

**OVERLAY CONSTRUCTION STRATEGIES FOR
PEER-TO-PEER LIVE STREAMING SYSTEMS**

ELISEU CÉSAR. MIGUEL

**OVERLAY CONSTRUCTION STRATEGIES FOR
PEER-TO-PEER LIVE STREAMING SYSTEMS**

Tese apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Doutor em Ciência da Computação.

ORIENTADOR: SÉRGIO VALE AGUIAR CAMPOS
COORIENTADOR: ÍTALO FERNANDO SCOTÁ CUNHA

Belo Horizonte - Minas Gerais

17 de novembro de 2017

ELISEU CÉSAR. MIGUEL

**OVERLAY CONSTRUCTION STRATEGIES FOR
PEER-TO-PEER LIVE STREAMING SYSTEMS**

Thesis presented to the Graduate Program
in Computer Science of the Universidade
Federal de Minas Gerais - Departamento
de Ciência da Computação in partial
fulfillment of the requirements for the
degree of Doctor in Computer Science.

ADVISOR: SÉRGIO VALE AGUIAR CAMPOS
CO-ADVISOR: ÍTALO FERNANDO SCOTÁ CUNHA

Belo Horizonte - Minas Gerais

November 17, 2017

© 2015, Eliseu César Miguel.
Todos os direitos reservados

Ficha catalográfica elaborada pela Biblioteca do ICEx - UFMG

Miguel, Eliseu César.

M636o Overlay construction strategies for peer-to-peer
live streaming systems / Eliseu César Miguel. — Belo
Horizonte, 2017.
xx, 75 f.: il.; 29 cm.

Tese (doutorado) - Universidade Federal de
Minas Gerais – Departamento de Ciência da Computação.

Orientador: Sérgio Vale Aguiar Campos
Coorientador: Ítalo Fernando Scotá Cunha

1. Computação - Teses. 2. Redes de computadores.
3. Sistemas de transmissão de dados. I. Orientador.
II. Coorientador. III. Título.

CDU 519.6*22(043)



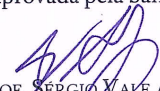
UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

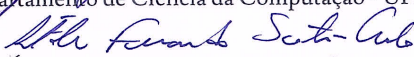
FOLHA DE APROVAÇÃO

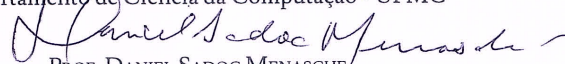
Overlay Construction Strategies for Peer-to-Peer Live Streaming Systems


ELISEU CÉSAR MIGUEL

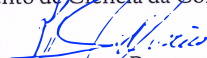
Tese defendida e aprovada pela banca examinadora constituída pelos Senhores:

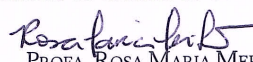

PROF. SÉRGIO VALE AGUIAR CAMPOS - Orientador
Departamento de Ciência da Computação - UFMG


PROF. ÍTALO FERNANDO SCOTÁ CUNHA - Coorientador
Departamento de Ciência da Computação - UFMG


PROF. DANIEL SADOC MENASCHE
COPPE - UFRJ


PROFA. JUSSARA MARQUES DE ALMEIDA GONÇALVES
Departamento de Ciência da Computação - UFMG


PROF. LEONARDO BARBOSA E OLIVEIRA
Departamento de Ciência da Computação - UFMG


PROFA. ROSA MARIA MERI LEÃO
COPPE - UFRJ

Belo Horizonte, 17 de Novembro de 2017.

Resumo

A transmissão de vídeo já representa o maior tráfego na Internet. Este tipo de transmissão pode ser realizado em grande escala em redes de distribuição de conteúdo (CDNs), que incorrem em custos significativos em sua construção e uso. A distribuição de conteúdo de vídeo por meio de sistemas par-a-par, por outro lado, reduz a dependência e o custo de CDNs.

Nas redes par-a-par, o conteúdo é compartilhado em uma sobreposição topológica na rede física. Isso é fundamental para o desempenho da rede. Infelizmente, a distribuição par-a-par é repleta de problemas de qualidade de experiência (QoE). Por exemplo, a chegada simultânea de um grande número de pares, conhecida como *flash crowd*, pode afetar a topologia da rede e interromper a transmissão de conteúdo. Além disso, em cenários em que os usuários têm uma largura de banda de contribuição limitada, os sistemas par-a-par precisam de mecanismos importantes para o incentivo de contribuição de mídia entre os pares.

Os algoritmos atuais que constroem e mantêm a topologia sobreposta em sistemas de transmissão par-a-par para mídia ao vivo geralmente enfrentam problemas de latência na reprodução e descontinuidade de mídia. Quando os pares estabelecem parcerias com um número elevado de vizinhos, pode ocorrer o aumento de troca de mensagem de controle entre os pares, bem como a necessidade de uso de técnicas sofisticadas de filtragem de vizinhanças de pares para garantir que a mídia seja distribuída sem interrupção de fluxo. Mais que isso, os problemas são particularmente desafiadores quando a porcentagem de *free riders* na rede é alta, um caso frequentemente.

Para lidar com esses desafios e ao mesmo tempo mitigar os efeitos negativos do comportamento egoísta de *free riders*, apresentamos a técnica chamada *Classificação de pares e restrição de parcerias*, do inglês *Peer Classification and Partnership Constraints* (2PC) que constrói e mantêm a topologia sobreposta do sistema par-a-par baseada no comportamento de configuração destes sistemas. 2PC estabelece o conceito de classes de pares em que os pares são agrupados com base na contribuição da mídia que cada um

oferece à rede. Essas contribuições são usadas para estabelecer restrições de parceria entre as classes, o que chamamos de *Restrições de Parceria do Peer*, do inglês *Peer Partnership Constraints* (PPC). Cada uma dessas classes configura seus pares com um número limitado de parceiros para permitir que os pares enviem pacotes de dados em ordem de chegada das solicitações e, assim, evitar a necessidade de qualquer técnica sofisticada de filtragem de vizinhança, o que reduz significativamente a complexidade do sistema. Além disso, as restrições de parcerias entre as classes de pares impedem a competição entre free riders e pares cooperativos. Esta falta de concorrência de parceria é fundamental para: (i) facilitar e acelerar o processo de ingresso de novos pares no sistema par-a-par; e (ii) ajuda o 2PC a diminuir a latência da reprodução e a descontinuidade da mídia, aproximando a classe dos pares que mais contribuem ao servidor de mídia enquanto empurra os free riders para a borda da rede. Nossos experimentos mostram que as redes que usam a estratégia do 2PC podem sustentar até 50% de *free riders*, o que não acontece sem o uso de 2PC, sem interferir nas parcerias entre os pares cooperativos presentes no sistema.

2PC exige baixa complexidade de implementação e, além disso, incorre em baixo acréscimo de sobrecarga de mensagens de controle trocada entre pares da rede'. Mais importante, uma vez que nossa solução é baseada no comportamento de configuração das redes par-a-par, nossa estratégia pode ser combinada com abordagens par-a-par atuais que lidam com *flash crowd* e *free riders* para construir e manter topologias de sistemas par-a-par com baixo custo e mais robustas.

Abstract

Video streaming now amounts to the majority of traffic on the Internet. Media streaming relies on large-scale content distribution networks (CDNs), that incur significant costs to build or use. Peer-to-peer distribution of video content reduces reliance on CDNs and costs.

In peer-to-peer networks, peers share content in a topological overlay above the physical network. This is fundamental to network's performance. Unfortunately, peer-to-peer distribution is fraught with quality of experience (QoE) problems. For example, the simultaneous arrival of a large number of peers, known as *flash crowd*, can affect network topology and disrupt content transmission. In addition, in scenarios where users have limited bandwidth to contribute to the overlay, peer-to-peer systems need important mechanisms for peer contribution incentive in order to deliver media content for all peers.

Existing peer-to-peer live streaming algorithms for constructing and maintaining network topology often face issues of high playback latency and media discontinuity problems. When peers achieve a larger number of partners, both control message overhead rises and sophisticated neighborhood filtering techniques are required to deliver media without disrupting the flow. Problems are particularly challenging when the percentage of free riders in the network is high, which is often the case.

In order to deal with these challenges while mitigating free rider negative effects, we present the *Peer Classification and Partnership Constraints (2PC)* that constructs and maintains the network topology focusing only on simple peer-to-peer network configuration. The algorithm establishes the concept of peer classes in which peers are grouped by their media contribution to the network. These contributions are used to establish partnership criteria among classes, what we call *Peer Partnership Constraints (PPC)*. Each of these classes sets up its peers with a limited number of out-partners in order to allow peers to send chunks in the request arrival order and avoid any sophisticated neighborhood filtering technique, significantly reducing system complexity. Moreover, constraints on peer classes prevent partnership competition

between free riders and cooperative peers. This lack of partnership competition is fundamental to: (i) facilitate and speed up the process of new peers joining the overlay; and (ii) help 2PC improve playback latency and media discontinuity by bringing the class of the most contributing peers closer to the media server while moving free riders to the network edge (Silva et al., 2008; Liu, 2007). Our experiments show that 2PC ensures that the network can sustain 50% of free riders without disturbing cooperative peer partnerships on the overlay.

2PC requires low implementation complexity and in addition, incur on the low overhead of exchange messages on the network. Most important, since we are basing our solution on peer-to-peer configuration behavior, our strategies may be combined with current peer-to-peer approaches which face flash crowd events and which handle free rider peers.

Palavras-chave: Peer-To-Peer, Networks, Flash Crowd, Free Rider, Mesh Overlay Topology, Overlay Construction.

List of Figures

2.1	Mesh-based overlay topology	12
2.2	Tree-based overlay topology	13
2.3	Circular Buffer $b(p)$	15
2.4	Results Analysis Example	22
3.1	Baseline Technique to Join Newcomer Peers	26
3.2	Performance for Batch Joining	27
3.3	Parallel Overlays Stages	28
3.4	Parallel Overlays to Join Newcomer Peers	28
3.5	Performance of Parallel Overlays Joining	29
3.6	Free Rider Slice to Join Newcomer Peers	30
3.7	Performance of Free Rider Slice Joining	31
3.8	Free Rider Slice to Joining Newcomer Peers (50% of Free Riders)	32
3.9	Performance of Free Rider Slice Joining (50% of Free Rider)	32
4.1	Performance of Classic and PPC for Balanced Overlays ($\rho \approx 1.0$)	39
4.2	Performance of Classic and PPC for Conservative Overlays ($\rho \approx 1.5$)	40
4.3	Performance of Classic and PPC for Aggressive Overlays ($\rho \approx 0.5$)	42
4.4	Performance of PPC for Aggressive Overlays ($\rho \approx 0.5$)	43
5.1	Performance of 2PC	48
5.2	Partnership Constraint Algorithm Class Inference.	49
5.3	Performance of Classic and 2PC. Join Phase [Warm Class] ($\rho \approx 1.0$)	51
5.4	Performance of Classic and b-2PC. Join Phase [Suggested Class] ($\rho \approx 1.0$)	52
5.5	2PC Class Inference. Join Phase [Warm Class] ($\rho \approx 1.0$)	53
5.6	b-2PC Class Inference. Join Phase [Suggested Class] ($\rho \approx 1.0$)	54
5.7	Performance of Classic and 2PC. Join Phase [Warm Class] ($\rho \approx 1.5$)	55
5.8	Performance of Classic and b-2PC. Join Phase [Suggested Class] ($\rho \approx 1.5$)	56
5.9	2PC Class Inference. Join Phase [Warm Class] ($\rho \approx 1.5$)	57

5.10 b-2PC Class Inference. Join Phase [Suggested Class] ($\rho \approx 1.5$)	58
-------------------------------------------------------------------------------------------	----

List of Tables

3.1	Network Peer Configuration Classes	24
3.2	Network Peers Configuration for 50% of Free Rider	31
4.1	Peer Classes and Output Partnership Configurations	37
4.2	Default Experiment Configuration ($N_{\text{in}}(p) = 20, \rho \approx 1.0$)	38
4.3	Default Experiment Configuration ($N_{\text{in}}(p) = 20, \rho \approx 0.5$)	41
5.1	2PC Experiment Configuration ($N_{\text{in}}(p) = 20, \rho \approx 1.0$)	47
5.2	2PC Experiment Configuration (random upload bandwidths)	50

List of Algorithms

1	Peer Classification algorithm	46
---	-----------------------------------------	----

Contents

Resumo	ix
Abstract	xi
List of Figures	xiii
List of Tables	xv
1 Introduction	1
1.1 Our Major Peer-to-Peer Challenges	4
1.2 Definition of the Problem	6
1.3 Work Contributions	7
1.4 Text Organization	8
2 Peer-to-Peer System Definitions and Configurations	11
2.1 Peer-to-Peer Networks	11
2.2 <i>Flash Crowd</i> on Peer-to-Peer Networks	15
2.3 <i>Free Rider</i> on Peer-to-Peer Networks	16
2.4 TVPP: A Peer-to-Peer Network Implementation	17
2.4.1 Media Streaming on TVPP	17
2.4.2 Bootstrap Server Reports	18
2.4.3 Peer Partnership Rules	19
2.4.4 Overhead of Our Techniques Implementation	19
2.5 General Experimental Method	19
2.6 Understanding the Result Charts	21
3 Effects of Partnership Constraints	23
3.1 Introduction	23
3.2 Experimental Method	24

3.3	Baseline: Batching Newcomer Peers	25
3.4	Increasing the Robustness of the Peer-to-Peer Overlay by Instantiating Parallel Overlays	26
3.5	Naïve Model: Implementing Partial Partnership Constraints	29
4	Full Partnership Constraints in Static Environments	35
4.1	Introduction	35
4.2	Peer Partnership Constraint	36
4.3	PPC Configuration	37
4.4	PPC Evaluation	38
4.4.1	Balanced Overlays ($\rho \approx 1.0$)	38
4.4.2	Conservative Overlays ($\rho \approx 1.5$)	40
4.4.3	Aggressive Overlay ($\rho \approx 0.5$)	41
5	Full Partnership Constraints in Dynamic Environments	45
5.1	Partnership Constraint Class Algorithm Concepts (2PC)	45
5.2	2PC Configuration	47
5.3	Comparative Evaluation between 2PC and PCC	48
5.4	2PC Evaluation	49
5.4.1	2PC and b-2PC Evaluation. ($N_{in}(p) = 20$ and $\rho \approx 1.0$)	50
5.4.2	2PC and b-2PC Evaluation. ($N_{in}(p) = 30$ and $\rho \approx 1.5$)	51
5.5	2PC Discussion	52
6	Related Work	59
6.1	Flash Crowds	60
6.2	Partnership Management, Overlay Construction and Selfish Behavior	60
7	Conclusion and Future Works	65
	Bibliography	67

Chapter 1

Introduction

Video content sharing among Internet users has become very attractive, being the dominant class of Internet data traffic (Cisco, 2014; ConsumerLab, 2014; Sandvine, 2014). According to Cisco, Internet video traffic will be more than 80% of total Internet traffic in 2019 (Cisco, 2017).

Nowadays, we have robust technologies to support the transmission of voluminous contents such as music, videos, and computer programs on the Internet. Peer-to-peer (P2P) systems have become a popular technology for this purpose, mainly because it supports a large number of users (peers) and requires a low operating cost compared to other options, such as the traditional client-server based model and Content Delivery Networks (CDN) (Ullah et al., 2013).

Live streaming is an important application of Internet content sharing. In 2009, Obama's inauguration reach the largest record of concurrent live viewership on a single day, with 7 million of active simultaneous streams (GIGAOM, 2009b) supported by peer-to-peer technology that has helped CDN networks. According to Turner, (...) *the resultant stresses on the various CDNs and the Internet in general would make it extremely difficult if not impossible to serve such an unprecedented audience without the use of peer-to-peer technologies.* (GIGAOM, 2009a).

Peer-to-peer networks allow live streaming to large audiences without relying entirely on (geographically-distributed) server upload bandwidth. In peer-to-peer networks, peers redistribute received content to other peers to improve system scalability and reduce infrastructure costs. Existing systems support thousands of simultaneous users in multiple media distribution channels (PPlive, 2013; UUSee, 2013; SopCast, 2013; PPS, 2017; TVU, 2017). Peers create partnerships in a decentralized way, forming overlay topologies over the physical network for exchanging media content.

However, live streaming challenges are still present. Recently, Trump's inauguration broke live video streaming records, peaking at 8.7Tb/s at 12:04 ET during the opening of President Trump's speech (TechCrunch, 2017). Although recent improvements in the Internet bandwidth, HD video quality transmission is increasing among Internet users and, also, the traditional TV broadcast has been losing the preference for online video streaming.

Also, many interesting challenges arise in overlay topology maintenance and media distribution strategies on peer-to-peer live streaming. For example: (a) peer *churn*, caused by peers joining and leaving the overlay, breaks partnerships and disrupts media distribution (Zheng et al., 2011); (b) *uncooperative* peers, which do not or cannot contribute with media redistribution (also known as *free riders*), increase resource competition in the overlay (Meulpolder et al., 2012); and (c) *flash crowd events*, that occur when a large number of peers join the peer-to-peer network in a short time period, which can disrupt the stability of media transmission. Known approaches to handle flash crowd events often involve sophisticated overlay maintenance (Payberah et al., 2011; Lobb et al., 2009) or peers need to implement media request scheduling strategies before sending the content to other peers (Wu et al., 2012; Silva et al., 2008).

Moreover, the delivery of media with low latency and low discontinuity demands sophisticated strategies typically based on incentive mechanisms for maximizing peer cooperation (Piatek et al., 2010). Basically, these strategies seek to establish promising relationships among peers and at the same time to bring cooperative peers close to the media server in order to construct a robust overlay topology (Payberah et al., 2011; Lobb et al., 2009; Fortuna et al., 2010). However, in addition to the cost of implementing incentive mechanisms, typical strategies for constructing and maintaining the overlay topology are not centralized, which requires the unwanted exchange of additional control messages among peers (Payberah et al., 2011; Lobb et al., 2009).

In a way to reach promising relationships among peers, typically peer-to-peer systems do not impose a limit of partnerships that each peer can established. As a consequence, peers may become unable to respond to all media requests (due to limited upload bandwidth). In order to address such issue, complex neighborhood filtering and chunk scheduling techniques are required to restrict and select a subset of partners at each peer to improve media distribution. According to Lobb et al. (2009), the larger the partnership gets at each peer, more complex it is to implement and maintain chunk scheduling techniques on the peer-to-peer system. Consequently, neighborhood filtering techniques are crucial to ensure the correct operation of peer-to-peer live streaming.

But, what happens when peer-to-peer networks are configured without sophisticated neighborhood filtering techniques? What peer-to-peer parameters are important to determine the behavior? Is it possible to avoid the neighborhood filtering techniques usage while keeping the media delivery for a large number of peers? Is it possible to construct a robust overlay topology without using media contribution incentive mechanisms?

In this thesis we investigate these questions in order to develop unsophisticated techniques of peer-to-peer overlay construction. We are interested in achieving robust overlays that support a large number of uncooperative peers and, at same time, that preserves the network against the negative effects of uncooperative peers presence. The strategies that we propose should be robust to permit a large fraction of uncooperative peers in the network only by configuring peer partnership rules.

Our objectives are to face peer-to-peer challenges by developing simple strategies with a focus on: (i) first, to provide media distribution among peers without interruptions (low discontinuity); (ii) second, to balance peers' upload bandwidth utilization in order to provide broad media distribution; and (iii) last, to keep a reduced difference on the media playback among peers (low media latency). Certainly, peer churn, free riding behavior and flash crowd events negatively affect our objective requirements.

We have developed an easy solution in order to restrict the partnerships among peers in the network, called *peer partnership constraints*. Peer partnership constraints are an important contribution of this thesis as a simple mechanism that allows the network to take advantage of the cooperative peers. Furthermore, the peer partnership constraints are a base of our mechanism to construct dynamically a robust peer-to-peer overlay topology.

Initially, we impose partnership restriction only between free rider and cooperative peers. Later, we extend this concept to all peers. We group peers by classes that define peers setup and partnership constraints of peers among existing classes. Our evaluation shows the proposed peer partnership constraints are sufficient to ensure the quality of peer-to-peer services without the complexity of neighborhood filtering techniques or the complexity of overlay construction mechanisms.

Our systems have been evaluated by imposing increasingly peer-to-peer constraints resources to attest its effectiveness. We have combined peer upload bandwidth constraint with a larger fraction of free rider peers in a way to obtain a severe peer-to-peer environment. In addition, we have applied the flash crowd events in order to evaluate the robustness of our overlay topologies.

As results, we show that our techniques provide stability of media distribution on the peer-to-peer systems for a larger fraction of free rider peers even facing the stresses of flash crowd events. We allow cooperative peers to answer its partners by sending the media in FIFO order without any sophisticated neighborhood filtering technique. This reduces CPU consumption, since the system implementation is simplified. In addition to avoid sophisticated neighborhood filtering techniques, application of peer partnership constraint constructs naturally a more robust overlay, placing peers closer to the media server or on the overlay's edge considering peer media contribution. Finally, we believe that our solution may be combined with known other techniques in order to reach more robust peer-to-peer systems.

In short, the main contribution of this thesis is a simple way to manage restrictions of partnerships between peers, *Peer Partnership Constraints* (PPC) presented in Chapter 4. Among peer-to-peer configuration parameters, the maximum number of partners that a peer can establish is the most important to implement PPC. Using PPC, we have shown that it is possible to construct a robust peer-to-peer overlay without neither sophisticated neighborhood filtering nor complexity contribution incentive mechanisms.

Next, Section 1.1 describes with more details our major peer-to-peer challenges. Section 1.2 presents the proposed work and goal, while Section 1.3 presents the thesis contributions. Finally, Section 1.4 presents the thesis organization.

1.1 Our Major Peer-to-Peer Challenges

Peer-to-peer networks are highly scalable structures (Piatek et al., 2009; Ullah et al., 2013). Live streaming, the focus of our work, is a way of content delivery that has become very attractive to Internet users and supported by peer-to-peer systems. According to Xiao and Ye (2008), peer-to-peer systems have proved to be robust for live streaming purpose by enabling large numbers of users simultaneously watching multiple video channels. However, even though peer-to-peer networks are robust, the configuration of these systems for live streaming is difficult since the content must be distributed in a short time to ensure the uninterrupted visualization by users. Failure to receive portions of media content, for example, may cause media playback interruptions since media viewing occurs at the time of content distribution (Locher et al., 2007).

Over time, several issues have emerged on peer-to-peer live streaming and we are interested on these challenges, that include:

1. *Free riding behavior*: In certain situations, some users may avoid sharing received content. These peers, known in the literature as *free riders*, hinder the content propagation among peers, since it decreases the content distribution opportunities. Many research have made efforts to understand and to curb this behavior (Pianese et al., 2006; Wang, Wenjie and Xiong, Yongqiang and Zhang, Qian and Jamin, Sugih, 2006; Locher et al., 2009; Piatek et al., 2010; Adar and Huberman, 2000; Moltchanov, 2011; Karakaya et al., 2009; Krishnan et al., 2004). However, there are situations where upload bandwidth constraint of peers imposes the free riding behavior of them, as occurs with mobile device users. In such cases curbing free rider peers is not desirable;
2. *Peer churn*: According to Cui et al. (2007), in this thesis we define *peer churn* as the action of a peer dynamically join or leave the streaming. Peer-to-peer systems must support high peer churn during the transmission. Peer churn is a factor that may degrade the quality of live streaming media (Zheng et al., 2011). On one hand, with the departure of some peers, new partnerships must be established to maintain data flow for peers joined on the network. On the other hand, the arrival of peers increases both the partnership request and the data competition until newcomer peers start sharing the received media. Several papers offer techniques for minimizing the problems caused by peer churn (Tran et al., 2003; Castro et al., 2003; Locher et al., 2009).
3. *Flash crowd events*: Flash crowd is a sample of event that happens suddenly. When an event of broad user interest is transmitted, such as a final of football world cup, a larger number of users may appear simultaneously requesting to join the network. The simultaneous new partnerships request may compromise the quality of service in the network. In this case, there are several techniques to control the joining rate dealing with the large number of newcomers peers without discouraging them to join the network (Liu et al., 2009; Chen et al., 2011; Li et al., 2008a; Liu et al., 2012).

Among our challenges, we consider free riding behavior the most important. Since typically upload bandwidth of mobile systems is reduced, these users become low cooperatives on peer-to-peer systems (i.e. free rider peers or peers of low media contribution). So, the growing number of mobile fraction users motivates to find solutions in order to improve the peer-to-peer live streaming stability to support a large number of uncooperative peers.

1.2 Definition of the Problem

In this section, we informally describe some peer-to-peer concepts in order to present our problem. Chapter 2 formally defines all concepts presented in this section.

Bootstrap server is a special server in peer-to-peer systems that allows peers to join the network. Peers contact the bootstrap server and receive from him a list of active peers in the network. With a list of active peers, a newcomer peer can establish its partnerships with other and starts the media sharing. If a peer receives the media with no discontinuity (or low discontinuity), we consider that peer is stable. However, high discontinuity may force a peer to leave the network in order to contact the bootstrap server again and redo the join phase (i.e. a non-stable peer). Consequently, if a large fraction of peers is stable, we consider the peer-to-peer network as stable, too. So, peer-to-peer systems stability is preserved on the streaming when the delivery of media content occurs with low discontinuity to a large fraction of peers in the network.

Preserving the peer-to-peer stability is a challenge mainly when the network constraints are severe (e.g. low surplus bandwidth upload). In this case, peers need to establish promising partnerships to preserve its stability and, also, peer-to-peer overlays need to be robust in a way to offer much media offers. There are some solutions in order to (i) incentive peer contribution, (ii) to help peers to find promising partnerships and (iii) to construct robust network topology overlay in order to preserve peer-to-peer network stability. However, despite these solutions, it is necessary to face the complexity of these approaches in order to apply them on existing peer-to-peer systems.

In this thesis, we propose to provide network stability on media delivery without the usage of sophisticated neighborhood techniques or complex peer partnership filtering. Instead, we try to understand the effects of peer-to-peer network parameters (e.g. number of peers partnerships and sets of peer partnership control) in order to define a robust peer-to-peer configuration that ensures network stability. Our solution is important to reduce complexity present in current sophisticated neighborhood techniques (both for in-partner or out-partner selection), as well as to decrease the overhead of control message exchanges between peers needed to the overlay.

Our problem consists of constructing robust overlay topologies that support severe resource constraints such as large number of uncooperative peers and flash crowd events. We are interested in discovering a way to establish partnerships between peers that avoids peer resource competition and, at same time, that ensure promising partnership to each peer considering peer's contribution behavior.

We investigate the effects of splitting the list of partners in two separate lists based on the peers contribution to the network. With two distinct sets of partners, each peer can handle cooperative and uncooperative peers in different structures. Our overlay control includes the number of cooperative and uncooperative partners over each peer eliminating the resource competition between them. The partnership separation is the main idea of our solution peer partnership constraint and, in addition, the most important contribution of this work.

The goal: Propose and develop dynamic mechanisms to improve the stability of live streaming systems even under resource constraints, such as limited upload bandwidth and selfish peer behavior.

In our study, we have configured severe peer-to-peer constraints composed by a large fraction of uncooperative peers and cooperative peers with low upload bandwidth. Our configuration was severe enough to prevent media transmission without advanced techniques in the presence of flash crowd events, even considering current techniques to handle flash crowds.

So, we have applied peer partnership constraint between peers considering the peer's chunk contribution capacity. As a result, peer partnership constraints avoid resource competition between peers with different contribution behavior. Naturally, peer partnership constraints are the key to the construction and maintenance of the desired overlay topology, where high upload bandwidth peers are kept near the network media server, while free rider peers are pushed to the topologies edge.

In addition, we believe that our solution can be combined with existing solutions without interference in order to offer more improvement on peer-to-peer systems stability.

1.3 Work Contributions

This section describes the most important contributions of this work:

1. We show that it is possible to mitigate the negative effects of free rider peers during the flash crowd event by imposing partnership constraints between cooperative and free rider peers. Furthermore, we show that these restrictions allow both cooperative and free rider peers join simultaneously to the network without disrupting the live transmission.

2. We present parallel overlays technique for handling flash crowd events. Even with resources constraints, a large fraction of free rider peers is able to join the network without resource competition with cooperative peers. Although parallel overlays are hard to configure and its concepts are not deeply studied yet, we believe that parallel overlays can be successfully applied to solve other issues of peer-to-peer systems. So, we consider parallel overlays technique as an important contribution.
3. We propose *Free Rider Slice* and *Peer partnership constraints (PPC)* as two new peer-to-peer overlay construction mechanisms to speed up peer joining on resource-constrained overlays during flash crowd events while preserving quality of experience (QoE) for peers already in the overlay. These are low complexity approaches that construct robust overlay able to support a large fraction of free rider peers (50% in our experiments) for networks with severe upload bandwidth constraints.
4. We present the *Partnership Class and Constraint Algorithm (2PC)*, a new approach for constructing and maintaining the overlay topology. 2PC is a solution with low complexity and low message overhead since its concepts are based on partnerships setup of peer-to-peer systems without sophisticated neighborhood filtering technique. So, 2PC can be combined with current overlay construction solutions in order to provide more robustness for peer-to-peer systems.

1.4 Text Organization

The rest of this thesis is organized as following:

Chapter 2, Peer-to-Peer System Definitions and Configurations, defines formally the concepts of peer-to-peer systems. We describe with more details both flash crowd events and free riding behavior and, in addition, our peer-to-peer implementation TVPP, used on our system executions. Finally, we define the default configurations for all experimentations and explain how our results of experiments are presented and interpreted.

In the Chapter 3, Effects of Partnership Constraints, we evaluate baseline experiments in order to compare with the results of our proposed solutions. Thus, in this chapter we confirm that the effects of restrict partnerships between free riders and cooperative peers are positive, based on results from two new techniques that we have proposed, Parallel Overlay and Free rider slice. Parts of the content of this chapter was published in (Miguel et al., 2016).

Chapter 4, Full Partnership Constraints in Static Environments, presents the Peer Partnership Constraint (PPC), a specialization of Free Rider Slice that imposes partnership constraint all peer classes. PPC provide a simple solution that allows that each peer to handle its partners without the need of sophisticated neighborhood filtering technique. PPC constructs robust overlay topologies to facing flash crowd events. Parts of the content of this chapter was published in (Miguel et al., 2017).

Chapter 5, Full Partnership Constraints in Dynamic Environments, presents the Partnership Constraint Class Algorithm Concepts (2PC), an algorithm to constructs and maintains the overlay topology based on PPC. 2PC is a dynamic and centralized solution with low complexity and no overhead of control message between peers. In our experiments, peer-to-peer systems using 2PC have demonstrated more stability and robustness with low cost of implementation if compared to systems without 2PC.

Chapter 6, presents the most important works that have incentivated our investigation and, finally, Chapter 7 concludes and presents future works.

Chapter 2

Peer-to-Peer System Definitions and Configurations

There are several technologies and strategies for the implementation of Peer-to-Peer networks (Lua et al., 2005b; Liu et al., 2008b; WikiBooks, 2017). The choice of the strategy to be used for building a peer-to-peer network depends on the network purpose, such as sharing stored content or streaming live media, for example. Moreover, aspects related to overlay topology construction and maintenance leads to large differences between implementations.

In this chapter, only important aspects of peer-to-peer network systems to live streaming are treated. Section 2.1 presents the peer-to-peer network systems definitions adopted in this work. In the following Sections 2.2 and 2.3 we define the flash crowd events and free riding behavior concepts applied in this work. We emphasize the readers that both flash crowd events and free riding behavior are mechanisms used such that to impose peer-to-peer network constraints in our experiments. Our peer-to-peer system environment for network experimentations, TVPP, is presented in Section 2.4, while Section 2.5 presents our generic peer-to-peer systems setup for our experimentations. Finally, Section 2.6 explains our readers how to understand our charts in this work.

2.1 Peer-to-Peer Networks

Let T be a peer-to-peer network for live media streaming that employs a mesh-based overlay topology. Let B be the bootstrap server, a server that allow peers to join in the T (described below). Let $P = \{S, p_1, p_2, \dots, p_n\}$ be the set of peers in T where S is a media server (described below) and p_i is a common peer. Let $R \rightarrow P \times P$ be the set of relationships among peers in P . Each peer $p \in P$ seeks to receive the distributed

content and it is expected that each p shares the received content with other peers that have established a relationship in R with it.

The media server S is a special peer that encodes the video, splits the video into chunks (a chunk can contain multiple frames), and starts the transmission. During the media transmission, is expected that each peer p receives all media chunks distributed by S in enough time to watch the live media.

T does not require an explicit authorization or messages sent to other peers from a peer $p \in P$ before p disconnects from T . Any peer can disconnect at any time. However, to join the peer-to-peer network T , a newcomer peer p' must know other peers in the T in order to establish its first partnerships. T has a special server B (bootstrap server) to allow peers join. An important function of B is to manage the active peers $p \in P$. In this way, B receives the join request from the newcomer p' and sends to p' a list $L(p') \subseteq P$ of active peers in T . In addition, B updates P as $P \cup \{p'\}$.

Each peer $p \in P$ manages a set of neighbor peers $\mathcal{V}(p)$ that contains all peers p knows. p updates $\mathcal{V}(p)$ constantly removing inactive peers (i.e peers that have left T) and inserting new peers according to periodically received $L(p)$ from B .

The overlay topology of T is determined by the set of all relationships (i.e. partnerships between peers) in R on the physical network. A peer-to-peer network can define a *tree-structured* overlay topology. In this case, B determines the elements of R forming an acyclic graph whose root is S . However, if each peer $p_i \in P$ is free to establish its own partnerships in R , the peer-to-peer system creates an overlay topology known as *mesh*.

Figure 2.1 shows mesh-based overlay topology peer-to-peer network.

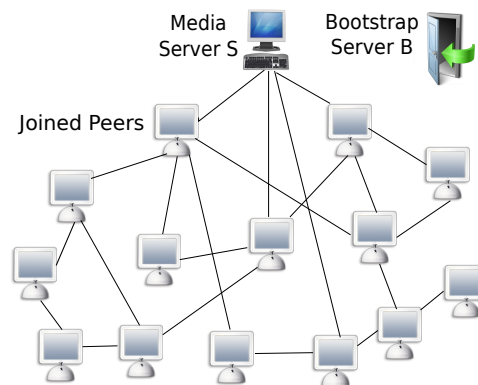


Figure 2.1: Mesh-based overlay topology

Mesh-based peer-to-peer overlay differs from tree-based overlay topology

peer-to-peer networks. In this second case, there must be network management algorithm to position each peer in the overlay topology (Magharei and Rejaie, 2007). This type of topology is more sensitive to peer churn, since it requires a topological restructuring based on a criteria to maintain the structure in the presence of peer dynamism, while mesh-based topologies suffer from the overhead of message to maintain data exchange between peers (Liu et al., 2008a; Venkataraman et al., 2006). In addition, hybrid-based topology that combine concepts of both mesh-based and tree-based topologies is known in the literature (Huang et al., 2007).

Figure 2.1 shows tree-based overlay topology peer-to-peer network.

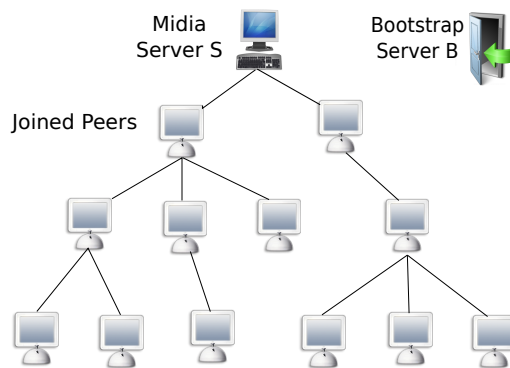


Figure 2.2: Tree-based overlay topology

In order to manage its partnerships, each peer $p \in P$ has a set of partners $\mathcal{N}(p)$ for exchanging video chunks. 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 that receive video chunks from p). In this case, $\mathcal{N}(p) = [\mathcal{N}_{\text{in}}(p) \cup \mathcal{N}_{\text{out}}(p)]$ and $\mathcal{N}(p) \subseteq \mathcal{V}(p)$. The maximum number of in-partners is denoted by $N_{\text{in}}(p)$. Similarly, the maximum number of out-partners is denoted by $N_{\text{out}}(p)$. For the media server S , $\mathcal{N}_{\text{in}}(p) = \emptyset$.

A relationship $(p, p') \in R$ where $p \in \mathcal{N}_{\text{out}}(p')$ requires other partnership $(p', p) \in R$ where $p' \in \mathcal{N}_{\text{in}}(p)$. Successfully established partnerships determine $\mathcal{N}(p)$. When p detects that one of its partners $p' \in \mathcal{N}(p)$ has been silent (control message absence or chunk not received) for longer than a predefined time period, p removes p' from $\mathcal{N}(p)$. Peer p periodically contacts the bootstrap server to obtain a new list $L(p)$ of potential partners to replace the lost partnership.

Metrics definition: Let $t_{(i)}$ be a time instant and let $ck_{(i)}$ be a media chunk delivered by server S at $t_{(i)}$.

Metric 1 Latency Metric: We define network latency (in seconds) as:

$$\text{Network Latency}(t_{(i)}) = \frac{\sum_{(p \in P)} (t_{(p)} - t_{(i)})}{|{}^+P|}$$

Where:

- $t_{(p)}$ is the time in which chunk $ck_{(i)}$ is received by p .
- $|{}^+P|$ is the number of active peers in P that have reported its performance logs.

Metric 2 Discontinuity Metric: In an interval of time $[t_{(i)}, t_{(i+x)}]$, server S generates the chunks $[ck_{(i)}, \dots, ck_{(i+x)}]$. Let $Nck([t_{(i)}, t_{(i+x)}])$ be the number of chunks generated by S in $[t_{(i)}, t_{(i+x)}]$. Let be $Rck_{(p)}([t_{(i)}, t_{(i+x)}])$ the number of chunks that a peer p received in $[t_{(i)}, t_{(i+x)}]$ at the playback time of p . We define discontinuity metric of peer-to-peer network (in percentage) at $[t_{(i)}, t_{(i+x)}]$ as:

$$\text{Network Discontinuity}([t_{(i)}, t_{(i+x)}]) = 1 - \frac{\sum_{(p \in P)} \frac{Rck_{(p)}([t_{(i)}, t_{(i+x)}])}{Nck([t_{(i)}, t_{(i+x)}])}}{|{}^+P|}$$

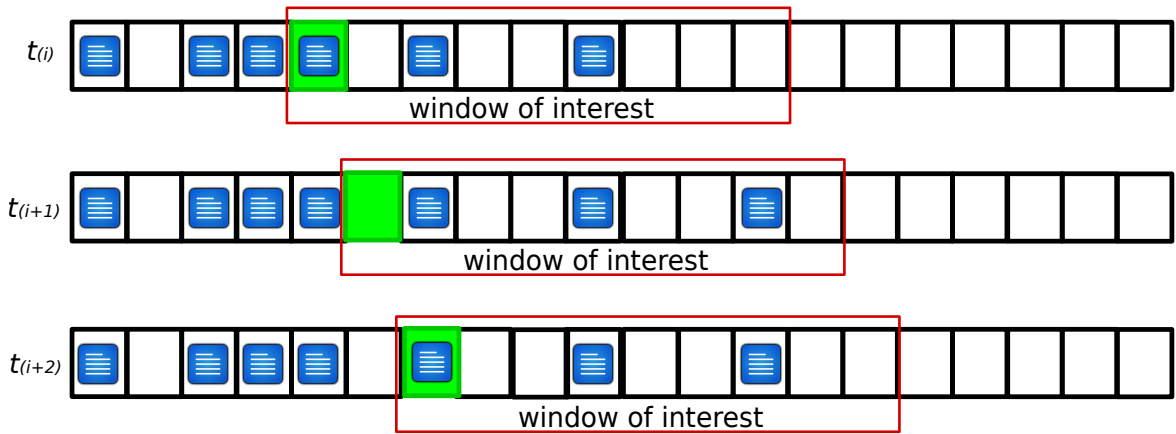
We calculate both latency and discontinuity metrics throughout the experiments at an interval of every 500 media chunks generated.

Each peer p has a buffer area $b(p)$ to store received *chunks*. b is a circular structure that stores only recent chunks. Playback quality is reached with low (or none) discontinuity in media, (i.e. the playback interruption in peer p is caused by the unavailability of some of the required chunks in $b(p)$). To avoid media discontinuity, each peer attempts to receive chunks in advance. Thus, we call *window of interest* of p the region of the $b(p)$ that should be filled relative to the current point of the playback.

When discontinuity stops the video playback for a long time, a peer p should be forced to leave T and ask to join again. We call this peer behavior as *peer join reboot*. Different from commercial live streaming where users can provoke its peer join reboot, in our experimentations the phase of peer join reboot is determined according to how much the window of chunk interest of a peer is empty after its playback was stopped.

Figure 2.3 shows a circular buffer of a peer p with its window of interest. In this figure, $t_{(i)}$ represents the playback time.

Let $t_{(i)}$ be a time instant. At $t_{(i)}$, a peer p is *joined* in the overlay if p 's playback does not motivate the *peer join reboot* of p . We consider that a live streaming network T is *robust* at $t_{(i)}$ if a large fraction of its peers p is joined. In this case, a robust peer-to-peer overlay support a desired media transmission.

Figure 2.3: Circular Buffer $b(p)$

For authors Traverso et al. (2012), about 5% is a satisfactory average of discontinuity on peer-to-peer live streaming networks. As opposed to determine how large should be the fraction of joined peer in order to define the robustness of the system, in our experiments we consider a peer-to-peer system robust when the system discontinuity is up to 5%.

We do not have included the latency metric in the robust overlay concept since high latency does not impose the *peer join reboot* and, consequently, does not affect the media transmission. However, we cite that Traverso et al. (2012) consider that 7 seconds is a satisfactory average reference of peer-to-peer latency metric.

2.2 *Flash Crowd* on Peer-to-Peer Networks

We define $L_{nc} \in B$ (bootstrap server) as the list of newcomer peers that have asked to join the peer-to-peer network and are waiting for the peer joining phase. Ordinarily, peer joining phase happens immediately after the newcomer peer joining request. However joining phase can take longer when occurs a flash crowd.

Flash crowd in peer-to-peer live streaming can be a sudden or an expected event. According to Chung and Lin (2011), flash crowd occurs when thousands of peers join a popular P2P IPTV channel in a short time, which can be expected. According to Chen et al. (2014), flash crowd is a sudden arrival of numerous peers at a system. When a flash crowd occurs, the sudden arrival of numerous peers may starve the upload capacity of the system, hurt its quality of service, and even cause system collapse. In this work, we consider flash crowd as an expected event with a defined time occurrence.

Our results, however, do not require knowing the flash crowd moment in advance.

In this work, we define the flash crowd event as the situation in T when $|L_{nc}| \gg |P|$ such that become a network stability threat. It is not our objective to study flash crowd events. Instead, we have applied flash crowd events in order to evaluate our strategies of overlay topology construction. We need, in this case, to provide enough large fraction of peers to provoke an effective flash crowd event (i.e. enough to collapse the peer-to-peer system).

However, for more information, Chen et al. (2014) provides a comprehensive study on the performance of peer-to-peer live streaming systems under flash crowds, including the mathematical model of the relationship between flash crowd size and system recovery time.

2.3 *Free Rider on Peer-to-Peer Networks*

Despite the significant research effort dedicated to the development of mechanisms to discourage uncooperativeness, it is common the presence of uncooperative peers in the peer-to-peer network systems without disrupting the media transmission.

Oliveira et al. (2013b) categorize peers using their *cooperation level* C , i.e., the ratio of a peer's average upload rate relative to the video bitrate throughout the experiment. Authors categorize peers as free riders if $C = 0$, uncooperative if $0 < C \leq 1$, cooperative if $1 < C \leq 5$, and altruistic if $C > 5$. Free riding behavior is the opposite of the cooperative peer category.

Further, Oliveira et al. (2013b) define two types of free rider, namely *conscious* and *oblivious*. Conscious free riders inform their partners that they are unwilling or unable to upload data. This behavior may be coded in the software or chosen by users. As a consequence, no peer ever wastes time and bandwidth sending requests to conscious free riders. On the other hand, oblivious free riders do not inform their partners that they are unwilling or unable to upload data. This behavior may happen if the software does not make provisions for free riders or due to malicious intent. Oblivious free riders request chunks as normal and advertise the chunks they have, but never upload data (i.e., never answer requests). Oblivious free riders may receive requests and degrade system performance, as their partners will have to retransmit chunk requests after waiting for answers that never arrive.

Although oblivious free riders degrade system performance more than conscious free riders, a simple mechanism was proposed in order to mitigate the negative consequences of the oblivious free riders. This mechanism is called *Simple Unanswered*

Request Eliminator (SURE), a modification to the chunk request scheduler that avoids wasting time and bandwidth by listing the peers that have no good chunk answer history. Using the SURE, experiments with oblivious free riders showed results quantitatively similar to experiments with conscious free riders. SURE works because peers may identify uncooperative peers with few interactions.

Since it is easy to configure a mobile peer-to-peer user on commercial systems to be a conscious free rider (and we are interested in providing a robust network that supports a larger number of free riders), we configure our experiments considering three categories of peers: conscious free riders (as mobile users); cooperative peers; and altruist peers. Moreover, we enable the SURE mechanism in all network systems.

2.4 TVPP: A Peer-to-Peer Network Implementation

TVPP (Oliveira et al., 2013a) is an open source system for peer-to-peer live streaming network execution. The TVPP topology is a mesh overlay that provides peer message exchanges through peer partnerships relations. As some TVPP advantages, we cite: (i) C++ implementation; (ii) easy parameters setup; (iii) similar to peer-to-peer commercial networks. Since TVPP encoding is object-oriented and implemented in a common programming language, it can be compiled for different operating systems and allows easy integration of new code modules.

TVPP implements the *bootstrap server* B , the *media server* S and *client peers*. B is a network access centralized server, and it manages active peers in the network. Media server S and client peers differ each other only by parameter configuration on client code. In the experiments, we defined a single S and a single B . TVPP defines message exchange rules among client peers and between client peers and B in order to ensure the live media streaming. Physical network peers exchange message on an overlay topology defined by the relationship among them.

2.4.1 Media Streaming on TVPP

The media generation is provided by VLC (VLC, 2013). VLC includes a easy-to-use streaming feature that can stream music and videos over a local network or the Internet. So, server S reads the media from the VLC streaming location, splits it into chunks, identifies each chunk with the current chunk ID (i.e. chunk identifier)

and distributes the chunks to its partnership peers. A peer P uses the chunk ID information to sort the media chunks in the $b(p)$ (i.e. p ' chunk buffer). Further, server S informs the current chunk ID to server B which distributes it for all newcomer peers in the joining phases. The current chunk ID exchange permits that a newcomer peer p defines its window of interest (to fill $b(p)$) before starting the media requests.

TVPP is implemented supported by client-oriented pull protocol (Cigno et al., 2008). Periodically, peer p exchanges its *buffer map* (i.e. the chunk ID map of available chunks in $b(p)$) with $\mathcal{N}_{\text{out}}(p)$ to inform them what chunks p has available. Each peer p periodically checks which chunks it needs, identifies which partners $\mathcal{N}_{\text{in}}(p)$ can provide missing chunks, and sends chunks requests accordingly. TVPP allows each peer p be configured to schedule requests depending on chunk availability (e.g., rarest chunk first), or they may schedule depending on playback time, (e.g., earliest deadline first). The rarest first policy tries to replicate a chunk as soon as possible while the earliest deadline policy tries to make playback smoother, among other.

In the scope of this work, we schedule chunk requests using the earliest deadline first policy together with the *Simple Unanswered Request Eliminator* (SURE) (Oliveira et al., 2013b). SURE makes peers prefer to request chunks to in-partners with less unanswered (timed-out) requests, balancing load, and reducing the number of unanswered requests. SURE favors requests to peers that have good answering history. A peer considers that a request has timed out if it is not answered within a determined period of time (500 milliseconds in our experiments). Finally, cooperative peers immediately serve received requests in order of arrival.

2.4.2 Bootstrap Server Reports

Peers send monitoring reports to server B every ten seconds. Reports currently include the number of chunks generated (only reported by the video server), sent, received, and the ones that missed their playback deadline; the number of requests sent, answered, and retried; the average forwarding path length, retry count, and time of arrival of received chunks; neighborhood size; and the number of duplicate chunks received.

Sent monitoring reports by the peers allows B to keep P updated with only active peers and each active peer receives an updated $L(p)$ from B . Server B organizes peer reports into two files: *peer performance*; and *overlay topology*. The first one is important to compute the performance metrics like chunk latency and chunk playback deadline miss rate. The last file is used in order to understand the overlay topology organization throughout the experiment execution time.

2.4.3 Peer Partnership Rules

Peers randomly choose in-partners from peer lists received from the bootstrap server B to connect to. To guarantee a new peer n is able to join the overlay, an existing peer p is able to accept an incoming partnership request even when $\mathcal{N}_{\text{out}}(p)$ is full. In this case, p disconnects a random out-partner $q \in \mathcal{N}_{\text{out}}(p)$ with less out-partnerships than the new peer n , i.e., $N_{\text{out}}(q) < N_{\text{out}}(n)$. Afterwards, peer p remains unable to disconnect more out-partners to accept incoming partnership requests during the next $\tau = 60\text{s}$ to prevent overlay instability.

2.4.4 Overhead of Our Techniques Implementation

In this section, we describe the overhead of message imposed on basic TVPP in order to support our proposed techniques. Subsequent chapters present our techniques with detail.

Parallel Overlay, presented in Chapter 3 includes one new control TCP/IP messages on TVPP, called *auxiliary source setup*. The new message is needed to allow server B to setup some peers in the overlay as temporary *auxiliary source servers*, described in Section 3.4. The messages of auxiliary source setup are used only one time in order to prepare the peer-to-peer overlay to support the flash crowd event. In our experiments, server B has sent only 6 message of auxiliary source setup, which implies a non significant additional overhead. Auxiliary source setup message transmits 8 bytes of information and this message extends the basic *Message Class* on TVPP, that contains 6 bytes.

Both Free Rider Slice, Chapter 3, and Peer Partnership Constraint (PPC), Chapter 4, techniques work on basic TVPP and without additional control message.

Finally, Partnership Constraint Class Algorithm Concepts (2PC), Chapter 5, has no additional message. However, the message of system control exchanged between peers includes two new fields, raising the basic TVPP message in 4 bytes. These 4 bytes are needed in 2PC because each peer $p \in P$ has to share $N_{\text{out}}(p)$ with its partners in order to allow disconnect behavior in the system, described above in Section 2.4.3.

2.5 General Experimental Method

We have run experiments on PlanetLab (PlanetLab, 2013) using TVPP. We use as many PlanetLab nodes as possible in our experiments (around 110, with slight variations between experiments). Peers in the overlay are subject to CPU and

bandwidth restrictions in the underlying PlanetLab node.

In order to limit the upload bandwidth to more realistic scenarios, we also apply a bandwidth limit, defined for each experiment. The bootstrap server B runs on our university’s network (UFMG - Universidade Federal de Minas Gerais, Brazil) and is not subject to bandwidth limitations (and has spare CPU capacity). The video server S streams a 420kb/s video. Each chunk is a MTU-sized packet, which gives around 40 chunks per second.

The experiment is composed of five runs, and we show average results over all runs. Experiment times are different among experiment sections. So, the reader can see the total time of each experiment on local *Experimental Method* section in each chapter and on presented charts. Each experiment consists of two phases, before and after the flash crowd event. During the first phase, we run a single TVPP client on each PlanetLab node (i.e., around 110 peers). On the second phase, we launch ten additional peers on each PlanetLab node to emulate a flash crowd event. To preserve packet loss and delay in a realistic scenario we do not allow peers’ communication within the same PlanetLab’s node.

Upload bandwidth distribution: In this thesis, first, we consider, a known upload bandwidth distribution for peers, and later a random upload bandwidth distribution, described in Chapter 5. Such as (Lobb et al., 2009) and (Traverso et al., 2015), we define four distinct upload bandwidth distribution, that configure peers as: Hot; Warm; Cold; and Free Rider peers. Both authors (Lobb et al., 2009) and (Traverso et al., 2015) have configured a fraction of 10% of its Hot peers with a higher upload bandwidth (5.0Mb/s). Similarly, we have setup 4.0Mb/s of upload bandwidth for Hot peers within the range of [9-11]%

In order to configure Free Rider peers, we use 0.0Mb/s of upload bandwidth and a large fraction of them (starting in 40%). Our fraction of Free Rider peers differ from the fractions of 10% and 0% defined in (Lobb et al., 2009) and (Traverso et al., 2015), respectively. In our systems, we setup the upload bandwidth for Warm and Cold peers, so that to provide the peer-to-peer systems with an upload bandwidth average equivalent to 1.13Mb/s, presented by Traverso et al. (2015) in their experiments.

Maximum number of in-partners and out-partners (N_{in} and N_{out}): According to Traverso et al. (2015), the goal of configuring $N_{in}(p)$ is to guarantee that a peer $p \in P$ has enough in-partners to sustain the stream download with high probability in face of peer-to-peer challenges. Glive (Payberah et al., 2011) configure peers’ in-partners $N_{in} = 8$ and Liu et al. (2012) suggest the value 20 for the number

of partners, but they do not explicitly separate their set of partner into in-partners and out-partners such as in this thesis. So, we configure N_{in} not less than 10 partners, being that in the majority of our experiments $N_{in} = 20$.

As described before in Section 2.1, a relationship $(p, p') \in R$ where $p \in \mathcal{N}_{out}(p')$ requires other partnership $(p', p) \in R$ where $p' \in \mathcal{N}_{in}(p)$. So, in our default configurations, we define the maximum number of out-partner N_{out} in a way to balance the $\sum_{(p \in P)} N_{in}(p)$ and $\sum_{(p \in P)} N_{out}(p)$.

2.6 Understanding the Result Charts

Figures from this section are used only in order to explain how we present and understand our experiment results. In this section, *Exp01* and *Exp02* are only a examples of experiments and we use both in order to explain the real experiments behavior.

As described before, our experiments are composed of five runs, and we show averaged results over all runs. However, not all charts display average information. For example, Figure 2.4(a) shows peers joining in the network. The purple (upper) points show the number of *incoming* peers, i.e., that are trying to join the overlay. The green (lower) and the red (lower) points show the number of peers that successfully *joined* the overlay, i.e., stables peers, on experiments *Exp01* and *Exp02*, respectively. In this case, we compute the number of incoming and joined peers in non-overlapping 10 seconds time intervals (x axis). Each chart aggregates results for five experiment runs, i.e., for every 10 seconds-period in the x axis, we show five numbers of incoming peers and five numbers of joined peers.

We are looking for peer-to-peer network configurations that preserve a large number of joined peers in the network during the whole experiment time. Figure 2.4(a) presents a situation where the overlay preserves only a low fraction of joined peers for the duration of the experiment (*Exp01*) and, on the other hand, an opposite situation (*Exp02*). For example, at time 850 seconds of the experiment the number of joined peers ranged between 450 and 650 depending on which of the five experiments is observed. In comparison, for *Exp02* the number of joined peers is stable at this time.

Thus, we consider that real experiments that provide results similar to the results of *Exp01* (Figure 2.4(a)) come from unstable networks and, consequently, these network configurations are undesired or unacceptable. On the other hand, experiments that provide results similar to the results of *Exp02* come from stable networks and these network configurations are our target, Figure 2.4(a), too.

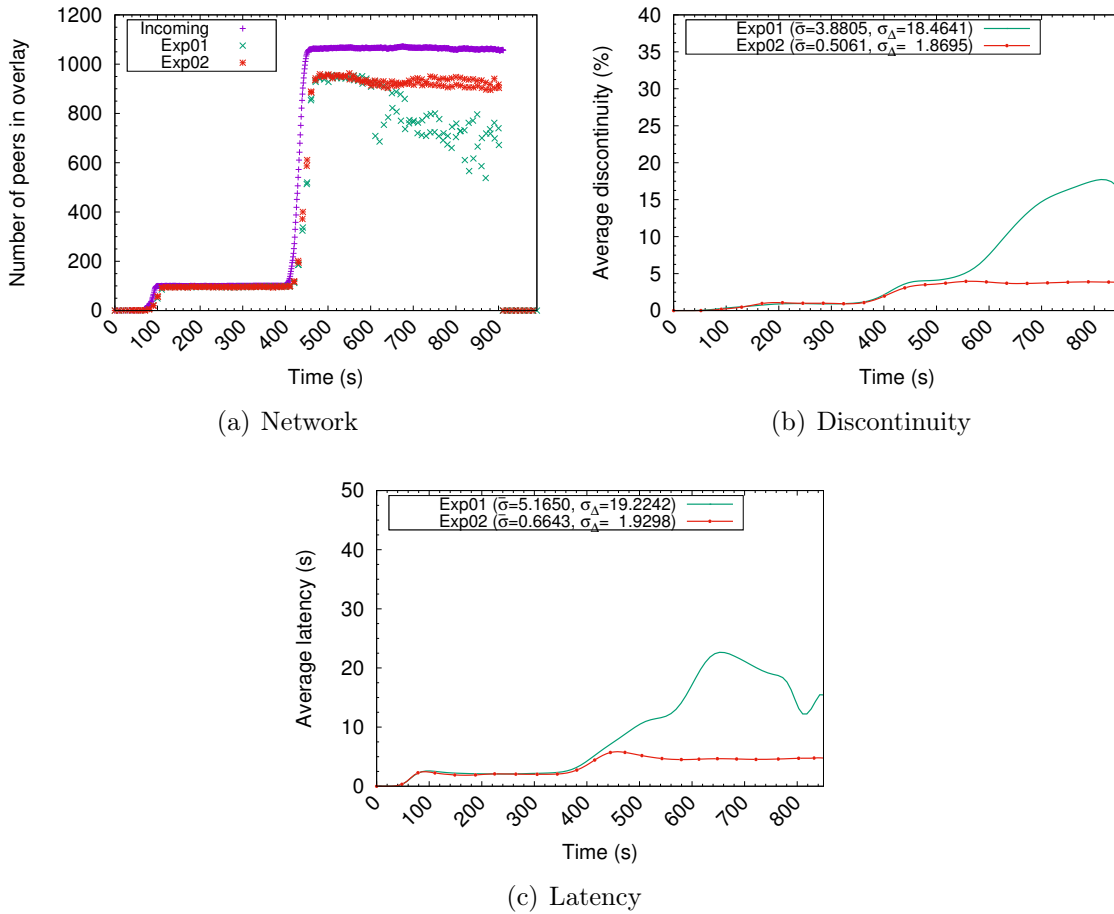


Figure 2.4: Results Analysis Example

Figures 2.4(b) and 2.4(c) show the average of both discontinuity and latency metrics for *Exp01* and *Exp02*. For a desired peer-to-peer setup, we want to reach up to 5% of discontinuity average and 7 seconds of latency average. Thus, besides the network configuration for *Exp01* does not preserve stable the joined peers, both discontinuity and latency metrics for *Exp01* are over our acceptable threshold. On the other hand, *Exp02* kept our imposed threshold, as shown in the charts.

We calculate both the average and the standard deviation for each metric every 10 seconds for the values obtained in each of the five experiments at that point in time. The average computed at each time is shown in the chart line. Each discontinuity and latency charts also present: (i) $\bar{\sigma}$ that is the value of the average of the standard deviation for the metric throughout the experiments; and (ii) σ_{Δ} that is the value of the maximum standard deviation during the experiments.

Chapter 3

Effects of Partnership Constraints

3.1 Introduction

In this chapter we present and evaluate an overlay maintenance strategy for integrating new peers into the overlay during challenging scenarios. In particular, we evaluate the performance of a resource-constrained overlay during a flash crowd event. We first present results for an existing mechanism to handle flash crowds introduced by Liu et al. (2012) in order to establish a baseline (Section 3.3). We then propose a new overlay maintenance strategy to join new peers in the overlay during flash crowd events, the parallel overlays technique (Section 3.4).

Using parallel overlays technique, even with resources constraints, a large fraction of free riders are able to join the network without resource competition with cooperating peers. This is possible because parallel overlays were set to not allow partnership formation between free rider peers that arrive during the flash crowd and cooperative peers that joined the network before the flash crowd event. Parallel overlays technique creates new overlay topologies to contain newcomer free riders without permitting them to initiate partnership with any peers from the master overlay. This mechanism shows been enough for limiting the risks offered by free riders exhausting network's resources.

In addition, we present the *free rider slice* (Section 3.5) as a simple way to restrict partnerships between free riders and cooperative peers without the need to implement parallel overlays. We show that it is possible to mitigate free rider negative effects during a flash crowd event by only setting restrictions to partnerships between free riders and cooperative peers using either free rider slice or parallel overlays technique. However, free rider slice is better than parallel overlays since free rider slice offers an easy implementation, faster peer joining, and works keeping only an existing mesh overlay.

The idea of constraining the use of network resources to increase the performance of admitted users is not novel (Adar and Huberman, 2000; Ma et al., 2006). Admission control, for instance, constrains the entrance of peers so as to guarantee service levels to those peers that effectively use the network. The constraints imposed on the connections presented in this work are a soft version of admission control, as they restrict the number of connections established by existing peers. In essence, such restrictions are imposed to prevent problems such as the tragedy of the commons (Hardin, 1968), wherein peers that don't contribute to the system degrade the performance of the whole population.

In order to study the flash crowd in this chapter, we define: (i) *master network*, the mesh overlay topology composed by the set of peers that joined on peer-to-peer network before the flash crowd's event; and (ii) *whole network*, the peer-to-peer network composed by the set of all active peers. Thus, before the flash crowd, *master network* and *whole network* are the same.

3.2 Experimental Method

Our experiments in this chapter were performed in 1350 seconds. The first 70 seconds initialize bootstrap and media servers. Then we construct a *master network* with a group of around 110 peers to support a flash crowd event of around one thousand of new peers, that happens at 350 seconds after each experiment beginning. In a total, we run around 1100 peer instances (i.e. 11 instances in each PlanetLab's node). We allocate peers to the four classes shown in Table 3.1 at random with a probability given by the *proportion* column. As described in Section 2.3, Chapter 2, since $N_{\text{out}}(p) = 0$ for each $p \in \textit{Free rider Class}$, it ensure that all free riders are *conscious*.

Table 3.1: Network Peer Configuration Classes

Peer Class	Upload (Mb/s)	Proportion	$N_{\text{in}}(p)$	$N_{\text{out}}(p)$
Hot peers	4.0	11%	10	23
Warm peers	2.5	22%	10	20
Cold peers	1.5	27%	10	09
Free rider	0.0	40%	10	00

3.3 Baseline: Batching Newcomer Peers

The authors Liu et al. (2012) present the most important solution to handle flash crowd event on mesh-based overlay for live streaming, becoming our baseline technique. To evaluate the behavior of our overlay configurations we implemented a simplified version of a technique to join newcomer peers during flash crowd events by Liu et al. (2012). This technique holds newcomer peers in a queue to pace the joining process. In each iteration i , the system evaluates the overlay’s resources and determines a subset \mathcal{R}_i of all newcomer peers that are to join the overlay in iteration i . The technique also determines the duration w_i of iteration i , which allows the overlay to integrate peers in \mathcal{R}_i and stabilize prior to the next iteration. Iterations occur until all newcomer peers join the overlay. We fix $w_i = 100seconds$ for all iterations, and consider three scenarios where 50%, 25%, and 18% of newcomer peers join the overlay in each iteration (leading to 2, 4, and 6 iterations, respectively).

Figure 3.1 shows results for the baseline technique. The blue (upper) points show the number of *incoming* peers, i.e., that are trying to join the overlay. The green (lower) points show the number of peers that successfully *joined* the overlay, defined as peers that are receiving at least 95% of all distributed video chunks prior to their chunk playback deadlines. We compute the number of incoming and joined peers in non-overlapping 10 seconds time intervals (x axis). Each graph aggregates results for five experiment runs, i.e., for every 10 seconds-period in the x axis, we show five numbers of incoming peers and five numbers of joined peers. A successful transmission will have the number of joined peers (green points) in a stable line close to the number of incoming peers (blue points).

We observe that some experiments have very low numbers of joined peers. This performance instability illustrates the negative effect of the flash crowd event on the peer-to-peer overlay: as a large number of peers are added to the overlay in a short period, partnerships are broken and chunk distribution is disrupted, ultimately leading to overlay collapse. However, we note batching newcomer peers gives time for the peer-to-peer overlay to stabilize and reduces the impact of the flash crowd.

Figure 3.2 shows the discontinuity and latency averaged across all 5 experiment runs shown in Figure 3.1(d), which had the best results. We show results for all peers in the network (blue line), and only for peers that joined the overlay prior to the flash crowd event, referred to as the *master overlay* (orange line). Figure 3.2 only includes peers that have successfully joined the overlay and are reproducing the video in the overlay, i.e., included in the green points in Figure 3.1(d).

We observe in both Figure 3.2(a) and Figure 3.2(b) that the overlay was robust

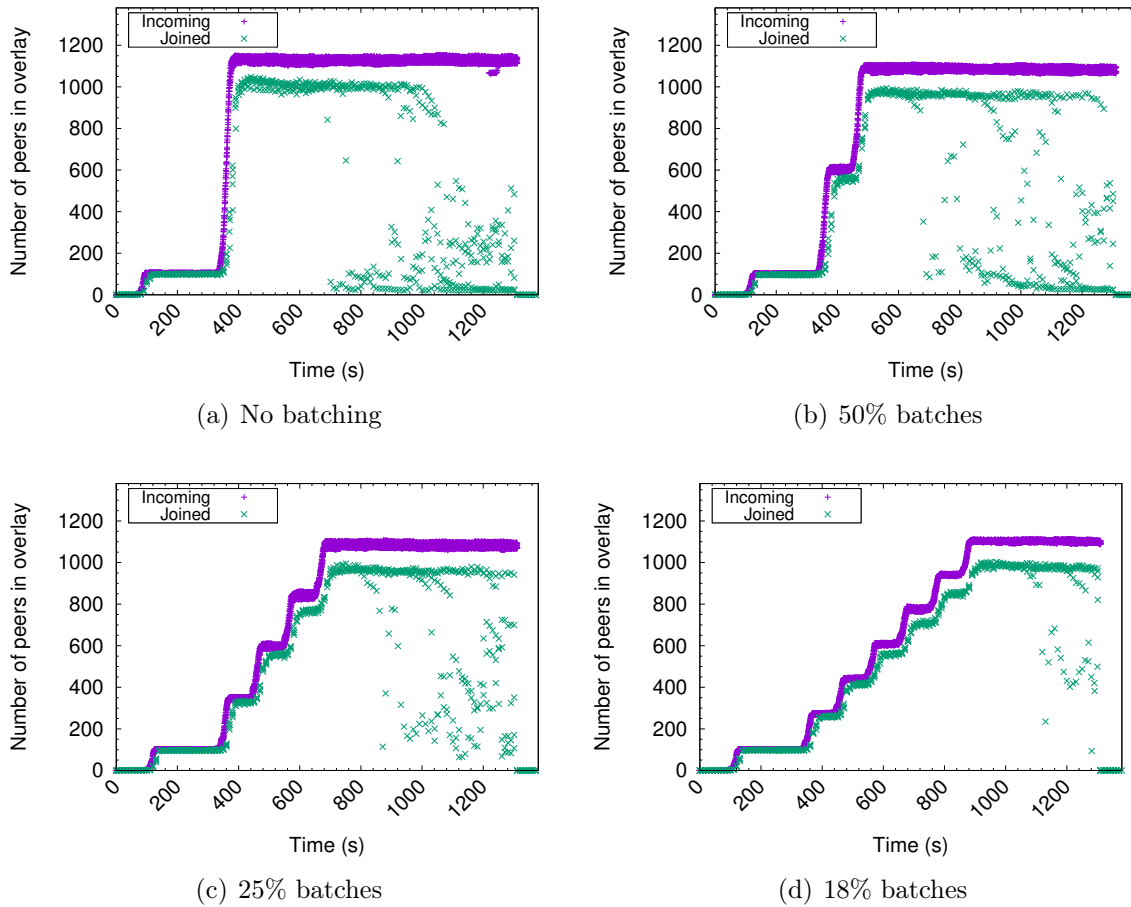


Figure 3.1: Baseline Technique to Join Newcomer Peers

until around 1000 seconds. After 1000 seconds chunk distribution was compromised and average discontinuity and latency increase. We also observe that peers that joined the overlay prior to the flash crowd event are impacted similar to peers joining the overlay during the flash crowd.

3.4 Increasing the Robustness of the Peer-to-Peer Overlay by Instantiating Parallel Overlays

Chung & Lin proposed to control the joining process during flash crowd events for a tree-based peer-to-peer live streaming system (Chung and Lin, 2011). 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.

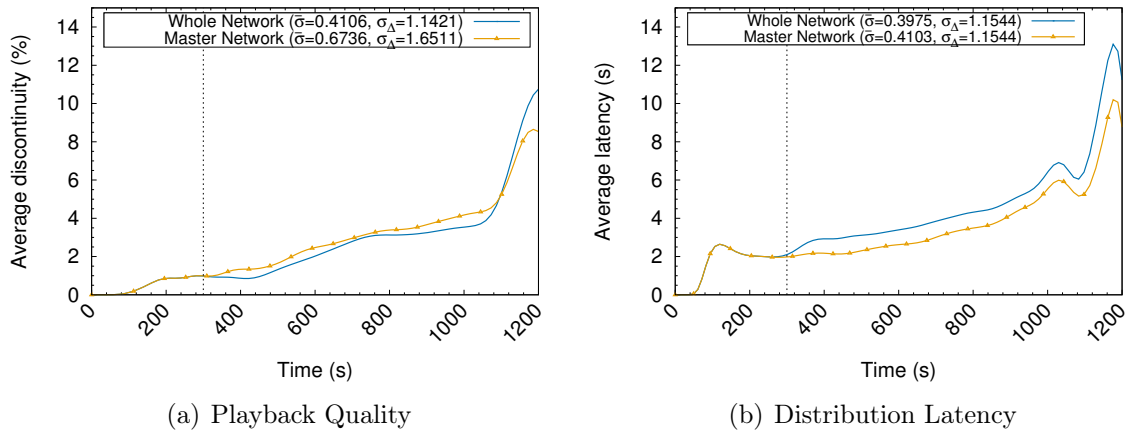


Figure 3.2: Performance for Batch Joining

We next present an adaptation of this idea to mesh-based peer-to-peer overlays. The main feature of our solution is that it requires no modification to most peer-to-peer peers; it is mostly controlled by the bootstrap server and incurs negligible no additional CPU or communication overhead.

Let *master overlay* be the peer-to-peer mesh overlay receiving media from the source S . At any point in time, the bootstrap server B selects a subset of peers \mathcal{S}_{aux} in the master overlay. Each $p \in \mathcal{S}_{aux}$ becomes an especial *auxiliary source*. Auxiliary sources receive media chunks from peers in the master overlay, but do not contribute back any chunks to the master overlay. This stage is called *auxiliary source isolation*.

Auxiliary source isolation prepares network for newcomer peer, including flash crowd events. When a newcomer peer n arrives, the bootstrap server only sends to n peer lists containing auxiliary sources and other newcomer peers (that have previously connected to auxiliary sources). This effectively establishes isolated parallel overlays for newcomer peers, as new partnerships are established only among peers in the same parallel overlay and their auxiliary source. As parallel overlays are disjoint with the master overlay, the distribution efficiency in the master overlay is not affected. This stage is called *parallel overlay construction*.

After a parallel overlay is built and partnerships have stabilized, the bootstrap servers proceeds to the *parallel overlay merge* stage. In this stage, the bootstrap server starts advertising peers in the master overlay to cooperative peers in the merging parallel overlay. As peers in the parallel overlay establish partnerships with peers in the master overlay, they become part of the master overlay. We find that new partnerships established during the merging stage do not disrupt the master overlay as peers already have a filled buffer and can immediately contribute to the overlay and

participate in chunk distribution after establishing new partnerships.

Figure 3.3 illustrates the stages of our solution. In the auxiliary source isolation stage, the master overlay is robust and the video source distributes chunks into the overlay. The bootstrap server selects auxiliary sources and isolate them from the overlay. In the parallel overlay construction stage, newcomer peers establish disjoint parallel overlays and do not communicate with peers in the master overlay. Finally, in the parallel overlay merge stage, peers in the parallel overlay establish partnerships with peers in the master overlay.

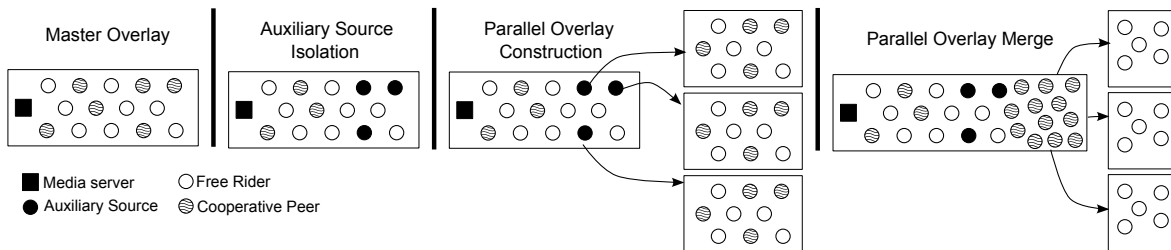


Figure 3.3: Parallel Overlays Stages

We run experiments and select six hot peers as auxiliary sources. Newcomer peers were uniformly split across auxiliary sources for constructing balanced parallel overlays. The bootstrap server waits 200 seconds for parallel overlays to stabilize, then starts merging one parallel overlay every 100 seconds. From the beginning of the experiment, the flash crowd event happens after 350 seconds, the first merge happens after 550 seconds, the second merge happens after 650 seconds, and so on, until the last merge happens after 1050 seconds.

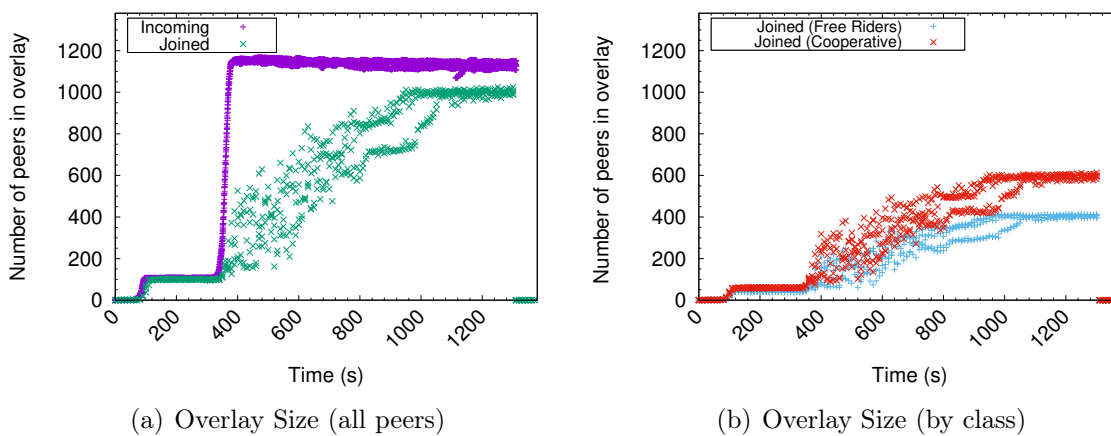


Figure 3.4: Parallel Overlays to Join Newcomer Peers

Figure 3.4 is similar to Figures 3.1. Figure 3.4(a) shows that, for the same pattern of incoming peers (blue points) as in Figure 3.1, parallel overlays can steadily and consistently distribute media to an increasing number of peers (positive trend in green points), at a pace similar to that of the 18%-batch baseline (Figure 3.1(d)) but without disruption to the master overlay. Figure 3.4(b) shows separated joined common peers and joined free rider. We observe balance in joining rate for all peer classes. The small joining gap between cooperative peers (i.e. Hot/Warm/Cold classes) and Free riders is explained because free riders are 40% versus 60% from common peers.

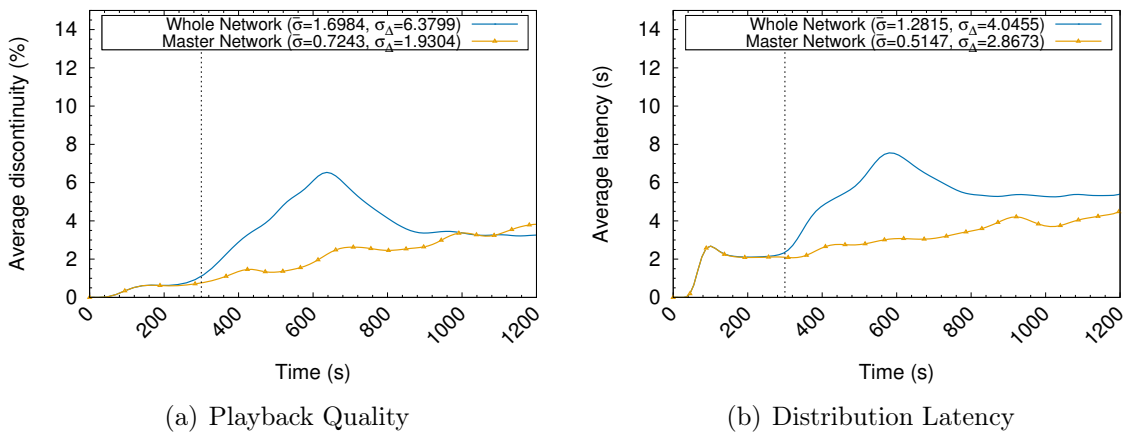


Figure 3.5: Performance of Parallel Overlays Joining

Figure 3.5 is similar to Figure 3.2. Figures 3.5(a) and 3.5(b) show that performance degradation is concentrated on parallel overlays. Even though latency and discontinuity slowly increase as the master overlay grows, this increase is small and steady.

3.5 Naïve Model: Implementing Partial Partnership Constraints

Based on results from parallel overlays in Section 3.4, we conclude that isolating peers unable to contribute from peers in the master overlay lead to more stable overlay distribution efficiency. Furthermore, parallel overlays make the bootstrap server significantly more complex (e.g., to choose the number and identities of auxiliary sources).

In this section, we employ our findings to devise a simple mechanism to increase the stability and efficiency of peer-to-peer overlays in general. Our idea is to decrease

competition among free rider and cooperative peers imposing a simple partnership constraint on peers. Therefore, by considering a peer’s classification, we define which partnerships are allowed and which partnerships are prohibited.

We limit peers to establish out-partnerships with other peers that have similar upload bandwidth. In particular, peers from the hot class (high upload bandwidth) are allowed to accept only hot, warm, and cold peers as out-partners. Peers from the warm class (average upload bandwidth) are allowed to accept any peer as out-partners. Finally, peers from the cold class (low upload bandwidth) are allowed to establish out-partnerships with free rider peers.

Figure 3.6(a) shows overlay efficiency when we restrict peer partnerships (without using parallel overlays), and Figure 3.6(b) is similar to Figure 3.6(a), but shows the number of joined peers for free riders and all cooperative peers classes combined. We assign classes to peers using the proportions in Table 3.1. We observe fast joins and robust overlay efficiency on all experiment runs. Apart from partnership restrictions, all peers are treated identically. Our partnership constraints pull cooperative peers close to the source and push free riders to the edge of the overlay. This overlay topology is desired, as it ensures peers with more upload capacity receive video chunks early and have more time to redistribute chunks to other peers in the overlay.

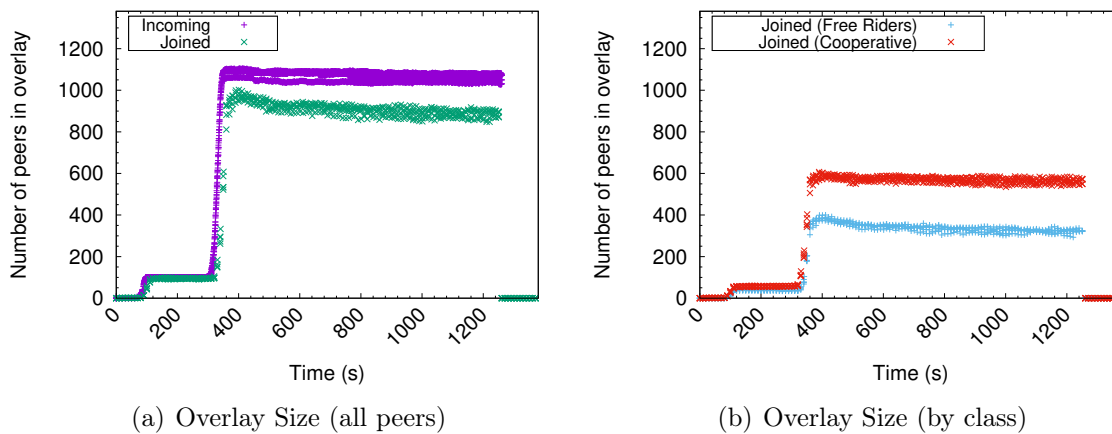


Figure 3.6: Free Rider Slice to Join Newcomer Peers

Figure 3.7(a) and 3.7(b) report improved distribution efficiency. We observe both discontinuity and chunk distribution latency remain stable and with low values for the whole overlay, including the master overlay. In particular, all peers achieve performance similar to that of the master overlay when using parallel overlays (Figure 3.5). We note that partnership constraints yield average discontinuity below 5% throughout the experiment, which previous work consider a target threshold for discontinuity (Traverso

et al., 2012).

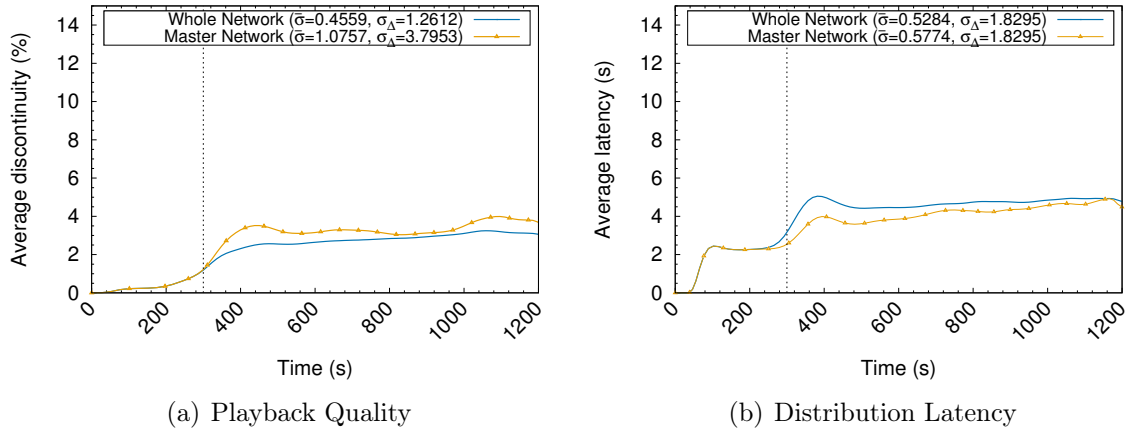


Figure 3.7: Performance of Free Rider Slice Joining

Imposing more network resource constraints: Motivated by the performance of partnership constraints, we consider a more resource constrained peer population to test the limits of our technique. We keep the same peer configurations shown in Table 3.1, but change peer proportions to 50% of free riders (+10%), 24% of cold peers (−3%), 17% of warm peers (−5%), and 9% of hot peers (−2%), shown in Table 3.2.

Table 3.2: Network Peers Configuration for 50% of Free Rider

Peer Class	Upload (Mb/s)	Proportion	$N_{in}(p)$	$N_{out}(p)$
Hot peers	4.0	09%	10	23
Warm peers	2.5	17%	10	20
Cold peers	1.5	24%	10	09
Free rider	0.0	50%	10	00

Figure 3.8(a) shows that overlay distribution efficiency remained stable throughout the experiment. Figure 3.8(b) (such as the Figure 3.6(b)) is similar to Figure 3.8(a), but shows the number of joined peers by class. We observe that partnership constraints guarantee higher stability among cooperative peers, which may serve as an incentive for peer contribution.

Figure 3.9 shows discontinuity metric and distribution latency. The smooth decline in the number of joined free riders in Figure 3.8(b) impacts the discontinuity

metric (Figures 3.9(a)). However, distribution latency line is constant after the flash crowd event, i.e., there is negligible impact on latency.

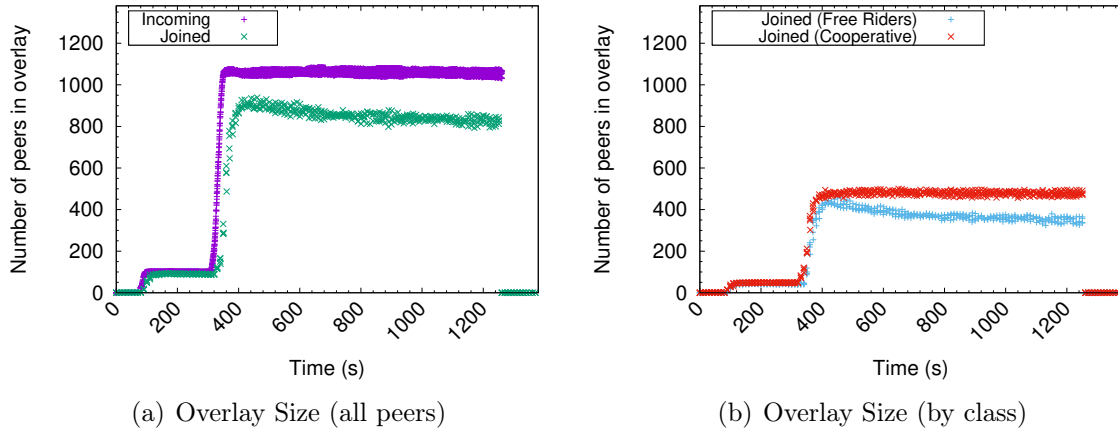


Figure 3.8: Free Rider Slice to Joining Newcomer Peers (50% of Free Riders)

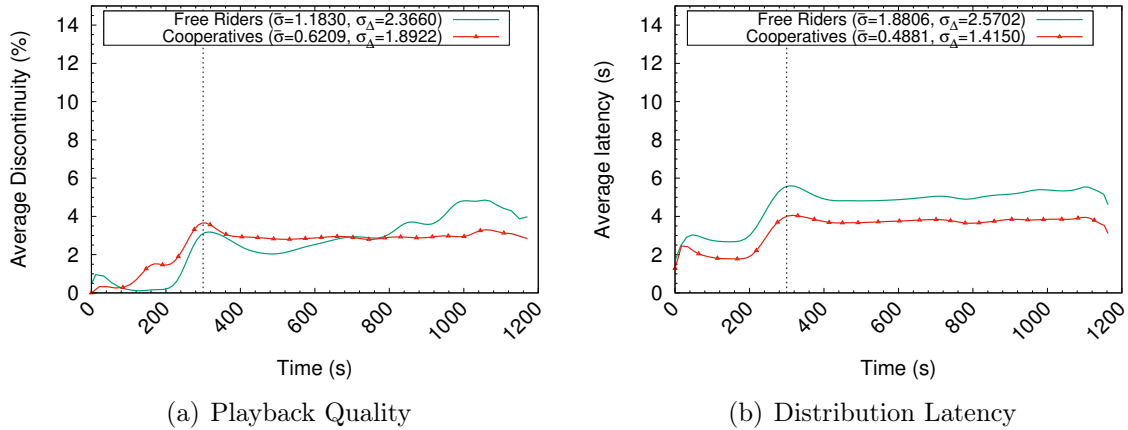


Figure 3.9: Performance of Free Rider Slice Joining (50% of Free Rider)

It is important to remember that our baseline technique failed to construct overlay when facing 40 % of free riders (Figure 3.1). It means that we define a severe peer-to-peer resource constraint. Thus, acceptable results from free rider slice when facing 50% of free riders make us realize that the partnership constraints between peers permit to construct a robust peer-to-peer overlay and should be more deeply investigated.

All models which we present in the next chapters are derived from free rider slice technique. Thus, we consider three weaknesses of free rider slice which we propose to solve: (i) peers from the hot class were configured such that to avoid relationship with

(only) free riders. Thus, peers with low upload bandwidth (from cold class) are allowed near the media server; (ii) since peers from the cold class should accept partnership only with free riders, the network may waste upload bandwidth when facing a small fraction of free riders; and (iii) since no partnership constraint is defined for peers from the warm class, free riders may experience a high level of instability. As explained in Section 2.4, Chapter 2, because cooperative peers have priority on new partnership connections, a peer p from warm class, even with a full set of out-partnership (i.e. $|\mathcal{N}_{\text{out}}(p)| = N_{\text{out}}(p)$), prefers randomly disconnect a free rider out-partner to accept a new partnership requisition from a cooperative peer.

Chapter 4

Full Partnership Constraints in Static Environments

4.1 Introduction

Chapter 3 discusses the initial idea on the partnership restriction between free riders and cooperative peers. Also, our naïve model free rider slice presents weaknesses that may compromise the peer-to-peer network stability. Based on free rider slice, we propose a new peer-to-peer overlay construction mechanism to speed up peer joining on resource-constrained overlays during flash crowd events while preserving quality of experience (QoE) for peers already in the overlay. Our technique, *peer partnership constraints (PPC)*, groups peers into classes by considering each peer’s contribution to media redistribution (upload bandwidth) and defines partnership constraints (more severe than imposed by free rider slice) for all peer classes.

PPC improves overlay distribution efficiency by bringing peers in classes with higher media redistribution scores closer to the server, as proposed in previous work (Payberah et al., 2011; Piatek et al., 2010; Lobb et al., 2009). However, unlike other works, PPC achieves this with simple mechanisms that do not increase communication overhead or implementation complexity. To alleviate the impact of uncooperativeness, PPC puts free rider peers in a special class that is pushed to the edge of the distribution overlay.

PPC speeds up peer joining rate during flash crowds by reducing competition for partnerships in the overlay. As each peer class can only establish partnerships with a few select peer classes, peers can promptly find peers to establish partnerships with. Moreover, as free rider peers join the overlay at its edge, they do not disrupt media

distribution for peers already in the overlay.

We present the technique, called *Peer Partnership Constraints* (PPC), in Section 4.2 and discuss its configuration in Section 4.3. We evaluate it in Section 4.4 and show PPC builds peer-to-peer overlays that are robust to flash crowds and efficiently distributes media even in the presence of a significant fraction of free riders.

4.2 Peer Partnership Constraint

One limitation of existing peer-to-peer systems is that peers share their out-partner slots $\mathcal{N}_{\text{out}}(p)$ with both cooperative peers and free riders. When a peer p shares out-partner slots between cooperative and free rider peers, the peer-to-peer overlay requires a mechanism to identify and push free riders to the edge of the overlay. Moreover, both free riders and cooperative peers compete for the same overlay resources. This is undesirable in general, but particularly serious during flash crowds, when a large number of free riders and cooperative peers try to join the overlay simultaneously.

To prevent competition between free riders and cooperative peers, we split each peer p 's out-partners $\mathcal{N}_{\text{out}}(p)$ into two sets: $\mathcal{N}_{\text{out}}^{\text{high}}(p)$, which contain out-partners with high upload bandwidth; and $\mathcal{N}_{\text{out}}^{\text{low}}(p)$, which contains out-partners with low upload bandwidth and free riders. Thus, a peer p accepts new cooperative peers in $\mathcal{N}_{\text{out}}^{\text{high}}(p)$ and new free-riding peers in $\mathcal{N}_{\text{out}}^{\text{low}}(p)$. $\mathcal{N}_{\text{out}}(p) = \mathcal{N}_{\text{out}}^{\text{low}}(p) \cup \mathcal{N}_{\text{out}}^{\text{high}}(p)$. The split of $\mathcal{N}_{\text{out}}(p)$ into $\mathcal{N}_{\text{out}}^{\text{low}}(p)$ and $\mathcal{N}_{\text{out}}^{\text{high}}(p)$ allows for more fine-grained peer partnership constraints; it allows not only to specify which partnerships are allowed, but how many of each partnership type are allowed.

Given the split of $\mathcal{N}_{\text{out}}(p)$, the bootstrap server can allow free riders to join the overlay immediately even during flash crowd events without incurring the risk of compromising overlay stability and efficiency. This is possible because the overlay is able to throttle the rate at which free riders and peers with low upload bandwidth join the overlay as a function of $\mathcal{N}_{\text{out}}^{\text{low}}(p)$ partnership slots available on cooperative peers p that have already joined the overlay. When 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 trying to join the overlay. PPC employs the same out-partner disconnection strategy described in Section 2.4, Chapter 2: a peer p with full $\mathcal{N}_{\text{out}}^{\text{high}}(p)$ (limited by $N_{\text{out}}^{\text{high}}(p)$) will disconnect one of its current cooperative out-partners in favor of a newcomer cooperative peer with higher upload bandwidth. The same occurs with $\mathcal{N}_{\text{out}}^{\text{low}}(p)$ set. As more cooperative peers join the overlay, they contribute additional $\mathcal{N}_{\text{out}}^{\text{high}}(p)$ and

$\mathcal{N}_{\text{out}}^{\text{low}}(p)$ slots and allow the overlay to grow further.

We need to split $\mathcal{N}_{\text{out}}(p)$ into $\mathcal{N}_{\text{out}}^{\text{high}}(p)$ and $\mathcal{N}_{\text{out}}^{\text{low}}(p)$ in a way that optimizes overlay efficiency. Based on previous work showing that pulling peers with high upload capacity close to the source improves overlay distribution efficiency (Lobb et al., 2009; Payberah et al., 2011; Piatek et al., 2010), we set $\mathcal{N}_{\text{out}}^{\text{high}}(p)$ proportional to a peer’s available upload bandwidth and $\mathcal{N}_{\text{out}}^{\text{low}}(p)$ inversely proportional to a peer’s available upload bandwidth. 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. For each free rider f , $\mathcal{N}_{\text{out}}(f) = \emptyset$ by definition.

4.3 PPC Configuration

PPC 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 PPC 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 4.1. The *contribution* column indicates whether a peer uses out-partnership slots in $\mathcal{N}_{\text{out}}^{\text{high}}(p)$ or $\mathcal{N}_{\text{out}}^{\text{low}}(p)$, defined by the maximum numbers $N_{\text{out}}^{\text{high}}(p)$ and $N_{\text{out}}^{\text{low}}(p)$, respectively.

Table 4.1: Peer Classes and Output Partnership Configurations

PEER CLASSES	SET SIZES		CONTRIBUTION	DESCRIPTION
	$N_{\text{out}}^{\text{high}}(p)$	$N_{\text{out}}^{\text{low}}(p)$		
Hot peers	> 0	$= 0$	High	High upload bandwidth
Warm peers	> 0	> 0	High	Average upload bandwidth
Cold peers	$= 0$	> 0	Low	Low upload bandwidth
Free riders	$= 0$	$= 0$	Low	Free riders

The hot class contains peers that have the highest potential for contributing to the overlay; hot peers only establish partnerships with peers from the hot and warm classes. The warm class contains peers with average potential for contributing to the overlay; warm peers can establish partnerships with peers from any class. The cold class contains peers with low potential for contributing to the overlay; cold peers can only establish partnerships with other cold peers and free riders. Free riders do not contribute to the overlay and have no out-partners.

Table 4.2: Default Experiment Configuration ($N_{\text{in}}(p) = 20, \rho \approx 1.0$)

Peer Class	Upload (Mbps)	Proportion	CONSTRUCTION STRATEGY SET SIZES				
			PPC		PPC-u		Classic
			$N_{\text{out}}^{\text{high}}(p)$	$N_{\text{out}}^{\text{low}}(p)$	$N_{\text{out}}^{\text{high}}(p)$	$N_{\text{out}}^{\text{low}}(p)$	$N_{\text{out}}(p)$
Hot	4.0	9%	46	0	26	20	46
Warm	2.5	17%	18	22	20	20	40
Cold	1.5	24%	0	38	18	20	38
Free rider	0.0	50%	0	0	0	0	0

4.4 PPC Evaluation

Table 4.2 shows the default configuration for our experiments. We consider four classes of peers with different upload bandwidths. We set the fraction of free riders to 50% of all peers to emulate a resource-constrained scenarios and stress the overlay. We distribute the remaining peers in the other three classes according to the *proportion* column. We configure the number of in- and out-partnerships so that the overlay is balanced, with $\rho = \sum_p N_{\text{in}}(p) / \sum_p N_{\text{out}}(p) \approx 1.0$ by default, but also evaluate conservative ($\rho \approx 1.5$) and aggressive overlays ($\rho \approx 0.5$). Results with the same aggregate upload bandwidth and lower fractions of free riders yield qualitatively *better* results (not shown), as the overlay has more uniformly-distributed upload bandwidth.

We compare PPC with two overlay construction strategies:

- **Classic:** Usual traditional strategy where peers have a single set of out-partners and there are not partnership constraints.
- **PPC-unconstrained:** A simplified version of PPC where out-partners are split into $\mathcal{N}_{\text{out}}^{\text{high}}(p)$ and $\mathcal{N}_{\text{out}}^{\text{low}}(p)$, but partnerships are not constrained. In other words, peers from all classes can establish partnerships with peers in any other class. The difference between Classic and PPC-unconstrained quantifies the impact of splitting $\mathcal{N}_{\text{out}}(p)$ and isolating free riders from cooperative peers. The difference between PPC-unconstrained and PPC quantifies the impact of partnership constraints.

4.4.1 Balanced Overlays ($\rho \approx 1.0$)

In this experiment we consider a balanced overlay where the total number of in-partnerships is the same as the total number of out-partnerships, as shown in Table 4.2. Figure 4.1 shows results averaged over the five runs. Figure 4.1(a) shows

the average number of incoming peers (blue points) as well as the average number of peers that successfully joined the overlay. The Classic construction strategy joins approximately 800 peers in the overlay with severe fluctuation on the number of joined peers. Both PPC-u and PPC support a steady 950 peers in the overlay, with negligible fluctuation.

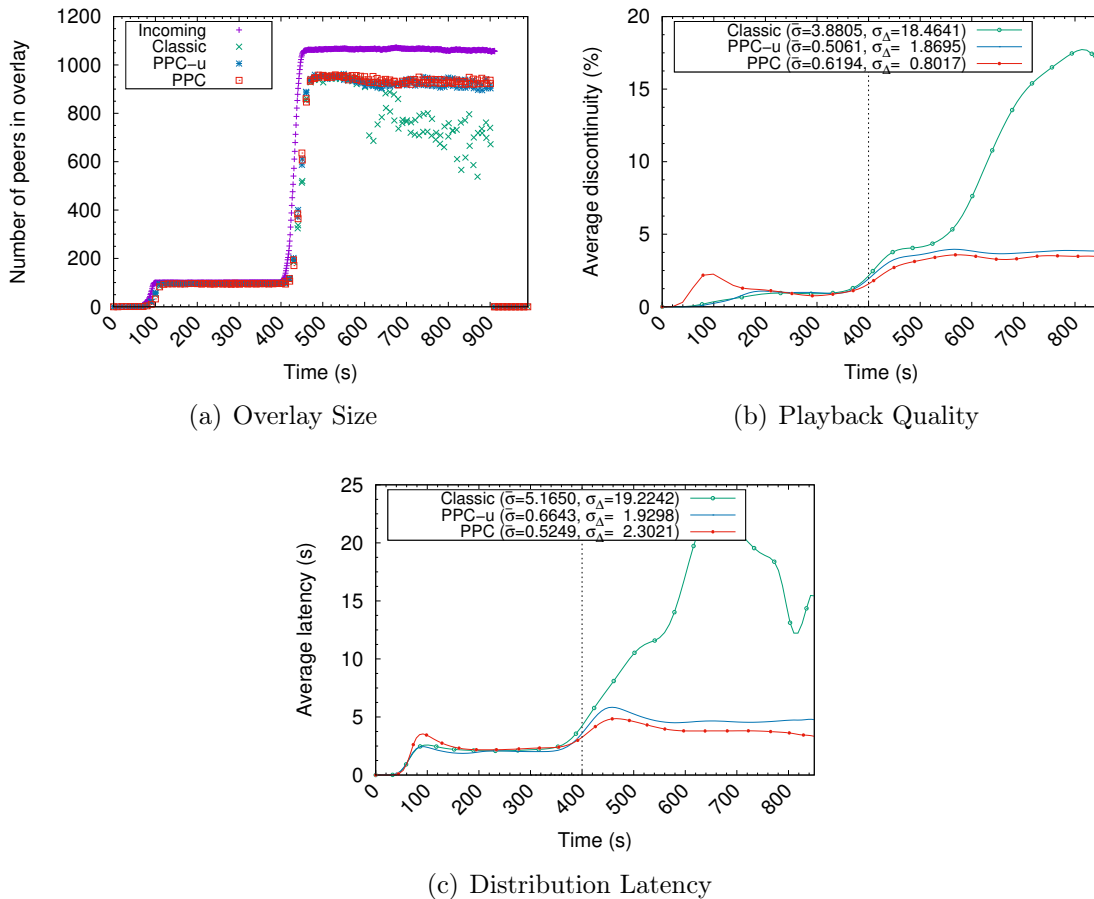


Figure 4.1: Performance of Classic and PPC for Balanced Overlays ($\rho \approx 1.0$)

Figure 4.1(b) shows the average discontinuity over all peers. We observe that PPC-u and PPC provide a discontinuity below 4% throughout the experiment, while Classic presents degraded performance after the flash crowd. Figure 4.1(c) shows the average chunk distribution latency for all peers in the overlay. Again, we observe PPC-u and PPC provide stable and reasonably low distribution latency, while Classic fails to distribute chunks in a timely fashion.

These results show that, while PPC-u and PPC achieve efficient and stable media distribution, the Classic overlay construction strategy leads to a disrupted overlay that fails at effective media distribution: fewer peers can reproduce the media, and the ones

that can do so with degraded quality of experience. The main reason for this is the competition between cooperative and uncooperative peers for out-partnerships, which disrupts the peer-to-peer overlay after the flash crowd event.

4.4.2 Conservative Overlays ($\rho \approx 1.5$)

In this experiment we consider a conservative overlay where we increase the total number of in-partnerships by 50%, setting $N_{\text{in}}(p) = 30$, $p \in P$. We keep all other parameters unchanged.

Figure 4.2(a) shows that all strategies reach approximately 950 joined peers, but the Classic strategy experiences some fluctuations. Since peers that have joined the overlay prioritize newcomer peers with higher upload bandwidth, peers with the largest potential for contributing to the overlay tend to successfully establish in-partnerships and join the overlay, which ultimately increases aggregate upload bandwidth in the overlay and improves stability for all strategies.

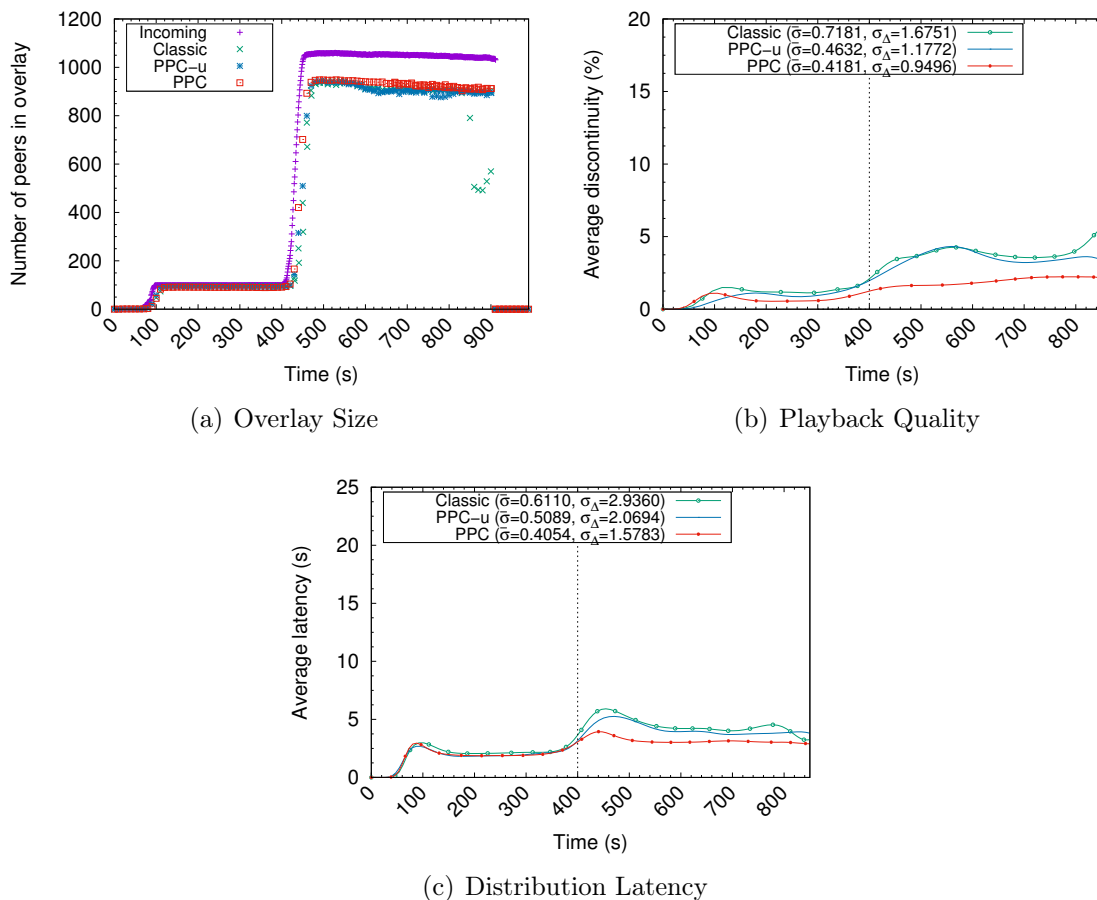


Figure 4.2: Performance of Classic and PPC for Conservative Overlays ($\rho \approx 1.5$)

Figure 4.2(b) shows the average discontinuity. We observe all strategies provide discontinuity below the acceptable threshold of 4%, but PPC presents the lowest discontinuity among evaluated strategies. Figure 4.2(c) shows the average latency, with similar results. Overall, we find a conservative overlay ($\rho \approx 1.5$) that pulls cooperative peers close to the source leads to more improved robustness to flash crowd events.

4.4.3 Aggressive Overlay ($\rho \approx 0.5$)

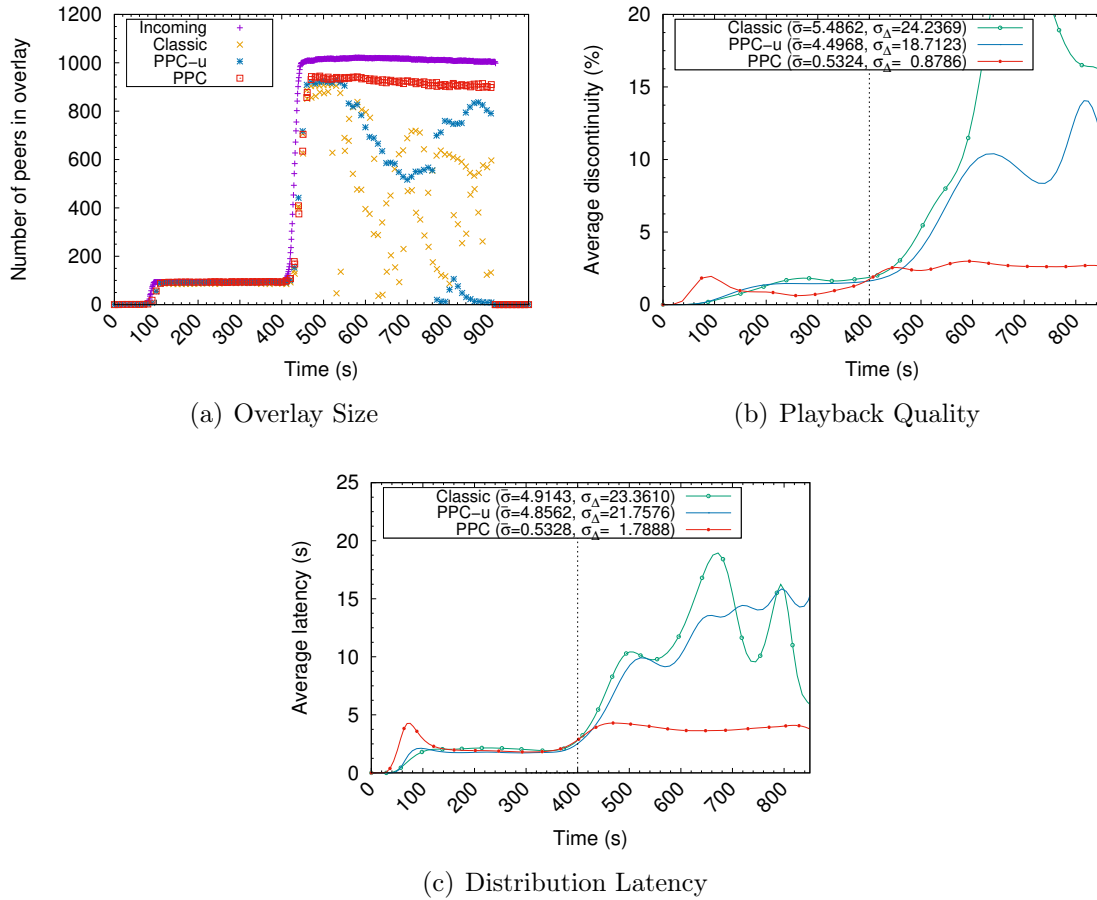
In this experiment we consider an aggressive overlay where we double the total number of out-partnerships in the overlay while keeping $N_{\text{in}}(p) = 20$, $p \in P$. Results for 50% of free riders, used in the previous sections, present very poor performance (not shown). Below, we consider an alternative configuration with more upload bandwidth, where the peer population has only 40% of free riders. The configuration is shown in Table 4.3.

Table 4.3: Default Experiment Configuration ($N_{\text{in}}(p) = 20, \rho \approx 0.5$)

Peer Class	Upload		CONSTRUCTION STRATEGY SET SIZES				
	(Mbps)	Proportion	PPC		PPC-u		Classic
			$N_{\text{out}}^{\text{high}}(p)$	$N_{\text{out}}^{\text{low}}(p)$	$N_{\text{out}}^{\text{high}}(p)$	$N_{\text{out}}^{\text{low}}(p)$	$N_{\text{out}}(p)$
Hot	4.0	11%	92	0	52	40	92
Warm	2.5	22%	36	44	40	40	80
Cold	1.5	27%	0	76	36	40	76
Free rider	0.0	40%	0	0	0	0	0

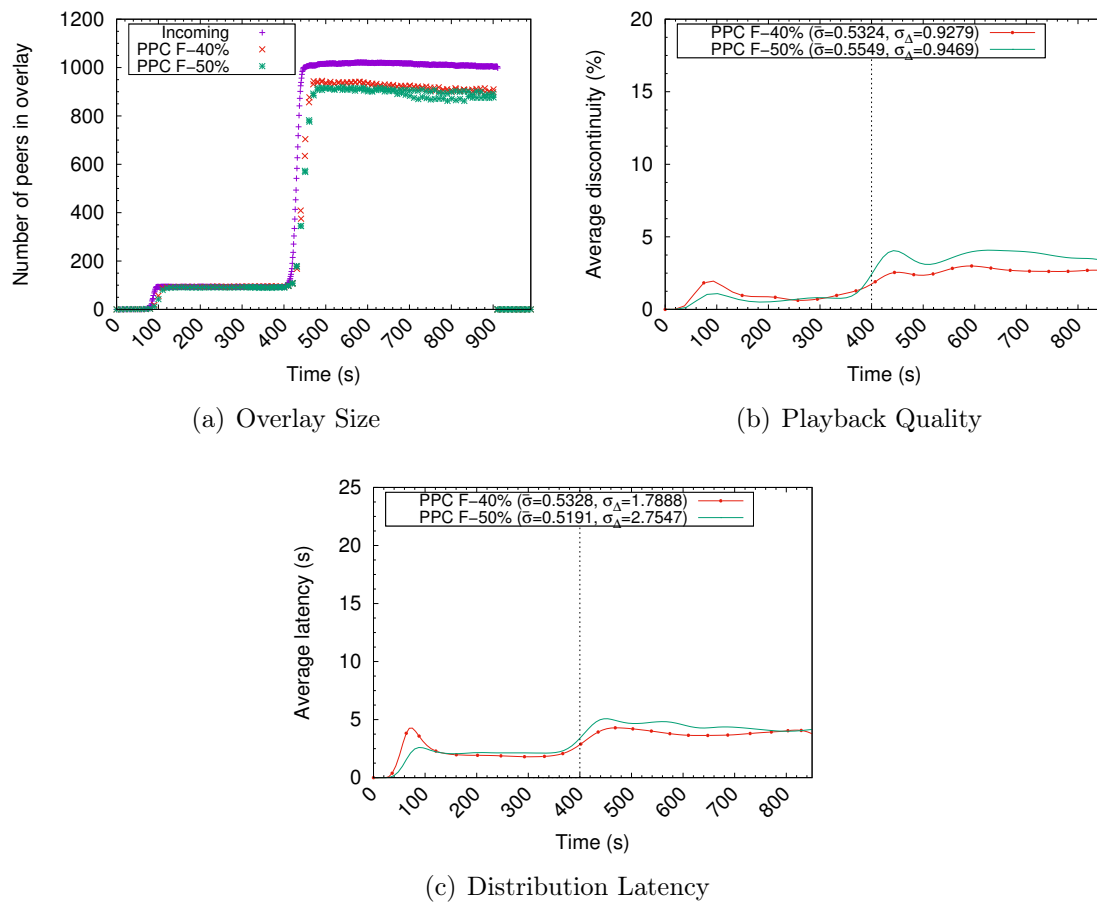
Figure 4.3(a) shows that both Classic and PPC-u build overlays that support less peers and are significantly more unstable. PPC builds a robust overlay that supports the same amount of peers as in the other configurations. In other words, PPC is less affected by peer populations, which makes peer-to-peer systems more robust to changing network conditions. As expected, Figs. 4.3(b) and 4.3(c) show that only PPC is able to keep low discontinuity and latency, while PPC-u and Classic result in very low quality of experience.

Finally, we show that PPC is able to support a higher fraction of free riders than PPC-u and Classic. Figure 4.4 shows results for PPC and aggressive overlays with 40% (same results as in Figure 4.3), as well as results for aggressive overlays (out-partnerships as in Table 4.3) with 50% of free riders, 24% of cold peers, 17% of warm peers, and 9% of hot peers. We observe that, even with 50% of free riders and decreased upload bandwidth, PPC still manages to support the same number of peers and maintain reasonable quality of experience, strengthening our point that PPC

Figure 4.3: Performance of Classic and PPC for Aggressive Overlays ($\rho \approx 0.5$)

builds more robust and effective overlays compared to PPC-u and Classic.

The success of both PPC-unconstrained and PPC on static scenario encourage us to propose a dynamic solution that combines them. In a real systems, before a peer p join in the network, it is impossible to determine the potential of p 's contribution. Thus, we propose to join newcomer peers in a warm class of PPC-unconstrained and, quickly, to consider peer chunk contribution behavior in order to classify dynamically each peer in the network in a way to construct an overlay topology based on PPC. In the next chapter we present our suggested model, that dispenses any prior information concerning peers' upload bandwidth since peers joined in the network with standard class setup and are reclassified after they start to report its chunk contribution.

Figure 4.4: Performance of PPC for Aggressive Overlays ($\rho \approx 0.5$)

Chapter 5

Full Partnership Constraints in Dynamic Environments

Chapter 4 presented Peer Partnership Constraints (PPC) and evaluated it under a static scenario, where a peer's class was fixed and known in advance. On real systems, however, it is impossible to know in advance how much a peer can contribute to the overlay. Even in cases where a peer knows and reports its available upload bandwidth (e.g., as configured in the client), the ability to contribute to chunk distribution may vary due to ISP broadband oversubscription, congestion in transit networks, high latency, or suboptimal partnerships. In practice, peer-to-peer systems need to estimate peer contribution dynamically (Guerraoui et al., 2010; Piatek et al., 2010). In this section, we present our Peer Classification and Partnership Constraints (2PC) algorithm, which classifies peers dynamically at run time and constrains partnerships to build robust peer-to-peer overlays.

5.1 Partnership Constraint Class Algorithm Concepts (2PC)

To provide all peers with the ability to contribute to the overlay, 2PC does not constrain partnerships prior to peer classification. After a peer joins the overlay, 2PC monitors how much it contributes to the overlay.¹ A peer can be promoted to a hotter class if its contribution is higher than that of its current class, or can be demoted to a colder class if its contribution is lower than that of its current class. As a special case, free riders can be immediately classified into their own class upon joining the overlay if they report their behavior to the bootstrap server.

Algorithm 1 Peer Classification algorithm

```

1: function RECLASSIFY(peers)
2:   additions[c]  $\leftarrow$  0 for each class  $c$ 
3:   active  $\leftarrow$  remove-new-peers(peers)
4:   sorted  $\leftarrow$  sort-by-decreasing-contribution(active)
5:   for peer  $p \in$  sorted do
6:     tc  $\leftarrow$  targetClass( $p$ )
7:     if tc is-hotter-than class( $p$ ) then
8:        $c \leftarrow$  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]  $\leftarrow$  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

```

Let $t_{(i)}$ be a time instant. Peer chunk contribution (i.e. peer contribution to the overlay) refer to the number of chunks that a peer p has sent to its out-partners in $[t_{(i)}, t_{(i+1)}]$ interval. We compute only sent chunks from an out-partner requests. As described in Section 2.4.2, Chapter 2, a joined peer p sends monitoring reports to server B every ten seconds containing, among others, its chunk contribution. Then, B sorts peers by its contribution.

Peer promotion or demotion might involve disconnecting existing partners. To minimize disruption to the overlay, 2PC only promotes or demotes peers one class at a time (e.g., from hot to warm, or from warm to cold). In this case, out-partners that contribute the least are disconnected first. Peer reclassification occurs periodically, once every μ seconds.

The peer routing for peer classification is shown in Algorithm 1. To give the system enough time to compute a peer's contribution with acceptable accuracy, 2PC only performs reclassification after a peer has been in the system for at least one complete reclassification period (line 3). In each period, the bootstrap server sorts peers by decreasing order of contribution to chunk distribution (line 4). The bootstrap

¹Contribution can either be reported by the peer, or cryptographic signatures can be used to track contributions in a verifiable manner Piatek et al. (2010).

servers then iterate on all peers, classifying their contribution in the current period (line 6), and checking whether the peer needs to be promoted or demoted (lines 7 and 15). To avoid degenerate overlays due to peer misclassification or skewed peer populations, 2PC limits the fraction of peers promoted and demoted into class c to a threshold $\Gamma(c)$ (line 9). Systems with many peer classes can have lower thresholds for Γ to spread peers across all classes.

Finally, the population in each class c is limited by a population limit $\mathcal{U}(c)$ (line 9). It is important determine $\rho \geq 1.0$ in the network (i.e.; $\sum_p N_{\text{in}}(p) / \sum_p N_{\text{out}}(p) \geq 1.0$) such that to avoid aggressive overlay configuration, as discussed in the Chapter 4. Note that since peers from warm class provide both $\mathcal{N}_{\text{out}}^{\text{high}}(p)$ and $\mathcal{N}_{\text{out}}^{\text{low}}(p)$ partnership sets, the overlay starts with a non aggressive configuration (i.e., the value of ρ is in the range between balanced and conservative overlay configurations, depending on the free rider fraction). Thus, we do not have imposed limit population for warm class and free rider class.

5.2 2PC Configuration

We evaluate 2PC in a peer population similar to that used to evaluate PPC in Chapter 4, shown in Table 5.1. In addition, for all 2PC experiments, we have configured $\Gamma(c) = 5\%$ (i.e., 5% of all cooperative peers) for all class c , and $\mu = 120$ seconds (i.e., 2PC iteration time). We consider the same four classes, and configure 2PC to add all new peers into the warm class. The warm class have weak out-partnership constraints: warm peers can establish out-partnerships with peers of any class. Temporarily classifying newcomer peers as warm enable them to freely establish out-partnerships while 2PC classifies them into their correct classes. As a result, we place no limit on what fraction of peers can be placed into the warm class.

Table 5.1: 2PC Experiment Configuration ($N_{\text{in}}(p) = 20, \rho \approx 1.0$)

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

Normally, a fraction below 10% of peers in the peer-to-peer live streaming are responsible for the high contributions of media delivery (Sacha et al., 2006; Oliveira,

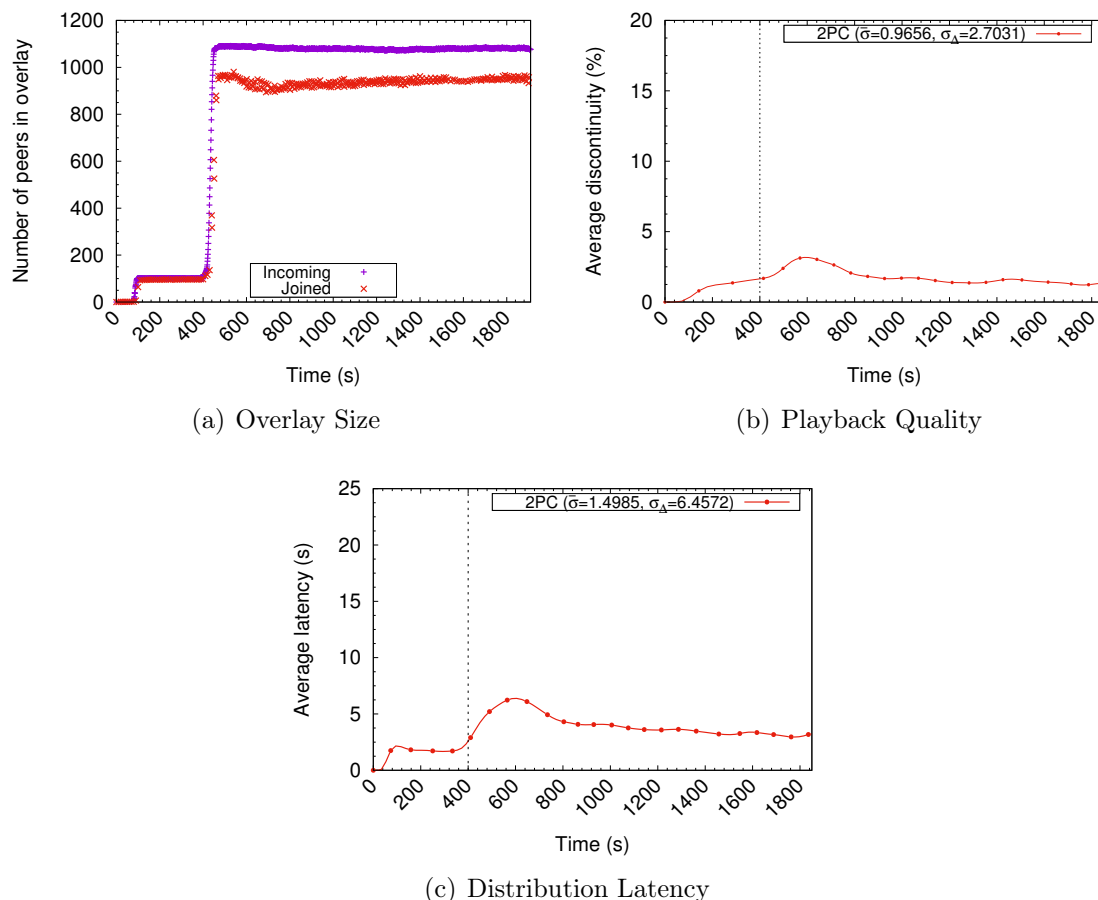


Figure 5.1: Performance of 2PC

2010). Then, we configure $\mathcal{U} = 15\%$ for Hot Class, that represents 7.5% of all peers in the network, and we configure $\mathcal{U} = 40\%$ (20% of all peers) for Cold Class in order to prepare the network to handle at least a fraction of 50% of free riders peers, Table 5.1.

5.3 Comparative Evaluation between 2PC and PCC

We evaluate 2PC, with the same peer classes configuration from Table 4.2, in order to compare performance of 2PC to the performance of plain PCC. Figure 5.1(a) shows that 2PC allows peers to join the overlay, while keeping the overlay robust during all experiments. Figures 5.1(b) and 5.1(c) show that average discontinuity and average chunk distribution latency also remain stable, with a slight increasing shortly after the flash crowd event. However, we observe that the overlay quickly recovers (results with identical configuration without 2PC are discussed in Figure 4.1, Section 4.4.1, Chapter 4).

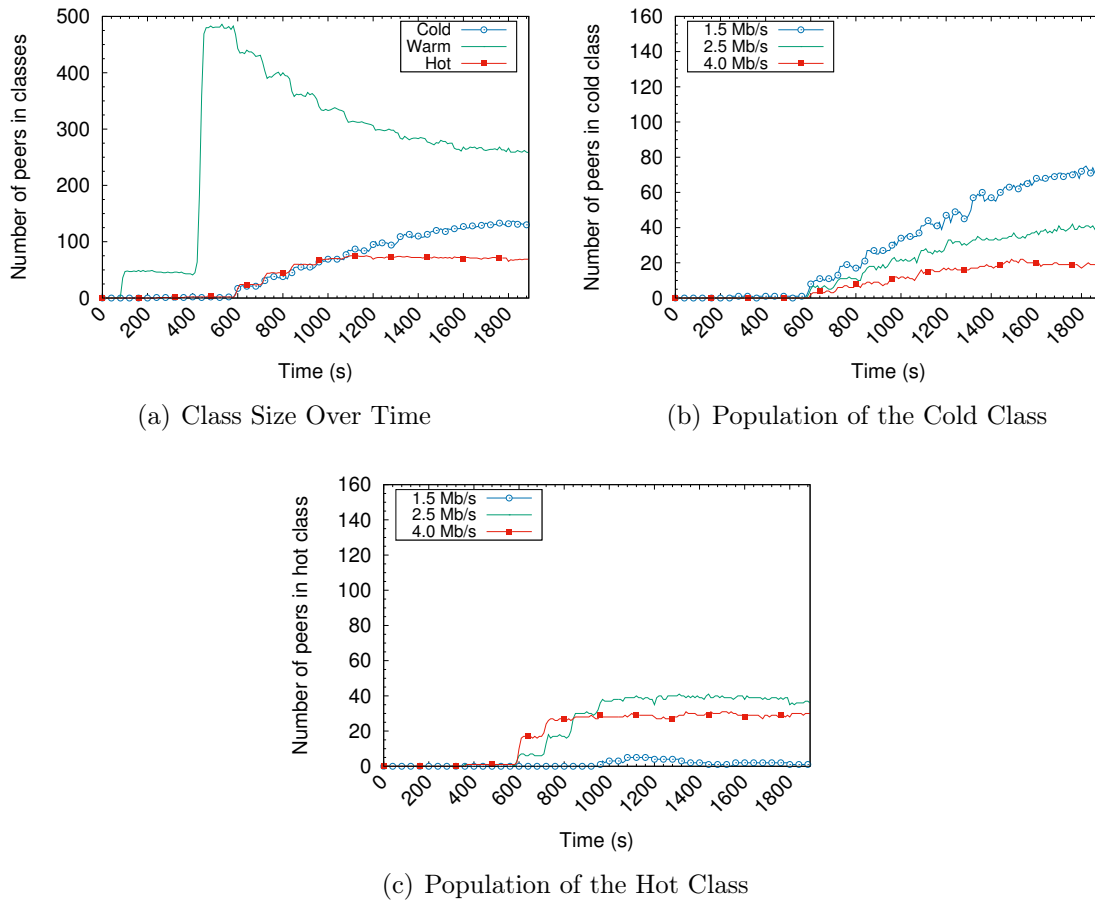


Figure 5.2: Partnership Constraint Algorithm Class Inference.

We also evaluate the classification algorithm. Figure 5.2 shows the number of peers in different classes over time. The flash crowd event starts after 400 seconds, and all peers quickly join the overlay in the warm class. The classification routine systematically promotes and demotes peers to hot/cold classes each iteration in order to adjust the overlay topology. In order to check whether 2PC is correctly promoting and demoting peers, we must take a look at Figures 5.2(b) and 5.2(c). These figures show the size of the cold and hot classes over time for different peer upload bandwidths. We observe that 2PC concentrates peers with 1.5Mbps bandwidth in the cold class, and peers with 4Mbps upload bandwidth in the hot class, as expected.

5.4 2PC Evaluation

The previous section assumed peer upload bandwidths were fixed at 1.5, 2.5, or 4Mbps depending on each peer's true class. In this section, we evaluate 2PC in a scenario

Table 5.2: 2PC Experiment Configuration (random upload bandwidths)

Peer Class	Upload (Mbps)	Class Population Limit \mathcal{U}	SET SIZES		
			2PC and b-2PC $N_{\text{out}}^{\text{high}}(p)$	$N_{\text{out}}^{\text{low}}(p)$	Classic $N_{\text{out}}(p)$
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 rider	0.0	-	0	0	0

where 50% of peers are free riders and the remaining 50% have upload bandwidth uniformly sampled between 1.0Mbps–3.5Mbps combined with $\rho \approx 1.0$ and $\rho \approx 1.5$.

We propose, in addition, the bandwidth-aware-2PC (b-2PC) as a variation of 2PC algorithm. In this case, we implement a functionality of 2PC in which the bootstrap server asks the peer’s upload bandwidth (in peer join phase) in order to classify all peer according to the suggested peer class configuration. In our experiments, we are interested in evidencing whether prior knowledge of peer’s upload bandwidth improves the 2PC algorithm performance.

Table 5.2 indicates the out-partnerships set sizes for each dynamically computed peer class. We compare both 2PC and b-2PC against the classic approach, which does not isolate out-partnerships nor employs partnership constraints.

5.4.1 2PC and b-2PC Evaluation. ($N_{\text{in}}(p) = 20$ and $\rho \approx 1.0$)

Figure 5.3 shows the averaged results for the Classic and 2PC strategies. We observe that 2PC joins a significant fraction of peers while maintaining low discontinuity and latency. The classic approach, however, cannot serve all peers and the peers that receive service often do so with poor quality of experience. Figure 5.4 compare Classic and b-2PC strategies and the conclusions are the same of the Classic and 2PC (Figure 5.3).

Figure 5.5 and Figure 5.6 show that both 2PC and b-2PC successfully classifies peers with more bandwidth into the hot class, and vice versa. Comparing Figure 5.5(a) and Figure 5.6(a) we see peer join considering the warm class (i.e. 2PC) or considering peer’s upload bandwidth (i.e. b-2PC) when occurs the flash crowd event. At 1700 seconds (near the ending of the experiments), the distribution of peer per class are the same in 2PC and b-2PC strategies.

When compared with static scenario for PPC in Section 4.4.1, Chapter 4, we conclude that results of both dynamic scenarios 2PC and b-2PC are similar to results from PPC.

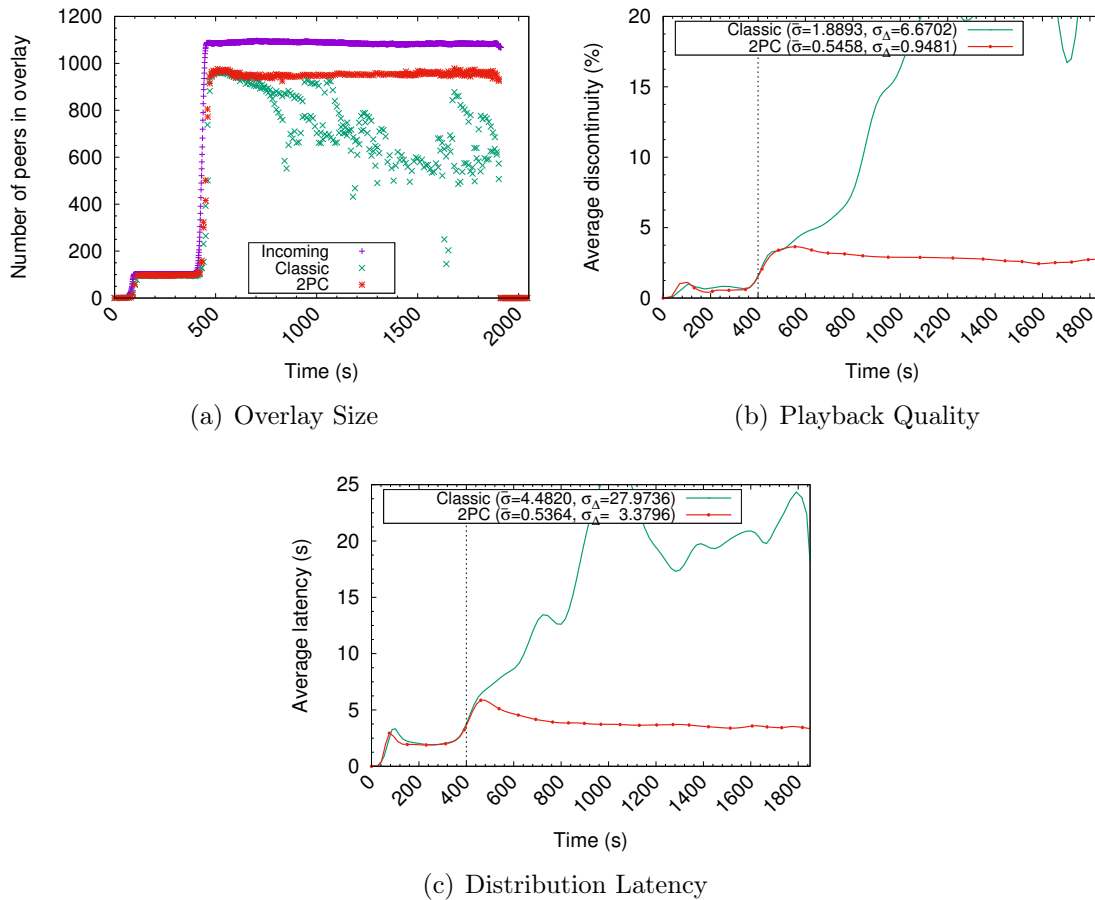


Figure 5.3: Performance of Classic and 2PC. Join Phase [Warm Class] ($\rho \approx 1.0$)

5.4.2 2PC and b-2PC Evaluation. ($N_{in}(p) = 30$ and $\rho \approx 1.5$)

In this section, we raise N_{in} parameter of peers from 20 to 30 in order to establish $\rho \approx 1.5$. Besides this, all configuration were the same as before for Section 5.4.1. We are interested in understanding whether 2PC and b-2PC can preserve the results of static PPC strategy with $\rho \approx 1.5$ presented in Section 4.4.2, Chapter 4.

We have to consider that high value of N_{in} parameter can promote competition in partnership between peers. It happens because a peer p wants to fill in its $\mathcal{N}_{out}(p)$. Thereby, peers that offer less chunk contribution are candidate to lose partnership for peers that offer high chunk contribution in the network. We remember that lower chunk contribution peers are important since they support free ride peers. So, instability caused by these peers' disconnections may cause network instability in the whole network, considering 50% of free rider peers.

Figure 5.7 shows result for 2PC and Figure 5.8 shows results for b-2PC strategies. Although discontinuity and latency metrics for b-2PC (Figure 5.8(b) and Figure 5.8(c))

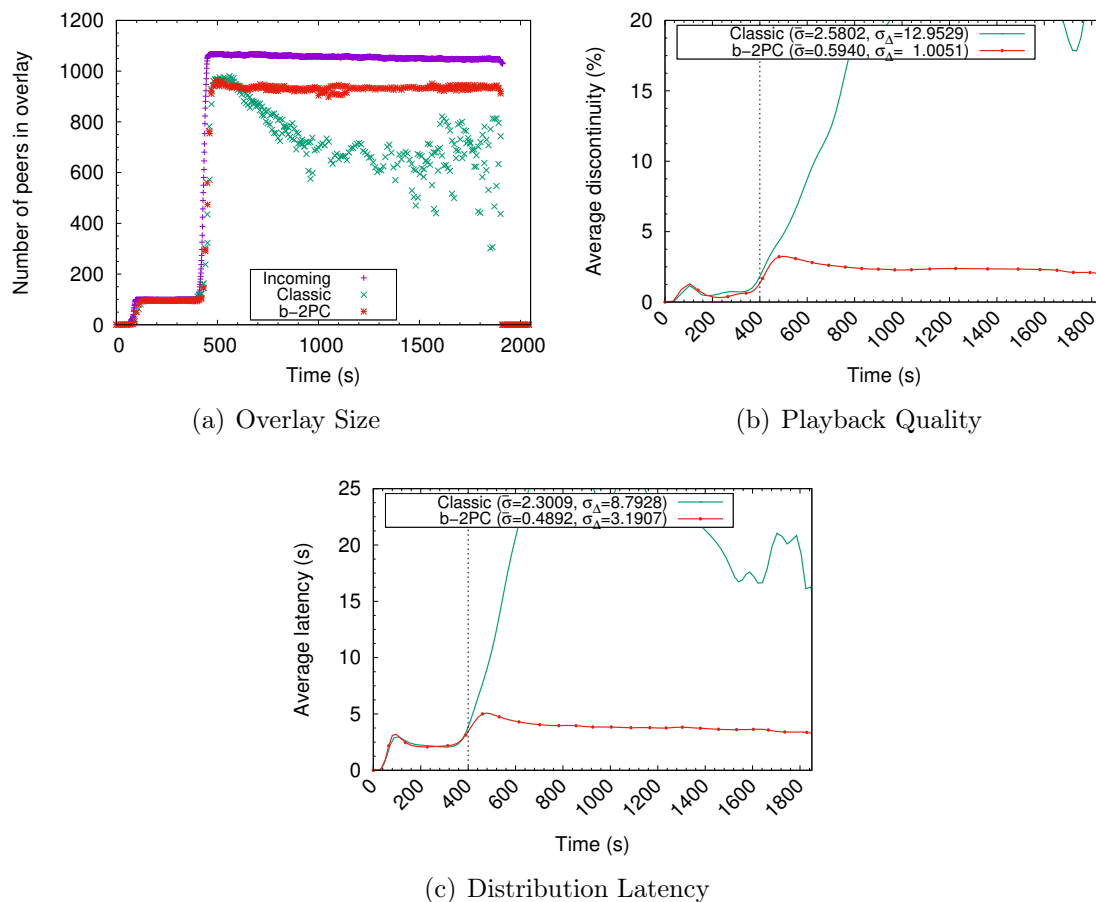


Figure 5.4: Performance of Classic and b-2PC. Join Phase [Suggested Class] ($\rho \approx 1.0$)

are relatively below the same metrics for 2PC (Figure 5.7(b) and Figure 5.7(c)), performance of b-2PC metrics are not significantly, since both strategies present acceptable levels of its metrics. The Classic strategy does not ensure the network robustness on facing the high resource constraints imposed.

As before, there is no significant difference on results of 2PC and b-2PC, as shown on Figures 5.9 and 5.10. Comparing results across this section and the last one, 2PC and b-2PC present the same results on different scenarios.

5.5 2PC Discussion

Figures comparing Classic and 2PC strategies (such as Figures 5.3 and 5.7) and Figures comparing Classic and b-2PC strategies (such as Figures 5.4 and 5.8) show an important consideration about Classic strategies. In the first case, both 2PC and Classic strategies join all peers in the same class (i.e. warm class) while in the second

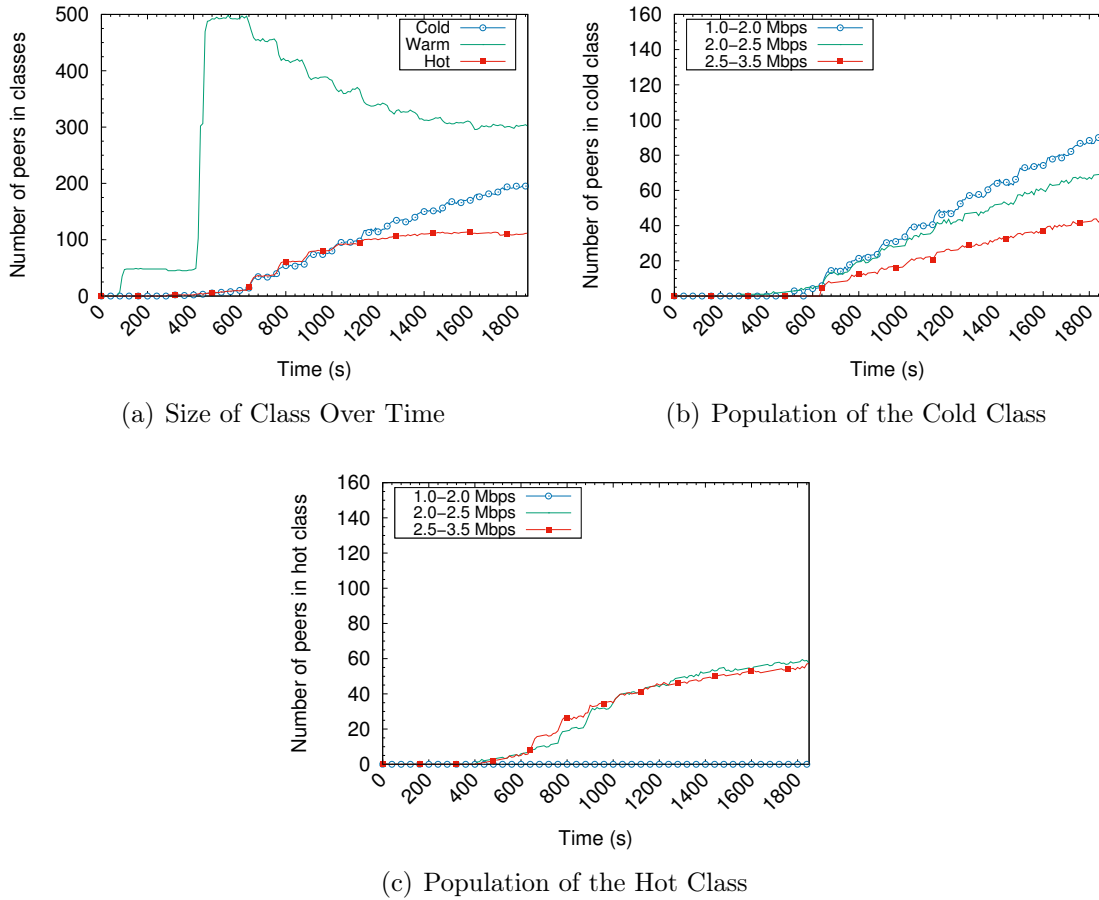


Figure 5.5: 2PC Class Inference. Join Phase [Warm Class] ($\rho \approx 1.0$)

case, both b-2PC and Classic strategies join each peer in a suggested class. Thus, comparing Classic strategies, in the second case, Classic strategy employs the same out-partner disconnection strategy described in Section 2.4, Chapter 2, which cannot occur in the first case. However, even with a more dynamic topology organization, all results for classic strategies are the same.

As occurred in the both Sections 5.4.1 and 5.4.2, when we compare 2PC strategies with the static scenarios for PPC in Section 4.4.2, Chapter 4, we conclude that results of both dynamic scenarios 2PC and b-2PC are similar to results from PPC. Thus, since results of PPC-u (a variation of PPC) for $\rho \approx 0.5$ were unaccepted (Section 4.4.3), we do not experiment 2PC algorithm for $\rho \approx 0.5$ configuration.

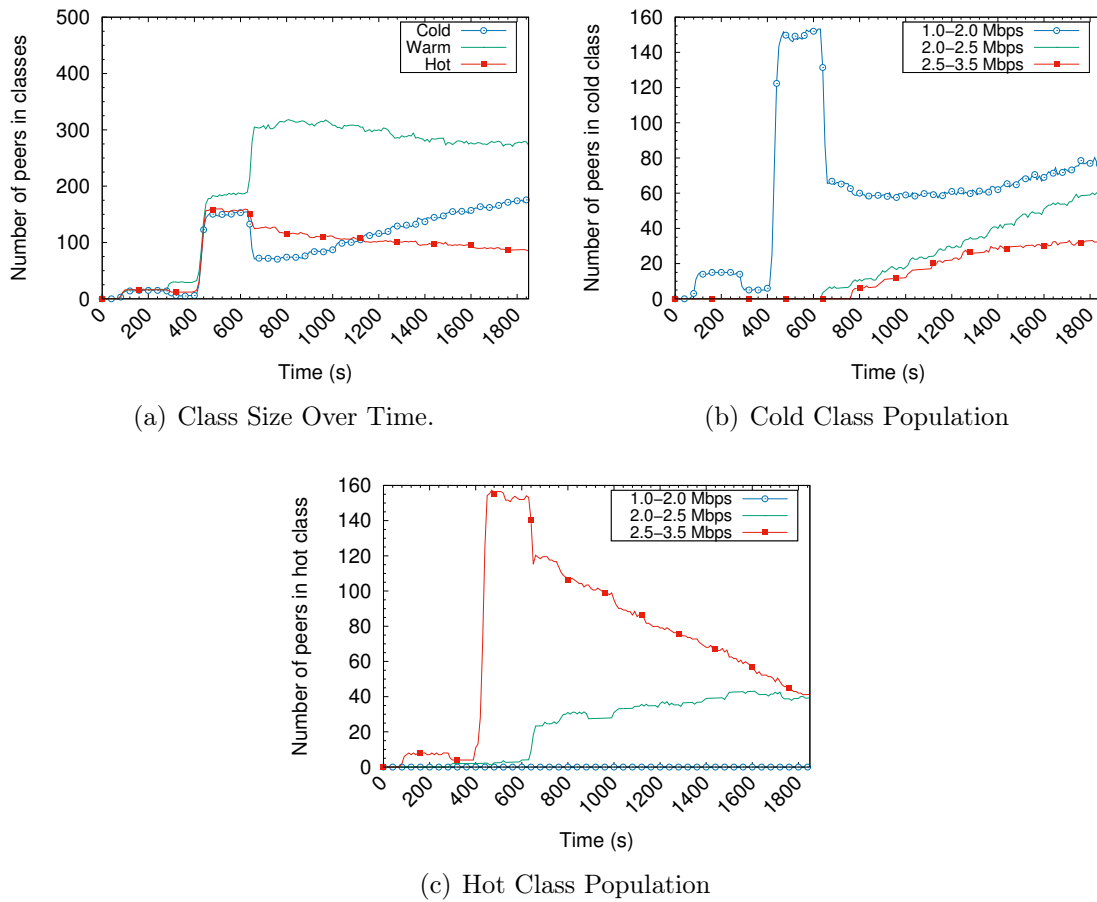


Figure 5.6: b-2PC Class Inference. Join Phase [Suggested Class] ($\rho \approx 1.0$)

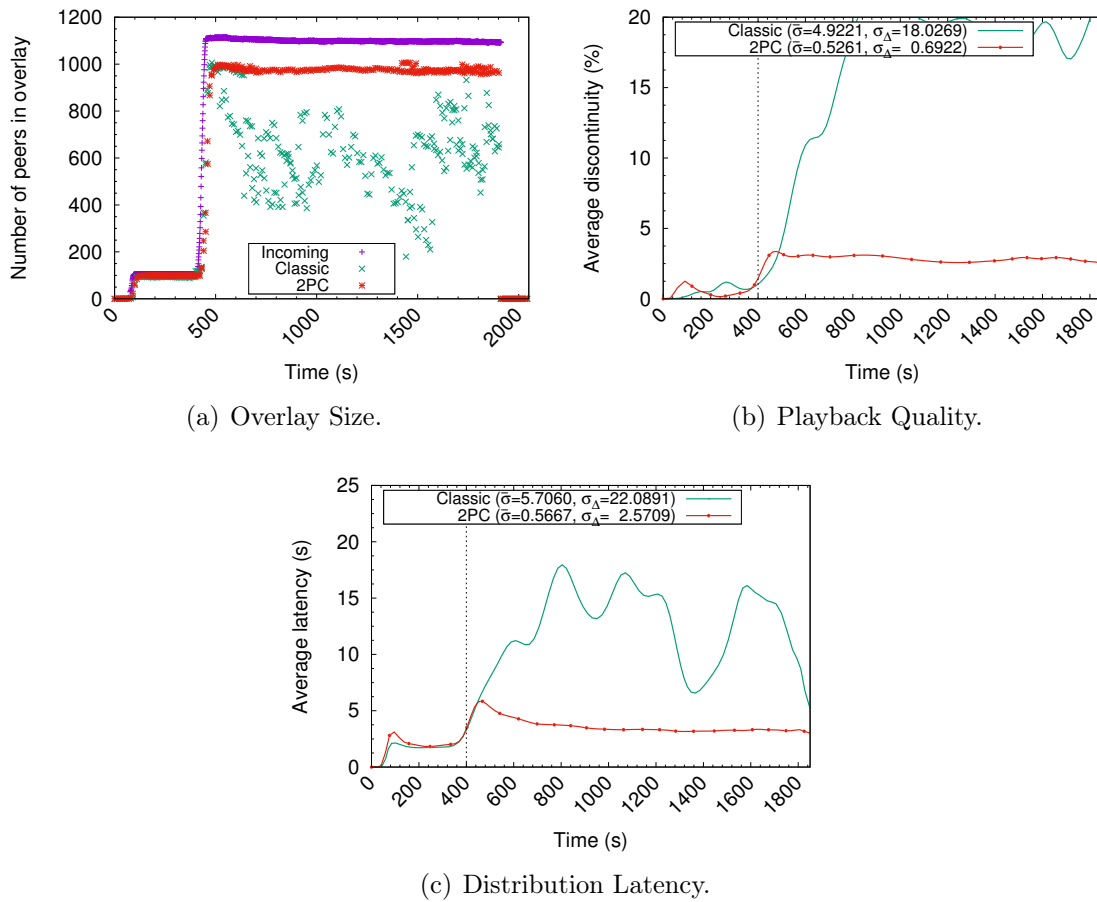


Figure 5.7: Performance of Classic and 2PC. Join Phase [Warm Class] ($\rho \approx 1.5$)

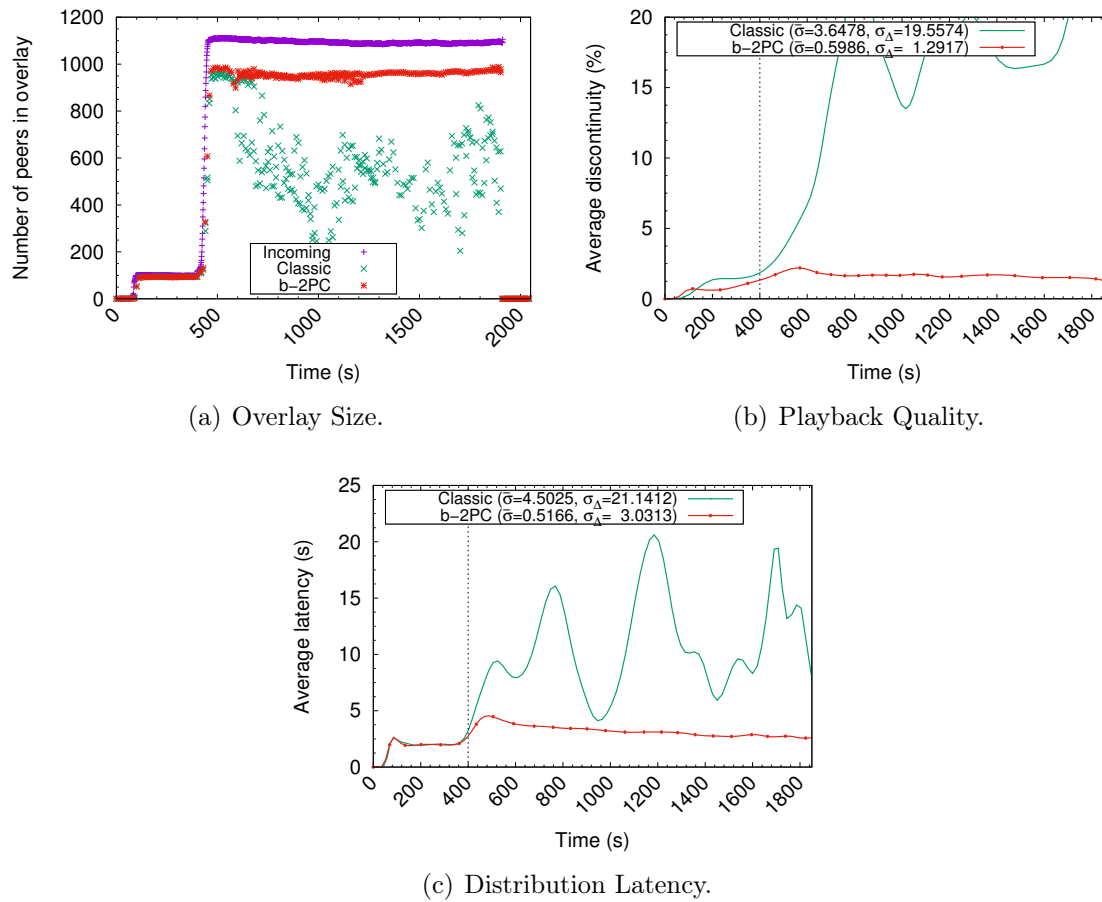
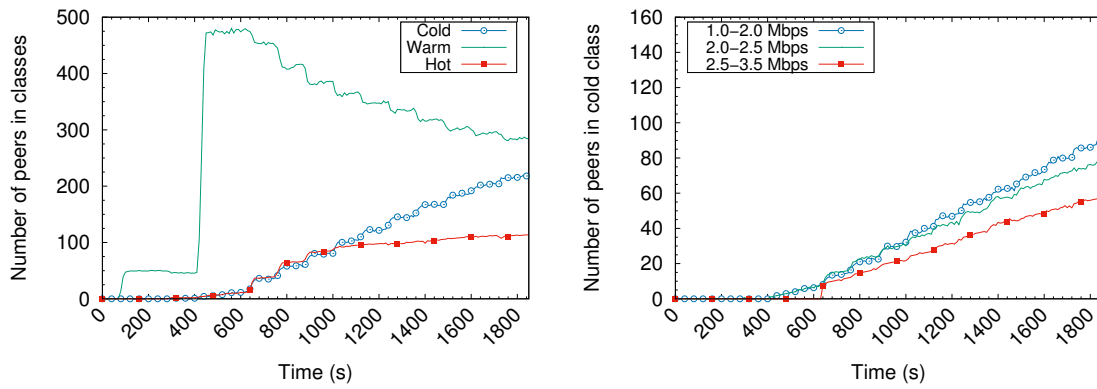
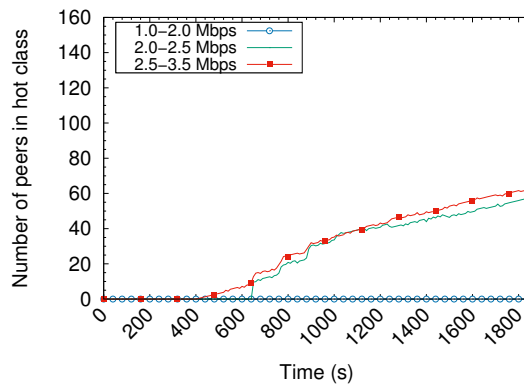


Figure 5.8: Performance of Classic and b-2PC. Join Phase [Suggested Class] ($\rho \approx 1.5$)



(a) Size of Class Over Time.

(b) Cold Class Population.



(c) Hot Class Population.

Figure 5.9: 2PC Class Inference. Join Phase [Warm Class] ($\rho \approx 1.5$)

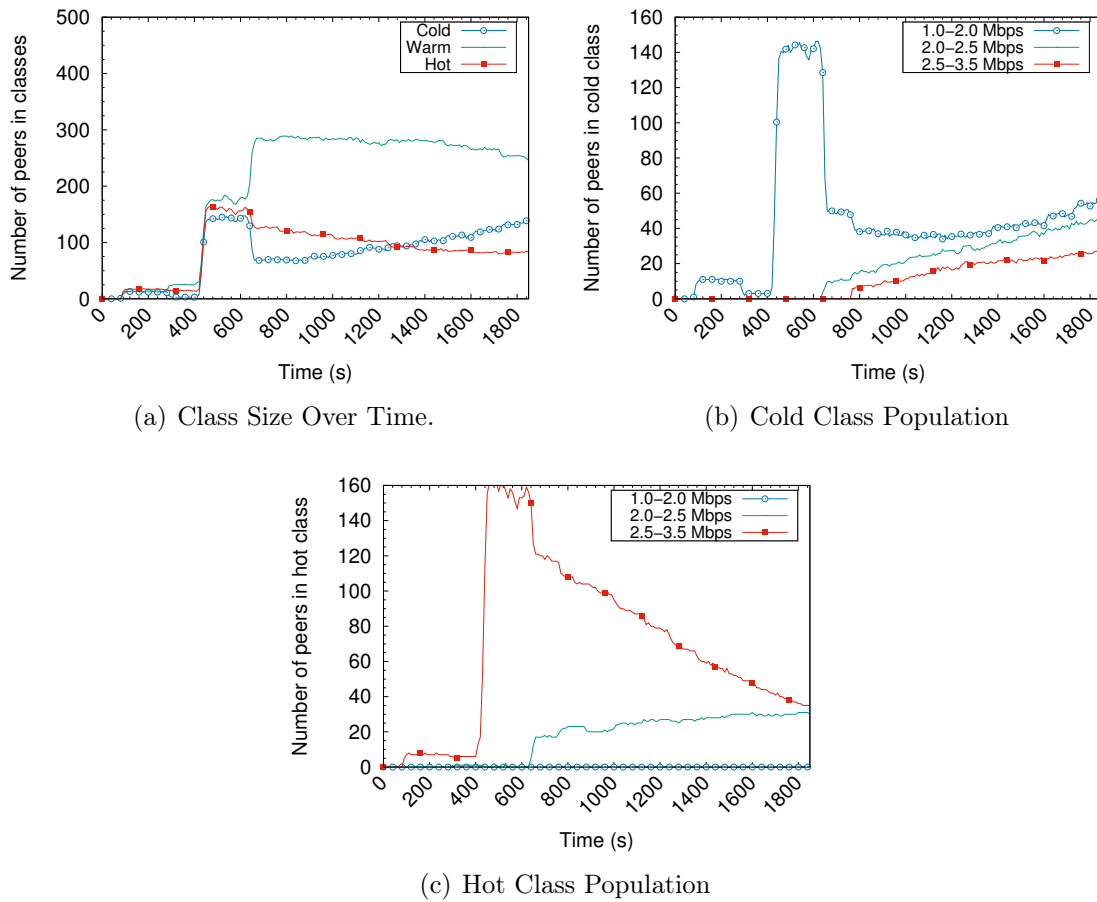


Figure 5.10: b-2PC Class Inference. Join Phase [Suggested Class] ($\rho \approx 1.5$)

Chapter 6

Related Work

Although large content providers such as Twitch, Google, Facebook, and Netflix have achieved great success using the classic client-server content distribution model (Tanenbaum and Steen, 2006), these companies deploy extensive content distribution infrastructure to support high upload bandwidth requirements for content distribution (see, e.g., (Jain et al., 2013; Schlinker et al., 2017; Yap et al., 2017; Adhikari et al., 2012)).

Compared to client-server content distribution, peer-to-peer content distribution allows media streaming without reliance on the content provider’s upload bandwidth. In peer-to-peer systems, peers establish partnerships in a distributed way (e.g., peer-to-peer mesh-based), or a centralized way (e.g., peer-to-peer tree-based) and form an overlay over the physical network to exchange content pieces. In addition, some peer-to-peer services such as media generate and peer joining are, usually, provided by centralized servers.

In this work, we have invested in a centralized approach to propose techniques such that avoid the overload of peer-to-peer control messages and moreover, considering that the extra costs to support few peer-to-peer centralized functionality are significantly smaller when compared to servers investment by several companies that employ the client-server approach.

We describe, in this chapter, issues of peer-to-peer systems related to the goals of this work, such as partnership management, chunk scheduling, overlay construction and maintaining, selfish behavior of peers, and flash crowd events. Media contribution incentive mechanisms and neighborhood filtering are several times combined in order to construct and maintain robust overlays.

6.1 Flash Crowds

According to Chen et al. (2014), when a flash crowd occurs, the sudden arrival of numerous peers may starve the upload capacity of the system, hurt its quality of service, and even cause system collapse. To mitigate flash crowd effects without applying advanced flash crowd handle techniques, streaming systems such as COOLSTREAMING or PPLIVE rely on a large number of dedicated servers or the use of CDNs Wu et al. (2012); Liu et al. (2012). Dedicated servers and CDNs increase the systems cost and do not ensure that an unforeseen flash crowd's event can be absorbed.

Several studies have proposed understanding and resolving the flash crowd event in live peer-to-peer (Chen et al., 2011; Chung and Lin, 2011; Jia et al., 2016; Li et al., 2008a). The majority of the research about the impact of flash crowds on peer-to-peer streaming systems do not focus on changing the overlay construction mechanisms (Liu et al., 2012, 2009). For example, Liu et al. propose a technique that increases overall peer join rate and distribution efficiency under flash crowds by batching joins to preserve network stability (Liu et al., 2009); Liu et al, in other work, keeping the batch joining technique, show that one important overlay constraint is the number of partnerships a peer maintains, but without investigating how to maintain the partnerships (Liu et al., 2012);

On the other hand, TOPT (Rückert et al., 2015) focus on preparing the system with a hybrid overlay to mitigate flash crowd negative effects. In order to determine whether a number of new peers represents a flash crowd event (and whether to apply a strategy to absorb the negative effects of the flash crowd), (Chen et al., 2014) provides a comprehensive study on the performance of peer-to-peer live streaming systems under flash crowds, including the mathematical model of the relationship between flash crowd size and system recovery time. Solutions proposed in this work are not a flash crowd handling technique, and are complementary to the techniques above. However, our experiments show that 2PC constructs robust overlay topologies that are effective at absorbing flash crowd events.

6.2 Partnership Management, Overlay Construction and Selfish Behavior

There are several approaches in order to construct peer-to-peer overlays (Lua et al., 2005a; WaysiAlTuhafi, 2013), but the widely used are tree-based (Castro et al., 2003; Mol et al., 2006; Payberah et al., 2010a,b) and mesh-based overlays (Li et al., 2008b;

Frey et al., 2010; Fortuna et al., 2010).

According to Magharei and Rejaie (2007), based-tree topologies establish partnerships between peers such that form a tree structures and use push-based content delivery. In this case, based-tree topology is more sensitive to peer churn, motivating solutions that construct robust tree topologies to face peer churn, e.g., (Tran et al., 2003; Castro et al., 2003). On the other hand, mesh-based overlay is more robust with respect to peer churn and usually implements pull-based content delivery. Unlike tree-based peer-to-peer systems, peers in mesh overlays often need some strategies that allow them to choose promising partnerships. In this work, we adopted pull-based mesh overlay.

A large body of work has been dedicated to building and evaluating algorithms and mechanisms for efficient peer-to-peer live streaming. Neighborhood filtering strategies for overlay construction are important to achieve media distribution efficiency and the literature presents several such strategies Payberah et al. (2011); Lobb et al. (2009); Ren et al. (2008); Li et al. (2008b); Liao et al. (2006); Magharei and Rejaie (2009); Silva et al. (2008); Wu et al. (2012). Contributions cover techniques to optimize peer-to-peer overlay topologies (Simoni et al., 2014; Payberah et al., 2012; Fortuna et al., 2010; Felber and Biersack, 2005), schedule chunk requests and transmissions (Vlavianos et al., 2006; Bonald et al., 2008; Liu, 2007; Silva et al., 2008; Picconi and Massoulié, 2008; Massoulie et al., 2007; Fortuna et al., 2010; Zhao et al., 2009), and topologies according to overlay and network conditions (Wichtlhuber et al., 2014). Supported by these strategies, low upload bandwidth peers can manage a large number of partners without cracking the video transmission.

Traverso et al. (2015) compares several neighborhood filtering strategies ranging from randomized approaches to very sophisticated strategies considering peer bandwidth and peer physical location. However, in most of these works, e.g., (Traverso et al., 2015) and (Lobb et al., 2009), as peers do not restrict the composition of the list of *output partnerships* (out-partners), i.e., the set of partner peers to which each peer redistributes media. According to Zhong et al. (2010), a large number of partners increases the overhead of control message exchanges between peers. In addition, 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 its out-partners. On the other hand, GLive (Payberah et al., 2011), a distributed market-based solution that builds a mesh overlay for peer-to-peer live streaming, imposes fixed peer's partnership limits. GLive employs the gossip-generated Gradient overlay network (Sacha et al., 2010, 2006) to enable peers to sample neighbors with similar upload bandwidth (that define its market model) in order to organize a desired

mesh overlay topology. Differently from such approaches, in this work we propose a centralized overlay construction algorithm based on random neighborhood filtering, allowing peers to forward content to all partners.

Additionally, many overlay construction strategies propose pushing free riders to the edge of the overlay to improve distribution efficiency. Ullah et al. (2013) propose an autonomous management framework that (i) enables peers to learn user behavior and organize themselves to improve streaming quality, and (ii) controls the topology of push-based systems through a stabilization process that continuously pushes unstable peers towards the leaves of the tree. This strategy is complementary and could be used in conjunction with our approach to push free riders to the edge of the overlay. Incentive mechanisms that aim to improve peer contribution identify and punish selfish behavior while improving quality of service for cooperative peers (Cohen, 2003; Piatek et al., 2010; Guerraoui et al., 2010; Locher et al., 2009; Gonçalves et al., 2014; Tan and Jarvis, 2008; Meulpolder et al., 2009; Mol et al., 2006).

A well-known incentive mechanism used in peer-to-peer file sharing is BitTorrent's tit-for-tat (Cohen, 2003). Unfortunately, tit-for-tat is inadequate for live streaming. Tit-for-tat forces balanced pairwise data exchanges, which are too restrictive to enable live streaming (Piatek et al., 2010). Some works have attempted to use tit-for-tat as a feature, where uncooperative peers experience degraded quality of service, even at overprovisioned scenarios (Locher et al., 2009). As examples of incentive mechanism for live streaming, Contracts (Piatek et al., 2010) identifies uncooperative peers auditing cryptographic receipts of chunk transfers while LiFTinG (Guerraoui et al., 2010) identifies uncooperative and malicious peers based on their partnerships.

Similar to our work, Oliveira et al. evaluate and show that peer-to-peer live streaming can support a large number of free riders (Oliveira et al., 2013b). However, they impose no bandwidth constraints and consider contributing peers have surplus bandwidth. Further, Oliveira et al. (2013b) define two types of free rider: *conscious*; and *oblivious*. Conscious free riders inform their partners that they are unwilling or unable to upload data. As a consequence, no peer ever wastes time and bandwidth sending requests to them. On the other hand, oblivious free riders do not inform their partners that they are unwilling or unable to upload data. This behavior may happen if the software does not make provisions for free riders or due to malicious intent. Oblivious free riders request chunks as normal and advertise the chunks they have, but never upload data (i.e., never answer requests), which degrades the system performance. Oblivious free riders negative effects can be mitigate by a simple mechanism called *Simple Unanswered Request Eliminator* (SURE), a modification to the chunk request

scheduler that avoids wasting time and bandwidth by listing the peers that have no good chunk answer history. Using the SURE, experiments with oblivious free riders showed results quantitatively similar to experiments with conscious free riders.

In this thesis, we configure all free riders as conscious and SURE is active on all experiments. Different from several works that propose contribution incentive, we developed a strategy in a way to handle a large fraction of free riders by establishing simple partnership constraints among peers that eliminate resource competitions between free riders and cooperative peers. Thus, we show that a peer-to-peer system can support a large fraction of free riders while imposing bandwidth constraints.

We set up strategies such as: random neighborhood filtering; first in first out for chunk scheduling when sending chunks; and SURE in order to avoid silent peers when requesting chunks. Although several works have proposed sophisticated neighborhood filtering and different chunk scheduling, we consider that our proposed solutions are complementary of current works and may be combined with them in order to achieve peer-to-peer systems more robust with low complexity.

Chapter 7

Conclusion and Future Works

Peer-to-peer live streaming systems, and their algorithms for constructing and maintaining the network overlay, often face issues of high playback latency and buffer underflows. In particular, as peers establish more partnerships to increase opportunity for exchanging media, the overhead of control messages increases and sophisticated neighborhood filtering techniques are required to maintain media distribution efficiency.

In this work, we study whether imposing partnership constraint between peers can improve peer-to-peer performance and stability without the need of sophisticated neighborhood filtering techniques, without increasing extra overhead of control messages and without imposing sophisticated peer cooperation incentive.

We present, first, the *Parallel Overlay Technique*. Parallel overlay allows constructing new overlay topologies where free riders peers are contained when a flash crowd event occurs. Thus, newcomer free riders are unable to start partnerships with cooperative peers near the server media. Parallel overlay technique shows an improvement of peer-to-peer stability promoted by free riders isolation. In addition to the complexity of implementation, concepts of parallel network are not deeply understood in order to make this technique a commercial alternative, but we believe that parallel overlay can be successfully applied to solve other issues related to the peer-to-peer systems.

In order to understand the effects of splitting cooperative peers and free riders into distinct classes of peers, we propose *Free Rider Slice*, as a simple strategy to impose constraint of peer partnerships, and its specialization *Peer Partnership Constraints (PPC)*. Free rider slice and PPC work by grouping peers with similar contributions into classes and define constraints for which classes can establish partnerships with each other class, but free rider slices differ from PPC, since PPC applies partnership constraints on all classes.

We evaluated PPC with a significant fraction of free riders, limited upload bandwidth, and different overlay characteristics. We have not imposed a sophisticated contribution incentive mechanism nor a neighborhood filtering technique. Free Rider Slice and PPC limit the number of allowed partnerships for peers, which allow cooperative peers to respond to their partner by sending media in FIFO order according to its upload bandwidth. Our results show that PPC builds robust and effective overlays that support more peers and achieve higher quality of experience compared to traditional overlay construction strategy.

Both Free Rider Slice and PPC are static solutions, where peers are classified once during peer join phase (i.e. a peer asks to join the media transmission and it is classified by the bootstrap server only at this moment, and nevermore). Thus, based on PPC strategy, we present the Peer Classification and Partnership Constraint algorithm (2PC), a simple strategy for constructing and maintaining a peer-to-peer mesh overlay network. Using PPC strategy, 2PC classifies peers on established classes in order to eliminate competition between more cooperative and less cooperative peers. 2PC dynamically reclassifies peers according to their contributions to chunk distribution. 2PC also employs partnership constraints to force the overlay into a robust and efficient topology. The resulting peer-to-peer overlay can simultaneously join a large number of cooperative and free-riding peers without disrupting chunk distribution even during flash crowd events.

Although 2PC is not specific for handling flash crowds, however 2PC constructs robust overlay topologies that present high performance even during these challenging events. Our evaluation also considered significant fraction of free riders and limited available upload bandwidth. Moreover, 2PC is simple to implement and can be combined with many other techniques already proposed for managing peer-to-peer overlays.

As future work, we enumerate the following: (i) we propose to study the behavior of Parallel Overlay Technique, establishing criteria in order to choose its auxiliary sources without compromising the system. In addition, it is need to understand other behaviors of this technique, such as the number of auxiliary sources and the time of system execution between its stages; (ii) we suggest to study, about 2PC, how to dynamically compute the number of classes and out-partnership set sizes to further improve performance; and (iii) we propose to combine 2PC with other peer-to-peer techniques in order to achieve more performance of peer-to-peer system with low complexity.

Bibliography

- Adar, E. and Huberman, B. (2000). Free Riding on Gnutella. *First Monday*, 5(10-2).
- Adhikari, V. K., Guo, Y., Hao, F., Varvello, M., Hilt, V., Steiner, M., and Zhang, Z. L. (2012). Unreeling Netflix: Understanding and improving multi-CDN movie delivery. In *Proceedings of IEEE INFOCOM (2012)*, pages 1620–1628. ISSN 0743-166X.
- Bonald, T., Massoulié, L., Mathieu, F., Perino, D., and Twigg, A. (2008). Epidemic live streaming: optimal performance trade-offs. In *SIGMETRICS '08 - ACM SIGMETRICS international conference on Measurement and modeling of computer systems - 2008*, pages 325–336, Annapolis, United States. ACM.
- Castro, M., Druschel, P., Kermarrec, A.-M., Nandi, A., Rowstron, A., and Singh, A. (2003). SplitStream: High-Bandwidth Multicast in Cooperative Environments. In *Proceedings of the Symposium on Operating Systems Principles*, pages 298–313. ACM.
- Chen, Y., Zhang, B., and Chen, C. (2011). Modeling and Performance Analysis of P2P Live Streaming Systems under Flash Crowds. In *Proceedings of the International Conference on Communications*, pages 1–5. IEEE.
- Chen, Y., Zhang, B., Chen, C., and Chiu, D. M. (2014). Performance Modeling and Evaluation of Peer-to-Peer Live Streaming Systems Under Flash Crowds. *IEEE/ACM Transactions on Networking*, 22(4):1106–1120.
- Chung, T. Y. and Lin, O. (2011). A Batch Join Scheme for Flash Crowd Reduction in IPTV Systems. In *Parallel and Distributed Systems (ICPADS), 2011 IEEE 17th International Conference on*, pages 823–828. IEEE.
- Cigno, R. L., Russo, A., and Carra, D. (2008). On some fundamental properties of P2P push/pull protocols. In *2008 Second International Conference on Communications and Electronics*, pages 67–73. ISSN .

- Cisco (2014). Forecast and Methodology, 2013–2018 - Cisco Visual Networking Index. Technical report.
- Cisco (2017). Cisco Visual Networking Index:Forecast and Methodology, 2016–2021. Technical report.
- Cohen, B. (2003). Incentives Build Robustness in BitTorrent. In *Workshop on Economics of Peer-to-Peer Systems*.
- ConsumerLab, E. (2014). Changing Consumer Needs are Creating a New Media Landscape - TV and Media 2014. Technical report.
- Cui, Y., Dai, L., and Xue, Y. (2007). Optimizing P2P Streaming Throughput Under Peer Churning. In *Proceedings of the Global Telecommunications Conference*, pages 231–235. IEEE.
- Felber, P. and Biersack, E. (2005). Cooperative Content Distribution: Scalability through Self-Organization. In *Self-star Properties in Complex Information Systems (2005)*, pages 343–357. Springer.
- Fortuna, R., Leonardi, E., Mellia, M., Meo, M., and Traverso, S. (2010). QoE in Pull based P2P-TV Systems: Overlay Topology Design Tradeoffs. In *Proceedings of IEEE P2P (2010)*, pages 1–10. IEEE.
- Frey, D., Guerraoui, R., Kermarrec, A. M., and Monod, M. (2010). Boosting Gossip for Live Streaming. In *2010 IEEE Tenth International Conference on Peer-to-Peer Computing (P2P)*, pages 1–10. ISSN 2161-3559.
- GIGAOM (2009a). CNN: Inauguration P2P Stream a Success, Despite Backlash. The industry leader in emerging technology research. <https://gigaom.com/2009/02/07/cnn-inauguration-p2p-stream-a-success-despite-backlash/>. Accessed October 05, 2017.
- GIGAOM (2009b). The Obama Inauguration Live Stream Stats. The industry leader in emerging technology research. <https://gigaom.com/2009/01/20/the-obama-inauguration-live-stream-stats/>. Accessed October 05, 2017.
- Gonçalves, G., Cunha, I., Vieira, A., and Almeida, J. (2014). Predicting the Level of Cooperation in a Peer-to-Peer Live Streaming Application. *Multimedia Systems*, pages 1–20.

- Guerraoui, R., Huguenin, K., Kermarrec, A.-M., Monod, M., and Prusty, S. (2010). LiFTinG: Lightweight Freerider-Tracking in Gossip. In *ACM/IFIP/USENIX International Conference on Middleware*.
- Hardin, G. (1968). The Tragedy of the Commons. *Science*, 162(3859):1243–1248.
- Huang, Q., Jin, H., Liu, K., Liao, X., and Tu, X. (2007). Anysee2: an auto load balance P2P live streaming system with hybrid architecture. In *Proceedings of the International Conference on Scalable Information Systems*, pages 1–2. ICST.
- Jain, S., Kumar, A., Mandal, S., Ong, J., Poutievski, L., Singh, A., Venkata, S., Wanderer, J., Zhou, J., Zhu, M., Zolla, J., Hözl, U., Stuart, S., and Vahdat, A. (2013). B4: Experience with a Globally-deployed Software Defined Wan. *SIGCOMM Comput. Commun. Rev.*, 43(4):3–14. ISSN 0146-4833.
- Jia, S., Zhang, R., Ma, Y., Zhong, L., and Xu, C. (2016). Modeling and optimization of bandwidth supply performance for cloud-assisted video systems under flash crowd. *China Communications*, 13(9):151–162.
- Karakaya, M., Korpeoglu, I., and Ulusoy, O. (2009). Free Riding in Peer-to-Peer Networks. *IEEE Internet Computing*, 13(2):92–98.
- Krishnan, R., Smith, M., Tang, Z., and Telang, R. (2004). The Impact of Free-Riding on Peer-to-Peer Networks. In *Annual Hawaii International Conference on System Sciences*.
- Li, B., Keung, G., Xie, S., Liu, F., Sun, Y., and Yin, H. (2008a). An Empirical Study of Flash Crowd Dynamics in a P2P-Based Live Video Streaming System. In *Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE*, pages 1–5. ISSN 1930-529X.
- Li, B., Xie, S., Qu, Y., Keung, G. Y., Lin, C., Liu, J., and Zhang, X. (2008b). Inside the New Coolstreaming: Principles, Measurements and Performance Implications. In *IEEE INFOCOM 2008 - The 27th Conference on Computer Communications*. ISSN 0743-166X.
- Liao, X., Jin, H., Liu, Y., Ni, L. M., and Deng, D. (2006). Anysee: Peer-to-peer live streaming. In *Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications*, pages 1–10. ISSN 0743-166X.

- Liu, F., Li, B., Zhong, L., Li, B., Jin, H., and Liao, X. (2012). Flash Crowd in P2P Live Streaming Systems: Fundamental Characteristics and Design Implications. *Transactions on Parallel and Distributed Systems*, 23(7):1227–1239.
- Liu, F., Li, B., Zhong, L., Li, B., and Niu, D. (2009). How P2P Live Streaming Systems Scale over Time Under a Flash Crowd? In *Proceedings of the International Conference on Peer-to-peer Systems*, pages 5–5. USENIX.
- Liu, J., Rao, S., Li, B., and Zhang, H. (2008a). Opportunities and Challenges of Peer-to-Peer Internet Video Broadcast. *Proceedings of the IEEE*, 96(1):11–24.
- Liu, Y. (2007). On the Minimum Delay Peer-to-peer Video Streaming: How Realtime Can It Be? In *Proceedings of the 15th ACM International Conference on Multimedia*, MM '07, pages 127–136, New York, NY, USA. ACM.
- Liu, Y., Guo, Y., and Liang, C. (2008b). A Survey on Peer-to-Peer Video Streaming Systems. *Peer-to-Peer Networking and Applications*, 1(1):18–28.
- Lobb, R. J., Couto da Silva, A. P., Leonardi, E., Mellia, M., and 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*, NOSSDAV '09, pages 31–36, New York, NY, USA. ACM.
- Locher, T., Meier, R., Schmid, S., and Wattenhofer, R. (2007). *Push-to-Pull Peer-to-Peer Live Streaming*, pages 388–402. Springer Berlin Heidelberg, Berlin, Heidelberg.
- Locher, T., Meier, R., Wattenhofer, R., and Schmid, S. (2009). Robust Live Media Streaming in Swarms. In *Proceedings of the International Workshop on Network and Operating Systems Support for Digital Audio and Video*, pages 121–126. ACM.
- Lua, E. K., Crowcroft, J., Pias, M., Sharma, R., and Lim, S. (2005a). A Survey and Comparison of Peer-to-peer Overlay Network Schemes. *Commun. Surveys Tuts.*, 7(2):72–93. ISSN 1553-877X.
- Lua, K., Crowcroft, J., Pias, M., Sharma, R., and Lim, S. (2005b). A Survey and Comparison of Peer-to-Peer Overlay Network Schemes. *Communications Surveys Tutorials*, 7(2):72–93.
- Ma, R. T. B., Lee, S. C. M., Lui, J. C. S., and Yau, D. K. Y. (2006). Incentive and Service Differentiation in P2P Networks: A Game Theoretic Approach. *IEEE/ACM Transactions on Networking*, 14(5):978–991. ISSN 1063-6692.

- Magharei, N. and Rejaie, R. (2007). Mesh or Multiple-Tree: A Comparative Study of Live P2P Streaming Approaches. In *Proceedings of the International Conference on Computer Communications*, pages 1424–1432. IEEE.
- Magharei, N. and Rejaie, R. (2009). PRIME: Peer-to-peer Receiver-driven Mesh-based Streaming. *IEEE/ACM Trans. Netw.*, 17(4):1052–1065. ISSN 1063-6692.
- Massoulie, L., Twigg, A., Gkantsidis, C., and Rodriguez, P. (2007). Randomized Decentralized Broadcasting Algorithms. In *IEEE INFOCOM 2007 - 26th IEEE International Conference on Computer Communications*, pages 1073–1081.
- Meulpolder, M., Meester, L., and Epema, D. (2012). The Problem of Upload Competition in Peer-to-Peer Systems With Incentive Mechanisms. *Concurrency and Computation: Practice and Experience*, 25(7):899–917.
- Meulpolder, M., Pouwelse, J., Epema, D., and Sips, H. (2009). BarterCast: A Practical Approach to Prevent Lazy Freeriding in P2P Networks.
- Miguel, E. C., Carvalho, F. C., Morgan, B., Junior, M. C., Cunha, Í., and Campos, S. V. (2016). Join Rate Improvements in P2P Live Streaming Based on Topological Aspects During Flash Crowds. In *SBRC 2017*.
- Miguel, E. C., Cunha, Í., Silva, C. M., Carvalho, F., and Campos, S. (2017). Resource-constrained P2P Streaming Overlay Construction for Efficient Joining Under Flash Crowds. In *22nd IEEE Symposium on Computers and Communication (ISCC 2017) (ISCC 2017)*, Heraklion, Greece.
- Mol, J. J. D., Epema, D. H. J., and Sips, H. J. (2006). The orchard algorithm: P2p multicasting without free-riding. In *Sixth IEEE International Conference on Peer-to-Peer Computing (P2P'06)*, pages 275–282. ISSN 2161-3559.
- Moltchanov, D. (2011). Service Quality in P2P Streaming Systems. *Computer Science Review*, 5(4):319 – 340.
- Oliveira, J., Viana, R., Vieira, A. B., Rocha, M., and Campos, S. (2013a). TVPP: A Research Oriented P2P Live Streaming System. In *Salão de Ferramentas. Anais do Simpósio Brasileiro de Redes de Computadores*.
- Oliveira, J. a., Cunha, Í., Miguel, E. C., Rocha, M. V., Vieira, A. B., and Campos, S. V. (2013b). Can Peer-to-Peer Live Streaming Systems Coexist With Free Riders? In *IEEE P2P 2013 Proceedings*, pages 1–5. IEEE.

- Oliveira, J. F. D. A. (2010). *Super Nós em Sistema P2P de Distribuição de Mídia ao Vivo*. Master's thesis, Universidade Federal de Minas Gerais.
- Payberah, A., Dowling, J., Rahimian, F., and H., S. (2012). Distributed Optimization of P2P Live Streaming Overlays. *Special Issue on Extreme Distributed Systems: From Large Scale to Complexity*, 94(8):621–647.
- Payberah, A. H., Dowling, J., and Haridi, S. (2011). GLive: The Gradient Overlay as a Market Maker for Mesh-Based P2P Live Streaming. In *Parallel and Distributed Computing (ISPDC), 2011 10th International Symposium on*, pages 153–162.
- Payberah, A. H., Dowling, J., Rahimian, F., and Haridi, S. (2010a). *gradientTv: Market-Based P2P Live Media Streaming on the Gradient Overlay*, pages 212–225. Springer Berlin Heidelberg, Berlin, Heidelberg.
- Payberah, A. H., Rahimian, F., Haridi, S., and Dowling, J. (2010b). Sepidar: Incentivized Market-Based P2P Live-Streaming on the Gradient Overlay Network. In *2010 IEEE International Symposium on Multimedia*, pages 1–8.
- Pianese, F., Keller, J., and Biersack, E. W. (2006). PULSE, a Flexible P2P Live Streaming System. In *Proceedings of the Global Internet Symposium in conjunction with International Conference on Computer Communications*. IEEE.
- Piatek, M., Krishnamurthy, A., Venkataramani, A., Yang, R., Zhang, D., and Jaffe, A. (2010). Contracts: Practical Contribution Incentives for P2P Live Streaming. In *Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation*, NSDI'10, pages 6–6, Berkeley, CA, USA. USENIX Association.
- Piatek, M., Madhyastha, H. V., John, J. P., Krishnamurthy, A., and Anderson, T. (2009). Pitfalls for ISP-friendly P2P Design. In *Proceedings of the Workshop on Hot Topics in Networks*. ACM.
- Picconi, F. and Massoulié, L. (2008). Is There a Future for Mesh-Based live Video Streaming? In *2008 Eighth International Conference on Peer-to-Peer Computing*, pages 289–298.
- PlanetLab (2013). An Open Platform for Developing, Deploying, and Accessing Planetary-Scale Services.
<http://www.planet-lab.org/>. Accessed June 2, 2013.
- PPlive (2013). PPlive. <http://www.pplive.com>. Accessed June 2, 2013.

- PPS (2017). PPStreaming. <http://www.pps.tv/>. Accessed September 3, 2017.
- Ren, D., Li, Y. T., and Chan, S. H. (2008). On Reducing Mesh Delay for Peer-to-Peer Live Streaming. In *IEEE INFOCOM 2008 - The 27th Conference on Computer Communications*.
- Rückert, J., Richerzhagen, B., Lidanski, E., Steinmetz, R., and Hausheer, D. (2015). TOPT: Supporting Flash Frowd Events in Hybrid Overlay-Based Live Streaming. In *2015 IFIP Networking Conference (IFIP Networking)*, pages 1–9.
- Sacha, J., Biskupski, B., Dahlem, D., Cunningham, R., Meier, R., Dowling, J., and Haahr, M. (2010). Decentralising a service-oriented architecture. *Peer-to-Peer Networking and Applications*, 3(4):323–350.
- Sacha, J., Dowling, J., Cunningham, R., and Meier, R. (2006). Discovery of Stable Peers in a Self-organising Peer-to-peer Gradient Topology. In *Proceedings of the 6th IFIP WG 6.1 International Conference on Distributed Applications and Interoperable Systems*, DAIS’06, pages 70–83, Berlin, Heidelberg. Springer-Verlag.
- Sandvine (2014). Fall 2014 Global Internet Phenomena Report. Technical report.
- Schlinker, B., Kim, H., Cui, T., Katz-Bassett, E., Madhyastha, H. V., Cunha, I., Quinn, J., Hasan, S., Lapukhov, P., and Zeng, H. (2017). Engineering Egress with Edge Fabric: Steering Oceans of Content to the World. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication (2017)*, SIGCOMM ’17, pages 418–431, New York, NY, USA. ACM.
- Silva, A. P., , Leonardi, E., Mellia, M., and Meo, M. (2008). A Bandwidth-Aware Scheduling Strategy for P2P-TV Systems. In *2008 Eighth International Conference on Peer-to-Peer Computing*, pages 279–288. ISSN 2161-3559.
- Simoni, G., Roverso, R., and Montesor, A. (2014). RankSlicing: A Decentralized Protocol for Supernode Selection. In *Proceedings of IEEE P2P (2014)*.
- SopCast (2013). SopCast. <http://www.sopcast.com>. Accessed June 2, 2013.
- Tan, G. and Jarvis, S. A. (2008). A Payment-Based Incentive and Service Differentiation Scheme for Peer-to-Peer Streaming Broadcast. *IEEE Transactions on Parallel and Distributed Systems*, 19(7):940–953.
- Tanenbaum, A. S. and Steen, M. V. (2006). *Distributed Systems: Principles and Paradigms (2Nd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA. ISBN 0132392275.

- TechCrunch (2017). Trump's Inauguration Broke Live Video Streaming Records. TC:TechCrunch. <https://techcrunch.com/2017/01/23/trumps-inauguration-broke-live-video-streaming-records/>. Accessed October 05, 2017.
- Tran, D., Hua, K., and Do, T. (2003). ZIGZAG: An Efficient Peer-to-Peer Scheme for Media Streaming. In *Proceedings of the Annual Joint Conference of the Computer and Communications*, volume 2, pages 1283–1292. IEEE.
- Traverso, S., Abeni, L., Birke, R., Kiraly, C., Leonardi, E., Cigno, R. L., and Mellia, M. (2012). Experimental comparison of neighborhood filtering strategies in unstructured P2P-TV systems. In *2012 IEEE 12th International Conference on Peer-to-Peer Computing (P2P)*, pages 13–24. ISSN 2161-3559.
- Traverso, S., Abeni, L., Birke, R., Kiraly, C., Leonardi, E., Lo Cigno, R., and 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.
- TVU (2017). TVU. <http://www.tvunetworks.com/>. Accessed September 3, 2017.
- Ullah, I., Doyen, G., and Gaiti, D. (2013). Towards User-Aware Peer-to-Peer Live Video Streaming Systems. In *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*, pages 920–926. IEEE.
- UUSee (2013). UUSee Inc. <http://www.uusee.com/>. Accessed June 2, 2013.
- Venkataraman, V., Yoshida, K., and Francis, P. (2006). Chunkyspread: Heterogeneous Unstructured Tree-Based Peer-to-Peer Multicast. In *Proceedings of the International Conference*, pages 2–11. IEEE.
- Vlavianos, A., Iliofotou, M., and Faloutsos, M. (2006). BiToS: Enhancing BitTorrent for Supporting Streaming Applications. In *Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications*, pages 1–6.
- VLC (2013). VideoLAN ORGANIZATION. <http://www.videolan.org/vlc/>. Accessed June 2, 2013.
- Wang, Wenjie and Xiong, Yongqiang and Zhang, Qian and Jamin, Sugih (2006). Ripple-Stream: Safeguarding P2P Streaming Against Dos Attacks. In *Proceedings of the International Conference on Multimedia and Expo*, pages 1417–1420. IEEE.

- WaysiAlTuhafi, A. (2013). A Review on Peer-to-Peer Live Video Streaming Topology. *International Journal of Computer Applications*, 68(5):6–14.
- Wichtlhuber, M., Richerzhagen, B., Rückert, J., and Hausheer, D. (2014). TRANSIT: Supporting Transitions in Peer-to-Peer Live Video Streaming. In *2014 IFIP Networking Conference*, pages 1–9.
- WikiBooks (2017). The World of Peer-to-Peer (P2P)/Building a P2P System. WikiBooks: Open books for an open world. [https://en.wikibooks.org/wiki/The_World_of_Peer-to-Peer_\(P2P\)/Building_a_P2P_System](https://en.wikibooks.org/wiki/The_World_of_Peer-to-Peer_(P2P)/Building_a_P2P_System). Accessed September 15, 2017.
- Wu, H., Liu, J., Jiang, H., Sun, Y., Li, J., and Li, Z. (2012). Bandwidth-Aware Peer Selection for P2P Live Streaming Systems under Flash Crowds. In *Performance Computing and Communications Conference (IPCCC), 2012 IEEE 31st International*, pages 360–367. ISSN 1097-2641.
- Xiao, Z. and Ye, F. (2008). New Insights on Internet Streaming and IPTV. In *Proceedings of the International Conference on Content-based Image and video Retrieval*, pages 645–654. ACM.
- 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., and Vahdat, A. (2017). Taking the Edge off with Espresso: Scale, Reliability and Programmability for Global Internet Peering. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication (2017)*, SIGCOMM '17, pages 432–445.
- Zhao, B., Lui, J., and Chiu, D. (2009). Exploring the Optimal Chunk Selection Policy for Data-Driven P2P Streaming Systems. In *Proceedings of IEEE P2P (2009)*.
- Zheng, Q., Long, Y., Qin, T., and Yang, L. (2011). Lifetime Characteristics Measurement of a P2P Streaming System: Focusing on Snapshots of the Overlay. In *9th World Congress on Intelligent Control and Automation (WCICA), 2011*, pages 805–810.
- Zhong, L., Dai, J., Li, B., Li, B., and Jin, H. (2010). Are a Few Neighboring Peers Good Enough? In *2010 IEEE Global Telecommunications Conference GLOBECOM 2010*, pages 1–5.