nguyen[1], Mathias Fischer[1], and Thorsten Strufe[2]

[1] Department of Computer Science, TU Darmstadt
{nguyen,fischer}@cs.tu-darmstadt.de
[2] Department of Computer Science, TU Dresden
thorsten.strufe@tu-dresden.de

Abstract. The robustness of pull-based streaming systems to node failure and churn has been extensively analyzed. Their resistance to sabotage, however, is not well understood, so far. Recent measurement studies on a large deployed pull-based system have discovered stable source-to-peer paths and the convergence of the content dissemination to rather static topologies over time. Thus, an attack on central nodes within these static topologies, which causes serious service disruptions, is feasible. This paper demonstrates attacks that significantly reduce the system's performance. As a countermeasure, we introduce a novel striping scheme, which decreases the dependencies between peers and thus the impact of attacks. A thorough simulation study indicates that our scheme achieves a high resistance against sabotage attacks at negligible overhead and performance penalties.

Keywords: Resilience, pull-based P2P streaming, DoS attacks.

1 Introduction

Peer-to-Peer (P2P) streaming has been becoming a viable solution to distribute live streaming content over the Internet. Systems following this paradigm incorporate peers in the content distribution and make use of their upload bandwidth. Therefore, the provision for server resources is reduced and the service scales with an increasing number of users.

Most popular P2P streaming systems in practical deployment, e.g., PPLive[1] and Sopcast[2], can be classified as pull-based. In such systems [5,8] the stream is divided into equally sized chunks and peers download and forward those chunks between each other. This requires that each peer establishes and maintains partnership with other peers via bidirectional connections. This results in an unstructured and randomized mesh overlay. Peers inform others about the chunks that are downloaded and stored in the video buffers via Buffer Maps (BMs). A BM is a signaling packet containing an array of binary-valued elements that indicates

[1] http://www.pplive.com
[2] http://www.sopcast.com

chunk availability. After receiving BMs from its partners, a peer needs to decide from which partners it requests which chunks, e.g., via a Rarest-First scheduling strategy. Hence, a randomized distribution tree is formed implicitly per chunk, but in case of failures the mesh topology provides redundant connectivity via alternative source-to-peer paths. Even when one or several partners fail, each peer can quickly react to failures by downloading the video chunks from other partners. For this reason, pull-based systems are inherently *robust* to node churn and peer failures.

However, measurement studies [6,13] of one of the largest pull-based P2P streaming systems reveal that peers form different *tiers* in terms of play-out lags. The *stable* tiering effect allows for inferring the flow of the video content distribution. As stable source-to-peer paths evolve, the topologies established for subsequent chunks become highly similar. This might not affect the robustness of these systems to node churn and failures, but is of concern during attacks, e.g., DDoS attacks, on the most relevant nodes in the content distribution. As in tree structures the majority of nodes is residing in leaf positions and close to them, random failures of nodes will affect only few other nodes in average. However, attacks on nodes in the tier close to the source of the stream will affect nearly the whole overlay. Trees are robust against random failures, but not very resilient against attacks that target the most relevant nodes, e.g., nodes adjacent to the source (so-called head nodes). For this reason, we suggest to study the *total resilience* which we define as the *robustness* to failures as well as the *resistance* to attacks. For that, we assume an attacker with global knowledge that attacks head nodes only. To the best of our knowledge, this paper is the first that addresses this problem.

Our contributions in this paper are two-fold: (*i*) First, we demonstrate that the performance of pull-based systems is significantly affected by practical and simple attacks. (*ii*) Second, we introduce a striping scheme that *enforces diversification* as a countermeasure. Aiming at *reducing the direct dependency* between peers, our scheme divides the video stream into several stripes and enforces each peer to request the stripes from diversified groups of partners. Simulation results indicate that the striping scheme effectively reduces the maximum and average chunk miss ratios by 50% and 30% respectively, even with a conservative number of two stripes.

The remainder of this paper is structured as follow: Section 2 discusses related work. The striping scheme is described in Section 3. After discussing the results in Section 4, Section 5 concludes the paper.

2 Related Work

Studies on the resilience of P2P streaming systems mostly address either failure recovery or the resistance to attacks.

To prevent overlay partitioning, Probabilistic Resilient Multicast (PRM) [1] allows for redundant connections alongside a single multicast tree. Each peer can establish additional connections, with a low probability, to a few others and forward video chunks to them. It has been shown that the whole system can maintain a high delivery ratio.

In FatNemo [2] nodes with higher bandwidth are placed closer to the source, while nodes with lower bandwidth are placed further away. The resulting tree topology is low and broad. Intuitively, the less number of predecessors a peer has the more likely the peer can receive a stable video stream.

The above approaches are not resistant to targeted attacks since they introduce relevant nodes that are close to the source. Attacks on them can disrupt the whole system.

DagStream [10] introduces directed links on top of a mesh overlay. Each peer separates its mesh partners into parents and children. The peer requests chunks from its parents and sends chunks upon requests from its children. To optimize the topology, each peer has a level that is calculated from the ones of its parents. The farther the peer is from the source, the higher its level is. To avoid loops, a peer has to find a parent whose levels are lower than its level. This way of ordering peers hinders the collaboration between peers when there is a disruption in the overlay. Furthermore, the parent selection policies in DagStream prioritize peers that are close to the source. As a result, many peers might depend on a few parents. Attacks on those nodes can cause a heavy impact on a large fraction of peers.

To tackle both problems of node failure and sabotage, systems such as [3,4] extend the publish-subscribe design and minimize the direct dependency between any two peers. Each peer has multiple parents. Those forward a fraction of the whole video stream, so-called a stripe, to their children. Peers are organized into inner-node disjoint spanning trees, each delivers the chunks in one stripe. The resilience of those systems was proven theoretically, but they have not been adopted in a wide real-world deployment.

To summarize the discussion, building a resilient P2P streaming system is an open question. One promising approach to achieve system's resilience to both random failure and targeted attacks is to: (*i*) leverage the resilience properties of pull-based systems with the mesh topology and (*ii*) reduce the direct dependency between peers.

3 Striping Scheme

The tiering effect in pull-based systems allows an outsider to gain information on the structure of the whole network and to infer the flow of video chunks between tiers. Attacks by shutting down peers on a certain tier can disrupt the flow of the video distribution. Furthermore, the damage can be severe when an attacker targets head nodes, which are the peers adjacent to the source in the overlay topology.

There are two potential approaches to mitigate the damage caused by attacking head nodes: (*i*) To decrease the direct dependency between peers by increasing the connectivity among them, which consequently increases the number of head nodes; and (*ii*) To remove the tiering effect completely. We reserve the second approach for future work and instead, in this paper, focus on the first approach which allows us to answer a more urgent question: *Assuming that the structure is revealed, what can we do to mitigate the damaging effect when head nodes are attacked?* We also assume that recovery measures, such as rejoining upon isolation or disconnection are always available.

Increasing connectivity among the source and peers is challenging due to resource constraints and inherent behavior of the pull-based protocol. Without increasing the bandwidth of the source and peers, the straightforward method that increases their number of partners does not work. Head nodes might gradually prefer to download chunks directly from the source due to its high availability of chunks. The higher the number of head nodes, the more likely that they have to compete with each other for a fixed source bandwidth. This leads to increasing delay and probably chunk miss since the source cannot response timely to all chunk requests.

In this section, we present our striping scheme for pull-based P2P streaming systems that reduces the direct dependency between peers. This scheme mitigates the negative effects of attacking on head nodes, which we demonstrate in Section 4. We begin with the idea of the scheme first. After that, we describe the design and the specification of the scheme.

3.1 Idea to Enforce Diversification by Out Striping Scheme

Current pull-based protocols do not diversify chunk requests exhaustively, i.e. peers can steadily download chunks from a few among several partners as long as they respond reliably. This leads to an implicit yet direct dependency between peers. To reduce this dependency, each peer needs to download video chunks from diverse partners. This implies that it needs to send chunk requests to more diverse partners. Towards this end, each peer enforces itself to request subsets of the required chunks from different groups of partners.

At this point there are three methods to diversify the requests. (*i*) *In the first method, each peer alternates between different groups of partners to request chunks at different scheduling cycles.* Over the long run, the average number of requested chunks per peer is reduced. However, certain peers might, by chance, receive many chunk requests in a short period. Consequently, local overloading might happen, which affects the overall chunk dissemination. (*ii*) *The second method is to split the needed set of chunks by their play-out deadlines, from most to least urgent.* On-time delivery of the most urgent chunks is more critical since there might not be enough time to request them again in the next scheduling cycles. When the peer requests the most urgent chunks from a subset of partners that is not reliable, the urgent chunks might not be delivered on time. This leads to more missed chunks. (*iii*) *The third method is to divide the video stream in an interleaved manner into stripes.* This way, diversification is achieved while avoiding the drawbacks of the above two methods.

Subsequently to dividing chunks into stripes, each peer needs to *locally* separate its partners into different groups. There are two methods: *(i) In the first method, the grouping is based on partners' identities, e.g., the IP addresses.*

This partner grouping is inflexible because it depends highly on the fluctuation of the partner list. A group might not have partners with certain identities among the peer's available partners. *(ii) In the second method, each peer assigns its partners into different logical groups, regardless of partners' identities.* Fluctuation of partners in each group can be quickly compensated by adjusting partners among the groups or even by reassigning.

We summarize our idea to enforce diversification as follows:

1. *The video stream is divided into stripes.*
2. *Each peer logically forms separate groups of partners.*
3. *Each peer enforces itself to request a certain stripe from a certain group of partners*

By doing that, the chunk downloading demand to a peer from its partners can be efficiently reduced. In conventional pull-based systems, a peer with m partners can, in principal, receive chunk requests of m times the streaming rate in the worst case. However, the demand for each peer with striping is reduced by a factor of k, the number of stripes, when the number of partners is fixed.

Consequently, the diversification enforcement allows a peer to have more partners, given the same upload bandwidth. Thus, the peer has more source-to-peer paths and at the same time avoids overloading itself. More importantly, the source can significantly increase its number of partners, or the number of head nodes, with the same upload bandwidth. The critical connectivity between the source and the peers is therefore enhanced, thus, potentially strengthens the resilience of the system against both failures and attacks.

In the coming section, we elaborate the idea of diversification enforcement into the design of the striping scheme.

3.2 Design of the Striping Scheme

Following the high-level sketch discussed in Section 3.1, this section details the design of the striping scheme. This includes the division of the video stream into stripes and the assignment of partners to different groups.

First, a stripe i consists of chunks whose sequence numbers equal to $i \bmod k$. Second, partners of a peer are assigned to k groups, each contains a subset of the partner list. This way, a peer requests chunks of the stripe i from partners of the group i. Figure 1 illustrates the design of our scheme for a generic peer. In this example, the video stream is divided into three stripes. Accordingly, seven partners are assigned to three groups.

The assignment of partners to groups has to satisfy several constraints. (i) First, every group has at least one partner to ensure the existence of chunk providers for the respective stripe. (ii) Second, all partners should be assigned to groups since partners that are not assigned to any group are not considered in requesting chunks. (iii) Third, the difference in the number of partners of any two groups should be minimized. Otherwise, chunks in the stripe whose respective group has very few partners have a lower chance to be requested and

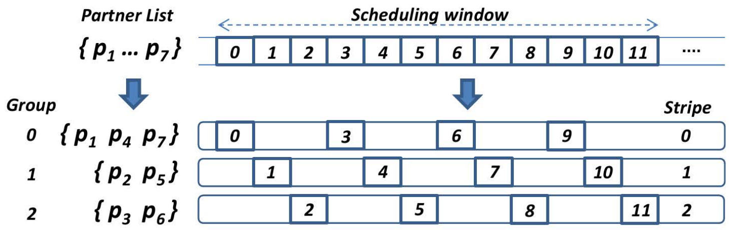

Fig. 1. An example of grouping and striping: partners in each group receives requests for chunks of the respective stripe

delivered successfully. (*iv*) Lastly, the assignment should minimize the difference between the number of groups assigned to each partner. Assigning a partner to several groups increases its chance to be requested more frequently, which might overloads it. We formulate the above assignment problem as follows.

Given the set of partners $P = \{p_1, ..., p_m\}$ and the set of groups $G = \{g_1, .., g_k\}$, and let $a_{ij} \in \{0, 1\}$ ($1 \leq i \leq k$ and $1 \leq j \leq m$) denotes the assignment of partner p_j to group g_i, where $a_{ij} = 1$ if group g_i has partner p_j and $a_{ij} = 0$ otherwise. Define $N_i^P = \sum_{j=1}^{m} a_{ij}$ as the total number of partners assigned to group g_i, and $N_j^G = \sum_{i=1}^{k} a_{ij}$ as the total number of groups to which partner p_j is assigned.

Consequently, the problem of assigning partners to groups is to find a_{ij} such that:

$$z = minimize \left\{ \sum_{i=1}^{k} \sum_{j=1}^{m} a_{ij} \right\} \tag{1}$$

$$s.t. \quad \sum_{j=1}^{m} a_{ij} \geq 1, \quad i = 1..k \tag{2}$$

$$\sum_{i=1}^{k} a_{ij} \geq 1, \quad j = 1..m \tag{3}$$

$$\underset{i}{\text{argmax}}\{\sum_{j=1}^{m} a_{ij}\} - \underset{i}{\text{argmin}}\{\sum_{j=1}^{m} a_{ij}\} \leq 1 \tag{4}$$

$$\underset{j}{\text{argmax}}\{\sum_{i=1}^{k} a_{ij}\} - \underset{j}{\text{argmin}}\{\sum_{i=1}^{k} a_{ij}\} \leq 1 \tag{5}$$

In this formulation, the objective function in (1) is to minimize the total number of assignments of partners to groups. The constraints in (2) & (3) ensure that every partner is assigned to groups and every group has partners. The constraints in (4) & (5) are used to prevent groups from having too many partners and assigning a partner to many groups.

With the *Round-Robin assignment* of partners to groups, we achieve the optimal solution [14] with $min\{\sum_{i=1}^{k}\sum_{j=1}^{m} a_{ij}\} = max(k, m)$.

Determining Parameters. Following the above design, in this section, we discuss the constraints for the two system parameters introduced in the design of the striping scheme.

The first parameter is the number of stripes k. Its value is constrained by the number of requested chunks C in each scheduling cycle which is asymptotically proportional to the streaming rate (in chunks per second) and the scheduling interval. When $k > C$, over-striping happens, i.e. one or more stripes contain no chunks. Consequently, there are groups of partners that are redundant for chunk request. The striping in this case is not efficient. At the other extreme, when $k = 1$, all partners are in the same group. The striping scheme operates similarly to a conventional pull-based system. The second parameter is the number of partners m. When m is too small, diversification is eventually limited because there are a few options for each chunks request. When m is too large, the necessary communication overhead can overload the system.

In the coming section, we refine the design of the striping scheme with specifications to integrate it into conventional pull-based systems.

3.3 Specification

Integrating the striping scheme into current pull-based protocols requires a few modification. At the source, the number of partners scales up with the number of stripes k. At each peer, the chunk scheduling operation does not search exhaustively available chunks from all partners. Instead, for a chunk with the sequence number s, the peer only considers partners in the group $s \ mod \ k$. Additionally, the assignment of partners to groups needs to be adapted when there are updates on the partner list. The adjustment should be fast to react promptly to the dynamics of peers to minimize the computational cost. The simple Round-Robin assignment introduces little computational cost. Even with a naive implementation when partners are re-assigned upon each update of the partner list, the cost would be negligible.

In the next section, we integrate the striping scheme with Round-Robin assignment into DONet – a conventional pull-based protocol and evaluate its performance.

4 Evaluation

In this section, the proposed striping scheme for pull-based systems is evaluated with respect to its provided resilience against attacks on head nodes. In addition, the efficiency of the proposed scheme is evaluated in detail. The evaluation aims at answering the following three questions:

1. *What damage does attacking head nodes cause to the performance of the conventional pull-based systems?*

2. *What resilience against the head node attacks is provided by our striping scheme?*
3. *What trade-offs in terms of signaling overhead does the striping scheme introduce?*

To answer the above questions, we need to identify metrics to quantify the performance of the streaming system and an accurate simulation model with realistic settings. Their in-depth discussions are presented next.

4.1 Metrics

In live streaming, timely delivery of video chunks is critical to ensure that the video stream can be played out smoothly. Therefore, each video chunk has its own play-out deadline for decoding. When a chunk arrives after its deadline it is considered missing. A missed chunk causes the video player to either stall or skip the chunk. In the former case, the smooth video play-out is not achieved. In the later case, the visual display of the video is impaired. Both cases reduce the perceived quality of the decoded video.

There are several methods to estimate the quality of a streaming system. Among them, the amount of missed chunks is one useful indicator because it tells how properly the system is working. It is also convenient since the calculation is straight-forward. The disadvantage of this method is that it hardly reflects the quality of experience (QoE) from users' perspective. To better quantify system's performance as perceived by users, studies in the literature [7] use QoE metrics, such as the Peak Signal-to-Noise Ratio (PSNR) which compares the decoded video at end users with the original one. However, this method has several drawbacks: First, calculating this metric is costly since it requires a considerable CPU power to decode the video. Second, the calculation depends on the video codec types and the benchmark video. It is therefore difficult to make general statements on the performance of a protocol.

From the above discussion, we select chunk miss to quantify system's performance for simplicity and flexibility. Specifically, we define the *chunk miss ratio* as the fraction of chunks that missed their play-out deadline divided by all chunks that should be played out. Note that the ratio is complementary to the Continuity Index that is commonly used in the literature. The chunk miss ratio is favored in this paper because it is more intuitive to quantify the damages caused by the attackers to the system's performance.

Furthermore, to comprehend the effects to the system after being attacked, we introduce three additional micro metrics to look at chunk miss ratio from three different perspectives:

– ***Average Miss Ratio*** which is the average miss ratio over a significant period of time after the attack.
– ***Maximum Miss Ratio*** which is the maximum instantaneous chunk miss ratio per second. It estimates the upper limit of damage that the attacks can cause to the system.

- ***Per-chunk Miss Ratio*** which is the fraction of peers missing a certain chunk. The metric quantifies how significantly missing a specific chunk can affect the system.

One immediate concern over the striping scheme is its signaling overhead. Due to an increased size of the partner list, additional overhead arises as (*i*) each peer sends more chunk requests to its partners and (*ii*) peers exchange more BMs with each other. Subsequently, we introduce the ***Signaling Overhead Ratio*** metric, which is the fraction of the volume coming from signaling packets divided by the total volume of both video and signaling packets.

4.2 Simulation Model

Simulation framework: To evaluate the resilience of pull-based P2P streaming against attacks, we developed OSSim, our generic simulation framework for P2P streaming, which is built on top of OMNeT++[3]. The framework allows packet-level simulations of different classes of P2P streaming systems. Its source code, including the one used in this study, is available online [4].

Representative pull-based P2P streaming system: Using the OSSim framework we developed DONet [16] – a popular deployed pull-based system in the literature. We select DONet as the benchmark system due to several reasons. (*i*) First, the design and protocol description of DONet is described in detail, which supports a verifiable implementation of the system in simulation. (*ii*) Second, its Rarest-First chunk scheduling strategy produces comparable performance to the state-of-the-art algorithms [17]. (*iii*) Third, the simulation model of DONet is also validated in our previous study [11].

Underlying networks: To emulate the characteristics of the underlying Internet, we used the GT-ITM [15] topology generator to generate a transit-stub core network consisting out of 20 core and 400 edge routers that are inter-connected by 1212 links. In particular, we use the following parameters for the topology generator: diameter 14, node degree 2.843, and path length 6.231. The latencies in the links connecting the routers are uniformly distributed in the range [1, 60] ms. Peers are randomly attached to the 400 edge routers at the beginning of each simulation.

Workload: We use a synthetic churn model from a measurement study in [12], in which the authors analyze traces from a popular live program over a period of 90 days. From this model, the distributions of inter-arrival times of users and session durations are Pareto and Lognormal respectively. Specifically, we use $a = 2.52$ and $b = 0.35$ for the Pareto distribution and $\mu = 1.44$ and $\lambda = 5.19$ for the Lognormal one. In addition, we allow leaving peers to rejoin the system after a random period, to maintain a rather stable peer population.

[3] `http://omnetpp.org`

[4] `http://www.p2p.tu-darmstadt.de/research/ossim`

Attack strategy: We assume that the strength of the attacker is represented by a budget (A). This tells the maximum number of head nodes the attackers can shut down simultaneously or in a relatively short period. We assume that the head nodes can be identified. For example, the attackers can apply a similar technique which was introduced in [6]. The implementation of such technique is, however, beyond the scope of this study. Given a list of head nodes, the attacker randomly selects nodes to shut down simultaneously. The attacker stops attacking when either all head nodes are shut down or the number of selected head nodes reaches its budget.

Parameters: In all experiments, the following parameters are used unless otherwise stated. The streaming rate is 400 kbps. Each video buffer stores up to 30 seconds of video chunks whose sizes are 2500 Bytes. A peer starts playing out video chunks when the downloaded chunks are equivalent to around six seconds. We simulate one source and 1000 peers. The upload bandwidth of the source and peers are 8 Mbps and 800 kbps respectively. Even though it is not realistic to assume that all peers have the same, it is reasonable as we are only interested in the resilience of pull-based systems against attacks. Using an homogeneous peer bandwidth eliminates the impact of the peers' characteristics in the results. The simulation duration is 1200 seconds. In the first 500 seconds, no data is collected to avoid unstable system behavior. We repeat each simulation setting at least 35 times.

4.3 Results

In the following, we summarize our main simulation findings and describe the effects of attacking head nodes first. Afterwards, we present a comparison of the striping scheme to a conventional protocol – DONet. Finally, we investigate the tradeoff of the striping scheme in terms of signaling overhead.

Effects of Attacking Head Nodes: In the following, we answer the question: *What damage does attacking head nodes cause to the performance of the conventional pull-based systems?*

In this experiment, the maximum number of partners of the peers and the source was eight and ten respectively. To perform the attack, all of the ten head nodes which are the partners of the source were shut down simultaneously at the 900th second, after the system has reached its steady-state. The instantaneous chunk miss ratio per second were plotted in dependence on time.

Figure 2 presents chunk miss ratio for DONet under the attack. For clarity, the figure includes only the relevant period after the attacks. It can be seen that, during the first 20 seconds, the chunk miss ratio remains as low as 1% . In the next 20 seconds, the miss ratio increases dramatically to reach its maximum of almost 35%. It then reduces quickly and remain low since the 960th second.

Intuitively, the results have agreed with expectation on the behavior of pull-based protocols and can be explained as follows: Head nodes serve as intermediate sources of video chunks for the rest of the peers. When they are shut down,

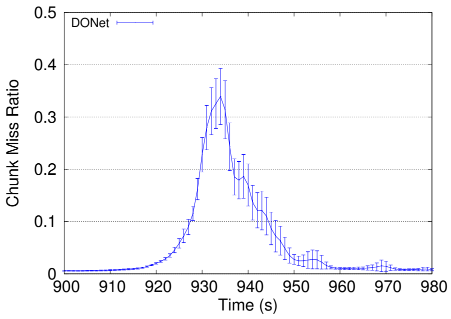

Fig. 2. Instantaneous chunk miss ratio of DONet versus time, after attacking all ten head nodes

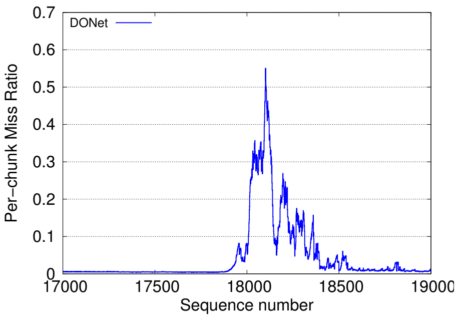

Fig. 3. Chunk miss ratio of DONet versus sequence number of chunks, after attacking all ten head nodes

their partners cannot request chunks from them. However, chunk missing does not occur immediately after the attack, because the peers have large buffers. Chunk missing might start from the peers connecting to the head nodes previously. It then spreads to other peers that locate further away from the source. Chunk missing reduces when the connections between the source and the peers are established again.

To estimate more precisely the impact of missed chunks to the system's performance we recorded the sequence number of all missed chunks and calculate the chunk miss ratio per chunk's sequence number. The results are plotted in Figure 3. The figure shows that the chunk miss ratio increases sharply from around 1% to a maximum of 55% in accordance with an increase in sequence number from around 18000 to 18100. It diminishes steadily for subsequent chunks and remains low for sequence numbers greater than 18500.

The spread of missed chunks to a significant fraction of peers as shown in Figure 3 indicates the strong and negative impact to the system's performance. A certain chunk can be more important than others, depending on whether it carries an intra-coded (I), a predictive-coded (P) or a bidirectionally predictive-coded (B) frame. A successfully decoded I-frame is required to decode the P-frames and B-frames in the same Group of Picture (GoP). Losing an I frame, therefore, fails the decoding of the GoP, which leads to a perceptible picture degradation at users. More severely, given the small number of head nodes a malicious party can periodically trigger attacks. Perceived quality by users in this case can be further degraded.

Comparing the Striping Scheme with a Conventional Protocol: The second question we studied was: *What resilience against the head node attacks is provided by our striping scheme?*

In this experiment, we compare the effect of attacking head nodes on the conventional DONet ($k = 1$) versus the adapted one with striping ($k = 2, 3, 4$). The number of partners of the source and peers in the striping scheme scales with the number of stripes. The attacker budget in terms of the number of nodes that is attacked varies between 5 and 60. Maximum chunk miss ratio and average chunk miss ratio over a period of 60 seconds have been recorded and plotted.

We expect that the impact of attacks with the same budget is stronger in the conventional DONet than in the striping scheme. Since the number of head nodes in DONet is less than in the striping scheme, the same number of attacked head nodes reduces a larger portion of the number of connections for distributing video chunks from the source to the peers. Consequently, chunk miss ratio in DONet is larger than in the striping scheme.

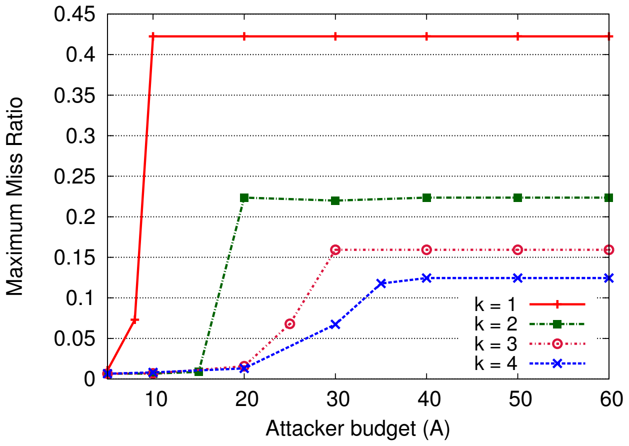

Fig. 4. Comparing the conventional DONet ($k = 1$) to the adapted one with Striping ($k = 2, 3, 4$) in terms of maximum chunk miss ratio after attacks

Figure 4 plots maximum chunk miss ratio of DONet and the striping scheme in dependence on the attacker budget. The results agree with our expectation. First, it can be observed that when more stripes are applied, the attacker needs significantly larger budget to achieve the same damage to the system. The miss ratios reach their maxima when the attacker budget equals to the number of head nodes, at least. Second, it is also shown that even a conservative diversification with two stripes can reduce the maximum chunk miss ratio by around 50%.

To comprehend better the effect of attacks on the system, we plot in Figure 5 the average chunk miss ratio over a period of 60 seconds after attacks in dependence on the attacker budget. The figure compares the conventional DONet ($k = 1$) versus the adapted one with striping ($k = 2, 3, 4$). As seen from the figure, the striping scheme reduces the maximum of the average chunk miss ratio from 30% to 50% when the number of stripes varies from two to four. Additionally, the attacker budget has to increase proportionally to the number of stripes to maximize the damage.

One unanticipated but interesting finding in Figures 4 and 5 was that the striping scheme effectively reduced the maximum damage even when all head

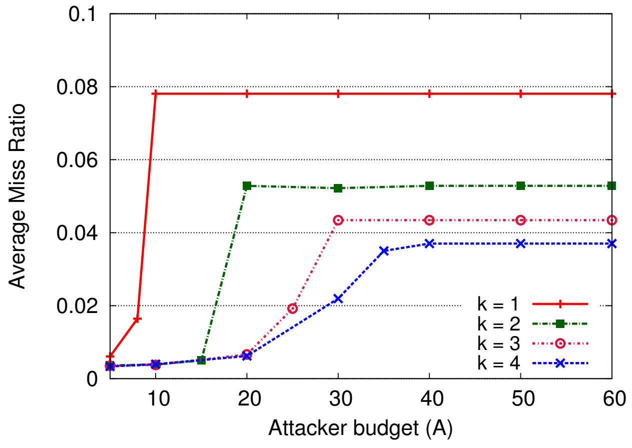

Fig. 5. Comparing the conventional DONet ($k = 1$) to the adapted one with Striping ($k = 2, 3, 4$) in terms of average chunk miss ratio over 60 seconds after attacks

nodes are attacked. Note that in all experiments we applied the same process and parameters for recovering a node from isolation. The finding can be explained by two reasons: (*i*) The striping scheme disseminates chunks better due to the diversification enforcement. When chunks are requested in small numbers from diverse partners, they can be quickly downloaded. Consequently, chunk availability in the whole network is improved. (*ii*) The increased number of partners that each peer has also allows it to connect to partners that have its required chunks with a higher probability.

Trade-off of the Striping Scheme. In this section, we studied the tradeoff of the striping scheme in terms of signaling overhead when there is no attacks. We varied the number of stripes from two to nine. We expect that the striping scheme produces more signaling overhead since each peer has more partners. The exchanged BMs are increased subsequently. Furthermore, each peer probably sends more chunk requests per scheduling cycle because the peer should diversify the requests to different partners.

Figure 6 shows the signaling overhead in dependence on the number of stripes. In this figure, the lower horizontal line represents the average signaling overhead of DONet. As expected, the overhead increases steadily with an increasing number of stripes. Specifically, for each additional stripe, the signaling overhead increases by around one percent. The total signaling overhead is less than 10% when the number of stripes equals to four, which significantly reduces damages caused by attacks. Note that a significant portion of the total signaling overhead would stem from exchanging BMs. This overhead, however, can be strongly reduced by techniques such as one described in [9].

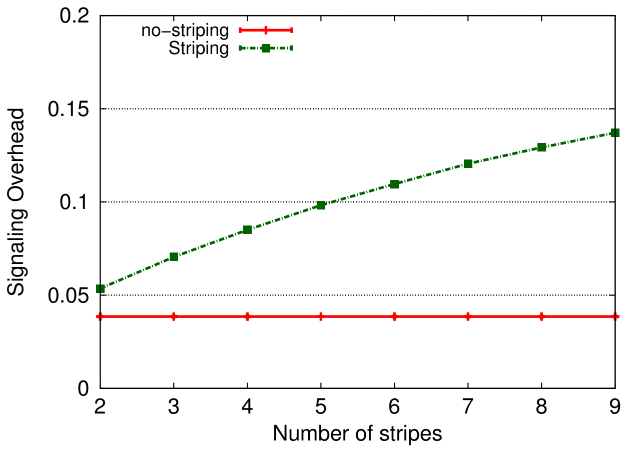

Fig. 6. Comparing the signaling overhead of the conventional DONet (no-striping) and the adapted one with striping when there is no attacks

5 Conclusion

In this paper, we investigated how attacks on head nodes affect pull-based P2P streaming systems. Results demonstrate that conventional pull-based streaming systems exhibit serious vulnerabilities to these rather simple attacks.

Subsequently, we introduced a striping scheme as a countermeasure against such attacks. The scheme enforces peers to request separate sets of chunks from diverse groups of partners. Simulation results reveal that the striping scheme effectively reduces both the maximum and average chunk miss ratios by 50% and 30% respectively. The scheme is light-weight and can be integrated easily to current pull-based systems with minimum modifications.

This study opens several interesting research questions that we would like to conduct. We plan to investigate the effect of the striping scheme on different chunk scheduling algorithms as well as the impact of varying partner assignment algorithms in our future work.

Acknowledgment. This work was supported by the German Academic Exchange Service (DAAD), Grant number A/09/97565.

References

1. Banerjee, S., Lee, S., Bhattacharjee, B., Srinivasan, A.: Resilient multicast using overlays. IEEE/ACM Trans. Netw. 14, 237–248 (2006)
2. Birrer, S., Lu, D., Bustamante, F.E., Qiao, Y., Dinda, P.A.: Fatnemo: Building a resilient multi-source multicast fat-tree. In: Chi, C.-H., van Steen, M., Wills, C. (eds.) WCW 2004. LNCS, vol. 3293, pp. 182–196. Springer, Heidelberg (2004)

3. Brinkmeier, M., Schafer, G., Strufe, T.: Optimally dos resistant p2p topologies for live multimedia streaming. IEEE Transactions on Parallel and Distributed Systems 20(6), 831–844 (2009)
4. Fischer, M., Grau, S., Nguyen, G., Schaefer, G.: Resilient and underlay-aware P2P live-streaming. Computer Networks 59, 122–136 (2014)
5. Hei, X., Liu, Y., Ross, K.: Iptv over p2p streaming networks: the mesh-pull approach. IEEE Communications Magazine 46(2), 86–92 (2008)
6. Hei, X., Liu, Y., Ross, K.: Inferring network-wide quality in p2p live streaming systems. IEEE JSAC 25(9), 1640–1654 (2007)
7. Kiraly, C., Abeni, L., Lo Cigno, R.: Effects of p2p streaming on video quality. In: IEEE ICC, pp. 1–5 (May 2010)
8. Li, B., Wang, Z., Liu, J., Zhu, W.: Two decades of internet video streaming: A retrospective view. ACM Trans. Multimedia Comput. Commun. Appl. 9(1s), 33:1–33:20 (2013)
9. Li, C., Chen, C., Chiu, D.: Buffer map message compression based on relevant window in p2p streaming media system. CoRR abs/1108.6293 (2011)
10. Liang, J., Nahrstedt, K.: Dagstream: locality aware and failure resilient peer-to-peer streaming. vol. 6071, p. 60710+. SPIE (2006)
11. Nguyen, G., Fischer, M., Strufe, T.: Ossim: A generic simulation framework for overlay streaming. In: Summer Computer Simulation Conference (2013)
12. Veloso, E., Almeida, V., Meira, W., Bestavros, A., Jin, S.: A hierarchical characterization of a live streaming media workload. In: Proceedings of the 2nd ACM SIGCOMM / IMW 2002, New York, NY, USA, pp. 117–130 (2002)
13. Wang, F., Liu, J., Xiong, Y.: Stable peers: Existence, importance, and application in peer-to-peer live video streaming. In: IEEE INFOCOM, pp. 1364–1372 (2008)
14. Wilkinson, B., Allen, M.: Parallel Programming: Techniques and Applications Using Networked Workstations and Parallel Computers. 2nd edn. Prentice-Hall, Upper Saddle River (2005)
15. Zegura, E.: Gt-itm: Georgia tech internetwork topology models (1996), `http://www.cc.gatech.edu/fac/Ellen.Zegura/graphs.html`
16. Zhang, X., Liu, J., Li, B., Yum, Y.S.: Coolstreaming/donet: A data-driven overlay network for peer-to-peer live media streaming. In: IEEE INFOCOM, vol. 3, pp. 2102–2111 (2005)
17. Zhou, Y., Chiu, D.M., Lui, J.C.S.: A simple model for chunk-scheduling strategies in p2p streaming. IEEE/ACM Trans. Netw. 19, 42–54 (2011)