Multimedia Tools and Applications (2021) 80:20255–20282
https://doi.org/10.1007/s11042-021-10604-w

Construction and maintenance of P2P overlays for live streaming

Eliseu C. Miguel[1] · Cristiano M. Silva[2] · Fernando C. Coelho[3] · Ítalo F. S. Cunha[3] · Sérgio V. A. Campos[3]

Received: 25 March 2020 / Revised: 16 October 2020 / Accepted: 25 January 2021 /
Published online: 5 March 2021

Abstract
P2P live streaming requires low latency and low media discontinuity to provide users good quality of experience. When peers are connected to a large number of partners, the communication overhead increases and sophisticated overlay maintenance strategies are required to maintain undisrupted media distribution. In order to deal with these challenges, we present the Peer Classification for Partnership Constraints technique for building and managing P2P overlays for live streaming. The proposed algorithm defines classes of peers based on their contribution to video chunk distribution. Classes are used to constrain partnerships among peers. The number of potential partners in each class is constrained, avoiding competition for partnerships between high-cooperation and low-cooperation peers. Since each peer has a given number of slots dedicated to high-cooperation and low-cooperation peers, we guarantee that the network keeps operating, even when incorporating a considerable share of free riders. The strategy is simple, significantly reducing system complexity. Moreover, it can also be used in conjunction with other strategies devised in the literature for greater gains in efficiency. Experiments show that our Peer Classification for Partnership Constraints technique allows a streaming system to handle 50% of free riders under flash crowd events with low latency and discontinuity.

Keywords P2P network · Overlay management · Live streaming · Flash crowd · Free rider

1 Introduction

Content distribution plays a major role in the Internet. For example, Netflix reported 139 million subscribers globally in 2018,[1] and Cisco estimates Internet video traffic will amount to 80% of all Internet traffic by 2021 [6]. Although large content providers have achieved

[1] https://edition.cnn.com/2019/01/17/media/netflix-earnings-q4/index.html

✉ Eliseu C. Miguel
eliseu.miguel@unifal-mg.edu.br

Extended author information available on the last page of the article.

success with the classic client-server model, this requires extensive infrastructure worldwide [1, 14, 35, 47].

Peer-to-peer (P2P) distribution facilitates streaming of media to large client populations without relying on content distribution networks. In P2P systems, clients establish partnerships and form an overlay over the physical network to exchange pieces of video content (*chunks*), reducing load on servers and increasing scalability [15, 16, 40]. P2P systems typically support thousands of simultaneous users in hundreds of channels for video distribution. Academic proposals include CoolStreaming [17], AnySee [18], SplitStream [3], and ChunkySpread [42]; and operational systems include UUSee Inc.,[2] Sopcast,[3] PPLive,[4] and TVU Networks.[5]

However, the distribution of content in P2P networks faces several challenges, which are amplified when considering live streaming, an application that demands very low latency and sophisticated strategies for managing the overlay. Existing strategies include a) incentive mechanisms to foster peer cooperation [11, 30]; b) techniques to optimize the P2P overlay considering the underlying physical network [25, 31, 46]; and c) specific purpose chunk exchange schedulers [10, 21, 22]; among others. Typically, these strategies increase system complexity, and may also require the exchange of additional control messages, incurring communication overhead [21, 40].

Additional problems can affect P2P live streaming systems, such as the sudden arrival of a large number of peers, known as flash crowd, or free riding. For instance, Liu et al. [19] propose a technique for dealing with flash crowds, by limiting the number of peers that can join simultaneously, creating batches in which peers join the channel at discrete time intervals. However, the presence of free riders and the overlay construction strategy are not considered. To increase overlay resources during a flash crowd, Wu et al. [44] computes the number of connections between new peers as a function of their upload bandwidth, but authors do not consider the overlay construction strategy. Several previous works focus on peer contribution incentives to mitigate free riding. However, these incentive mechanisms for peer contribution may punish peers unable to cooperate (e.g., mobile users on metered connections) even if the overlay has enough capacity to deliver media to them, pushing users our of the overlay unnecessarily.

In this work, we identify an important cause of reduced P2P live streaming overlay distribution efficiency: uncontrolled competition for available chunk upload bandwidth between high-cooperation and low-cooperation peers. Our goal is to develop simple strategies for constructing and maintaining an overlay for P2P live streaming that supports free rider peers while preventing disruption of media distribution.

We define classes of peers based on their cooperation to video distribution. Each peer in the overlay is configured to limit the number of partners from different classes. As more cooperative peers join the overlay, more free rider peers are allowed to join the overlay and receive the stream. However, if a large number of free rider peers try to join the overlay at the same time, our techniques ensure undisrupted transmission by limiting the number of free riders joining the overlay.

Our first proposed technique, Full Partnership Constraints (FPC), restricts partnerships among peers in the overlay according to their classes. Full Partnership Constraints bring

[2] http://www.uusee.com/

[3] http://www.sopcast.com

[4] http://www.pplive.com

[5] http://www.tvunetworks.com/

high-cooperation peers closer to the streaming source, while low-cooperation peers are pushed to the edge of the overlay. Complementary, Peer Classification for Partnership Constraints (2PC) combines the idea in FPC in a strategy to build and manage P2P overlays without prior information about peers. The algorithm groups peers into classes by the cooperation of each individual peer, then applies different partnership constraints on each class. Peer Classification for Partnership Constraints is a centralized algorithm that runs on the streaming system's server, and manages all peers and peer classes in the overlay.

We evaluate the proposed strategies in several scenarios. To quantify the efficiency of the strategies, we perform real experiments on PlanetLab where we systematically reduce the peer upload bandwidth in order to quantify the overlay's distribution efficiency under stressful conditions. We combine peers with limited upload bandwidth to a large set of free riders in order to create very challenging scenarios. We also emulate flash crowd events to study the robustness of the overlay to extreme peer churn. By using Peer Classification for Partnership Constraints, the P2P network becomes able to handle cooperative peers and free riders during flash crowds with reduced impact on media distribution.

The proposed techniques make minimal assumptions on a streaming system's operation, can be combined with several other existing techniques, with their advantages combined for possibly even more performant media distribution. We compare our proposed techniques with a baseline overlay construction strategy to focus on their gains and their impact if integrated with other solutions. Our experiments show that the Peer Classification for Partnership Constraints outperforms the baseline overlay construction strategy across all evaluated scenarios. At the extreme, Peer Classification for Partnership Constraints allows the P2P system to handle 50% of free riders under flash crowd events, with latency and discontinuity similar to those of the baseline strategy with only 30% of free riders and stable peer population.

This work is organized as follows. Section 2 presents related work. Section 3 describes the P2P live streaming prototype developed for running the experiments. Section 4 discusses the experimentation environment used in this work. Section 5 presents the partnerships constraint techniques developed in this work. Finally, Section 6 concludes the work.

2 Related work

P2P is a well-studied subject counting on extensive literature. In this section, we present a selection of related work in partnership management, overlay construction, addressing free riders, and flash crowd management in order to contextualize this work.

Partnership management several works are dedicated on building and evaluating algorithms and mechanisms for efficient P2P live streaming. Contributions cover techniques to optimize P2P overlay topologies [9, 10, 29, 38], scheduling chunk requests and transmissions [10, 48], and adapting topologies according to the overlay and network conditions [43]. Neighborhood filtering strategies for overlay construction are important to achieve media distribution efficiency, and the literature presents several efforts towards this direction [24, 33, 37, 45].

Some neighborhood filtering strategies limit the number of partners that peers in the P2P system can have, such as in [28], while other approaches dynamically set the number of partners for each peer considering the peer's contribution [21]. The first strategy is easier to deploy, but the second improves chunk dissemination capacity in the system by making better use of available bandwidth at peers. To combine the ease of deployment with the improved capacity of these strategies, in this work we establish a limit number of partners but dynamically establish partnerships depending on a peer's contributions. Supported by these strategies, peers with low upload bandwidth can manage a large number of partners without compromising the video transmission.

Traverso et al. [40] compare several neighborhood filtering strategies ranging from randomized approaches to very sophisticated strategies considering peer bandwidth and peer physical location. However, in most of these work, peers do not constrain the establishment of *output partnerships* (i.e., the set of partners to which a peer redistributes media). In realistic scenarios where the upload bandwidth of peers is restricted, such strategies require peers to answer requests and forward content to a subset of out-partners. Differently from such approaches, here we propose an overlay construction algorithm based on random neighborhood filtering allowing peers to forward content to all partners.

Overlay construction and free riders many overlay construction strategies propose pushing free riders to the edge of the overlay to improve efficiency of media distribution. Ullah et al. [41] propose an autonomous management framework that enables peers to learn the user behavior and self-organize through a stabilization process that continuously pushes unstable peers towards the leaves of the distribution tree. Incentive mechanisms that aim to foster the contribution of peers try to identify and punish selfish behavior, while improving the QoS for very cooperative peers [7, 11, 13, 23, 30].

A well-known incentive mechanism used in P2P file sharing is BitTorrent's tit-for-tat [7]. Tit-for-tat forces balanced pairwise data exchanges, which are too restrictive to enable live streaming [30]. Some works have attempted to use tit-for-tat as a feature, where uncooperative peers experience degraded QoS, even at over-provisioned scenarios [23]. As examples of incentive mechanisms for live streaming, Contracts [30] identifies uncooperative peers auditing cryptographic receipts of chunk transfers, while LiFTinG [13] identifies uncooperative and malicious peers based on their partnerships. These strategies are complementary to our work and can be used in conjunction with our proposals to improve live streaming delivery.

Shahriar et al. [36] investigated the effect of the presence of free riders on the performance of the P2P live streaming and developed a discrete-time stochastic model that can inform new incentive mechanisms. However, incentive mechanisms may punish peers unable to cooperate even if the overlay has spare capacity to deliver media to them.

Oliveira et al. [26] observe two kinds of free riders. *Conscious* free riders inform their partners that they are unwilling or unable to upload data. Conscious free riders have small impact over P2P overlay distribution efficiency as no peer ever wastes time and bandwidth sending requests to them. On the other hand, *oblivious* free riders do not report that they are unwilling or unable to upload data, which results in unanswered video chunk requests and degradation of chunk distribution efficiency.

Unlike other overlay maintenance algorithms that identify and exclude free rider peers from the overlay, we instead propose a P2P overlay maintenance algorithm that accepts (conscious) free riders if the overlay has spare capacity and that provides a simple incentive mechanism to foster peer contribution.

Flash crowds according to Chen et al. [4], when the flash crowd occurs, the sudden arrival of numerous peers may starve the upload capacity of the system, reduce the QoS, and even cause the collapse of the system. In order to mitigate the effect of flash crowds (without relying on very sophisticated flash crowd handle techniques), streaming systems such as CoolStreaming or PPLive rely on a large number of dedicated servers forming Content

Delivery Networks (CDNs) [19, 45]. However, CDNs increase the costs without guarantees that all flash crowds will be properly handled.

Budhkar et al. [2] propose an overlay management strategy that organizes peers in the overlay based on their serviceability. In this case, peers are arranged in a hybrid tree-mesh overlay at the edges of a CDN in which peers with higher upload capacity are part of an extended CDN tree. Complementary, our approaches may be combined with this work to construct a robust overlay around the CDN. In addition, our approach can identify high-contribution peers in the tree-mesh overlay to support its maintenance.

The majority of the research about the impact of flash crowds on P2P streaming systems do not focus on changing the overlay construction mechanisms [19, 20]. For instance, Liu et al. [20] propose a technique that increases overall peer join rate and distribution efficiency under flash crowds by batching joins to preserve network stability. The same authors [19] (in follow-up work) show that one important overlay constraint is the number of partnerships that peers maintain. TopT [34] focuses on building a system with a hybrid overlay to mitigate the side effects of flash crowds. In order to determine the number of peers that represent a flash crowd, Chen et al. [4] provide a comprehensive study on the performance of peer-to-peer live streaming systems under flash crowds, including analytical models relating flash crowd and time for network recovery.

Chung & Lin [5] proposed to control the joining process during flash crowd events for a tree-based peer-to-peer live streaming system. They propose newcomer peers first join a subtree that branches off from the existing master tree. This protects partnerships in the master tree and prevents performance degradation for peers that have already joined the overlay. In Section 5 we present an approach similar this work for mesh-based overlay.

In this work, we complement these studies with an evaluation on the impact of flash crowds in real scenarios, and we propose practical modifications to overlay construction mechanisms that naturally lead to robust P2P overlays that can quickly join newcomer peers during flash crowds.

3 P2P Live streaming systems

In this section we describe general characteristics of P2P live streaming systems, introducing definitions and parameters relevant to this paper. We also describe how these characteristics manifest in our research-oriented, general-purpose live streaming P2P prototype, called TVPP [27], that we use for evaluation.

P2P live streaming can be defined as the realtime transmission of a live video feed to a set of *peers* that collaborate with each other to disseminate the content. The *source* S is a special peer that encodes the video, splits the video into *chunks* (a chunk may contain multiple video *frames*), and starts the distribution.

Each peer p has a set of *partners* $\mathcal{N}(p)$ that p connects to and exchanges video chunks with. To get more control over partnerships, $\mathcal{N}(p)$ is split between two subsets of partner peers: $\mathcal{N}_{\text{in}}(p)$ containing *in-partners* (partners that provide chunks to p) and $\mathcal{N}_{\text{out}}(p)$ containing *out-partners* (partners receiving chunks from p).

The maximum number of in-partners is given by $N_{\text{in}}(p)$, while the maximum number of out-partners is denoted by $N_{\text{out}}(p)$. For the source, $\mathcal{N}_{\text{in}}(S) = \emptyset$. In order to join a live streaming channel, a peer p registers itself at a centralized *bootstrap* server B, which returns to p a subset of all peers currently active in the system as potential partners.[6] Peer p selects peers from this subset and tries to establish partnerships with them.

Successfully established partnerships determine $\mathcal{N}(p)$, and a relationship $p \in \mathcal{N}_{\text{out}}(p')$ implies $p' \in \mathcal{N}_{\text{in}}(p)$. When p detects that one of its partners $p' \in \mathcal{N}(p)$ has been silent for longer than a predefined time period, p removes p' from $\mathcal{N}(p)$. However, peer p periodically contacts the bootstrap server to obtain a new list of potential partners to replace lost partnerships. The values of $N_{\text{in}}(p)$ and $N_{\text{out}}(p)$ are determined for each peer p. In TVPP, we set $N_{\text{in}}(p) = 10$ or 20, and set $N_{\text{out}}(p)$ proportional to p's available upload bandwidth. This configuration is compatible with Liu et al.'s findings [19] that the number of partnerships (i.e., $N_{\text{in}}(p) + N_{\text{out}}(p)$) should be at least 10.

Peers randomly choose in-partners from peer lists received from the bootstrap server. However, to guarantee that a new peer n is able to join the overlay, an existing peer p employs a mechanism to accept new out-partners even when $\mathcal{N}_{\text{out}}(p)$ is full. In TVPP, a peer p disconnects a random out-partner $q \in \mathcal{N}_{\text{out}}(p)$ expected to cooperate less upload bandwidth than the new joining peer n. In particular, p disconnects a random peer q such that $\mathcal{N}_{\text{out}}(q) < \mathcal{N}_{\text{out}}(n)$. After disconnecting a peer in this way, peer p may become unable to process new out-partner requests for some time period τ to prevent overlay instability. In this work, we set $\tau = 60s$, which is large enough to prevent instability (particularly during flash crowds) while still allowing the overlay to expand.

Each peer has a local buffer to store video chunks. Periodically, each peer exchanges *buffer maps* with their out-partners to advertise available chunks. Additionally, peers keep track of chunks not yet received in order to identify the in-partners able to serve those chunks. P2P systems also employ a chunk scheduler that decide the order in which requests should be served, and which requests should be served in case of insufficient upload bandwidth.

In the scope of this work, we schedule chunk requests using the earliest deadline first scheduler extended with *Simple Unanswered Request Eliminator* (SURE) [26]. SURE makes peers prefer to send chunk requests to the in-partner with the least number of unanswered (timed-out) requests, balancing load and reducing the number of unanswered requests. A peer considers that a request has timed out if it is not answered within 500ms [26]. Finally, cooperative peers serve requests in order of arrival at a rate defined by their available bandwidth.

To support the evaluation required in this research, TVPP also monitors peer behavior. Peers send monitoring reports to the bootstrap server every 10s. The reports include, among other information: the number of chunks generated (only reported by the source S), the number of received chunks, the number of transmitted chunks, and the number of chunks that have missed their playback deadline.

4 Experimentation environment

We run experiments on PlanetLab [32] using customized TVPP client. During the experiments, we use the maximum number of available PlanetLab nodes (approximately 110 nodes, with slight variations across experiments). PlanetLab provides high bandwidth between nodes, and we constrain the available bandwidth at each peer using TVPP to emulate different network scenarios. The bootstrap server is located at our laboratory with plenty

[6]The bootstrap server is also commonly referred to as a *tracker*.

computing power and network bandwidth. The source streams video at a constant bitrate of 420kbps (approximately 35 chunks per second).

Each run lasts between 900s and 1900s, a sufficient period for building and exercising the P2P overlay, allowing observation of the streaming system's behavior and performance. Experiments start with one single peer running on each PlanetLab node (around 110 peers in the overlay). The initial overlay containing the initial set of peers stabilizes in less than 100s (Section 5). After 350s to 400s of simulation, which allows for observation of the performance of the initial overlay, we launch 10 additional peers on each PlanetLab node to emulate a flash crowd event, and observe the performance of the overlay under more challenging scenarios. We limit the number of peers on each PlanetLab node to 10 due to bandwidth restrictions on PlanetLab. We repeat each experiment five times to compute average values for the metrics and estimate variance.

We keep each peer's upload bandwidth, the number $N_{\text{in}}(p)$ of in-partners, and the number $N_{\text{out}}(p)$ of out-partners constant throughout an experiment. Across multiple experiments, we vary the number of in-partners and the upload bandwidth which, in turn, determines the number of possible out-partners. This way, each experiment covers a different deployment scenario and network properties. In particular, we study the ratio between the number of in-partners and out-partners:

$$r = \frac{\sum_p N_{\text{in}}(p)}{\sum_p N_{\text{out}}(p)}. \tag{1}$$

Values of $r < 1$ correspond to an overlay where peers have out-partners than in-partners, and may be found on aggressive configurations, where peers have a high number of out-partnerships but may lack bandwidth to answer all chunk requests. On the other hand, values of $r > 1$ may be found on conservative configurations, where peers have a low number of out-partnerships but are more likely to have enough bandwidth to answer all chunk requests. Finally, we also experiment with balanced overlays where $r = 1$.

In order to evaluate the topological features of the overlay, we distribute peers into four disjoint classes: (a) hot, (b) warm, (c) cold, and (d) free riders. Peers are classified according to the amount of upload bandwidth they have available, configured in TVPP. Peers belonging to the same class have the same upload bandwidth. A significant share of peers is defined as free riders (i.e., unwilling or unable to upload content). Free riders have no out-partners, and never advertise chunks.

We evaluate two metrics to capture the performance of media distribution: discontinuity and chunk distribution latency. *Discontinuity* is computed as the fraction of chunks not received in time for playback (in other words, the fraction of chunks that missed their playback deadline). Discontinuity leads to glitches or hiccups and significantly reduces quality experienced by users [12]. *Chunk distribution latency* denotes the time interval between the first transmission of the chunk by the source S until its reception by peers. We report the average chunk distribution latency across all peers in the network.

5 Partnership constraint techniques

In this section, we present our partnership constraint approaches and their evaluation. In Section 5.1 we present our first solution to mitigate competition between cooperative and free rider peers. When a larger number of peers ask to join the P2P system, we isolate newcomer peers from existing peers by instantiating new, smaller, and temporary *Parallel*

Overlays supported by the master overlay. This approach imposes partnerships constraints among peers, limiting newcomer peers to establishing connections to other newcomer peers in the same parallel overlay, which prevents resource competition and quality of service degradation on the master overlay. With Parallel Overlays we improve our P2P overlay to support an amount of free riders, even during flash crowds, that is not supported by the baseline approach [19]. However, Parallel Overlays are difficult to configure, and their maintenance incurs significant complexity and communication overhead, limiting their use.

Section 5.2 presents the *Partial Partnership Constraints* approach. Partial Partnership Constraints is based on the peer classes proposed earlier, but is easier to configure and supports more free riders than Parallel Overlays. However, as the name suggests, Partial Partnership Constraints does not isolate peer classes and permits some competition among cooperative and free rider classes.

Section 5.3 presents the *Full Partnership Constraints* (FPC) approach, which extends the Partial Partnership Constraints approach to completely isolate cooperative and free rider peers. We propose and test several configurations of FPC and the results show higher overlay distribution efficiency when compared with the Partial Partnership Constraints approach. FPC, however, assumes peer classes are known and static over the period while they participate in the overlay.

Finally, Section 5.4 presents *Peer Classification for Partnership Constraints* (2PC), where we present a solution to dynamically classify peers by their level of cooperation throughout the period while they participate in the overlay. 2PC allows dynamic execution of FPC in real deployments. If peer behaviour changes, for example, a peer's available bandwidth is temporarily reduced, 2PC can identify this and adjust the peer's class, which then triggers FPC to adjust the overlay. Full Partnership Constraints and Peer Classification for Partnership Constraints techniques are the major contributions of this work.

5.1 Temporary parallel overlays for constraining peer partnerships

We propose a strategy for imposing partnership constraints among peers by instantiating parallel overlays composed of newcomer peers. Our goal is to minimize the risk of collapsing the overlay due to the large amount of new peers joining the network abruptly. The proposed strategy is derived from the work presented by Chung and Lin [5]. These authors handle flash crowds in tree-based overlays by proposing that new peers should first join a subtree that branches off from the existing tree as a way to protect the partnerships already established.

Here, we present an adaptation of this idea to mesh-based P2P overlays. The main feature of the proposed strategy is requiring no modification to peers: the formation and maintenance of parallel overlays is managed entirely by the bootstrap server, and imposes minor communication overhead. In order to handle new peers, the bootstrap server selects a subset of the peers from the **master overlay** (the master overlay is the one holding those peers already connected to the overlay before the flash crowd event). Peers are selected based on their cooperation level, and each selected peer becomes a special **virtual source**. For our purposes, virtual sources receive media chunks from peers participating in the master overlay, but do not cooperate back: they cooperate with newcomer peers in the parallel overlay they serve as a source for.

The isolation of virtual sources prepares the network for receiving the burst of new peers from the flash crowd event. Arriving peers receive from the bootstrap a set of virtual sources and peers in a parallel overlay for establishing partnerships. Arriving peers create new temporary branches (isolated parallel overlays). Since the bootstrap server ensures partnerships are only established among peers participating in the same overlay, the master overlay is protected from disruption induced by joining newcomers.

Whenever the parallel overlay reaches stability (in terms of number of peers and partnerships), the bootstrap server merges the parallel overlay into the master overlay by allowing partnerships between peers from both overlays. This is accomplished by having the bootstrap server advertise peers in the master overlay to the parallel overlay, and *vice versa*. This eventually leads to both overlays merging.

The key to the merge process is that the bootstrap allows only cooperative peers from the parallel overlay to establish new partnerships with cooperative peers in the master overlay. This way, free riders keep their partnerships stable and are passive during the merge, avoiding competition for partnerships and disruption of chunk distribution. The isolation between free riders and cooperative peers from master overlay based on the parallel overlay is our first peer partnership constraint evaluate.

Partnerships established during the merging process have a low risk of compromising the master overlay because the parallel overlay can already efficiently distribute video chunks prior to starting the merge. In particular, peers in the parallel overlay have full buffers and functioning in- and out-partnerships. The formation and merging of parallel overlays is presented in Fig. 1.

The formation and merging of parallel overlays require coordination by the bootstrap server to constrain the partnerships that arriving peers can establish. This simply changes the content of peer advertisements made by the bootstrap server, but does not incur any additional communication costs. The formation of parallel overlays also requires that virtual sources terminate their partnerships in the master overlay before they can start establishing new partnerships in the parallel overlay. This requires a small modification to clients and minor communication overhead.

Despite the difficulty of configuring parallel overlays, this work contributes by proposing an adaptation of the original tree-based overlay approach to the construction of P2P mesh-based overlay.

5.1.1 Experimental setup

We present experiments designed to evaluate the efficiency of adopting parallel overlays for imposing peer partnership constraints in the P2P system. As baseline, we consider the interesting strategy presented by Liu et al. [19]. These authors propose handling flash crowds through a queue to pace the joining process of new peers. At each iteration, the strategy

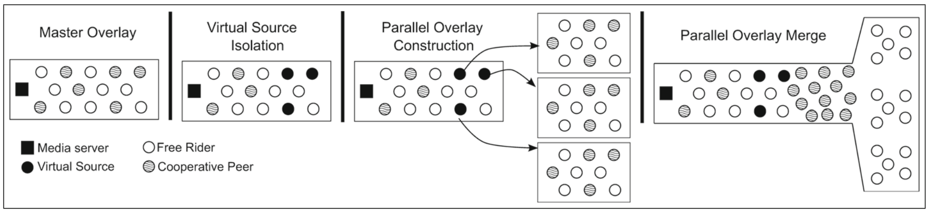

Fig. 1 Steps for building parallel overlays. In order to preserve the master overlay, new peers first join "temporary" overlays that branch off from the master overlay. A subset of well-connected peers from the master overlay are assigned as temporary servers acting as *virtual sources* of chunks. virtual sources remain linked to the master overlay and forward chunks to the temporary parallel overlays. As soon as the parallel overlays stabilize, they are merged into the master overlay without disrupting it

evaluates the overlay in order to determine the subset of newly arrived peers that will join the network. No restrictions for free riders joining the overlay are imposed.

By monitoring the overlay (and the joining process), Liu at al. define the time duration of each iteration, allowing the network to reach stability before iterating again. In order to compute the experimental results of the our baseline, we implement a simplified version of Liu et al.'s strategy in TVPP (our prototype) where iterations happen until all newcomers join the overlay. TVPP considers static time windows of 100s for all iterations, and three P2P configurations for joining the new peers: a) all at once; b) spread across four iterations; and c) spread across six iterations. Along this set of experiments, based on [19] and explained in Section 3, we set up $N_{\text{in}}(p) = 10$ and randomly allocate peers according to the four classes presented in Table 1.

5.1.2 Baseline: Joining new peers into the master overlay using queues for pacing the joining process

Figure 2a–e show the results obtained after simulating the baseline strategy proposed in [19] where 1,000 new peers go through a queue to pace the joining process. We consider that successful peers are the ones receiving, at least, 95% of all distributed video chunks prior to the chunk playback deadline. In Fig. 2a–c, the y-axis indicates the number of incoming and joined peers in non-overlapping time intervals of 10s. The successful network is the one presenting the number of joined peers (green) in a stable (flat) line close to the number of incoming peers (purple). The x-axis indicates the experiment time.

Figure 2a shows the scenario where peers join all at once. Figure 2b uses four iterations for joining all peers (25% of peers joined each iteration), while Fig. 2c indicates the scenario where new peers are joined through six iterations (≈18% of new peers are joined each iteration). We point out that the overlay collapses in the first scenario (Fig. 2a) after around 1,000s. We also notice several peers failing to receive the media in Fig. 2b–c. We observe that some experiments have very low numbers of joined peers (green dots close to the x-axis). The instability shown in these figures illustrates the negative side of flash crowds: when a very large number of peers are added to the network in a short period of time, partnerships break and the distribution of chunks becomes disrupted leading to the collapse of the overlay.

Although the baseline failed to successfully join the peers for 1, 4, and 6 iterations (Fig. 2b–c), we limited our experiments to 6 interactions so that newcomers did not take too long to join the overlay. In this case we are not looking for the number of the iterations that preserves overlay stability when facing the flash crowd. Our goal is to understand the behavior of the overlay when applying a classic technique to identify a resource-constrained and challenging scenario to use in the evaluation of our techniques.

Table 1 Distribution of peers into classes of connectivity

Peer Class	Upload (Mb/s)	Share	$N_{\text{in}}(p)$	$N_{\text{out}}(p)$
Hot peers (high upload bandwidth)	4.0	11%	10	23
Warm peers (average upload bandwidth)	2.5	22%	10	20
Cold peers (small upload bandwidth)	1.5	27%	10	09
Free riders (no upload bandwidth)	0.0	40%	10	00

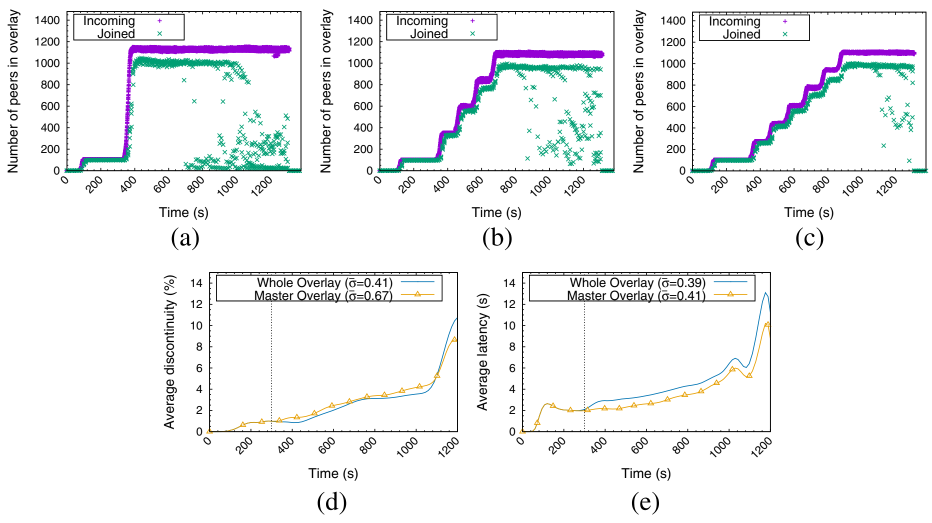

Fig. 2 Performance using queues for controlling the joining of new peers. Results obtained after simulating the strategy proposed by Liu at al. [19] where new peers first go through a queue to pace the joining process. In all figures, the x-axis indicates time (seconds). In Fig. 2a–c, the y-axis indicates the number of peers in the overlay. The purple dots indicate the number of incoming peers trying to join the overlay, while green dots show the number of peers that successfully joined. Figure 2a shows the scenario where all peers are joined at once. Figure 2b uses four iterations for joining all peers (25% of peers joined each iteration). Figure 2c uses six iterations for joining all peers (≈18% of new peers are joined each iteration). We can notice the overlay collapsing in Fig. 2a–c, when time > 1,000s. Figure 2d presents (y-axis) the average discontinuity (share of chunks not received in time for playback). The yellow line indicates the discontinuity for peers that have successfully joined the master overlay (before the flash crowd). The blue line indicates the discontinuity for all peers in the overlay (i.e., peers that joined before and during the flash crowd event). Figure 2e presents (y-axis) the average latency for chunk distribution. The latencies for the master overlay and the whole overlay are presented by the yellow line and the blue line, respectively

Now, we present discontinuity and latency for the less challenging scenario simulated for the baseline (scenario where peers join the network through six iterations) shown in Fig. 2c. The discontinuity is presented in Fig. 2d. The y-axis indicates the share of chunks not received in-time for playback, while the x-axis indicates the time (s). The yellow line indicates the discontinuity for peers in the master overlay, while the blue line indicates the discontinuity for the whole overlay (all peers in the overlay, i.e., peers that joined before and during the flash crowd event). Both yellow and blue lines are the average of all experiments shown by the *Bézier curve*. The legend also presents the standard deviation for the discontinuity.

We notice the discontinuity rises linearly before 1,000s. After that, the discontinuity increases rapidly, indicating that the network is collapsing, no longer able to deal with the large amount of free riders. Complementary, Fig. 2e presents the latency for chunk distribution (also considering the scenario where peers join the network in six iterations). The yellow and blue lines have the same meaning as previously. Finally, we notice the latency increasing very high after 1,000s.

Figure 2 shows that the overlay is stable before 1,000s. After that, the network becomes compromised and several peers are no longer able to receive the media timely. This issue is indicated by very high discontinuity and very high latency shown in Fig. 2d–e after 1,000s of operation. Interestingly, we also notice that peers that joined the network before the flash crowd are also negatively impacted (yellow line). Our techniques using parallel overlays preserve the master overlay during flash crowds, which we evaluate next.

In conclusion, the negative effects of the flash crowd identified in our baseline evaluation show that unrestricted upload bandwidth allocation among peers (Table 1) compromises the construction of the overlay with the classic technique. Thus, we will start our studies of overlay construction imposing the worst case of the baseline (peers joining at once Fig. 2a) while aiming to obtain better latency and discontinuity values than the best case of the baseline (Fig. 2c).

5.1.3 Evaluating the application of parallel overlays

Here, we repeat the same experiments, but now considering the application of parallel overlays for handling 1,000 new peers. We select six high-cooperation peers as virtual sources. Newcomer peers are uniformly distributed over the available virtual sources for constructing well-balanced parallel overlays. In this case, six parallel overlays with a proportional number of peers for each baseline iteration (Fig. 2c) will be constructed, however at the same time.

The simulation starts at time $t = 0$s. The flash crowd starts at $t = 350$s when all parallel overlays are formed. The bootstrap server waits (arbitrarily) 200s for the stabilization of parallel overlays, then it starts merging the parallel overlays into the master overlay. The first overlay starts the merge when $t = 550$s. Each one of the following overlays are merged in subsequent time windows starting with the delay of 100s (so, the second parallel overlay starts the merge when $t = 650$s, the third starts at $t = 750$s, and so on, until the sixth overlay starts the merge at $t = 1050$s).

Figure 3a shows that, for the same pattern of incoming peers (purple points) in Fig. 2a, the application of parallel overlays provide a steadily and consistently distribution of media to an increasing number of peers (green). In fact, since we are using six parallel overlays, we achieve a similar result to the one presented in Fig. 2c, but the parallel overlays prevent the network from collapsing (differently from what we see in Fig. 2c).

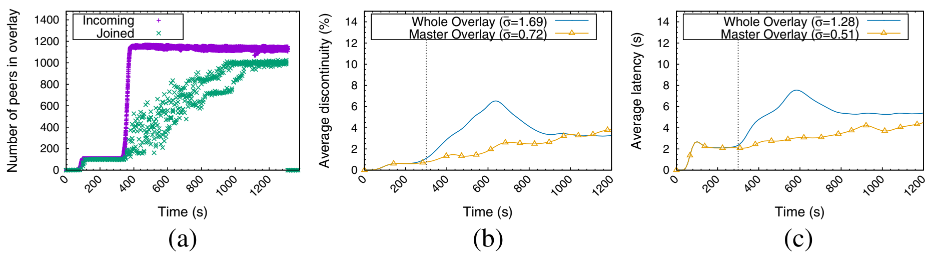

Fig. 3 Performance of parallel overlays during flash crowd. Figure 3a shows the number of peers in the overlay after receiving a flash crowd of 1,000 peers in time=350s. The x-axis indicates time, while the y-axis indicates the number of peers in the overlay. Purple dots indicate the number of incoming peers trying to join the overlay, while green dots show the number of whole peers that successfully joined the overlay. Figure 3b presents (y-axis) the average discontinuity (share of chunks not received in time for playback). The yellow line indicates the discontinuity for peers that have successfully joined the master overlay (before the flash crowd). The blue line indicates the discontinuity for all peers in the overlay (i.e., peers that joined before and during the flash crowd event). Figure 3c presents (y-axis) the average latency for chunk distribution. The latencies for the master overlay and the whole overlay are presented by the yellow line and the blue line, respectively

Figure 3b shows the discontinuity. We notice that the master overlay (yellow) is preserved, while the overall discontinuity (considering all parallel overlays shown in blue) peaks around time 650s. Complimentary, Fig. 3c shows the latency for chunk delivery, and we also notice the master overlay is preserved, while the latency in the parallel overlays peaks when t=600s. Although latency and discontinuity increase as the overlay grows, such increase is small and steady, not putting the master overlay in danger (differently from results for the baseline strategy in Fig. 2d–e).

The success of parallel overlays over the baseline can be explained because the parallel overlays preserve the master overlay by creating virtual sources. As a consequence, free riders are contained in special slots (for maintaining their old partnerships) after the merger of the parallel overlays. Therefore, we aim to propose new techniques that impose partnerships between cooperative peers and free riders without the need for virtual sources, as proposed by parallel overlays.

5.1.4 Remarks

In Section 5.1, we compared two approaches for handling flash crowds: i) the use of queues to pace the joining process; ii) the use of parallel overlays. Our results show that parallel overlays are effective at isolating the master overlay from disruption caused by flash crowds. In particular, although latency and discontinuity increase as the overlay grows, the increase is small and predictable, not putting the master overlay in danger. However, the implementation of parallel overlay incurs complexity at the bootstrap server, particularly to select virtual sources and decide when to merge parallel overlays into the master overlay. The next section presents an alternative approach that is simpler, easier to deploy, and more effective.

5.2 Partial partnership constraints

In this section we devise a simple mechanism to increase the stability and efficiency of P2P overlays. The basic idea is to reduce the competition among free riders and cooperative peers by imposing partnership constraints on peers during in the master overlay, without the parallel overlays. In other words, we group peers into classes according to their cooperation to the network, and we use such grouping for defining the kind of partnerships that will take place.

5.2.1 Experimental setup and experiments

As before, we consider four peer classes (Table 1). Partial partnership constraints among peers from different classes are shown in Fig. 4. We assign classes to peers using the shares defined in Table 1. Figure 5a shows the overlay efficiency when we restrict the partnership of peers. We observe fast joins and the number of joined peers (green) in a stable (flat) line close to the number of incoming peers (purple) on all experiment runs. The partnership constraint strategy pulls cooperative peers close to the source, and pushes free riders off to the edge of the overlay. Such behavior is desired since it ensures that peers with more upload capacity receive video chunks earlier, having more time to redistribute the chunks to other peers in the overlay.

Figure 5b–c show that the efficiency for distributing chunks is improved. We observe that the discontinuity and the latency for chunk distribution remain stable and with low values for the whole overlay, including the master one. In particular, all peers achieve performance similar to that of the master overlay when using parallel overlays (see Fig. 3a).

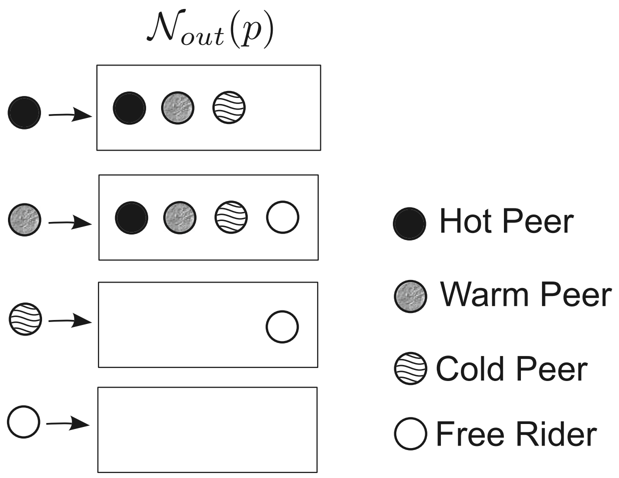

Fig. 4 Partial partnership constraints. Hot peers (high upload bandwidth) are allowed to accept out-partners from classes hot, warm, and cold. Warm peers (average upload bandwidth) are allowed to accept peers from any class as out-partners. Cold peers (low upload bandwidth) are allowed to establish out-partnerships only to free riders

We notice that the partnership constraints yield average discontinuity below 4% throughout the experiment, a value considered in [39] as a target for discontinuity.

In addition, as some partnerships between cooperative and free rider peers in the master overlay may be broken in favor of the cooperative peers from flash crowd (Section 3), we observe in Fig. 5b a slight increase in the discontinuity average (yellow) in the master overlay. We note that this discontinuity is concentrated on free rider peers (not shown).

Motivated by such gains in performance achieved using the partnership constraints, we restrict the resources of the peer population to test how the strategy would respond. We

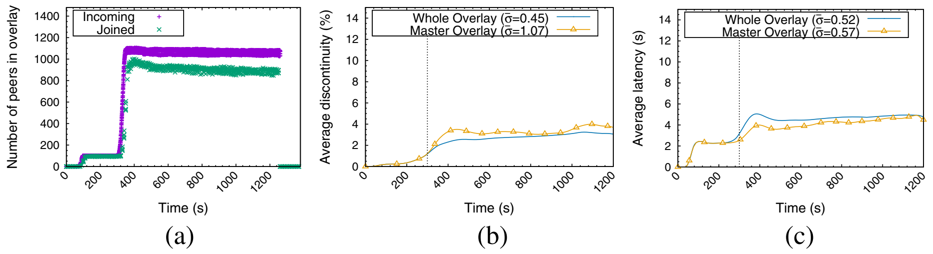

Fig. 5 Performance of partial partnership constraints for 40% of free riders. Figure 5a shows the number of peers in the overlay after receiving a flash crowd of 1,000 peers in time=350s. The x-axis indicates time, while the y-axis indicates the number of peers in the overlay. Purple dots indicate the number of incoming peers trying to join the overlay, while green dots show the number of peers that successfully joined the overlay. Figure 5b presents the average discontinuity (share of chunks not received in time for playback). The y-axis indicates the discontinuity, while the x-axis indicates the time. The yellow line indicates the discontinuity for peers that have successfully joined the master overlay. The blue line indicates the discontinuity for all peers that joined the network. Figure 5c shows the latency for chunk distribution. The y-axis indicates the average latency, while the y-axis indicates the time. Latency for peers that joined before and after the flash crowd event is presented by the yellow line and the blue line, respectively. Figure 5b-c show that partial partnership constraints preserves low discontinuity and latency

keep the same setup shown in Table 1, but we increase the share of free riders from 40% to 50% (+10%). When we increase the number of free riders, we must decrease the share of peers in the remaining classes in order to preserve the same amount of peers: class 'hot' (-2%), 'warm' (-5%), and 'cold' (-3%).

Figure 6a shows that the efficiency of the overlay remained stable throughout the experiment. Figure 6b shows the number of free riders and cooperative peer joined in the overlay. We observe that the partnership constraints guarantee higher stability among cooperative peers, which may serve as an incentive for peer cooperation. The smooth decline in the number of joined free riders in Fig. 6b impacts the discontinuity metric shown in Fig. 6c. However, the latency for chunk distribution (not shown) remains very similar to the one presented in Fig. 5c (i.e., negligible impact on latency).

5.2.2 Remarks

Partial Partnership Constraints is a simple approach that can increase the scalability of the network. As a drawback, the partnership constraints are applied only between free riders and cooperative peers. Since peers from the cold class can establish out-partnership only with free riders, overlays having a small share of peers from the cold class may lead to low use of the upload bandwidth. However, such findings lead us to the intermediary model presented along the next section.

5.3 Full Partnership Constraints

Competition between free riders and cooperative peers for the same resources in the overlay is not desirable, specially when dealing with live stream delivery. In order to avoid such competition, we extend our Partnership Constraints approach to split the list of out-partners of any given peer into two subsets:

– $\mathcal{N}_{\text{out}}^{\text{high}}(p)$ contains high bandwidth out-partners;

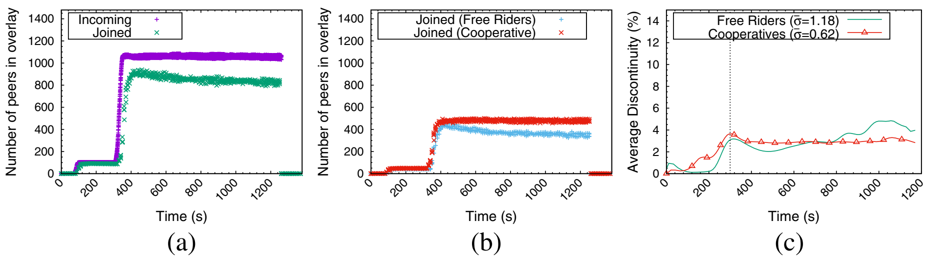

Fig. 6 Performance of the partial partnership constraints for 50% of Free Riders. Figure 6a shows the number of peers in the overlay after receiving a flash crowd of 1,000 peers at time 350s. The x-axis indicates time, while the y-axis indicates the number of peers in the overlay. Purple dots indicate the number of incoming peers trying to join the overlay, while the green dots show the number of peers that successfully joined the overlay. Figure 6b shows the number of free riders and cooperative peer that joined the overlay. partial partnership constraints guarantee higher stability among cooperative peers. The smooth decline in the number of joined free riders impacts the discontinuity metric shown in Fig. 6c. Figure 6c presents the average discontinuity over time (share of chunks not received in time for playback). The y-axis indicates the discontinuity, while the x-axis indicates the time. The red line indicates the discontinuity for cooperative peers that have successfully joined the overlay. The green line indicates the discontinuity for free riders that have successfully joined the overlay

– $\mathcal{N}_{\text{out}}^{\text{low}}(p)$ contains low bandwidth out-partners.

Splitting up the sets of out-partners allows more control over partnerships. The bootstrap server can allow free riders to join the overlay immediately, even during flash crowds, without taking the chance of compromising the stability of the overlay. This is possible because the overlay becomes able to throttle the rate at which free riders (and peers with low upload bandwidth) join the overlay in terms of the number of slots available on cooperative peers participating in the overlay.

Whenever peers with high upload bandwidth try to join the overlay, $\mathcal{N}_{\text{out}}^{\text{high}}(p)$ slots are available regardless of the number of free riders also trying the join. In the (comfortable) scenario that all $\mathcal{N}_{\text{out}}^{\text{high}}(p)$ slots are full, Full Partnership Constraints (FPC) considers that peers can disconnect one of their out-partners to make room for a new peer with higher upload bandwidth. The disconnected peer may be in the set of high-cooperation out-partners ($\mathcal{N}_{\text{out}}^{\text{high}}(p)$) or the set of low-cooperation out-partners ($\mathcal{N}_{\text{out}}^{\text{low}}(p)$), with the restriction that the disconnected peer has lower upload bandwidth than the new peer. As more cooperative peers join the overlay, they cooperate with additional slots for holding more peers, increasing capacity in the overlay and allowing the overlay to grow. Although the disconnected peer needs to find a new in-partner, we note a peer has $N_{\text{in}}(p)$ in-partners, and will have opportunities to find new ones.

Complementary, we also provide slots for low-cooperation peers inversely proportional to the available bandwidth of the peer. This groups peers with high upload bandwidth together and pulls them close to the source, while pushing free riders and peers with low upload bandwidth to the edge of the overlay.

5.3.1 Setting up the full partnership constraints strategy

FPC may be configured with any number of classes and with different sizes for $\mathcal{N}_{\text{out}}^{\text{high}}(p)$ and $\mathcal{N}_{\text{out}}^{\text{low}}(p)$ within each class. This allows FPC to be tailored to specific peer populations or application requirements. In this work, we consider the four classes of peers and set sizes in Table 2. Future work can investigate the effect of setting different class numbers for FPC.

Table 2 shows the classes of peers. The Hot class contains peers that have the highest potential for cooperating to the overlay. Hot peers only establish partnerships with peers from the hot and warm classes (defined by $N_{\text{out}}^{\text{high}}(p) > 0$ and $N_{\text{out}}^{\text{low}}(p) = 0$). The Warm

Table 2 Classes of peers and output partnerships

PEER	SET SIZES			
CLASS	$N_{\text{out}}^{\text{high}}(p)$	$N_{\text{out}}^{\text{low}}(p)$	CONT.	PEER BANDWIDTH
Hot	> 0	$= 0$	High	High upload
Warm	> 0	> 0	High	Average upload
Cold	$= 0$	> 0	Low	Low upload
Free R.	$= 0$	$= 0$	Low	Free riders

The *peer class* column indicates the level of chunk upload cooperation of a given peer p. The second column indicates the size of the set of out-partners (number of slots) for high-cooperation peers ($N_{\text{out}}^{\text{high}}(p)$) and low-cooperation peers ($N_{\text{out}}^{\text{low}}(p)$). The peer contribution column (*cont.*) indicates whether a given peer q uses out-partnership slots in $\mathcal{N}_{\text{out}}^{\text{high}}(p)$ or $\mathcal{N}_{\text{out}}^{\text{low}}(p)$ when q asks for partnership to p. Finally, *peer bandwidth* column describes the population of peers expected in each class

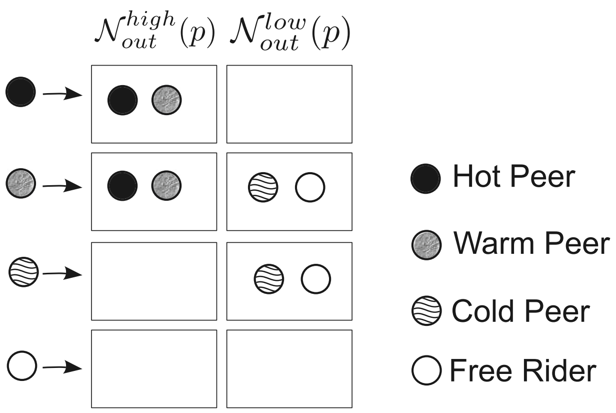

Fig. 7 Full Partnership Constraints. Hot peers (high upload bandwidth) are allowed to accept out-partners from classes hot and warm. Warm peers (average upload bandwidth) are allowed to accept peers from any class as out-partners, but hot and warm out-partners of a warm peer p are placed in $\mathcal{N}_{\text{out}}^{\text{high}}(p)$ set, while cold and free riders out-partners are placed in $\mathcal{N}_{\text{out}}^{\text{low}}(p)$ set. Cold peers (low upload bandwidth) are allowed to establish out-partnerships only to cold peers and free riders

class contains peers with average potential for cooperating to the overlay. Warm peers can establish partnerships with peers from any class. The Cold class contains peers with low potential for cooperating to the overlay. Cold peers can only establish partnerships with other cold peers and free riders (recall that free riders don't cooperate to the overlay and, thus, do not have out-partners). Figure 7 is a graphical representation of Table 2.

5.3.2 Experimental setup

The setup used along the set of experiments presented here is shown in Tables 3 and 4. We consider four classes of peers with different upload bandwidths. Free riders correspond to 50% of all peers, allowing us to emulate resource-constrained scenarios. We compare the Full Partnership Constraints strategy to two other strategies for building overlays: i) **classic-partnership sets approach**; and ii) **naive** Full Partnership Constraints.

The **classic-partnership sets approach** (classic for short) count only on a traditional single set of out-partners and doesn't provide partnership constraints to peers. On the other hand, the **naive** Full Partnership Constraints (Naive-FPC or N-FPC) employs a simplified version of the Full Partnership Constraints (FPC) strategy where out-partners are divided into the sets of high-cooperation peers and low-cooperation peers, but partnerships are not constrained (i.e., all cooperative peers can establish partnerships with peers in any other class and are classified as *warm peers* described in Table 2). The difference between classic

Table 3 Distribution of peers

Peer Class	Upload (Mbps)	Share
Hot	4.0	9%
Warm	2.5	17%
Cold	1.5	24%
Free rider	0.0	50%

Table 4 Configuration of classes in terms of the number of high-cooperation out-partners ($N_{\text{out}}^{\text{high}}(p)$), low-cooperation out-partners ($N_{\text{out}}^{\text{low}}(p)$), and the total number of out-partners ($N_{\text{out}}(p)$)

Peer	FPC		Naive-FPC		Classic
Class	$N_{\text{out}}^{\text{high}}(p)$	$N_{\text{out}}^{\text{low}}(p)$	$N_{\text{out}}^{\text{high}}(p)$	$N_{\text{out}}^{\text{low}}(p)$	$N_{\text{out}}(p)$
Hot	46	0	26	20	46
Warm	18	22	20	20	40
Cold	0	38	18	20	38
Free R.	0	0	0	0	0

approach and naive-FPC allows us to qualify the impact of splitting the set of out-partners and isolating free riders from cooperative peers versus one approach without out-partner separation. Since FPC may be combined with other techniques, (for instance with our baseline presented in Section 5.1), we do not improve our classic approach in order to understand the real impact of the splitting the set of out-partners. Moreover, the difference between naive-FPC and FPC quantifies the impact of the partnership constraints across peer classification.

5.3.3 Experiments

We consider three overlay setups, according to the relation of in-partners and out-partners:

- **balanced overlays:** overlays where the number of in-partners is approximately equal to the number of out-partners ($r = \sum_p N_{\text{in}}(p) / \sum_p N_{\text{out}}(p) \approx 1.0$);
- **conservative overlays:** number of in-partners is (approximately) 1.5 times the number of out-partners ($r = \sum_p N_{\text{in}}(p) / \sum_p N_{\text{out}}(p) \approx 1.5$);
- **aggressive overlays:** number of in-partners is (approximately) 0.5 times the number of out-partners ($r = \sum_p N_{\text{in}}(p) / \sum_p N_{\text{out}}(p) \approx 0.5$).

a) *Balanced overlay: number of in-partners approximately equal to the number of out-partners ($r \approx 1.0$)*

Here, we consider a balanced overlay where the total number of in-partners is (approximately) equal to the total number of out-partners. In this configuration, we setup $N_{\text{in}}(p) = 20$ for all peers and $N_{\text{out}}(p)$ as shown in Table 4.

Figure 8a shows the average number of incoming peers (blue) and the average number of peers that have successfully joined the overlay. The classic approach joins (approximately) 800 peers in the overlay with severe fluctuation in the number of joined peers. On the other hand, the Naive-FPC and the FPC support 950 peers in the overlay, with negligible fluctuation.

Figure 8b presents the average discontinuity, and we can see that Naive-FPC and FPC achieve efficient and stable media distribution (for this scenario), while the classic approach (green) leads to high discontinuity (just a small set of peers reproduce the media, and they do with degraded quality of experience).

Figure 8c presents the latency for the distribution of chunks, and we notice the classic approach incurring in high latency for distributing the video. In summary, the competition

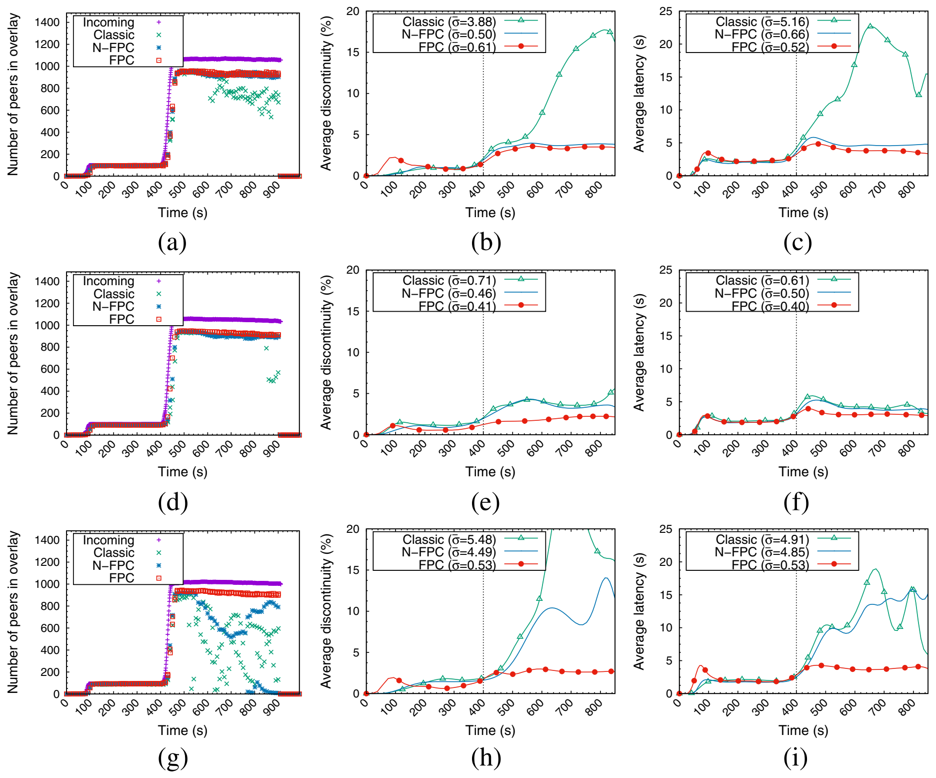

Fig. 8 Performance of the classic approach *versus* Naive-FPC *versus* FPC (Full Partnership Constraints). Figure 8a-c show the scenario for balanced overlays ($r \approx 1.0$). Figure 8d–f show the same analysis for a conservative overlay ($r \approx 1.5$). Figure 8g–i show the scenario for aggressive ($r \approx 0.5$). Figure 8a, d, and g show the number of peers in the overlay after receiving a flash crowd of 1,000 peers. The x-axis indicates time, while the y-axis indicates the number of peers in the overlay. Purple dots indicate the number of incoming peers trying to join the overlay. Figure 8b, e, and h compare the average discontinuity (share of chunks not received in time for playback). The y-axis indicates the discontinuity, while the x-axis indicates the time. Figure 8c, f, and i show the latency for chunk distribution. The y-axis indicates the average latency, while the y-axis indicates the time

between cooperative and uncooperative peers for out-partnerships disrupts the P2P overlay after the flash crowd.

b) *Conservative overlay: number of in-partners approximately equal to 1.5 times the number of out-partners ($r \approx 1.5$)*

Now, we focus on a conservative overlay where the number of in-partners is 1.5 times the number of out-partners. We consider the number of in-partners $N_{\text{in}}(p) = 30$ for all peers and $N_{\text{out}}(p)$ we keep unchanged (Table 4).

Figure 8d shows that all three strategies reach around 950 joined peers (however, the classic approach experiences higher fluctuations than the other two). Naive-FPC and FPC present more stability in the overlay because they prioritize newcomer peers with more upload bandwidth to join the overlay. Such peers have more potential for cooperating to the overlay, supporting the distribution of chunks, leading to higher aggregate upload bandwidth in the overlay.

Figure 8e shows the average discontinuity. We observe all strategies providing discontinuity below the acceptable threshold of 4%, while FPC presents the lowest discontinuity among the evaluated strategies. Figure 8f shows the average latency. FPC provides smaller latency for chunk distribution, while Naive-FPC and the classic approach show similar results. In summary, the Naive-FPC and FPC are also able to improve the live streaming delivery in conservative overlays.

c) *Aggressive overlay: number of in-partners approximately half the number of out-partners ($r \approx 0.5$)*

Now, we turn our attention to an aggressive overlay where the number of in-partners is half the number of out-partners (preserving the number of in-partners $N_{\text{in}}(p) = 20$). We consider 40% of free riders. Remaining classes are updated to hot=11%, warm=22%, and cold=27%. The configuration of partnerships is presented in Table 5.

Figure 8g presents the number of peers that have joined the overlay (y-axis) versus the time in the x-axis. FPC strategy builds a more stable overlay (red), while the classic approach and the Naive-FPC fail to incorporate several peers when time is > 500s. In other words, the FPC strategy is less affected by flash crowds.

Figure 8h plots the average discontinuity (y-axis) versus the time (x-axis). It indicates that only the FPC strategy is able to keep low discontinuity. Moreover, Fig. 8i plots the latency for chunks distribution (y-axis) versus time (x-axis), showing that FPC also keeps low latency, indicating that the strategy is able to support more free riders than the Naive-FPC and the classic approach.

5.3.4 Remarks

Full Partnership Constraints improves on our basic Partnership Constraints by enforcing stricter rules for partnership establishment. Our approach of disconnecting a peer in favor of other peers with more upload bandwidth ensures the overlay always accepts (and quickly integrates) these peers. The constraints pull high-contribution peers close to the source, and push low-contribution and free rider peers to the edge of the overlay, which improves distribution efficiently.

5.4 Implementing full partnership constraints in dynamic environments

The previous section discussed the Full Partnership Constraints strategy assuming the upload bandwidth of each peer is known and static (i.e., does not change over time).

Table 5 Configuration of classes in terms of the number of high-cooperation out-partners ($N_{\text{out}}^{\text{high}}(p)$), low-cooperation out-partners ($N_{\text{out}}^{\text{low}}(p)$), out-partners ($N_{\text{out}}(p)$) considering the number of in-partners $N_{\text{in}}(p) = 20$ and the ratio $r \approx 0.5$

Peer	FPC		Naive-FPC		Classic
Class	$N_{\text{out}}^{\text{high}}(p)$	$N_{\text{out}}^{\text{low}}(p)$	$N_{\text{out}}^{\text{high}}(p)$	$N_{\text{out}}^{\text{low}}(p)$	$N_{\text{out}}(p)$
Hot	92	0	52	40	92
Warm	36	44	40	40	80
Cold	0	76	36	40	76
Free R.	0	0	0	0	0

However, real networks require strategies for classifying peers on-the-fly, as peer upload bandwidth may change over time due to factors such as cross traffic or network congestion. Even when peer report their available upload bandwidth, the effective ability to distribute chunks may vary according to several practical issues (Internet Service Provider, number of network applications running at the peer or in the same home, among others).

We present the Peer Classification for Partnership Constraints (2PC) to classify peers dynamically (on-the-fly). After any peer joins the network, the strategy monitors the cooperation of the peer to the overlay. Peers climb up the hierarchy of classes by contributing more to the network. Periodically, peers forward a report stating their latest cooperation (in terms of chunks) to the bootstrap server. The bootstrap server periodically reclassifies peers using information in the reports. In order to provide all peers with the ability to contribute to the overlay, the proposed strategy doesn't restrict partnerships before a peer is classified.

Algorithm 1 Reclassification of peers.

```
 1: function RECLASSIFY(peers)
 2:     additions[$c$] ← 0 for each class $c$
 3:     active ← remove-new-peers(peers)
 4:     sorted ← sort-by-decreasing-cooperation(active)
 5:     for peer $p \in$ sorted do
 6:         tc ← targetClass($p$)
 7:         if tc is-hotter-than class($p$) then
 8:             c ← promote($p$)
 9:             if (c $\neq$ warm) $\wedge$ (additions[c] $\geq \Gamma$(c) $\vee$ population(c) $\geq \mathcal{U}(c)$) then
10:                 {Cannot move $p$ into $c$, revert promotion.}
11:                 demote($p$)
12:                 continue
13:             end if
14:             additions[c] ← additions[c] + 1
15:         else if tc is-colder-than class($p$) then
16:             {Same as above, but swapping demote and promote.}
17:         end if
18:     end for
19: end function
```

The reclassification of peers may involve disconnecting of some partners. In order to minimize the risk of disrupting the overlay, the strategy only reclassifies peers in to a neighboring class at each round (e.g., reclassify a hot peer as warm, or a warm peer as cold). The routine for classifying peers is shown in Algorithm 1, while Table 6 presents the algorithm's elements. The routine's input is a list of active peers in the P2P system, maintained by bootstrap server. To give the system enough time to compute the cooperation of peers with acceptable accuracy, the routine only performs the reclassification after the peer has been in the system for at least one complete reclassification period (line 3).

Periodically, the bootstrap server sorts peers by decreasing order of cooperation in terms of chunks (line 4). The bootstrap server iterates over all peers, quantifying their cooperation (line 6), and checking whether the peer needs to be reclassified (lines 7 and 15). To avoid degenerating the overlay due to peer misclassification or skewed peer populations, the strategy limits the share of peers able to be reclassified in each iteration to a threshold $\Gamma(c)$ (line 9). Finally, the population in each class c is limited by a population limit $\mathcal{U}(c)$ (line 9). It is important to have the ratio $r \geq 1.0$ in order to avoid aggressive overlays, defined in Section 4. Peer reclassification occurs periodically every μ seconds and, as a result, the routine's output is a list of new $N_{\text{out}}^{\text{high}}(p)$ and $N_{\text{out}}^{\text{low}}(p)$ configurations for each class of peers.

Table 6 Parameters and Inputs of Algorithm 1

Parameter/Input	Description
additions[c]	Number of peers reclassify to class c in the current iteration.
active	List of active peers in the P2P system.
sorted	List of active peers sorted by their chunk contribution.
$\Gamma(c)$	Share of peers able to be promoted or demoted.
population(c)	Size of peer population in the class c.
$\mathcal{U}(c)$	Limit of shared peers per class.

Table 6 helps us to better understand the elements of the Algorithm 1. The variable additions[c] handles the number of peers that can be reclassified per iteration, limited by the parameter $\Gamma(c)$ (Algorithm 1, line 9). The variable population(c) handles the number of peers in each class per iteration, limited by the parameter $\mathcal{U}(c)$ (Algorithm 1, line 9). Finally, *active* and *sorted* are auxiliary lists of peers.

5.4.1 Experimental setup

We evaluate the 2PC using the same four classes of peers, according to Table 7. In addition, we setup $\Gamma(c) = 5\%$ (i.e., 5% of all cooperative peers) for all classes, and $\mu = 120$ seconds (i.e., 2PC iteration time). Values of $\Gamma(c)$ and μ are combined to limit the number of partnership disconnection caused by peer reclassification.

Arriving peers firstly join the warm class, because it provides weak constraints for out-partners: peers in the warm class can establish out-partnerships with peers from any other class, making the network responsible for reclassifying those peers.

Figure 9a shows that the 2PC (red) keeps the overlay stable during the experiment, even in the presence of flash crowds. Moreover, Fig. 9b–c show that the average discontinuity and latency (for distribution of chunks) also remain stable, with a slight increasing shortly after the flash crowd. However, we notice a quick recovery. (Recall that results using the same setup without Full Partnership Constraints in dynamic environments are presented in Fig. 8).

We also evaluate the classification algorithm. Figure 9d shows the number of peers in different classes over time. The flash crowd starts after 400s of operation, and all peers

Table 7 Experimental Setup

Peer	Upload		Population	Set Sizes	
Class	(Mbps)	Share	Limit $\mathcal{U}$	$N_{\text{out}}^{\text{high}}(p)$	$N_{\text{out}}^{\text{low}}(p)$
Hot	4.0	9%	15%	46	0
Warm	2.5	17%	-	18	22
Cold	1.5	24%	40%	0	38
Free R.	0.0	50%	-	0	0

Number of in-partners ($N_{\text{in}}(p)$) equal to 20 and ratio $r \approx 1.0$ ($r = \sum_p N_{\text{in}}(p) / \sum_p N_{\text{out}}(p)$). In this table, the upload bandwidth distribution and peer share are similar the Table 3, and 'Set Sizes' column is similar the Table 4. Column 'Class Population' define the limit of peer in classes Hot and Cold. Warm and Free rider classes are free

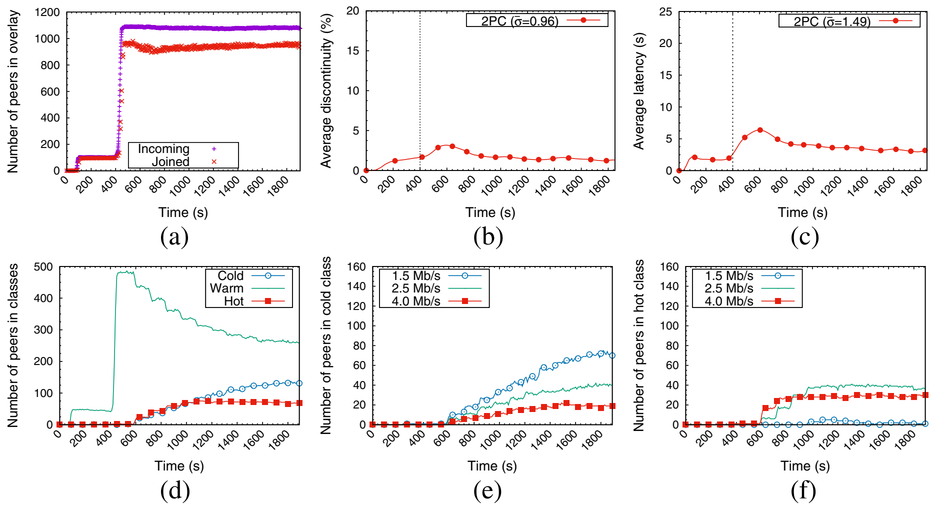

Fig. 9 Performance of 2PC (Peer Classification for Partnership Constraints). Figure 9a–c show the performance of 2PC for balanced overlays ($r \approx 1$) of configuration from Table 7 and $N_{\text{in}}(p) = 20$ for all peers. In this case, the algorithm performance can be compared with the classic approach, Naive-FPC and FPC shown in Fig. 8a–c. Figure 9a shows the number of peers in the overlay after receiving a flash crowd of 1,000 peers. The x-axis indicates time, while the y-axis indicates the number of peers in the overlay. Purple dots indicate the number of incoming peers trying to join the overlay, while red dots indicate the number of joined peers. Figure 9b shows the average discontinuity (share of chunks not received in-time for playback). The y-axis indicates the discontinuity, while the x-axis indicates the time. Figure 9c shows the latency for chunk distribution. The y-axis indicates the average latency, while the x-axis indicates the time. Figure 9d–f show the Class Inference. Figure 9d shows the peer distribution in each class throughout the experiment, limited by $\mathcal{U}$ parameter. Figure 9e shows the peer population in Cold Class per peer upload-bandwidth. Figure 9f shows the peer population in Hot Class per peer upload-bandwidth

quickly join the overlay in the warm class. The network systematically reclassifies peers to neighbor classes (hot/warm/cold) at each iteration in order to improve the topology of the overlay.

In order to audit the reclassification routine, we present Fig. 9e–f. They show the number of peers in classes cold and hot over time for different peer upload bandwidths. We observe that the reclassification algorithm concentrates peers having bandwidth equal to 1.5Mbps in class cold, and peers with bandwidth equal to 4.0Mbps in class hot. So far, the experiments have assumed the upload bandwidth with only three possibilities: 1.5Mbps; 2.5Mbps; or, 4.0Mbps. Now, we evaluate the 2PC with upload bandwidth uniformly distributed between 1.0Mbps–3.5Mbps.

Table 8 indicates the setup of out-partners for each dynamic class. We compare the 2PC to the classic approach (strategy that doesn't isolate out-partnerships nor employs partnership constraints).

Figure 10a shows that 2PC is able to incorporate a significant share of peers, while maintaining low discontinuity and latency (shown in Fig. 10b–c). On the other hand, the classic approach fails to serve several peers (and those peers receiving chunks show poor quality of experience to users). Figure 10d–f show that the proposed strategy successfully reclassifies peers into the proper class of cooperation.

Table 8 2PC Experiment Configuration on a random upload bandwidths setup

Peer	Upload	Population	SET SIZES		
			2PC		Classic
Class	(Mbps)	Limit $\mathcal{U}$	$N_{\text{out}}^{\text{high}}(p)$	$N_{\text{out}}^{\text{low}}(p)$	$N_{\text{out}}(p)$
Hot	3.0 - 3.5	15%	46	0	46
Warm	2.0 - 2.5	-	18	22	40
Cold	1.0 - 1.5	40%	0	38	38
Free R.	0.0	-	0	0	0

Different from the Table 7, in this case the upload bandwidth for all peers are select by a normal random distribution, shown in column ‘Upload(Mbps)’. The other columns are similar to the respective columns in Table 4 (for classic approach) and Table 7 (for 2PC)

5.4.2 Remarks

Peer Classification for Partnership Constraints runs at the bootstrap server and (re)classifies peers dynamically using periodic contribution reports. While 2PC and peer reclassification can induce “partnership churn” in the P2P overlay as reclassified peers may need to

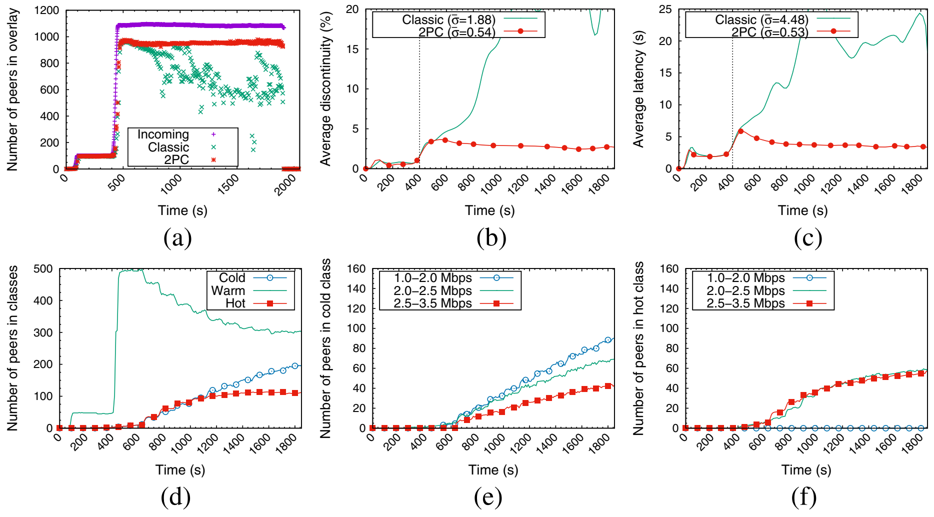

Fig. 10 Performance of the classic approach *versus* 2PC Class Inference on a random upload bandwidth distribution setup. Figure 10a–c show the performance of the classic approach *versus* 2PC for balanced overlays ($r \approx 1.0$) of configurations from Table 8 and $N_{\text{in}}(p) = 20$ for all peers. Figure 10a shows the number of peers in the overlay after receiving a flash crowd of 1,000 peers. The x-axis indicates time, while the y-axis indicates the number of peers in the overlay. Purple dots indicate the number of incoming peers trying to join the overlay, while red and green dots indicate the number of joined peers. Figure 10b shows the average discontinuity (share of chunks not received in-time for playback). The y-axis indicates the discontinuity, while the x-axis indicates the time. Figure 10c shows the latency for chunk distribution. The y-axis indicates the average latency, while the x-axis indicates the time. Figure 10d–f show the 2PC Class Inference. Figure 10d shows the peer distribution in each class throughout the experiment, limited by $\mathcal{U}$ parameter. Figure 10e shows the peer population in Cold Class per peer upload-bandwidth. Figure 10f shows the peer population in Hot Class per peer upload-bandwidth

terminate some partnerships, this churn is small compared to the intrinsic churn in the overlay (peers are frequently changing partners). We find that only around 5% of peers are reclassified every 120 seconds in our experiments (and this fraction is bounded by Γ). The overlay is mostly unaffected by reclassification, and peers that are reclassified need to reestablish only a fraction of their partnerships, and can usually do so without experiencing discontinuity. The dynamic overlay organization by the 2PC algorithm was evaluated in the work [8]. The authors analyzed the migration of cooperative peers close to the server, while free riders were pushed to the edge of the overlay during a 2PC execution. Overall, we find 2PC allows deployment of Full Partnership Constraints in real scenarios, with significant overlay efficiency improvements.

6 Conclusion

In this work, we present a comprehensive study on strategies for building and managing the overlay in P2P live streaming systems. Several ideas and premises are incrementally tested and evaluated, until they are formalized in the combination of Full Partnership Constraints and Peer Classification for Partnership Constraints. The algorithm dynamically classifies peers according into a set of classes to eliminate competition between high-cooperation, low-cooperation, and free-riding peers. By eliminating such competition, the system ensures that chunks are efficiently distributed in the network. The resulting P2P overlay can, simultaneously, accept a large number of cooperative peers and free riders without compromising the distribution of chunks, even during flash crowds. Although the proposed strategy is not specifically designed for handling flash crowds, it builds robust topologies that perform well. The experiments in PlanetLab consider various scenarios with low peer upload bandwidth, a large share of free riders, and flash crowds.

In summary, the strategies proposed in this work build effective and resilient overlays, have simple implementation, and are compatible with many other techniques already proposed in the literature for managing P2P overlays. As future work, we intend to study how to dynamically set the number of peer classes and the size of partnership sets to further increase overlay distribution efficiency, and combine these strategies to ones presented by the traditional literature.

Acknowledgments This research was partially funded by the CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico) grant 303933/2017-8, CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior), and FAPEMIG (Fundação de Amparo à Pesquisa do Estado de Minas Gerais) grant APQ-02145-18.

References

1. Adhikari VK, Guo Y, Hao F, Varvello M, Hilt V, Steiner M, Zhang ZL (2012) Unreeling Netflix: Understanding and improving multi-CDN movie delivery. In: Proceedings of IEEE INFOCOM (2012), pp 1620–1628
2. Budhkar S, Tamarapalli V (2019) An overlay management strategy to improve qos in cdn-p2p live streaming systems. Peer-to-peer networking and applications
3. Castro M, Druschel P, Kermarrec A, Nandi A, Rowstron A, Singh A (2003) SplitStream: High-bandwidth multicast in cooperative environments. In: Proceedings of the nineteenth ACM symposium on operating systems principles, SOSP ’03. ACM, New York, pp 298–313
4. Chen Y, Zhang B, Chen C, Dah MC (2014) Performance modeling and evaluation of peer-to-peer live streaming systems under flash crowds. IEEE/ACM Trans Netw 22(4):1106–1120

5. Chung TY, Lin O (2011) A batch join scheme for flash crowd reduction in IPTV Systems. In: Proceedings of IEEE parallel and distributed systems, ICPADS (2011). IEEE, pp 823–828
6. Cisco (2019) Cisco visual networking index: Forecast and trends, 2017–2022 Technical report
7. Cohen B (2003) Incentives build robustness in bitTorrent. In: Workshop on economics of peer-to-peer systems (2003)
8. dos Santos AC, Silva CM, Miguel EC (2020) Overlay Convergence Analysis in P2P Networks: An Assessment of the 2PC Algorithm. In: 2020 International Conference on Innovation and Intelligence for Informatics, Computing and Technologies (3ICT), pp 1–6. https://doi.org/10.1109/3ICT51146.2020.9311950
9. Felber P, Biersack E (2005) Distribution: Cooperative content scalability through self-organization. In: Self-star properties in complex information systems (2005). Springer, pp 343–357
10. Fortuna R, Leonardi E, Mellia M, Meo M, Traverso S (2010) QoE in pull based P2P-TV systems overlay topology design tradeoffs. In: Proceedings of IEEE P2P (2010)
11. Gonçalves G, Cunha I, Vieira A, Almeida J (2014) Predicting the level of cooperation in a peer-to-peer live streaming application. Multimedia Systems:1–20
12. Guarnieri T, Cunha I, Almeida JM, Drago I, Vieira AB (2017) Characterizing QoE in large-scale live streaming. In: Proceedings IEEE Globecom, Singapore
13. Guerraoui R, Huguenin K, Kermarrec A, Monod M, Prusty S (2010) LiFTinG lightweight freerider-tracking in gossip. In: ACM/IFIP/USENIX international conference on middleware
14. Jain S, Kumar A, Mandal S, Ong J, Poutievski L, Singh A, Venkata S, Wanderer J, Zhou J, Zhu M, Zolla J, Hölzle U, Stuart S, Vahdat A (2013) B4: Experience with a globally-deployed software defined WAN. SIGCOMM Comput Commun Rev 43(4):3–14
15. Kumar R, Liu Y, Ross K (2007) Stochastic fluid theory for P2P streaming systems. In: Proceedings of IEEE INFOCOM (2007), pp 919–927
16. Li B, Keung GY, Xie S, Liu F, Sun Y, Yin H (2008) An empirical study of flash crowd dynamics in a P2P-based live video streaming system. In: Proceedings of IEEE GLOBECOM (2008), pp 1–5
17. Li B, Xie S, Qu Y, Keung GY, Lin C, Liu J, Zhang X (2008) Inside the new coolstreaming principles, measurements and performance implications. In: Proceedings of IEEE INFOCOM (2008)
18. Liao X, Jin H, Liu Y, Ni LM, Deng D (2006) AnySee: Peer-to-peer live streaming. In: Proceedings of IEEE INFOCOM (2006), pp 1–10
19. Liu F, Li B, Zhong L, Li B, Jin H, Liao X (2012) Flash crowd in P2P live streaming systems fundamental characteristics and design implications. EEE Trans Parallel Distrib Syst 23(7):1227–1239
20. Liu F, Li B, Zhong L, Li B, Niu D (2009) How P2P live streaming systems scale over time under a flash crowd? In: Proceedings of the 8th international conference on peer-to-peer systems, IPTPS'09, pp 5–5, Berkeley, CA, USA. USENIX Association
21. Lobb RJ, Silva AP, Leonardi E, Mellia M, Meo M (2009) Adaptive Overlay Topology for Mesh-based P2P-TV Systems. In: Proceedings of the 18th international workshop on network and operating systems support for digital audio and video (2009), NOSSDAV '09. ACM, New York, pp 31–36
22. Locher T, Meier R, Schmid S, Wattenhofer R (2007) Push-to-pull peer-to-peer live streaming. In: Pelc A (ed) Proceedings of distributed computing: 21st international symposium (2007). Springer, Berlin, pp 388–402
23. Locher T, Meier R, Wattenhofer R, Schmid S (2009) Robust live media streaming in swarms. In: NOSSDAV 2009. ACM, pp 121–126
24. Magharei N, Rejaie R (2009) PRIME: Peer-to-peer receiver-driven mesh-based streaming. IEEE/ACM Trans Netw 17(4):1052–1065
25. Magharei N, Rejaie R, Rimac I, Hilt V, Hofmann M (2014) ISP-friendly live P2P streaming. IEEE/ACM Trans Networking 22(1):244–256
26. Oliveira J, Cunha I, Miguel EC, Rocha MV, Vieira AB, Campos SV (2013) Can peer-to-peer live streaming systems coexist with free riders? In: Proceedings of IEEE P2P (2013). IEEE, pp 1–5
27. Oliveira R, Viana J, Vieira AB, Rocha MV, Campos SV (2013) TVPP: A research oriented P2P live streaming system. In: SBRC Salão de Ferramentas (2013)
28. Payberah AH, Dowling J, Haridi S (2011) GLive: The gradient overlay as a market maker for mesh-based P2P live streaming. In: Proceedings of IEEE parallel and distributed computing, ISPDC (2011), pp 153–162
29. Payberah A, Dowling J, Rahimian F, Seif H (2012) Distributed optimization of P2P live streaming overlays. Special Issue on Extreme Distributed Systems: From Large Scale to Complexity 94(8):621–647
30. Piatek M, Krishnamurthy A, Venkataramani A, Yang R, Zhang D, Jaffe A (2010) Contracts: Practical contribution incentives for P2P live streaming. In: Proceedings of the 7th conference on networked systems design and implementation USENIX (2010), NSDI'10. USENIX Association, Berkeley, pp 6–6

31. Piatek M, Madhyastha HV, John JP, Krishnamurthy A, Anderson T (2009) Pitfalls for ISP-friendly p2p design. Avaliable at https://homes.cs.washington.edu/tom/support/pitfalls.pdf
32. PlanetLab (2009) An open platform for developing, deploying, and accessing planetary-scale services. http://www.planet-lab.org/
33. Ren D, Li YT, Chan SH (2008) On reducing mesh delay for peer-to-peer live streaming. In: Proceedings of IEEE INFOCOM (2008)
34. Rückert J, Richerzhagen B, Lidanski E, Steinmetz R, Hausheer D (2015) TOPT: Supporting flash frowd events in hybrid overlay-based live streaming. In: Proceedings of IEEE IFIP networking conference (2015), pp 1–9
35. Schlinker B, Kim H, Cui T, Katz-Bassett E, Madhyastha HV, Cunha I, Quinn J, Hasan S, Lapukhov P, H. Zeng. (2017) Engineering egress with edge fabric: Steering oceans of content to the world. ACM, New York, pp 418–431
36. Shahriar I, Qiu D, Jaumard B (2017) Modeling of free riders in P2P live streaming systems. In: 2017 international conference on computing, networking and communications (ICNC), pp 729–734
37. Silva AP, Leonardi M, Mellia E, Meo M (2008) A bandwidth-aware scheduling strategy for P2P-TV systems. In: Eighth international conference on peer-to-peer computing (2008), pp 279–288
38. Simoni G, Roverso R, Montresor A (2014) RankSlicing: A decentralized protocol for supernode selection. In: Proceedings of IEEE P2P (2014)
39. Traverso S, Abeni L, Birke R, Kiraly C, Leonardi E, Lo Cigno R, Mellia M (2012) Experimental comparison of neighborhood filtering strategies in unstructured P2P-TV systems. In: Proceedings of IEEE P2P (2012), pp 13–24
40. Traverso S, Abeni L, Birke R, Kiraly C, Leonardi E, Lo Cigno R, Mellia M (2015) Neighborhood filtering strategies for overlay construction in P2P-TV systems design and experimental comparison. IEEE/ACM Transactions on Networking (TON) 23(3):741–754
41. Ullah I, Doyen G, Gaïti D (2013) Towards user-aware peer-to-peer live video streaming systems. In: Proceedings of IFIP/IEEE international symposium on integrated network management (2013). IEEE, pp 920–926
42. Venkataraman V, Yoshida K, Francis P (2006) Chunkyspread: Heterogeneous unstructured tree-based peer-to-peer multicast. In: Proceedings of the 2006 IEEE international conference on network protocols, pp 2–11
43. Wichtlhuber M, Richerzhagen B, Ruckert J, Hausheer D (2014) TRANSIT: Supporting transitions in peer-to-peer live video streaming. In: Proceedings of IEEE IFIP networking (2014)
44. Wu H, Jiang H, Liu J, Sun Y, Li J, Li Z (2011) How P2P live streaming systems scale quickly under a flash crowd? In: 30th IEEE international performance computing and communications conference, pp 1–8
45. Wu H, Liu J, Jiang H, Sun Y, Li J, Li Z (2012) Bandwidth-aware peer selection for P2P live streaming systems under flash crowds. In: Proceedings of IEEE performance computing and communications conference, IPCCC (2012), pp 360–367
46. Yang P, Xu L (2010) On tradeoffs between cross-ISP P2P traffic and P2P streaming performance. In: Proceedings of IEEE GLOBECOM (2010), pp 1–6
47. Yap K, Motiwala M, Rahe J, Padgett S, Holliman M, Baldus G, Hines M, Kim T, Narayanan A, Jain A, Lin V, Rice C, Rogan B, Singh A, Tanaka B, Verma M, Sood P, Tariq M, Tierney M, Trumic D, Valancius V, Ying C, Kallahalla M, Koley B, Vahdat A (2017) Taking the edge off with espresso: scale, reliability and programmability for global internet peering. ACM, New York, pp 432–445
48. Zhao B, Lui J, Chiu D (2009) Exploring the optimal chunk selection policy for data-driven P2P streaming systems. In: Proceedings of IEEE P2P (2009). IEEE, pp 271–280

Affiliations

Eliseu C. Miguel[1] · Cristiano M. Silva[2] · Fernando C. Coelho[3] · Ítalo F. S. Cunha[3] · Sérgio V. A. Campos[3]

Cristiano M. Silva
cristiano@ufsj.edu.br

Fernando C. Coelho
fccoelho@dcc.ufmg.br

Ítalo F. S. Cunha
cunha@dcc.ufmg.br

Sérgio V. A. Campos
scampos@dcc.ufmg.br

[1] Departamento de Ciência da Computação (DCC/UNIFAL-MG), Universidade Federal de Alfenas, Alfenas, Brazil

[2] Departamento de Tecnologia (DTECH/UFSJ), Universidade Federal de São João del Rei, São João del Rei, Brazil

[3] Departamento de Ciência da Computação (DCC/UFMG), Universidade Federal de Minas Gerais, Minas Gerais, Brazil