THESIS

Presented on the 19 november at

Universidad de La República, UdelaR

TESIS DE DOCTORADO EN INFORMÁTICA
PEDECIBA

de

Pablo Gabriel ROMERO RODRÍGUEZ

Institute : Laboratorio de Probabilidad y Estadística - IMERL
Departamento de Investigación Operativa - INCO
University :
UNIVERSIDAD DE LA REPÚBLICA
FACULTAD DE INGENIERÍA

Title of the Thesis :

Mathematical Analysis of Scheduling Policies in Peer-to-Peer Video Streaming Networks

Commitee:

Dr. Franco	ROBLEDO	Advisor
Dr. Pablo	RODRIGUEZ-BOCCA	Advisor
Dr. Martín	VARELA RICO	Rapporteur
Dr. Bruno	TUFFIN	Rapporteur
Prof. Ing. Omar	VIERA	President
Dr. Raúl	URES	Invited
Dr. Francisco	BARAHONA	
Dr. Alejandra	BEGHELLI	
Dr. Fernando	PAGANINI	
Dr. Raúl	TEMPONE	

Reconocimientos

Los reconocimientos representan, a mi juicio, lo más importante de esta tesis, y permiten valorar el camino mucho más que la meta. Al repasar el trayecto de esta tesis, me sorprendo de tantas personas que me acompañaron, promovieron y motivaron, desde los primeros pasos. Con mis queridos tutores, Franco Robledo y Pablo Rodríguez, he contado en las buenas y las malas. Me han enseñado de los pequeños detalles hasta lo realmente significativo; desde los medios y herramientas hasta la pasión y el amor por la investigación. Gracias a ellos ya no tengo trabajo, sino placer. Siempre voy a extrañar las reuniones compartidas con Darío Padula y Elisa Bertinat. De ellas nacieron ideas reflejadas en el Capítulo 5 de este escrito, pero muy por encima, el valor de la amistad y la solidaridad. Claudia Rostagnol y Pablo Rodríguez han compartido muy amablemente conmigo su ámbito de investigación. Ellos me invitaron a extender sus logros, y señalaron para dónde queda el norte. El Capítulo 4 es producto, en buena parte, de sus enseñanzas. Gracias a Claudia Rostagnol, Daniel de Vera, Andrés Barrios, Matías Barrios y nuevamente Pablo Rodríguez, los resultados de esta tesis fueron probados en "vida real". Ellos saben que me entretengo con un lápiz y papel, pero cuando las predicciones se comprueban en la práctica, ya vale mucho más que el doble. Ellos fueron entonces quienes mutiplicaron el modesto valor del producto final.

El tiempo vale oro y, cuando se entrega generosamente, más aún. En los momentos críticos de escritura de esta tesis, Franco me ha llamado telefónicamente todos los días para alentarme y saber cómo estoy. Pablo ha complementado respondiendo muy sabiamente mis extensos y a veces torpes correos, en todas sus oportunidades con brevedad, calidad y celeridad. La naturaleza precoz de liderazgo en ellos debe a su humanidad. La generosidad entregada por ellos no la puedo cuantificar. Los admiro.

Agradezco a Alejandra Beghelli, que me ha dado bienvenida en Chile y presentado a su equipo. A Raúl Ures, cuyo apoyo económico hizo posible esta visita. Los frutos de un proceso fueron recogidos en la Universidad de Santa María de Valparaíso, con nuevos sabores, olores y colores. El mayor esfuerzo durante la escritura de esta tesis lo ha llevado mi pareja y filósofa de la vida: Ana Alvarez. Mientras escribía en Chile, ella lidió con obras en nuestro hogar, pese a la distancia. Inconmensurable es todo su apoyo sentimental, que enciende mi motor para hacer las cosas mejor. Mi agradecimiento es también para mi abuela, que con sus juegos me enseñó a sumar y restar, tarea hasta el momento imposible para mi maestra de primaria. De allí en más los números representaron un curioso juego para mí. Gracias a mi madre, Virginia Rodríguez, que hizo un admirable esfuerzo para criarme, pese a diversas adversidades.

Agradezco a Gerardo Rubino que me ha dado la bienvenida en Rennes. Poquitos encuentros con él me permitieron constatar que el humor es su gran legado familiar, y su humildad digna de mención. No es casual que Franco Robledo y Pablo Rodríguez hayan sido orientados por él. Agradezco a Bruno Tuffin y Martín Varela, que de forma completamente altruista han aceptado el rol de Reporteros de esta tesis. Mis tutores han logrado reunir a un equipo de investigadores soñado, en un mismo tribunal. Agradezco a Martín Varela, Bruno Tuffin, Omar Viera, Fernando Paganini, Raúl Tempone, Francisco Barahona, Alejandra Beghelli y Raúl Ures, por brindarme el honor de juzgar esta tesis. Advierto desde ya que los errores son todos míos, mientras que los aciertos se deben a la calidez humana de compañeros con quienes he compartido esta pasión.

Contents

INTRODUCTION

0.1 Abstract

Peer-to-peer networks are self-organizing communities developed at the application layer over the Internet infrastructure, in which users (called peers) share resources (bandwidth, memory, CPU time), in order to meet a common interest. The most challenging application due to bandwidth constraints is video distribution. There are basically three video streaming modes. The most simple is called file sharing, where the video is owned by one or several source nodes, and must be completely downloaded before its playback. A second streaming mode is video on-demand, where peers join a virtual network once a video content is requested, and begin a progressive download. The last streaming mode is called live-streaming, where the video is generated, distributed and played simultaneously. These streaming modes sound similar from a user's viewpoint, but present different design issues. In this thesis we study design aspects for on-demand and live video distribution.
The contributions of this thesis are two-fold. On one hand, we analyze the stability and capacity of a swarm-assisted video-on-demand peer-to-peer network. Peers start one or several concurrent downloadings and disconnect when they wish. The expected peer evolution is predicted assuming poissonian arrivals and exponential departure rates, with a deterministic fluid model. The system turns to be stable under practical scenarios, and via Little's law we can find closed expressions for the expected peer-excursion time. We theoretically prove that the peer-to-peer philosophy outperforms traditional Content Delivery Networks. A combinatorial optimization problem (COP) is introduced. The existence of a feasible solution is an NP-Complete decision problem. The issue is to store different video-types in caching nodes, trying to minimize the mean peer-excursion times. The nature of this problem is similar to the Multi-Knapsack Problem, where the knapsack capacities are represented by storage caching capacity, and items are video-types, which have different sizes. A greedy randomized resolution is here presented in order to define an optimal caching policy. This technique is applied into a real-life scenario, which is based on log traces taken from YouTube. The results reveal that the peer-to-peer distribution is economically attractive. On the other hand, a mathematical model for cooperation in live-streaming networks is here deeply analyzed. In this cooperative system, a source node broadcasts an unlimited video channel, and users store and forward video chunks. They wish to display simultaneously the same video stream with no cuts and reduced buffering times. The quality of experience is first captured by a COP. The issue is to choose the order in which video-chunks should be requested, to maximize the quality of experience. The nature of the problem is similar to finding the cheapest tour (node-permutation) of an Asymmetric Traveling Salesman Problem (ATSP). Hence, the problem is then translated into a suitable ATSP, which is inside the class of NP-Complete computational problems. The latter problem is heuristically addressed with an Ant-Colony Optimization approach. Its resolution has a direct interpretation in the design of chunk scheduling policies in live-streaming. Finally, the results state that this new policy outperforms widely used chunk policies, for instance the Rarest First and Greedy strategies. Finally, an Extended Model is introduced, discussing heterogeneity and free-riding effects. The results remark the importance of contribution-awareness to design highly resilient live streaming systems. *Keywords*: Peer-to-peer; Combinatorial Optimization Problem; Video-on-demand; Live-streaming.

0.2 Publications

List of publications issued from this thesis work:

2012 *"Analysis and Design of Peer-Assisted Video On-Demand Services"*.
Pablo Romero, Franco Robledo, Pablo Rodríguez-Bocca, Claudia Rostagnol.
To appear in International Transactions in Operational Research.

2012 *"Stability and Capacity of Peer-to-Peer Assisted Video-on-Demand Applications"*.
Franco Robledo, Pablo Rodríguez-Bocca, Pablo Romero, Claudia Rostagnol.
To appear in Proceedings of the fourth International Congress on Ultra Modern Telecommunications and Control Systems Saint Petersburg, Russia, October 3-5, 2012.

2012 *"A new caching policy for cloud assisted Peer-to-Peer video-on-demand services"*.
Franco Robledo, Pablo Rodríguez-Bocca, Pablo Romero, Claudia Rostagnol.
Proceedings of the 12 IEEE International Conference on Peer-to-Peer Computing. Tarragona, Spain, September 3-5, 2012. To appear in IEEE P2P 2012 Conference Proceedings and IEEEXplore.

2/2012 *"A Cooperative Model For Multi-Class Peer-to-Peer Streaming Networks"*.
Pablo Romero, Elisa Bertinat, Darío Padula, Pablo Rodríguez-Bocca, Franco Robledo.
Proceedings of the 1st International Conference on Operations Research and Enterprise Systems (ICORES'12). Algarve, Portugal, February 4-6, 2012. Pages 274-282.

2012 *"An Ant-Colony approach for the design of optimal Chunk Scheduling Policies in live Peer-to-peer networks"*. Pablo Romero, Franco Robledo, Pablo Rodríguez-Bocca.
To appear in International Journal of Metaheuristics (IJMHeur).

2012 *"Optimum Piece Selection Strategies for A Peer-to-Peer Video Streaming Platform"*.
Pablo Romero, Franco Robledo, Pablo Rodríguez-Bocca.
To appear in Computers & Operations Research.

2/2011 *"A Simple Proactive Provider Participation Technique in a Mesh-Based Peer-to-Peer Streaming Service"*. María Elisa Bertinat, Darío Padula, Franco Robledo, Pablo Rodríguez-Bocca, Pablo Romero.
Proceedings of the 6th International Conference on Hybrid Artificial Intelligence Systems (HAIS'11).
Wroclaw, Poland, May 23-25, 2011. Springer Lecture Notes in Artificial Intelligence, v. 6679, part II, pp. 42-50.

6/2011 *"Optimal Bandwidth Allocation in Mesh-Based Peer-to-Peer Streaming Networks"*.
Pablo Romero, Franco Robledo, Pablo Rodríguez-Bocca, María Elisa Bertinat, Darío Padula. Proceedings of the 5th International Network Optimization Conference (INOC'11).
Hamburg, Germany, June 13-16, 2011. Lecture Notes in Computer Science, v. 6701, pp. 529-534.

3/2010 *"A Cooperative Network Game Efficiently Solved via an Ant Colony Optimization approach"*. Pablo Romero, Franco Robledo, Pablo Rodríguez-Bocca, María Elisa Bertinat, Darío Padula.
Proceedings of the 7th International Conference on Swarm Intelligence (ANTS'10).
Brussels, Belgium, September 8-10, 2010. Lecture Notes in Computer Science, v. 6234, pp. 336-343.

12/2009 *"A COP for Cooperation in a P2P Streaming Protocol"*. Bertinat, M.E. Padula, D. Robledo, F., Romero, P. De Vera, D. Rodríguez-Bocca, P. Rubino, G.
International Conference on Ultra Modern Telecommunications & Workshops (ICUMT'09).
St. Petersburg, Russia, October, 3-5, 2012.
IEEE Computer Society, pp: 1-7.

12/2009 *"Optimum Piece Selection Strategy in GoalBit, a BitTorrent-based streaming system"*. Bertinat, M. E., Padula D., Romero P. Regional Conference in Applied Mathematics, (MACI'09). Rosario, Argentina, 14-16 December, 2009.

9/2009 *"GoalBit: The First Free and Open Source Peer-to-Peer Streaming Network"*. Elisa Bertinat, Daniel De Vera, Darío Padula, Franco Robledo Amoza, Pablo Rodríguez-Bocca, Pablo Romero, Gerardo Rubino.
Proceedings of the 5th Latin American Networking Conference (LANC '09).
Pelotas, Brasil, September 24-25, 2009.
ACM Digital, New York, USA, pp. 49-59.

9/2009 *"Systematic Procedure for Improving Continuity and Latency on a P2P Streaming Protocol"*. Bertinat, E. Padula, D. Robledo, F. Romero, P. De Vera, D. Rodríguez-Bocca, P.
IEEE Latin-American Conference on Communications (IEEE LATINCOM'09).
Medellín, Colombia, September 8-11, 2009. IEEEXplore Digital Library, pp. 1-5.

Basic Terminology

In order to assist the readability of this written, a summary of key concepts of peer-to-peer networks is here included:

- *Peer-to-peer*: Self-organized virtual communities developed on the Internet Infrastructure, where users, called peers, share resources (bandwidth, CPU-time, memory) to others, basically because they have common interests.
- *File sharing*: a streaming mode in which peers must completely download the file before its playback (in some related literature file sharing is not considered a streaming service).
- *On-demand video streaming*: the streaming mode in which users start a progressive download whenever requested.
- *Live video streaming*: in this streaming mode, the video is simulaneously generated, distributed and played by peers. It has the hardest real-time requirements, and peers must play the video stream synchronously.
- *Swarm*: a connected component of a peer-to-peer network, in which all users download the same stream.
- *Swarming*: Once a peer joins the network, it should discover peers with common interests. This is usually called swarming.
- *Swarm-assisted peer-to-peer network*: A P2P network organized in swarms, that contains one or several distinguished nodes, called node-servers or super-peers. Those nodes have higher resources than normal peers, thus contribute with higher streaming rates.
- *Chunk*: In order to distribute a video streaming (or file) to end-users, the video must be chopped-into several blocks. These blocks are called video chunks, or just chunks.
- *Chunk scheduling policy*: in order to get the full video streaming, peers exchange video-chunks. The strategy used to propagate video-chunks to end-users is called chunk-scheduling policy. In its most structural way, it is the permutation-order in which the numbered pieces should be requested.
- *Peer selection policy*: a characteristic element of peer-to-peer networks is cooperation between peers. The way in which peers are chosen in order to cooperate in groups is known as the peer selection policy.

- *Pull-process*: There are two elementary modes of cooperation (push or pull-based). When requesting peers force the communication asking for video chunks, we have a pull-process.

- *Push-process*: When transmitter peers choose useful video-chunks from receivers, we have a push-process. There is a rich number of mixed techinques, sometimes named pull-push.

- *Random policies*: we refer to a random peer (chunk) selection policy whenever the peer (or video chunk) is picked uniformly at random.

- *Churn*: Peers can join or leave the virtual network when they wish. Peer-to-peer networks suffer from node-churn, which represents the effects determined by peers because of their unpredictable arrival and departures. Well-designed P2P networks must be resilient to node-churn.

- *Resilience*: the ability of effective accommodation to unpredictable environmental disturbances.

- *Scalability*: The non-degradation property of a network, when the scale of the network (number of peers, arrival rate) is increased. The non-degradation property covers a level of performance and life of the network.

- *Seeders*: Peers that completely downloaded the streaming rate or file, and eventually feed peers within the network.

- *Leechers*: Peers who download one or several concurrent streams (from souce servers or seeders).

- *Free-riders*: Non-altruistic peers. They try to download as much files as possible, but do not contribute with the system.

- *Lifetime*: Several peer-to-peer networks are only available temporarily, and their life depends on the popularity of a file or stability of seeders. We say a peer-to-peer network dies when joining users cannot download the file (or stream) completely (for instance, by a fail in many seeders or mismatches)

- *Starvation*: A peer suffers from starvation when it cannot download one or several chunks. In file sharing, the result is catastrophic: that peer completely looses the file, and it dies. The complete unavailability of a data chunk in a swarm implies the death of that swarm in a file sharing service. In on-demand streaming, it means higher buffering times, whereas in live-streaming the user will have either a frame loss or image freezing.

- *Younger/Older peers*: Peer A is younger than Peer B when A joined within the network later than B. In that case, B is older than A. This concept is widely applied in video on-demand services, where there is no synchronization, and younger peers usually do not have resources.

- *Prefetching*: in on-demand video streaming, younger peers can rapidly insert into the community not fetching the current video-chunks, but later chunks, that are desired by older peers in the network. This technique trades buffering for bandwidth efficiency.

- *Video Cassette Recorder (VCR)*: the user of a VoD service expects basic interactivity options of the system. One of the most elementary interactions is the one offered by a simple Video Cassette Recorder (VCR), i.e. pause, forward and rewind options.

- *Anchor point*: the interactivity of a VoD system will succeed thanks to anchor points, which are pre-specified instants of a movie. The media player is re-directed to the nearest anchor point whenever the user skips or goes back to a chosen point of the playback, thus saving buffering times.

- *In-order (or sequential) chunk policy*: the video player in a VoD service requires video chunks in-order to achieve continuous playback. A greedy scheduling policy is hence to request for chunks sequentially.

- *Rarest First policy*: this policy tries to maximize the availability of data items, by means of requests/sendings of the rarest chunks in the swarm (or connected neighbors). Given that the neighbors are not always the swarm, this policy is sometimes called Local Rarest First. The idea is applied to all services.

- *Multiple Video Recorder (MVR)*: it is a caching technique frequently used in VoD systems, in which users can store multiple videos in their caches (even videos that they do not watch or intend to watch).

- *Single Video Recorder (SVR)*: the user in a VoD system only stores one movie in its cache.

Introduction

Our visual system has several limitations. For example, we cannot see ultraviolet nor infrared waves. Moreover, our eyes do not detect very fast movements, a fact which is unfortunately exploited by pickpockets, but essential for video deployment. This latter limitation has already been understood by the Chinese since the second century [145, 150]. However, the first moving pictures had to wait until the end of the nineteenth century.

Euclid and Aristotle supported that the light comes from the eyes toward the objects. The Arab mathematician Alhacen proved that the opposite is correct, and introduced the word camera for the first time (basically a cubic box with a small hole, which reflects an inverted image on a face). The simple idea of the dark camera was highly sophisticated and improved during years. The first moving pictures had to wait until 1888 by the hands of Eadweard Muybridge, with "The moving horses". This work founded empirical evidences of the discrete-continuous visual dilemma. Seven years later, the brothers Lumière invented in 1895 the first cinematographic camera, based on photographic concepts, and formally presented in Paris simple scenes on that year. They understood their product for scientific purposes. Paradoxically, they said that their invention had no future... Studios like Metro-Goldwin-Meyer and 20th Century Fox found the business face to the cinematographic camera. The cinema played an important social place for entertainment, specially in Hollywood and then all over the world. A huge step towards to deploy a visual service at home was the first commercial monochromatic television in 1928 and the NTSC color-compatible norm widely accepted for television broadcasting since 1953 [28].

On the other hand and in an apparently disconnected area, the ARPANET had its beginnings in the need of globally military strategies from the Pentagon in the second half of the century [176]. The first computer connection was possible via the public switched telephone network, and the first try was a login that failed and stucked in LOG... Where is the link between this fact and television? This experimental network after some decades covered the planet and so interconnected millions of personal computers. The invention of the World Wide Web in 1989 by Tim Berners permitted to achieve a "dream", that is, to link different web sites: Internet navigation via hyper-links [13]. As a paradox, that original experimental platform designed for military purposes with a very complex operation was transformed in the 90's into a friendly-guided navigation tool, which showed to be useful to communicate, learn, listen to music and see videos, among many others. The client-server architecture was the dominating scheme for file sharing. The users simply centralized requests to a server, which returned

the desired file. This architecture has some benefits. The service is both simple and highly predictable. But what happens if a lot of users try to get several files at the same time? A similar routine situation occurs if 100 people wait for the same bus with a capacity of 50 people. Needless to say, better luck next time. The first idea to attack this *scalability* problem was based on the content's popularity. The most popular contents can be obtained from the users! As a consequence, the server invites users to communicate and offer those contents which are normally replicated in the network. This idea was developed by Napster, the first structured and centralized peer-to-peer network [30]. Users were able to download MP3 music files and many others. The Recording Industry Association of America (RIAA) highly criticized these peer-to-peer networks complaining that the sales went down. Definitely, the digital music was the killer application of Napster, which ended in the thumb after a legal process. Clearly, the hierarchical structure of Napster avoided a key element in the design of peer-to-peer networks: the *anonymity*. However, the new paradigm woke-up the imagination of platform providers and users as well. Gnutella arrived with a very different concept. Now, all peers are clients and servers at the same time, normally called *servents* [3]. These peers find neighbors with elementary primitives of communication, and look for new contents via flooding. This completely distributed architecture showed its strength to disseminate popular contents, via the cooperation of neighbors. However, flooding was not effective for rare contents. To make things worse, the names could incidentally crash for different files, users were not forced to cooperate, and malicious peers could even conspire with one peer, disturbing its neighbors [175]. Many other structured network overlays were deployed in the beginnings of this century: CoolStreaming [229], Emule/Edonkey [165] and BitTorrent [44] for instance. BitTorrent, created by Bram Cohen, is an unstructured network overlay designed for fast distribution and replication of media contents [44]. The new concept is inspired in *incentives*: "give to get". The tit-for-tat solution of a game theory problem (the Iterated Dilemma's Prisoner) was included in this new design philosophy, and promotes an altruistic behavior of players [9]. Peers are self-organized via *swarms*. Once they enter the network, they are included in a swarm. With a pull-based technique, peers can get the rarest pieces of the file content first, in order to maximize the file availability. It exploits the robustness of random topologies, and peers cooperate with each other as soon as possible (in previous networks peers could contribute only after complete downloading).

BitTorrent faces many challenges, including the last piece problem [89, 139], fairness [72], streaming rate bandwidth adaptation [193], free-riding [127], churn [204] and faking files [162], among many others. However, it had enormous success, mainly because its usability for offline downloading and on-demand services. Its applicability cannot be easily extended to live-streaming applications, a hot topic that wakes the curiosity of the scientific community. When designing live-streaming peer-to-peer services, the natural approach is to extend or at least adapt the BitTorrent protocol. This is not a capricious idea: it is easier to extend than to re-invent. Moreover, BitTorrent works fine in offline and on demand streaming applications, where in the latter suffers subtle modifications, specially in its incentive-based mechanism.

The new *dream* is to have a triple-play system compatible with all streaming modes, covering offline downloading, on-demand and live-streaming. In the video market, the aim is to

obtain High-Definition Television (HDTV) in a traditional PC, in a cost-effective way (not wasting bandwidth resources) for a massive number of viewers. The nature of client-server systems does not adapt to this purpose, because the capacity of servers does not increase proportionally with respect to the number of entities in the network. An encouraging alternative is peer-to-peer networks. They are self-organized communities developed at the application layer and installed in the Internet infrastructure, in which the entities (usually named *peers*) act both as clients and servers. They are strongly based on cooperation. Peers communicate basically in a three-level based policy. In the first one, peers discover others interested in the same content, and is called swarming. Then, peers must select the best ones to cooperate, what is called neighboring. Finally, peers cooperate sending data-chunks to each other, and the planning must attend the *chunk scheduling policy* (or piece selection strategy in some literature).

Several scientific works suggest that the chunk scheduling policy is a key aspect of cooperation in live peer-to-peer networks [1, 34, 57, 214, 233, 234]. The most widely spread peer-to-peer systems are currently based on BitTorrent, whose chunk scheduling policy was not originally designed for real time streaming urgencies. In this thesis new chunk policies are designed, which outperform previous policies both theoretically and empirically, in the lights of the GoalBit platform.

On the other hand, a caching policy should be exploited in on-demand scenarios, where the video content is fully stored beforehand. Currently, YouTube offers video on-demand to end-users following the traditional Content Delivery Network (CDN) architecture, and comprises nearly 10% of the total traffic of Internet is due to YouTube videos. What is more, Google pays more than one million dollars per day in bandwidth access to Internet Service Providers [40, 96]. In this thesis we suggest YouTube could save even a 90% of bandwidth costs with a smart caching policy combined with peer assistance, exploiting user's resources. The two following sections contain the manuscript organization, production and main results of this thesis. The remaining sections of this introduction present the common causes of concern in the design of peer-to-peer networks, which show to be a promising alternative to offer bandwidth-sensitive services as video streaming, in a highly scalable fashion.

0.3 Manuscript Organization

This thesis is structured in three parts, subdivided in chapters. Part I summarizes the State of the Art, describing the main challenges and mathematical understanding of peer-to-peer networks, focused in the design of core resilience mechanisms. Part II contains the main contributions of this thesis, whereas Part III remarks the main conclusions, open problems and trends for future work.

More specifically, this introduction defines the basic terminology, exposes different video-streaming techniques and discusses possible architectures when choosing the taxonomy of a peer-to-peer network. Chapter 1 details the performance analysis of file sharing, with scope in the design aspects adopted in other streaming modes. BitTorrent shows to be an inspira-

tional system. Chapter 2 discusses the main design issues related with on-demand streaming services. The consequences of asynchronous playback of different peers combined with node churn and need for interactivity adds complexity to the network design (for instance, to fast forward or rewind like a traditional Video Cassette Recorder, VCR). A historic-driven motivation is provided, as well as a thorough real design in PPLive-VoD [98], covering those issues. This chapter shows hints of economical savings using the peer-to-peer architecture versus traditional content delivery networks (CDNs). Chapter 3 discusses the main challenges and trade-offs in the design of peer-to-peer networks for live streaming services. The original BitTorrent ideas are projected and extended in a live video-streaming platform, named *GoalBit* [18]. A conceptual exposition of the GoalBit protocol is included, and useful to understand the contributions of this thesis, found in Part II. A characteristic element in the design of live streaming is the importance of chunk scheduling policies, as can be revealed from the related literature [1, 34, 57, 214, 233, 234]. It closes with a discussion of widely adopted claims that have in fact a mathematical foundation. They include the fact that the local Rarest-First chunk policy works appropriately in sharing, but this success cannot be extended to live streaming systems. Those claims are motivation and point of departure for this thesis.

Part II has the main contributions of this thesis, which are two-fold. On one hand, Chapter 4 presents a cooperative fluid model for peer assisted on-demand video streaming. There, the stability and performance of P2P and CDN architectures are analyzed. We theoretically prove that the P2P approach is globally stable and always outperforms pure CDNs. In the peer-assisted architecture, peers cooperate and the system contains resourceful peers with large life-time, called super-peers. These nodes are hosted in the cloud and managed by the service provider. The issue is to choose whether a specific video-item should be stored in the a certain cache-node (super-peers), in order to minimize the expected excursion time of peers. We present a combinatorial optimization problem, called the Caching Problem, which captures this objective. This problem turns out to be NP-Complete, and its nature is similar to the Multi Knapsack Problem [131]. A greedy randomized heuristic is developed to solve the Caching Problem. Finally, the new caching policy is introduced in a real-life scenario, regarding more than 59000 video items taken from a passive YouTube-Crawler. The results confirm both the consistency and cost-effectiveness of peer-assisted architectures, versus pure CDNs.

On the other hand, Chapter 5 is intended for the mathematical analysis of live-streaming services in peer-to-peer networks. A novel chunk scheduling policy is designed, which outperforms classical policies, like Rarest First and Greedy. This new policy is introduced in the GoalBit platform, showing advantages when regarding playback delivery ratio and buffering times. The chunk-scheduling design is evaluated in the lights of a simple cooperative model for peer-to-peer streaming networks [233]. The reasons for this option are multiple. First, its simplicity: it is the first tractable model for analyzing chunk scheduling policies. Second, it captures the essential aspects of live-streaming systems: cooperation and synchronization. It also captures the most shocking video-factors for quality of experience: playback continuity and buffering times [182]. Last but not least, the model is proved to be highly robust [63]. We analytically show hints for the poor performance of classical scheduling policies. Therefore, an ideal design is first attempted to capture low buffering times and high playback continuity.

It fails but gives a preliminary experience for the design of feasible solutions. A single-score is defined trying to capture both playback continuity and latency. Then, a Combinatorial Optimization Problem (COP) is proposed. The issue is to find the best order in which video-chunks should be requested, and an exhaustive search among possible feasible solutions is computationally prohibitive. The COP is translated into a suitable Asymmetric Traveling Salesman Problem (ATSP), and the latter solved heuristically following an Ant-Colony approach. The result is a permutation-based chunk policy which outperforms classical policies, both theoretically (in the lights of the cooperative model) and empirically, by emulations in the GoalBit platform. A simple extension of the original model for cooperation is also introduced and discussed. It covers new ingredients: peers heterogeneity and presence of non-altruistic peers, or *free-riders*. We analyze different cooperation strategies in multiple scenarios. The results are intuitive, and confirm that when free riders represent a high proportion of this network, the server should be able to recognize them as well as other peers in order to deploy a highly scalable system. An optimistic conclusion is that the awareness of non-altruistic peers and bandwidth resources help to punish free-riding behavior and make the system scalable. Essentially, these results highlight the importance of contribution awareness in live streaming systems, and the natural tree-based organization in order to speed-up the chunk propagation when resourceful and altruistic peers can feed peers with limited bandwidth access.

Part III contains conclusions, trends for future work and an appendix. The concluding remarks and trends for future work are summarized in Chapter 6, whereas related open problems are presented in Chapter 7. The Appendix contains algebraic details as well as a template for the GRASP metaheuristic to address combinatorial optimization problems, which is used to solve the Caching Problem of Chapter 4.

0.4 Production and Main Results

The production of this thesis is basically summarized in the following items:

1. Mathematical modeling of peer-assisted architectures for on-demand video streaming, via fluid models.
2. Analysis of the expected evolution of peer-assisted video on-demand services.
3. Analysis of stability for on-demand video streaming systems.
4. Theoretical prove that peer-assistance always outperforms raw CDN architectures.
5. Combinatorial specification of a Caching-Problem, to decide the video-items stored in cache-nodes in order to minimize the expected waiting times for end-users in peer-assisted networks.
6. Real-world simulations of caching-mechanism regarding traces taken from a YouTube crawler, getting information of more than 59000 video items.
7. A mathematical analysis of chunk scheduling policies, in a pull-mesh cooperative system for live video streaming.
8. Design of the best chunk scheduling policies so far, in the lights of the first tractable mathematical model.
9. Introduction of feasible chunk scheduling policies in the GoalBit Platform, outperforming classical policies.
10. Design of an Extended Model, including free-riding effects and node-heterogeneity.

Items 1 to 6 are presented in Chapter 4, whereas Chapter 5 contains items 7 to 10. In the whole research process, accuracy is slightly compromised to gain simplicity. This is a natural fact inherent to the art of mathematical modeling, where the results give an overview of the system's behavior, and suggest hints for the network design.
As main conclusions, this thesis promotes the inclusion of peer-assistance in VoD-CDN architectures like YouTube (corollary of Items 3, 4 and 6), who comprises nearly 10% of all traffic on the Internet, and spends millions of dollars per month in bandwidth from ISPs [40, 96]. We confirm the strength of the peer-to-peer philosophy to address flash crowds and offload a cluster of servers. In fact, we predict nearly 90% of bandwidth savings by means of peer-assistance in YouTube for popular video-items, promoting the introduction of a hybrid architecture in YouTube and similar on-demand streaming services.

In live streaming, the Rarest First policy from BitTorrent is not suitable (Items 7, 8 and 9), and we just give hints of high-performing chunk scheduling policies. We introduce new scheduling policies in the GoalBit platform, and show the buffering times can be reduced to five seconds or less (Item 9), which is clearly acceptable for most purposes. Here we addressed a static and structural analysis, but a dynamic buffer adaptation is a trend for future

research. We believe a closed-look at successful proprietary networks such as PPLive would give additional hints, which could be combined with the mathematical results here provided. Additionally, we highlight valuable benefits of contribution and bandwidth awareness under heterogeneous networks under presence of potential free-riders with the development of an Extended Model (Item 10). Indeed, the cooperative system is highly scalable when the server is able to discriminate different entities in the network. However, if free-riders are frequently *gifted peers* (i.e. the server chooses them to send video-chunks several times), the performance drops dramatically.

These contributions have been disseminated in refereed proceedings as well as international journals. The point of departure of the research process in caching policies for P2P-VoD systems is [183], where my colleagues Pablo Rodríguez-Bocca and Claudia Rostagnol propose a combinatorial optimization problem to minimize the expected downloading times for end-users. Some inconsistencies were detected in the model, specially in the treatment of bandwidth bottlenecks (upload or download) and network stability, which was used there with no proof. The model suffered improvements, and the peer-assistance is proved to outperform the traditional CDN architecture under quite general scenarios in a short paper [177]. However, the stability of the fluid model was, up to that moment, an open problem. The peer-to-peer philosophy is proved to always outperform CDN systems, and the first stability results were included in a full paper [178]. The Sequential Fluid Model is finally proved to be globally stable, and the combinatorial optimization problem (the Caching Problem) is inside the class of NP-Complete problems, also proved for the first time in the journal [188].

A preliminary understanding of the cooperative chunk scheduling model for live streaming, including a Follower System for the design of ideal policies, was first presented in [16]. The analytical expression for the expected extension of a peer-to-peer request and combinatorial optimization problem, including an Ant-Colony-based resolution was introduced in [17]. The main ingredients and problem translation to an Asymmetric Traveling Salesman Problem (ATSP) is summarized in [187]. The GoalBit architecture is first presented to the research community in [18], which represents the benchmark that supports the experimental results of this thesis. Two compilation works, which include a historical revision of scheduling policies and results in the real GoalBit platform are disseminated in journals [184, 185]. Specifically, an in-detail design of the ant-colony exploration is presented in [184], whereas the compilation work [185] is more suitable for an operational research audience. An extension of the cooperative model is introduced for the first time in [186], where the goal is to study the impact of heterogeneity and free-riding effects. We highlight valuable benefits of contribution and bandwidth awareness under heterogeneous networks under presence of potential free-riders. The cooperative system is highly scalable when the server is able to discriminate different entities in the network, even under presence of free-riders. The overlay is naturally organized in a tree-based structure, where resourceful peers are parents of normal peer (or free-riders). It is worth to remark that the network performance is dramatically deteriorated when the server cannot recognize the different entities in the system (normal peers, double peers, free-riders and super-peers). In fact, if free-riders are frequently *gifted peers* (i.e. the server chooses them to send video-chunks several times), the performance drops dramatically.

The remaining sections describe video streaming modes over the Internet infrastructure, and cover essential challenges for the deployment of high-scalable peer-to-peer architectures, as well as outstanding concepts to face a massive video streaming service in the Internet.

0.5 Three Video Streaming Modes

A video can be delivered from a source-node using the Internet infrastructure via three streaming mechanisms, that differ in the generation, distribution and synchronization of media players. The simplest streaming mode is *file sharing*, in which the file is first generated in the source, then distributed and finally played back. In file sharing, the user has to download completely the video-file before its use. The second mode is called *on-demand* video streaming. The server forwards the video content whenever users demand it. Users can play the video online, while downloading it. In this streaming mode the user's media players are not necessarily synchronized. In fact, users connect independently and asynchronously, and the asymmetric nature of video on-demand services makes cooperation a challenging task, basically because younger users do not have fresh information to serve older users, and request for the beginning of the video stream (which is useless to several older peers).
The hardest real time constraints are imposed by the third mode, called *live streaming*, with simultaneous generation, distribution and playback. All users must be synchronized in the playback. From the user's viewpoint, on-demand video and live video are quite similar services. However, they are extremely different for design purposes. A remarkable difference is that the video content is fully stored before its dissemination in video on-demand, whereas the video stream is simultaneously generated, disseminated and played, with hard real-time constraints in the distribution of live-streaming. As a consequence, the operator of a cluster of servers can decide the video items stored in each server in an on-demand service beforehand.

Part I of this thesis is organized covering different streaming modes in each chapter. Specifically, Chapters 1, 2 and 3 cover file sharing, video on-demand and live streaming respectively, in turns. Although the design of live streaming distribution imposes hardest requirements than other modes, the related problems in each chapter are described looking for ease of readability, trying to elucidate the most challenging problems connected with the contributions from Part II, that explores live and on-demand services. File sharing is not included in Part II of this thesis. Nevertheless, Chapter 1 helps to understand the origins and mathematical foundations of peer-to-peer computing, and learn smart solutions that are adopted from file sharing to other streaming techniques.

0.6 Content Delivery Networks vs Peer-to-Peer

Consider a traditional Content Delivery Network (CDN) with m servers, each one working at its full Shannon capacity C_i [195]. Suppose in a certain instant, those servers should offer a video content to M concurrent users. The minimum streaming rate to assure the customer expectations is R (in compatible units). As a consequence, every user must receive a streaming

of R if we wish to maintain a minimum quality of experience, and the sum-capacity of servers must be higher than the sum-rate of users:

$$\sum_{i=1}^{m} C_i = C > MR. \tag{1}$$

It is clear that each server can feed a limited population of peers, and the minimum streaming rate cannot be kept under the presence of flash crowds (neither with some increase over its limit $M_{max} = \lfloor \frac{C}{R} \rfloor$). As a consequence, the performance of a CDN dramatically deteriorates when the population exceeds a threshold. This argument is an evidence that a new paradigm is needed for the delivery of highly popular contents. In Peer-to-peer networks, peers provide an additional uploading capacity, and if the cooperation is effective, we say the network *scales well*. A network is scalable when the quality of experience is unaffected with respect to the size of the population. The interaction of peers inside a peer-to-peer system is so complex that it is not practicable to determine on-line. Established providers need to understand the performance of such a system before its deployment, as a frequent loss in quality would jeopardize their reputation [206].

For those reasons, diverse mathematical models were developed, trying to understand scalability, fairness in the solution between peers, measures of altruism-greediness in the network and the presence of free-riders (i.e. peers that do not cooperate), among many other aspects. The common design approach is to model the network considering one or more possible scenarios. The model gives hints for the network design, and some prediction of the users behavior or network performance. Then, it is time to incorporate elements to the protocol and content delivery mechanism. A cyclic-design is pushed by technology and users needs, making the mathematical modeling a powerful tool.

There is an awesome mismatch between offer and demand when we refer to video distribution over the Internet. Indeed, the volume of video on the Internet doubles every year, while the demand is increased by a factor of three [1]. Currently, most of the video traffic carried over the Internet is managed by Content Delivery Networks (CDN). In these networks, a set of servers absorbs the load of the system, and are responsible of the distribution to end-users. Guided by the offer-demand mismatch and the network effect [151], there is a fierce competition between CDNs for the lucrative video distribution market.
There are basically two paradigmatic designs of CDNs. One is ISP-friendly, and driven by locality. Customers are served with the closest server with some metric (geographical, IP-based or latency based, for instance), and some issues of major concern are the managing and network maintenance in such a highly distributed architectural design. A relevant representative of these kind of CDNs is Akamai [4, 61]. A second approach is sometimes called ISP to home, where a few locations are strategically chosen to concentrate cluster servers, usually connected with high speed private connections. This second approach trades lower maintenance costs and overhead, at the price of possible higher delays to end-users. This approach is typified by Limelight [117].

[1] See http://www.researchandmarkets.com.

An alternative viewpoint to cope with massive demands and scalability is the peer-to-peer philosophy. Completely distributed P2P systems (Gnutella-like) suffer from illness like uncontrolled flooding, high overheads and network dynamics (variable resources and node churn). Regarding the great deployment of CDNs and recognizing their static resources which limit scalability, an evident possibility is to *combine the best of both worlds*, extending CDNs in an hybrid CDN-P2P architecture, as suggested by many researchers. The reader can find inspirational works on hybrid architectures in [97, 101, 156, 180, 191, 222]. In Chapter 4, we will study the performance and scalability of pure CDNs and Hybrid P2P-CDN technologies for on-demand video streaming services, showing clear advantages of the latter distribution engine. The interested reader can find a thorough revision of video delivery networks in [182].

0.7 Design Challenges for Resilient P2P Systems

The resilience of a network represents its capability to adapt under dynamic environments (network failures, variable resources, etc.). The peer-to-peer paradigm presents evident advantages when we compare the global resources versus a traditional CDN. However, the design of a resilient P2P network comes at a cost. The high penetration of ADSL services in the Internet makes matching difficulties between neighboring peers, given that the upload capacity of peers is usually five times smaller than their download capacity [21]. Peers arrive and depart the system when they wish, producing variability in global network resources, a phenomenon called *node-churn*. Some peers exploit network resources but do not contribute with the system (they do not share resources), called *free-riders*. The network is developed at the application layer, but network failures damage the service via congestion, network losses, link or router failures, among many others. There is an explicit trade-off between the full knowledge of the network (topology, peers bandwidth and file availability) and payload, which directly impacts in the throughput and network performance. In networks with stringent real-time constraints such as live video streaming, there is an additional challenge related with freshness of information. Packets that out of date are indeed discarded, meaning either a playback stall or frame losses.

0.7.1 Bandwidth Availability

From the beginnings of this century, broadband DSL systems were spread all over the world, specially ADSL. The asymmetric concept of ADSL provides a higher range of frequency spectrum for downloading, because normally users are expected to download more contents than what they upload [21]. This fashion imposes a top-uploading capacity in a peer-to-peer system. Moreover, the network in a P2P system is exposed to network failures. In a common BitTorrent-based network, every peer posses a list between fifteen and twenty other peers. As a consequence, if those peers do not need video content from the local peer, the uploading bandwidth is wasted, a phenomenon called *content bottleneck*. A third element which reduces the uploading resources still more is the malicious behavior of peers.

0.7.2 Churn

Peer-to-peer networks are open systems. Peers connect and disconnect freely, a fact which makes the network an attractive tool for them, but at the same time imposes many challenges in the topological network design and information availability. The unpredictable and uncontrolled peer-arrivals and departures is usually known as *node-churn*. In peer-to-peer networks, node-churn represents variability of network resources and more: losses of information. A related concept is *flash-crowd* which means an instantaneous and massive peer-departure or arrival. The consequences of a flash-crowd in ill-designed networks can be catastrophic. Just in file sharing, if the flash-crowd means that a piece of information cannot be recovered, it triggers the network's death. When several resourceful peers depart the system, a topological re-organization could be a solution. It is worth to notice that an intensive arrival rate in a traditional CDN represents either a degradation of quality, or even worse, a collapse of the system.

0.7.3 Malicious Peers

There are peers who do not contribute with the network (i.e. do not upload), and are called *free riders*. Free riding is a major cause of concern in peer-to-peer systems, and the network operator (or cooperative protocol) should promote incentives to avoid or control this undesired phenomenon. In related literature, incentives represent a cross-layer resilience mechanism to encourage peers to contribute with the global system (cross-layer because the peers' behavior is in the User Layer, but the application of incentives is in the overlay). Three alternative incentives mechanisms are paying, punishing or service differentiation [1, 88, 159]. A common approach is to press the nodes of the system with a give-to-get policy, so peers cannot departure the network without sharing resources.
In unstructured flat systems like Gnutella, we can find other malicious behaviors. Contents are shared via flooding, and if the neighbors of a certain peer do not share, it dies. This behavior is called *asphyxia*. Fake peers and crashed-names are additional difficulties, specially when the network lacks of a centralized entity (for example, a tracker). When the network presents a centralized entity like a tracker, another malicious behavior is the misreport or report of incorrect persuasive information in order to have better opportunities.

0.8 Centralized vs Distributed P2P Architectures

The design of peer-to-peer networks has an explicit trade-off between information availability and control, churn robustness and the underlying infrastructure (i.e. pure or unstructured, structured or hybrid). The history shows that in pure (totally decentralized) networks there is little or nothing to control, and free-riding and malicious peers are major causes of concern. On the other hand, a completely centralized network must have a special set of cluster servers, and the stability of the service strongly depends on the peer-population, usually falling in bottlenecks. These two opposite designs are typified by Gnutella and Napster respectively, which established the beginnings of the deployment of peer-to-peer networks. The deficiencies of both systems promoted the design of alternatives, like Distributed Indexing with Hashing ar-

chitectures. The concept is to distribute among peers a hash table, which maps keys (usually representing files) onto peers, promoting scalability, typified by CAN [170], Tapestry [230], Chord [203] and Kademlia [140], among many others (discussed in Chapter 1).

An inspirational system for fast dissemination and file propagation is called BitTorrent, created in 2003 by Bram Cohen [44]. In fact, both hashing and flooding architectures were partially displaced by BitTorrent-based networks, which constructs one swarm with random topologies for each torrent file. The BitTorrent's success for file sharing applications is proved repeatedly by several researchers, both theoretically and via experimental setup. However, it is not suitable in its original specification for live streaming, due to its large latencies. Nowadays, most of the peer-to-peer platforms are BitTorrent-based. For instance, GoalBit is the first open source platform that widely offers live and on-demand video streaming to end users [18]. A succinct description of the most determinant elements for the BitTorrent success are discussed in Chapter 1. The GoalBit protocol specification and key ideas to extend its success to live streaming are considered in Chapter 3.

0.9 Tree-Based vs Mesh-Based P2P Topologies

The topological structure of the network overlay is highly related with the core mechanism for the design of resilient video stream. A possible topology is to construct a tree-based overlay, always rooted in seeders (i.e. peers who fully own the file). Random neighboring leads to an alternative mesh-based topology. The most popular peer-to-peer networks are mesh-based. There are several reasons for that. The root-to-leaf delay is linearly increasing with respect to the height of the tree. Therefore, in order to design short trees, high node-degrees must be used. However, this solution carries new problems. First, the tree looses robustness, given that a node disconnection means several components to be immediately repaired. Second, the in-degree rate of a node must be divided into all successor peers, and a low capacitated peer imposes the bottleneck of several branches. The system is highly sensitive to node-churn. A third problem inherent to the design of single-rooted topologies is that leaf nodes never upload information. To make the things worse, short trees with high order always have an important number of leaf nodes, that are forced to free-ride. In multi-tree structures, leaf nodes are encouraged to be in some branch of other node, in order to contribute with the system. The design of multi-tree structures imposes many challenges, in order to recover from the previous issues and guarantee network reliability, high throughput and reduced latencies as well. Examples of multiple-tree designs are SplitStream [31] and CoopNet [155], both with complicated maintenance algorithms which give raise to cross layer techniques. The reader can find a compressed treatment with valuable references in [1], and a challenging related network design with bounded degree in [122]. There, they reveal solved problems and try to find the capacity of a node-degree constrained network, via multiple sub-trees. On the other hand, mesh-based approaches enjoy the advantages of low costs and simple maintenance of the network structure. Additionally, they are more resilient to node failures and departures. However, a clear disadvantage is that peers need to periodically exchange buffer-maps, leading to higher global overhead in the whole system. The trade-offs between network efficiency

and latency is a main problem in mesh based approaches. We will explore this trade-off in Chapter 5, where a simple cooperative model [233] is analyzed to address the playback-delay trade-off. A comparative study of tree-mesh topologies for live streaming purposes can be found in [132]. There are several realizations of mesh-based peer-to-peer network, for instance PPLive [163], PRIME [133], CoolStreaming [229] and GoalBit [18], which will be described in Chapter 3.

Part I

STATE OF THE ART

Chapter 1

File Sharing

1.1 Introduction

File sharing represents the most "rudimentary" method for peer-to-peer file propagation. The file is owned by a set of (or a single) source content nodes, or *seeds*, and they will try to disseminate this content as soon as possible to a dynamic set of peers, usually called *leechers*. An essential difference with respect to other streaming modes is that the file is limited, and completely stored by the content source server beforehand. In this way, the distribution of the content has a *lifetime*, which presents phases according to the file popularity and peers dynamism.

Furthermore, users do not enjoy the content before complete downloading. Hence, the most adopted measure of performance is the expected download time, or the worst waiting time among peers. Other relevant parameters are the network capacity under flash crowds (i.e. its survival properties), the payload and streaming rates, which are directly related with the minimum download time, called the *makespan* in related literature. The major causes of concern are churn, free-riding effects and malicious peers, trade-offs between fairness and performance, the last piece syndrome or starvation, mismatches between the logical topology and underlying physical resources, unbalanced upload-download capacities and push-to-pull mismatch between neighboring peers. In order to address those issues, several strategies were proposed, sometimes via a careful design of the network overlay and logical neighboring structure, choosing outstanding peers to share messages, others maximizing the total availability of information, grouping users in accordance with bandwidth resources, even sometimes inviting powerful and most contributing peers with incentives to join the network. The related literature is vast, and surveys do not reveal the entire design aspects of file sharing dissemination in peer-to-peer networks. In this chapter we will discuss the mathematical foundations of file sharing revealing the most referenced works, and a brief revision of real implementations specially designed for file sharing purposes. The intention is to capture the main design challenges, and suggest ingredients which are inherited in other services.

1.2 Mathematical Foundations of File Sharing Systems

Xiangying Yang and Gustavo de Veciana analyze the performance of file sharing systems under transient and steady state regime [225]. The first one is specifically the survivability and service capacity of the system under flash crowds, and the second is performance under variability on demand. An exponential growth can be appreciated in the transient, in which parallel one-to-one download and pipelining are remarkable survival elements. The expected latency is obtained for different scenarios. The size of a piece is a critical design element: smaller pieces suggest fast downloads, but they introduce overload to the system. In the transient analysis they propose a deterministic model, where a closed network has n identical nodes in the network with capacity b bps, and only one out of the n nodes owns a file of size s, measured in bits. Peers can contribute with the system uploading the file only after full download. Intuitively, the best cooperative strategy is one-to-one feeding. Specifically, the seeder first sends the file to another peer; then both feed two different peers and so on. The authors state without proof that the latter cooperative strategy is the best. The proof is simple, and we will include it here. First note that in the one-to-one strategy, we can study the evolution in slots, where the seeder takes $\tau = s/b$ seconds to feed another peer. Note that exactly 2^i peers fully download the desired file in time $t_i = i\tau$. Without loss of generality assume that $n = 2^k$. Then, the average download time is [225]:

$$\overline{T} = \frac{1}{n}\sum_{i=1}^{k} 2^i t_i = \frac{\tau}{n}\sum_{i=1}^{k} i2^i = \tau\left(log_2(n) - 1 + \frac{1}{n}\right) \tag{1.1}$$

In a client-server architecture, users should increase the waiting times linearly with the numbers of users. Equation (1.1) gives a hint of possible advantages to cooperative systems.

Proposition 1.2.1 *The one-to-one strategy is the best.*

Proof. Given that peers cannot cooperate unless they own the file completely, partial downloadings are never useful. All peers that fully downloaded the file must cooperate; otherwise the performance of the system will be deteriorated. Therefore, we will assume that the seeder equally sends the file to $u - 1 \geq 1$ users simultaneously. Then, in the second slot, each one of the u seeders send to $u-1$ different peers, and so on. Each time slot lasts $\tau' = (u-1)s/b = u\tau$. Then, exactly $u^i - u^{i-1}$ peers fully download the file in time $t_i = i\tau'$. The number of stages x for the makespan must comply that $1 + \sum_{i=1}^{x}(u^i - u^{i-1}) = u^x \geq n$, so the average download time is:

$$\begin{aligned}
\overline{T_u} &= \frac{(u-1)\tau}{n}\sum_{i=1}^{x}(u^i - u^{i-1}) = \frac{\tau}{n}\frac{(u-1)^2}{u}\sum_{i=1}^{x} iu^i \\
&= \frac{\tau}{n}\frac{(u-1)^2}{u}\frac{u-(x+1)u^{x+1}+xu^{x+2}}{(u-1)^2} = \frac{\tau}{n}(u^x(ux-1-x)+1) \\
&\geq \frac{\tau}{n}(n(ux-1-x)+1) = \frac{\tau}{n}(nx(u-1)-(n-1)) \\
&\geq \frac{\tau}{n}(n\log_2(n)-(n-1)) = \overline{T},
\end{aligned}$$

where the last inequalities use that $u^x \geq n$.

Q.E.D.

In a closed network, this highlights the idea that peer-to-peer networks can be effective to reduce expected waiting times experienced by end-users. Yang and de Veciana also suggest that multi-part downloading can substantially improve the performance of the system. In fact, under the same scenario, if we assume that the file is divided into m parts of equal size, and peers can share parts whenever they own it, the average download time is now [225]:

$$T'(m) = \frac{\tau}{m}\left(\log_2(n) + \frac{2m-1}{2}\right) \tag{1.2}$$

It is interesting to check that $T'(m)$ tends to τ when the number of parts increases to infinity. This ideal result would say that all peers wait the same as if only one peer were within the network! Naturally, this is far from practice. The headers of TCP packets would be dominant in the file size when the number of parts exceed a certain threshold [135]. Moreover, the statistics of network failures would assure losses with massive retransmissions. In all cases, real systems are exposed to node-churn, and the *best strategy* from Proposition 1.2.1 looses dramatically its quality, because a failure in the starting nodes has enormous impact in the average waiting times, or even worse, in the network's life. The authors translate the deterministic approach into a stochastic branching process, and discuss whether the network survives or not, in terms of the number of branches, including node-churn in a simplistic but illustrative fashion. They also propose a Markovian model to describe the peer evolution, which established the basis of further research. Empirical results highly validate the common conclusions obtained from both deterministic and stochastic models, and reinforce the fact that multi-part combined with parallel downloading are desirable to handle bursty demands.

Another foundational paper has credits to Qiu and Srikant [169]. There, the authors model BitTorrent-based networks under steady state and its variability, with empirical validation as well. A steady state analysis is first presented with a simple fluid model, in which the peer evolution is captured by poissonian arrivals and exponential departures in the system. They consider homogeneous peers, and find a closed expression for the average downloading time. The fluid model is captured with the two following differential equations, where the parameters are summarized in Figure 1.1:

$$\begin{cases} \frac{dx}{dt} = \lambda - \theta x(t) - \min\{cx(t), \mu(\eta x(t) + y(t))\}, \\ \frac{dy}{dt} = \min\{cx(t), \mu(\eta x(t) + y(t))\} - \gamma y(t). \end{cases} \tag{1.3}$$

The essential idea is to treat the number of peers and seeds in the system as a dynamic fluid inside a recipient, where λ represents the fluid entry rate, θ and γ the respective peer and seed's output rates. Additionally, the minimum function plays the role of the bottleneck, which is either in the download ($cx(t)$ in peers per second) or global upload $\mu(\eta x(t) + y(t))$, which takes into account seeders as well as incomplete information owned by peers (captured by an effective factor $\eta \in [0, 1]$). The authors find first the rest point for the system (solving $\frac{dx}{dt} = \frac{dx}{dt} = 0$) and by means of Little's law they relate the average download times T with the number of peers under steady state:

$$T = \frac{1}{\theta + \beta}, \tag{1.4}$$

being $\frac{1}{\beta} = \max\{\frac{1}{c}, \frac{1}{\eta}(\frac{1}{\mu} - \frac{1}{\gamma})\}$. A sensitivity analysis of Expression (1.4) with respect to different design parameters offers one of the first valuable insights of the BitTorrent's correctness [169]. Surprisingly, the download time is independent of the arrival peers rate, giving hints for the BitTorrent's scalability. The download time monotonically decreases when η is increased, and T increases when γ does. Another valuable remark is that the performance improves when the download bandwidth is increased, but after certain threshold β does not depend on c, neither the average download T. A similar argument holds for μ.

$x(t)$	number of downloaders at time t.
$y(t)$	number of seeders at time t.
λ	arrival rate of peers.
θ	departure rate of peers.
γ	departure rate of seeders.
c	downloading bandwidth of each peer.
μ	uploading bandwidth of each peer.
η	sharing effectiveness of file sharing ($\eta \in [0, 1]$).

Figure 1.1: Parameters in the inspirational model from Qiu and Srikant [169]

A special treatment is included for the file sharing efficiency between peers, and confirm that random peer selection is translated into a high file sharing efficiency. Indeed, Qiu and Srikant also find a rough expression for the file sharing effectiveness η, which is defined as the probability of not needing any piece from all other directly connected peers, assuming uniform piece distribution in the system (exactly what BitTorrent tries with its Rarest First policy):

$$\eta \approx 1 - \left(\frac{logN}{N}\right)^k, \tag{1.5}$$

being N the number of pieces in the system, and k the number of neighbors of a given peer. The steady state of the system given in (1.3) is partially characterized as locally stable, and they conjectured that it is globally stable as well. The authors also include a game-theoretical approach and discuss incentives effects. Specifically, they model BitTorrent as a competitive system, and prove the system achieves a Nash equilibrium point, in which free-riders are punished with lower rates than cooperative peers. This work is directly complemented by Tewari and Kleinrock [206], Qiu and Sang [168], and cited by more than one-thousand other works. Tewari and Kleinrock easily show that with peer groups higher than 15-20, the file sharing efficiency increases with respect to the number of chunks, strongly based on Expression (1.5) and simulations. BitTorrent tries to maintain 20 peers, so Tewari and Kleinrock support the protocol [206]. Qiu and Sang prove that the conjecture of global stability for (1.3) is true, which suggest both consistency of the results given in [169] and stability of peer-to-peer networks. They identified the fluid model as a switched linear dynamical system. As a consequence, local stability is easily proved, but global stability has several tricks. They discovered a Lyapunov function in order to complete the global stability. The reader can find a friendly approach to the stability of dynamic systems in [129], and an in-depth study of linear switched systems in [116]. These works have been extended in this thesis in Chapter 4, where the model is

adapted to on-demand video streaming, adding node churn, special caching nodes and multiple concurrent video transmissions. The system is proved to be globally stable as well, and a novel caching technique is applied into the first free and open-source live peer-to-peer platform called GoalBit, currently offering on-demand services as well [11, 18].

The models developed by Laurent Massoulié and Milan Vojnović are similar in spirit than the ones of Yang-de Veciana [225], and Qiu-Srikant [169], via *coupon replication systems*, with rigorous mathematical results [137]. They propose a deterministic fluid model for a large scale limit, in both layered and flat systems. In a layered system, users exchange coupons only if they have the number of coupons, whereas peers are uniformly selected at random in the flat system. In order to understand the flash crowd, a closed system is considered, and Poissonian processes are used for a steady state analysis. The models assume that joining peers are awarded from the server one random coupon. A remarkable conclusion is that both flat and layered systems are stable even under random blind coupon selection, so the Rarest First policy, though intuitive does not represent the reason of robustness in BitTorrent. An additional analysis of the complete layer system with all but one coupon missing shows that the population geometrically decreases, strengthening the robustness of swarming systems that were previously predicted by Yang and de Veciana [225].

Di Wu details a trade-off between fairness and performance of peer-to-peer streaming systems [218]. Experimental analysis show that the most resourceful and altruistic peers normally upload nearly ten times more than other peers, but download the similar streaming rates. It sounds unfair to keep altruistic peers as long as possible in the system in order to achieve better performance, and a fairness index is considered in his exposition. Using only traffic conservation and statistical assumption of poissonian arrivals, an average streaming rate is found with heterogeneous peers (with different uploading and downloading capacities). The discussion is then focused on the fairness score under given performance, and vice-versa. A convex optimization problem is presented, and a water-filling approach[1] gives the best trade-off (found via the Karush-Kuhn-Tucker optimization technique). There, users with uploading rates lower than a certain threshold must contribute at their full capacity, whereas the other peers must stream with the same rate (the threshold streaming rate). The introduction of this cooperative scheme in a real platform should use a tracker, who must estimate peer resources, find a threshold and return the task to the peer. However, the author points out it is not practical, just because selfish peers might not report their statistics, and the centralized tracker could collapse in a flash crowd. Besides, an efficient perfect-matching is assumed (where no uploading resources are wasted). A similar trade-off is studied by Fan, Lui and Chu, where the service capacity is measured with the total download time [73]. The authors prove that the best performance is achieved when the strongest peer (with the highest download bandwidth) downloads a limited rate, lower than its full capacity, whereas all the other peers must download as fast as possible. This is extremely unfair (the best peer can even download at the lowest rate). Then, a min-max problem is proposed in order to find trade-offs, falling again in an optimum water-filling technique. The authors argue that BitTorrent works in a feasible space of rates, which are between

[1] Curiously, water filling is the best strategy to share information with multiple uncorrelated independent Gaussian channels subject to additive white Gaussian noise. See [46] for an excellent book covering network information theory.

fairness and performance optimality. A design using *knobs* is considered to build up a protocol including those outstanding points, which can be implemented in a distributed way. The authors explain that BitTorrent is just one point between the whole space, which is not necessarily fair.

Minghua Chen et al offer a survey dedicated to single source multicast scenarios over P2P networks, with perfect links and node upload capacity constraints [36]. They do not treat network coding, and argue that it is not applicable unless extreme changes in both IP routers (which need to encode) and end-hosts (to decode). An optimal solution can be obtained when no degree constraints, with the presence of helpers [115] or without them [107, 136]. The structure of both solutions are multi-tree based, rooted on the source node. An extension of this problem is to limit the per-tree degree, which sounds more realistic in peer-to-peer networks, given that peers cannot maintain connection with all the other peers. If all nodes are identical, it is illustrated that CoopNet and SplitStream achieve optimal solutions (in a static network without node-churn), via multiple trees where each node is interior only once, called interior-node-disjoint trees. The complexity of the problem increases dramatically in the general heterogeneous network with bounded node-degree. In fact, the determination of the streaming capacity is NP-Complete [122], and the Bubble algorithm proposed by the authors achieves an approximation factor. If the uploading peer capacities are assumed to be a random variable with first and second moment known, then a pragmatic grouping algorithm is designed, whose streaming capacity is near the mean peer upload capacity. The solution assumes that the bound degree is logarithmic in the number of peers within the network, and also proposes a merge-break scheme to re-organize under node-churn. A similar work not covered by the survey is conducted by Mundiger, Weber and Weiss [147, 148]. There, the authors study the minimal makespan, that represents the necessary time to share M messages from a source node of upload capacity C_S (measured in MBps) to N end-users, with limited uploads $C_1, \ldots, C_N$ but unlimited downloads. In the special case of identical peer capacities, the minimum number of rounds required to complete the dissemination of all file parts is $M + \log_2(N)$. Each round takes $\frac{1}{M}$ units of time, and then the minimal makespan for all M and N is:

$$T = 1 + \frac{\log_2 N}{M} \tag{1.6}$$

Basically, the idea of the proof is to note that the server requires M rounds to distribute all parts, and the last part (owned first by a single peer and the server) will take $log_2(N)$ additional rounds to distribute each part to the whole population. A complete proof including a construction of the distribution is detailed in [148], and reveals the optimum technique is quite similar to the BitTorrent matching for exchange. There, the authors also prove two lemmas that permit to treat distribution with slots (in a discretized fashion). The heterogeneous case is more complex. They prove that the minimal makespan can be obtained by an analogous discretized problem - a Mixed-Integer Linear Problem - in which new uploads to occur only after other uploads finish. When the number of messages M is large in relation with the number of end-users N, then the minimal makespan can be obtained via a limit fluid model:

$$T^* = \max\left\{\frac{1}{C_S}, \frac{N}{C_S + \sum_i C_i}\right\}. \tag{1.7}$$

In a similar fashion, if every end user has a file to share of size F_i, $i = 1, \ldots, N$ (with no distinguished server), the minimal makespan is:

$$T^* = \max \left\{ \frac{F_1}{C_1}, \ldots, \frac{F_N}{C_N}, \frac{(N-1)\sum_i F_i}{\sum_i C_i} \right\}. \tag{1.8}$$

The authors are interested in finding makespan of real-life platforms in order to measure possibilities of optimization. Additionally, they propose decentralized solutions to be carried out in a practical network, and confirm that random peer selection possesses high performance.

However, real platforms are exposed to node churn, and the models here developed do not include departures nor arrivals[2].

Kumar and Ross observe that closed expressions for the makespan have only been obtained for highly simplified scenarios, with infinite download bandwidth or identical peers [108]. They consider a peer-assisted distribution problem, in which there are S seeds with streaming rates r_s that completely own a file of size F, and L leechers with (possibly different) downloading capacities d_l and uploading rate r_l. Their objective is to find the makespan, regarding users that depart the system immediately when they fully download the file. Three elementary remarks characterize a lower bound for the makespan:

1. A leecher cannot download at rates faster than d_l, so $T_{min} \geq \frac{F}{d_l}$ for all leechers.
2. A leecher cannot receive more than the whole available resources: $T_{min} \geq \frac{F}{\sum_i r_s}$.
3. The L files are downloaded with the global uploading resources: $T_{min} \geq \frac{LF}{\sum_s r_s + \sum_l r_l}$.

A lower bound can be trivially obtained from those remarks. A hard task is to prove that the equality holds [108]:

$$T_{min} = \max \left\{ \frac{F}{d_l}, \frac{F}{\sum_i r_s}, \frac{LF}{\sum_s r_s + \sum_l r_l} \right\} \tag{1.9}$$

Kumar and Ross compare Equations (1.9) and (1.6), with a single seed and equal capacities. The difference is due to their continuous model versus the chunk-model developed by Mundiger, Weber and Weiss. The relative error is $\frac{log_2(L)}{M}$. Kumar and Ross conclude this relative error is negligible (one percent or less) considering realistic scenarios. Specifically, the typical chunk size in BitTorrent is 256 KBytes [110], and with less than 10000 leechers and file sizes of 350 MBytes (or more), the relative error is not bigger than one percent. Lingjun Tsang and King-Shan Lui further explore the makespan (1.9) obtained by Kumar and Ross, designing a grouping problem. They try to find k packs of seeds and leechers such that the following conditions are met [120]:

$$T_{min}(S_i, L_i) \leq T_{min}(S, L), \forall i \in \{1, \ldots, k\}$$
$$T^G_{avg} < T_{min}(S, L),$$

[2] It is interesting to notice that their results extend telephonic models [74], proving results in a brief and elegant way.

being $T_{min}(S_i, L_i)$ the makespan for a group with $S_i \subseteq S$ seeds and $L_i \subseteq L$ leechers, and T^G_{avg} the averaging among all those makespan (S_i, L_i). If there is such partition G they call it a *grouping*. They prove that no grouping exists when the equality for T_{min} holds in Cases 2 or 3. However, when the equality holds for Case 1 (i.e. $T_{min} = \frac{F}{d_l}$ for some leecher l), the existence of a grouping is an NP-Complete decision problem, as can be proved with a reduction from Set Partition [120]. The authors also design an algorithm that looks for feasible groupings and re-organizes grouping when a leecher failure occurs. The results of the grouping algorithm shows to be encouraging when compared with the BitTorrent protocol via simulations. Note that free-riding is completely neglected in most works, and the focus in just a few. Free-riding is a major cause of concern in peer-to-peer networks. However, it is rare to find works that measure the streaming capacity of the system under free-riders. Creus, Casadesus and Hervas propose a game with elementary rules and two entities: free-riders and sharers [47]. Entities have scarce unit download and infinite upload, and choose a single target sharer to download. If s and f denote the number of sharers and free-riders respectively, they show that the average download among all stable topologies is tightly similar to $s/(s + f)$, being a topology stable whenever a peer cannot improve re-connecting to another sharer. The model is extremely simple, and does not consider heterogeneity, file sharing efficiency nor topological constraints. The reader can have an overview of works on free-riding for file sharing in the references therein [47].

Other works focus on a phenomenon sometimes present in BitTorrent-based networks, called *starvation*: a peer owns the whole file but a missing data-chunk, waiting very long to get that chunk, or never getting it. Mathieu and Reynier explain the hazard of starvation under flash-crowd in a simple way [139]. They propose an upload-oriented approach for a swarm-based network with P peers, S seeders and K data-chunks to be shared. Specifically, the sender chooses the chunk to be uploaded. The upload-oriented model is there justified because an uploader peer is altruistic, hence able to choose the best piece and peer to upload in order to help the collectivity (and the pragmatic assumption that the upload is the bottleneck but the download). They further assume identical peers with unit uploading capacity, no exogenous arrivals, and instant departures once the file is fully downloaded. Under globally random strategies, the network is either expected to reach a safe regime or starvation. In the former, the seed can fail and the network survives. Otherwise, under a starvation scenario the network *dies* if the seed departs the system, given that a missing chunk cannot be recovered. Starvation can be triggered in BitTorrent if the entry rate is high, peers depart before they are *healed*, and starving peers accumulate. The authors conclude that discriminant chunk-oriented policies are more resilient than globally random policies, giving credits to BitTorrent (when compared with eDonkey). Bruce Hajek and Ji Zhu develop a Poissonian model so understand the missing piece syndrome [89]. They theoretically prove that the missing piece syndrome occurs almost surely (i.e. with unit probability) whenever the seeder rate λ_S is lower than the arrival rate λ, so the bottleneck for stability is the upload capacity of the seed. On the other hand, if the arrival rate of new peers is less than the seed upload rate, the Markov Chain is ergodic, and the mean number of peers in the system in equilibrium is finite. They assure that the result holds for random useful piece (in which the uplink or downlink models are equivalent), but for the Rarest First and Greedy policies as well. When using network coding with Galois fails F_q, the stability condition is $\lambda < \lambda_S(1 - \frac{1}{q})$. Nevertheless, network coding has

the advantage that no exchange of state information among peers is needed because there is no need to identify useful pieces. They conjecture the tit-for-tat policy greatly helps to tackle the missing piece syndrome, and turns the system stable, but the mathematical tools developed do not cover tit-for-tat nor peer-selection. The stability for different two-chunk based models is presented in [152]. There, the authors show 5 different strategies to download a file with only two chunks, focusing on the stability, and suppose that entering peers are awarded with one chunk once they join the network.

1.3 Inspirational Systems for File Sharing

It is instructive to understand the origins and inspirational platforms of peer-to-peer networks. We briefly visit the first widely spread networks: Gnutella and Napster, discussing their benefits and drawbacks. Hashing architectures were designed to address the major problems in both, distributing a hash table among peers in order to facilitate the routing and item availability. Then, a description of BitTorrent, the most successful protocol for file sharing purposes, is here included, discussing experimental results.

1.3.1 Napster, Gnutella and Hashing Architectures

In Napster, a centralized entity invites users to communicate and offer those files which are normally popular in the network. Napster represents the first structured and centralized peer-to-peer network [149]. Users were able to store several MP3 music files and retrieve them with the help of Napster. As soon as this network achieved an enormous success with hundreds of millions of users sharing MP3 files, it became a threat against traditional business models supported by the record companies [30]. The Recording Industry Association of America (RIAA) highly criticized peer-to-peer networks, complaining that the sales went down. Definitely, the digital music was the killer application of Napster, which ended in the thumb after a legal process. Clearly, the hierarchical structure of Napster avoided a key element in the design of peer-to-peer networks: the *anonymity*. However, the new paradigm woke-up the imagination of platform providers and users as well.
Gnutella is the first unstructured peer-to-peer network. The developers (fans of a dessert named Nutella, and followers of the GNU project) introduced the concept of *servents*, in which nodes are servers as well as clients [3]. Via flooding or back-propagation, servents communicate exchanging very basic primitives: ping-pong, query and get-push messages. Studies of social dilemmas have shown it is hard to generate spontaneous cooperation in large anonymous groups [99]. Via experimental measurements, Adar and Hubermanthe show that Gnutella is not an exception [3]. They found that nearly 70% of Gnutella users share no files, whereas the half of all responses are returned by the top 1% of sharing hosts. Due to its power-law topology [45], Gnutella is robust to random disconnections. However, this network is vulnerable to well-planned attacks, and flooding carries enormous overheads. Furthermore, an entropy-based measure highlights the fact that the shape of this self-organized network is independent of (hence, it does not match with) the underlying physical infrastructure [175]. Another critic for Gnutella is presented in [228]. There, the authors use a sophisticated crawler, and point

out that the overload due to ping-pong messages is enormous, occupying a 63% of the network utilization. Summing up, the great freedom of peers and random topology in Gnutella is translated into free-riding and malicious peers, payload in handshaking and evident mismatch between the logical topology and the underlying network resources. An incentive-based mechanism is needed in order to deter free-riders and keep strong peers alive. Regarding Gnutella's deficiencies, new architectures were proposed, for example Chord, CAN, Pastry and Tapestry. They represent a third architecture, called Distributed Indexing with Hashing. The concept is to distribute among peers a hash table, which maps keys (usually representing files) onto peers, promoting scalability. The Distance Vector (DV) and Link State (LS) were IP-routing algorithms at disposal in that moment. However, they require every router to have a global level of information about the topological structure of the network level [205]. Chord is a distributed lookup protocol to efficiently locate the node that stores a particular data item [203]. Peers in Chord can find a desired file in a path with logarithmic length with respect to the number of nodes, in a similar way than a bipartition search and with smart memory savings. Chord lacks anonymity, and does not exploit network locality for its routing. However, its correctness is robust in the face of partially incorrect routing information. The heart of Chord is the fast distributed computation of a hash function mapping keys to nodes responsible for them. The load balancing is achieved with Consistent Hashing [114]. CAN (Content Addressable Network) uses a dynamic discretization of a d-dimensional torus in the euclidean space to address churn and item location [170]. Peers are assigned one or several zones of the torus, and they have logical neighbors in accordance with adjacent zones. The item search is based on request to logical neighbors. The authors propose several design extensions in order to improve round-trip times (RTT), average path length and robustness. The node updating costs $O(d)$, and the lookup cost is $O(dN^{1/d})$, being N the number of peers in the network. The dimension of the torus d can be chosen beforehand. However, the peer population should be predicted to choose the best dimension d. The hash table's size managed by a CAN node does not depend on the network size, but the lookup cost increases faster. Note that the routing table has neighbors, and does not increase with the network size (it is twice the dimension of the torus or less). CAN requires an additional maintenance protocol to periodically remap the identifier. Globe is similar to a DNS lookup, and exploits network locality, but works in a hierarchical manner [212]. Pastry is completely decentralized, and exploits network locality minimizing the number of IP-hosts. The average number of IP-hops is logarithmic in the number of nodes. Each peer has an identifier, and forwards messages to one out of k peers, whose keys are similar to its identifier. The node-ids are randomly assigned, so peers with similar keys are likely to have diverse geographical distance. Nevertheless, a smart heuristic for routing is proposed, selecting the nearest peer with high probability [189]. Tapestry is designed in a similar spirit, and trades-off complexity and anonymity [230].

In [81], the authors develop mathematical models to understand different peer-to-peer architectures. They consider three different infrastructures: Centralized Indexing Architecture (typified by Napster), Distributed Indexing with Flooding Architecture (eg. Gnutella) and Distributed Indexing with Hashing Architecture (Chord, Pastry, Tapestry and CAN, for instance). They assume a closed queuing system with multiple classes, and find approximate expressions for service rates and probability of fail during a query. They also include free-riding effects, and conclude that the system does not suffer much performance degradation, in contrast with pre-

vious works. A system with flooding cannot exploit the full capacity of peer-to-peer systems, whereas centralized architectures achieve acceptable performance in a limited population of users. The authors argue the framework is flexible enough to analyze further features in file sharing. Both the centralized and hashing architectures can support a much larger ratio of freeloaders than the flooding mechanism in pure decentralized networks. A widely used DHT-based system is Kademlia [140]. The routing structure resembles the algorithms of Pastry and Tapestry. However, Petar Maymounkov and David Mazières pointed-out that those algorithms require a secondary routing table whose size is increased with the network scale. Kademlia is a DHT-based system which includes a novel XOR metric to facilitate search and lookup, in a symmetric node relation. This solution permits to learn useful routing information, hence avoids the rigidity of node's finger and static routing in Chord. Routing in Kademlia is latency-based, and parallel queries can be asynchronously sent from a peer to a several target receivers. Kademlia is strongly based-on the XOR metric, which is the bitwise exclusive or, interpreted as an integer written in binary. It is clearly a non-Euclidean metric (associativity implies the property $(x \bigoplus y) \bigoplus (y \bigoplus z) = x \bigoplus z = d(x, z)$, and the triangular inequality). As in Chord, it is *unidirectional*: given a certain distance $\Delta > 0$ and a 160-bit x, there is only one 160-bit y such that $d(x, y) = x \bigoplus y = \Delta$. The unidirectionality property ensures that lookups for the same key converge to the same path, and caching $< key, value >$ pairs alleviate hotspots. Each node keeps for each integer $i : 0 < i \leq 160$ a list, called *bucket*, which contains nodes who are both alive and the XOR distance is between 2^i and 2^{i+1} from the local Node-ID. All buckets have a top-size k, and the members of the buckets are periodically updated using alive messages as well as full information of queries. A curious element in Kademlia is that older alive peers are preferred to be in the buckets, because previous studies in Gnutella show they are more stable peers [192]. Another incidental advantage is that the system is resistant to certain DoS attacks, given that fresh nodes play a secondary role. Kademlia has provable performance: the order of operations is logarithmic with the number of nodes, and lookups returns a key with overwhelming probability. See [140] for details of the lookup algorithm and sketch of the proof.
Some modern BitTorrent clients support distributed hash-table extensions implementing a Kademlia-style structure overlay network. The interested reader can find details and concerns in the design of Kademlia-style structured overlay networks in [49].

1.3.2 BitTorrent

In this section we will briefly describe the BitTorrent protocol and give an insight of the different experimental setups and results, in order to complement the previous mathematical understanding of file-sharing systems.

1.3.2.1 Protocol

Bram Cohen, the developer of BitTorrent, affirms that BitTorrent includes the tit-for-tat policy seeking for Pareto efficiency (i.e. trying to reach a stable configuration in which no two counterparts can trade and be both better than before). He also assures its success is not only due to the core design, but the simple interface [44]. Users are forced to use the BitTorrent client (that

must be downloaded before its first use) once a publisher decides to upload a .torrent file in a web server, presented as a hyperlink. The user just clicks on the link and is presented a Save-as dialog. Once it is confirmed with OK, two progressive bars appear: one for download and other for upload. The .torrent file contains the file name, size, a SHA1 hash code, the IP address of a tracker, and more. Peers cooperate with the help of a tracker, that sends a random list of other peers interested in the same file (typically a random list of 50 peers). Normally the client seeks to maintain communication with 20-40 other peers. The bandwidth requirements of the tracker and web server are very low, but the seed must send out at least one complete copy of the original file. The pure tit-for-tat strategy was first designed by Axelrod, who proposed it to solve the Iterated Dilemma's Prisoner [9]. His program consisted of a three-line BASIC code, and has been proved to be optimal, winning that competition (other proposals were scripted in hundreds or thousands of lines!). In peer-to-peer networks, a pure tit-for-tat policy would mean that peers would share the file only with others who serve it. The swarming of BitTorrent combines tit-for-tat with a special ingredient, called *optimistic unchoking*. The peer's handshaking is managed over TCP connections. Peers can either upload or choke neighboring peers. A choke is a temporary refusal to upload. If two peers are getting poor downloading rates in relation with their upload, they can both cooperate and find better exchange. Note however that the TCP connection is still active, and the other part can still upload data.

From a single peer's viewpoint, all peers are choked but four peers (some recent BitTorrent versions slightly modify the choking process [112]). These peers are the ones with the highest download counting the last twenty seconds. Once these four peers are chosen, the choke state remains for at least 10 seconds, in order to saturate the TCP connections at their full capacity. Every 10 seconds, a peer re-evaluates the upload rates for all the peers that transfer data to him. Then, the choking algorithm is repeated in cycles. Additionally, peers can discover new opportunities via one optimistic unchoke every thirty seconds. It is strictly related with the first step in the dilemma's prisoner: always cooperate in the first step. In this way, entering peers are normally welcome.

Now we concentrate in the piece selection policy of BitTorrent. A file is normally divided into pieces of size a quarter of a megabyte. Each piece is also subdivided into blocks, typically of 16 kilobytes. A peer always chooses to download the rarest piece, which is the less available among its neighbors. This is called the Local Rarest First policy. In this way, there is a uniform balance of availability in the network. There are exceptions to the rarest first rule: at the beginning and at the end of a file download. At the beginning, a peer joins the network with no piece, and this peer will not be capable to cooperate unless a fast download takes place. To that purpose, a random piece is first selected, which is possibly more available than the rarest and hence several peers are able to send blocks to the new peer. This random first policy is applied just to download the first piece; then switched to the rarest first policy. At the block level, BitTorrent uses a *strict priority policy*, which means that once a block of a piece is downloaded, the other blocks of that piece are requested with the highest priority. A second rule that escapes the rarest first is the last piece. When a peer has all the file but one piece, that peer sends requests to all peers to recover that piece, so its download velocity is increased. That peer is now a seed, and chooses to upload to the most contributing peers. Seeds can leave the system when they wish. However, if the user is polite, it will remain within the system for some

time, improving the system performance. A nice and more-detailed description of BitTorrent written by Bram Cohen can be found in [44]. The reader can find further information and description of other BitTorrent flavors in [100, 112, 169, 218].

1.3.2.2 Validation

Several experimental works validate the BitTorrent's success for file sharing purposes. Here we enumerate a non-exhaustive list of empirical and related theoretical results, to illustrate the BitTorrent performance.
Izal et. al. assess the performance of the algorithms used in BitTorrent through several metrics, concluding that both a high throughput per client during download and the ability to sustain high flash-crowds [100]. Arnaud Legout et. al. spread all the success of choking algorithms via several experimental setups [111]. Specifically, the choking algorithm enables clustering of similar-bandwidth peers, fosters effective sharing incentives by rewarding peers who contribute, and achieves high peer upload utilization for the majority of the download duration. Additionally, a bad-provisioned seed can be compensated by clustering, which is achieved by the BitTorrent-choking algorithm. Legout, Urvoy and Michiardi present a further experimental work on BitTorrent. They guarantee close to ideal piece diversity with Rarest First, and network resilience to free-riders thanks to choke algorithms. Therefore, the authors conclude it is not necessary to apply network coding or other cooperative scheme different to Rarest First [112]. Tewari and Kleinrock study the file-sharing efficiency of BitTorrent, extending the original work [169] of Yang and de Veciana. They easily show that with peer groups higher than 15-20, the file sharing efficiency increases with respect to the number of chunks. BitTorrent try to maintain 20 peers, so they support the protocol [206]. Qiu and Srikant [169] analyze file-sharing systems via fluid models and game-theory, providing a strong mathematical support of BitTorrent. Jiadi Yu et. al. develop a fluid model to study the evolution of free-riders and normal peers [227]. They find closed expressions for the average waiting times of normal peers and free-riders, under the assumption of regime. The results show BitTorrent is resilient against free-riding effects, and the performance of normal peers is not significantly affected by the presence of free-riders. The stability of the fluid there developed is an open problem. The authors Chehai, Xianliang and Hancong propose a mathematical framework to maximize the global availability in the network [35]. They measure availability to the capacity of exchanges. The problem turns to be NP-Complete. However, they prove that random overlays are likely to achieve near-optimal availability. The lower bound of 4 peers in BitTorrent is supported with a rigorous mathematical model for stratification [79]. P. Pouwelse et. al. study the flash-crowd phenomenon presented in the .torrent from the file "The Lord of the Rings III", and confirm the global BitTorrent/Supernova components can handle very large flash-crowds efficiently [162]. With the help of moderators in the Supernova website, fake-files are avoided. Cameron Dale and Jiangchuan Liu take logs from both the Internet and PlanetLab. Their results validate that the downloading policy of BitTorrent is quite effective from a piece distribution and evolution perspective [55]. The proportion of leechers with some piece increases with time, and follows a normal law. They assure enhancements are still possible to achieve the ideal piece distribution. David Erman, Dragos Ilie and Adrian Popescu present a statistical analysis of a BitTorrent session [70]. They considered a goodness-of-fit test more robust to parametrization errors for

large sample sizes than the classical Kolmogorov-Smirnov and Kramér-von Mises [54]. They conclude that session interarrivals can be accurately modeled by the hyper-exponential distribution, while session durations and sizes can be reasonably well modeled by the log-normal distribution, with a significance level of $\alpha = 5\%$. Independently, Daniel Stutzbach and Reza Rejaie developed a statistical analysis of BitTorrent inter-arrival and session lengths in order to understand node-churn in BitTorrent-based networks [204]. They assure exponential inter-arrival fit quite well in some torrents (but in Debian, for example), and the random variable representing session lengths is better described by Weibul or log-normal distributions.

Works that point-out BitTorrent's weaknesses are though not absent. However, some corporations support their business with traditional client-server architectures, and the results could not be completely objective. Tian, Wu and Ng empirically observe the network sometimes dies when the last seed departs the system [207]. In order to address this issue, the authors propose a trade-off between file availability and file-sharing efficiency, that replaces the tit-for-tat policy. Via simulations they show the new proposal helps to extend the lifetime of the system. Ashwin R. Bharambe from Carnegie Mellon, together with Cormar Herley and Venkata Padmanabhan from Microsoft Research also agree that the seed bandwidth is critical to conserve when it is scarce. They add that the Local Rarest First policy is critical in eliminating the last block problem and ensuring that arriving leechers quickly have something to offer other nodes [20]. The same authors conducted experiments and conclude BitTorrent is near optimal, but tit-for-tat is not as effective, failing to prevent unfairness [19]. Eytan Adar from Hewlett Packard considers a simplified BitTorrent-like system with one seed that owns M pieces, and leechers arriving/departing the system stochastically [2]. He measures the performance of the system by seed overload, upload/download peer ratio and average download time, and suggests that the poorest first strategy (i.e. always to send piece to the peer with less information) outperfoms both random and other preferential strategies, concluding that the BitTorrent protocol deserves further improvements. The literature even reveals careful design of publicly available malicious clients, trying to show some BitTorrent deficiencies. Michael Piatek et. al. develop a strategic free-rider for BitTorrent-based networks, called BitTyrant, which is publicly available [160]. The title is a challenging question, replying to Brahm Cohen [44]. Thomas Locher et. al. develop BitThief [127], which is available in the web [22]. The authors affirm they can outperform the streaming rate of other clients while not contributing even a single bit to the system. The paper motivates to feel guilty when sharing because of the distribution of copyrighted media content is unlawful in certain countries, hence promoting the use of centralized systems. It is interesting to mention that one of the authors works for Google corporation. A counterattack is presented by Nikitas Liogkas et. al. The authors present three BitTorrent clients that attempt to abuse existing protocol mechanisms in order to achieve higher download rates. Although in some cases the exploits delivered significant benefits, BitTorrent proved to be quite robust against them [121].

1.4 Conclusions

The makespan (minimum download time) for cooperative sharing swarms is a logarithmic function with the population size, versus a linear relation in traditional client-server systems. Additionally, peer-to-peer networks show to be resilient against flash crowds.
Via multi-part downloading the cooperation is more dynamic and possible as soon as one part is owned by a peer. Therefore, real-life platforms split files into different pieces, called chunks. Pure peer-to-peer networks (completely decentralized) rely on flooding to sustain file search, and their overload has serious drawbacks, without mentioning the problem of malicious peer. On the other hand, centralized systems suffer from limited scalability. The presence of a tracker in BitTorrent facilitates the coordination of incentives to promote peers to cooperate. The local Rarest First policy is intuitive, in order to balance the chunk availability in the network. It has received merits from both mathematical models (showing near-optimal performance) and experiments. The tit-for-tat and optimistic unchoking policies are hard to model. However, it is proved experimentally that they work quite well, and have game-theoretical reasons for its use. Random networks have strong connectivity properties. Free-riding and starvation are special causes of concern, and BitTorrent addresses them via incentives and the end-game mode. It can be observed that few models take the underlying network failures into consideration. The assumption of perfect network is widely adopted, mainly because it is negligible with respect to the challenging design issues of peer-to-peer networks.
Summing up, there is already a vast comprehension of the BitTorrent's success for file sharing purposes. Some elementary ideas are useful for all streaming modes, as multi-part downloading, pipelining, early cooperation, incentives to encourage peers to cooperate and the trade-off between payload and overlay information. All these tools are naturally exploited in live and on-demand streaming services, and its main features and flavors are discussed in Chapters 2 and 3.

Chapter 2

On-Demand Video Streaming

2.1 Introduction

In file sharing, a user should wait until the file is fully stored in its personal computer. The behavior of a user under a video on-demand (VoD) service is quite different. The user pretends to choose some movie, enjoy it once he executes a click on it, and pause, fast-forward, rewind (as in a Video Cassette Recorder, VCR), disconnect or switch to other movies when he or she wishes, with a corresponding fast answer of the video player. The network design must be effective, bandwidth efficient and transparent to the user.

There are several challenges shared in off-line downloading and video on-demand. For instance, node churn, heterogeneity and malicious behaviors are causes of concern in both services. However, there are additional challenges inherent to on-demand video streaming. Users join the system at different times, so their video players work in different times (they see different parts of the movie in a certain instant). This asynchronous nature of VoD imposes a difficult cooperative scheme in peer-to-peer architectures. The amount of video content in the user's storage is asymmetric. Therefore, younger users (the ones that recently joined within the system) cannot contribute to the system. Moreover, older users will surely watch advanced instants, so they will never receive useful information from a younger user, unless a clever strategy is applied to accelerate the growth of joining peers. The information usually describes directed forests, rooted in the server. The bandwidth of the youngest users is a resource sometimes hard to use. A remarkable corollary is that the tit-for-tat policy widely applied in BitTorrent is not practical, and the introduction of peer incentives to cooperate must be re-designed (considering paying, punishing, quality differentiation or a combination). Additionally, the video player must receive chunks in-order. However, a greedy in-order chunk scheduling policy does not maximize the availability of information, because younger users will never upload resources. In VoD, the Rarest First policy is highly related with a concept called *caching with prefetching*, which is basically to request for chunks far away from the current video playback (which are usually the rarest in the system). Peers that are not necessarily on the same part of the video may have chunks to exchange because of the data stored in the cache. Clearly, prefetching facilitates availability and cooperation, and trades delivery ratio for buffering times.

However, the dynamic behavior of users sometimes makes the prefetching technique a wastage (because of departures). The Rarest First policy carries unacceptable buffering times, as several works confirm. Summing up, the major concerns in a P2P VoD service is the caching policy, the overlay building to facilitate chunk diversity and the chunk scheduling policy to both attend urgencies and assure the global diversity of the whole buffer map within the nodes. The result form the user's viewpoint should be a high quality of experience (related with low buffering times and continuous media playback).

This chapter is structured as follows. Section 2.2 presents a historic-driven motivation of the inclusion of peer-to-peer architectures for the deployment of on-demand video streaming, discussing outstanding contributions in this area. The chore design and operation of VoD-PPLive is illustrated in Section 2.3. Section 2.4 contains a revision of mathematical modeling of peer-assisted video on-demand systems. Concluding remarks are included in Section 2.5.

2.2 Historic Motivation

On-demand video streaming was born in the 90's, with one-to-many architectures as the client-server and IP-multicast [92]. The evident weaknesses of those systems (single-point of failure and non-scalability) became obvious with the increasing popularity of video contents through the Internet. The requests in IP multicast are assumed synchronous, but VoD is asynchronous in nature. Batching [56] and Patching [87] were two alternative proposals. Batching groups requests to execute them together, carrying high latencies. In Patching clients receive a patch of the video stream by means of a unicast server, which can easily be overwhelmed with patching requests [41].

To cope with network layer problems, application layer multicast was introduced, in an infrastructure or peer-to-peer version [95]. The infrastructure base is supported by overlay routers, which need to have maintenance. Chaining is the first collaborative system that offers video streaming to end-users, developed in 1996 [196]. The main idea is to construct chains so that users can relay the media stream. The developers of Chaining do not explicitly mention the concept peer-to-peer, and they even do not include a failure recovery. Nevertheless, that work is extremely mature.

Mohamed Hefeeda, Bharat Bhargava and David Yau describe unicast systems (single-source, proxies and CDNs) and multicast systems (IP multicast, application level multicast) in detail [92]. They conclude the load distribution of those systems is not acceptable for high-scalable on-demand media streaming. Therefore, they propose an efficient distribution of media streaming by means of a *hybrid* architecture, combining scalability aspects of peer-to-peer computing with swarm assistance provided by powerful super-peers, which help with item search and forwarding. This hybrid architecture can handle flash-crowds in a cost-effective fashion, as experiments confirm. The power of peer-assistance for on-demand services became obvious, but the design imposes several trade-offs, and the user behavior should be first understood.

Hongliang Yu, Dongdong Zheng, Ben Zhao and Weimin Zheng offer a complete statistical analysis of the user behavior in a high scale on-demand system from China Telecom in 2004,

with a client-server architecture and a total of 1,5 million users [226]. Quite predictably, the users connect massively during break-times and after work (7 p.m. - 10 p.m.), but practically do not exploit the VoD system between 0-7 a.m. The joining and departure rates have a periodic pattern, with a period of one day. The arrival process is non-Poissonian, but the authors propose a modified Poisson distribution. They could also confirm the 80-20 Pareto behavior in popularity: an 80% of requests correspond to the most popular 20% of video files. Besides, the video popularity matches the Zipf distribution, as shown via the Kolmogorov-Smirnov goodness of fitness for one sample, with a level of significance of 5% [32]. The users take their time to scan several items in order to select one, and the most popular are usually the shortest video clips. The authors discovered there are external factors (recommended videos, new available videos) that greatly impact to define the video popularity. Interestingly enough, the operator has some control on the popularity.

Both analytical and experimental results converge to the fact that peer-assistance offloads the server and provides high scalability. Paradoxically, the most successful VoD service in the world is expected to work far away from optimal. Established in 2005, YouTube has become the most successful Internet site providing a new generation of short video sharing service, comprises approximately nearly 10% of all traffic on the Internet [40]. However, the network access is yet working with a client-server architecture, and the operator (Google corporation) must afford more than one million dollars per day just for bandwidth requirements, which is a clear motivation to add cooperation in YouTube's users [96].

The first analytical model for a mesh peer-to-peer VoD systems is presented by Yue Lu, Jan David Mol, Fernando Kuipers and Piet Van Mieghem [128]. With a similar spirit to other fluid models, they study an on-demand peer-to-peer system in phases: start-up, first seed appearance, first seed departure and steady state. The authors confirm the seed departure rate is critical for the life and stability of the system.

A valuable step towards the deployment of VoD services with the peer-to-peer philosophy is the experimental exposition of benefits obtained from Microsoft Research. The authors Cheng Huang, Jin Li from Microsoft Research and Keith W. Ross from the Polytechnic University of Brooklyn, present the first measurement of a large-scale video on-demand system, using real traces taken from a client-server from MSN videos, and simulations of a peer-assisted network [96]. This paper supports the peer-to-peer architecture is promising to save bandwidth even in an ISP-friendly way, providing better streaming rates. Developers had that prediction clear, since several P2P-VoD systems were currently deployed at the same time of the publication. The authors measure the 95 percentile of bandwidth usage from nine months of study, counting 52 million video requests. They found a 95 percentile bandwidth of 2.2 Gbps in December 2006, which potentially could be reduced to 79 Mbps with a peer-to-peer architecture combined with prefetching, as simulations predict. The simulated strategy uses prefetching when there are idle uploading resources, which are distributed either from older to younger generations or via water-filling. The authors also prove both strategies achieve near optimal results, in the lights of a simple mathematical model and simulations. An ISP-friendly P2P system can both reduce cross-traffic and save bandwidth.

Several real peer-assisted platforms were developed for VoD services at the beginning of the 21st century [39, 41, 57, 69, 87, 90, 113, 146, 154, 161, 172, 214], with variable complexity and

self-sustainability. Anwar Al Hamra, Ernst W. Biersack and Guillaume Urvoy-Keller present a pull-based approach (PBA) [90]. A joining user sends requests to the server, unless it is overloaded, in which case the server responds with a set of servants. The authors defended the algorithm is easy, but it has clear weaknesses inherent of centralized architectures.

Lei Guo, Songqing Chen and Xiaodong Zhang present PROP, which is not entirely peer-to-peer [113]. They argue that the disadvantage of a client-based P2P system is its unreliable quality of streaming media delivery due to the dynamic nature of peers. P2Cast is a patching scheme for peer-to-peer networks, which disseminates fragments in an application level multicast tree structure, looking for the best fit failure recovery. An entering peer is a leaf-node of the tree, and obtains a *patch* (the first video chunk). However, the server both distributes the streaming and keeps track of the clients offering the first path, becoming a single contact point [87]. Similarly, PALS [172] and CoopNet [154] rely on the server too often, hence do not exploit the cooperation efficiently. On the other hand, Cui et. al. propose oStream, a multi-source distribution to alleviate the server, and performance via simulations [50]. Chen et al. describe a topology-aware algorithm that encourages peers to contribute providing economic incentives [37]. Several works try to extend the BitTorrent success from file sharing to VoD services. However, they either publish ad-hoc solutions or give partial information, via simulations. BitOS defines a nice trade-off between urgency and rarest pieces, but lacks of a backing server [214]. The purpose of Toast (Torrent Assisted Streaming) is to reduce the burden of a VoD server with a BitTorrent extension [41]. Their experiments confirm the Rarest First policy is not suitable to address time urgencies. Though its simplicity of extension, the authors conclude that Toast can alleviate the load of a VoD server between 70% and 90%. BitTorrent-Assisted Streaming System (BASS) is similar to Toast but more conservative in its extension [57]. The server disseminates the stream using a rarest-first policy, which induces buffering times. Despite, the authors show a linear load of the server better than a pure client-server architecture. Another BitTorrent-based system is called BEST (BitTorrent Extension for Streaming). The authors compare chunk selection policies via simulations, and the VoD design is partially covered [69].

A new peer-selection policy for mesh-based P2PVoD systems is introduced in 2010 by Igor Moraes and Otto Carlos Duarte [146]. The authors explain the random peer selection in a mesh-based VoD is not as efficient as in live streaming, by the asymmetric state of peers. They propose a LIfetime-based Peers Selection (LIPS). The goal is to maximize the probability of choosing partners interested in the same chunks. The parents are chosen with similar lifetime. Therefore, it is highly probable they should be able to cooperate, and they can also discard the old cache, thus reducing the cache needs in peers. The authors show improvements when compared only with a random selection policy in simplified simulations. However, to their knowledge, they claim to propose the first implementation of a peer selection policy in a VoD system. A nice concept to tackle free-riding is Give To Get (G2G [144]). The authors propose a peer selection policy to encounter an analogous for “tit-for-tat”, in a video on-demand system. Basically, every peer has a neighbor list and builds a ranking with them, according to the number of video chunks offered to the system (a level of altruism). In order to avoid lies, the local peer does not ask directly to their children (potential downloaders), but to their grand-children. The local peer pushes video chunks to the most scored peers first, and updated the

neighbor list in a BitTorrent-based fashion (it periodically selects the best 4 in the ranking, with an optimistic unchoking). Therefore, peers have incentives to cooperate, and the G2G policy works indirectly. Three chunk policies are considered, with low, mid or high priority to the order. The G2G system is compared with BiTOS via simulations, with video clips of 5 minutes and Poisson peer arrivals (departures at the end of the playback). It can be noticed that G2G punishes free-riders, and its performance is better than BiTOS when measuring both buffering times and playback losses. However, G2G introduces higher overheads, that were neglected in the simulations. Additionally, the performance degradation is unavoidable under the presence of free-riders (representing a 20% of the population), requiring 33 seconds of buffering versus 14 seconds without free-riders. However, these non-cooperative peers had to wait 89 seconds of buffering time on average. The authors added G2G into Tribler and contrasted this peer selection policy with a proportional uploading (the notion of tit-for-tat), showing important benefits of the new proposal [53].

A third alternative to mesh or tree-based topology is Rindy [39]. The main idea is to construct a logical ring-based concentric topology, with power law radius, and peers within a same radius share the same length of buffer video. Rindy is designed to offer VCR functions with low overhead and therefore reduced buffering times. Neighbors periodically exchange announce messages with a snapshot of their buffers, and forward messages to send video chunks. With the help of simulations the authors show a reduced overhead when compared with tree-based systems, a server offload of 90% and better start-up latencies than pure client-server architectures.

Another sophisticated design is called P2VoD, which tries to be a fault tolerant peer-to-peer VoD service [62]. It assumes a single unicast server, and introduces the concept of *generations* of peers. Two peers are in the same generation only if they posses the same older video chunk. Each peer sends requests to their parents: peers from the nearest older generation. In this way, the data traffic travels in a multi-tree topology. Peers cache only the latest chunks, hence it works with limited memory resources. The peer selection policy is not periodic, and uses polling: round robin, least latency or least loaded parent. The three peer selection policies have similar performance, and the peer selection with least latency has been chosen. Children must know the IP addresses of the previous generation of peers. In order to recover from network failures, P2VoD applies the same technique proposed in P2Cast, based on primitives that are used to communicate children with their parents (a test message, recover and wait). The protocol tries to minimize the load of the main server. Simulations were carried out considering Poissonian arrivals of peers randomly distributed in the network, and entities of the underlying network (backbones, routers and shortest routing). They show that the server is more stressed when the workload is reduced. This counter-intuitive result is a corollary of the multi-tree structure: more joining users will contact the server when they have no parents (a low arrival rate is translated into high distant peers in the genealogical tree). Simulations suggest the new proposal outperforms P2Cast.

A considerable different approach based-on network coding is given in [197]. The authors (from Microsoft and Telefonica) support that mesh-based P2P systems perform poorly when used for VoD. However, PPLive-VoD does not include network coding, and seems to be an

exception to that rule.

There are at least three insightful reasons to understand PPLive-VoD. It is the first high-scalable peer-to-peer VoD system reported in the literature. Its design is so complex that it revisits practically all challenges in the VoD design. Additionally, the live service of PPlive is proprietary. The study of their VoD service give perhaps some hints of the design complexity of PPlive, whose success for live streaming is nowadays a mystery for the scientific community. Section 2.3 describes the PPLive-VoD architecture and its main design ingredients.

2.3 PPLive-VoD

This section presents a summary of the work from Yan Huang, Cheng Huang (from PPlive), Tom Z. J. Fu, Dah-Ming Chiu and John C. S. Lui from the Chinese University of Hong Kong [98]. They achieved millions of concurrent users enjoying a VoD service with some interactivity, acceptable quality of experience and reduction in the burden of servers.

The authors first explain basic components present on the PPLive-VoD system: a bootstrap with a distributed hash table, trackers, servers and peers. The data is encapsulated with high level of granularity. Video chunks are divided in pieces, which are subdivided in sub-pieces. The system exploits the user's cache massively. Peers should have 1 GB of free-space to start the VoD-PPLive client, and the system works via Multiple Video Cache (MVC), meaning that peers could be downloading and contributing with movies that are not being watched (the cache accepts multiple movies). A user can surprise to notice that after a session, he or she has another movie in his hard drive disk. Curiously, they do not use any pre-fetching technique. The authors have at least two reasons to support this decision. It is known that most peers do not connect more than one hour. Therefore, pre-fetching could waste precious bandwidth resources with high probability. Moreover, more than one-half of the connections from China are ADSL, and the upload capacity is affected by concurrent downloading. A related issue is which movie we must discard if the cache is full. Common approaches tend to discard the least recently used (LRU) or least frequently used (LFU) movie. At the beginning PPLive-VoD used LRU, but after further studies they changed by a weighted function, which is inversely related with the *needs* of that movie. The authors introduce an Availability to demand ratio (ATD): if c peers own movie m and n peers need it, then $ATD(m) = c/n$. The weights are not specified by the authors, but they assure a movie might be discarded without losses when $ATD > 8$. The best change in design so far has been this weighted discard, reducing the load from 19% to 7-11%. The content discovery (item lookup and advertising) strategy adopted in PPLive-VoD is both complex and instructive. The authors observe that all peer-to-peer networks use at least one of the following rules for content discovery:

- Tracker (sometimes called *super-node*)
- Table (DHT), or
- Gossip protocol.

The different strategies offer different level of robustness and freshness. Interestingly enough, PPLive-VoD combines the three strategies for different purposes. The node-trackers are used

to track the movies owned by different peers, as well as to measure the *health* of the system and statistics. The buffer-map is exchanged among neighboring peers to find useful chunks, with gossiping in a pull-based process. Finally, trackers (and also peers in the current version) are assigned movies with a distributed hash table (DHT). It is worth to mention that there is a hidden notion of redundancy: if a tracker fails, peers are able to find the movie switching to the gossip strategy to that purpose.
The piece selection policy applies first greedy requests (sequential download governed by urgency) followed by rarest first requests. This policy seems to be in agreement with BitOS, but the authors do not give more details of this trade-off. Other systems study the user's behavior to determine outstanding clips in a whole video and predict in advance a high probable skip of parts of a certain movie (in other words, a fast-forward), to support VCR. Once a user skips a part, the system should track a pre-specified near point (called *anchor points*) to re-start buffering, as fast as possible. PPLive-VoD does not apply anchor, because the user interactivity is not often (just 1,8 times per movie), and the user should wait 18 seconds in average for each jump [1].

There are aggressive transmission techniques (request the same chunk to all neighbors) and conservative (non-concurrent requests). Naturally, the latter does not exploit the bandwidth resource efficiently, whereas the former carries unnecessary retransmissions. PPLive uses an intermediate approach, in which different contents are simultaneously requested to different peers. The mass of requests is proportionally shared regarding round-trip-times (i.e. better RTT means more requests). The authors heuristically found their optimal size for the neighbor list, between 8-20 for streaming rates of 500 Kbps, and 16-32 for 1Mbps. Data integrity is added at a chunk level by message digest, to avoid pollution attacks [60]. Recognition of NAT peers and Firewall is extremely useful (between 60% and 80% users are behind NAT). A weaker form of piece-level authentication is applied in PPLive-VoD. The quality of experience is poorly captured via a concept of *fluency*, which represents the ratio of watching time and connection time, using statistics taken from the trackers. They finally measure the health of the system, success of the replication strategy and user's behavior. Most users download less than ten minutes per content (because of scanning). Despite most users have high fluency, one-fifth of them have poor fluency. The authors argue the buffering times are long for some users, and this point need further research. The reader can find experimental results and discussion in the paper [60].

2.4 Mathematical Models for On-Demand Video Streaming

The first analytical model for a mesh-based P2P-VoD system is introduced in 2008 by Jue Lu et. al. [128]. It is similar in spirit to the fluid model from Qiu and Srikant [169], but proposes some ingredients inherent to VoD systems, with different file size, variable arrival rates and seeders aborting the system as a function of time, which makes it non-linear. They find closed expressions for the average excursion of different downloaders, and study the peer evolution in phases: *i)* start-up, *ii)* first seed appearance, *iii)* first seed departure and *iv)* steady state.

[1] The reader might wonder if he would accept 18 seconds to re-start watching the movie. The system here presented refers to 2008, and it is not rare to guess several enhancements up to date.

Peers cannot skip parts of the movie or download different video-types. The authors prove the equivalent linearized version of the model is exponentially globally stable. However, the linear version is not suitable for aggressive seed departure. The authors conclude the seed departure plays a critical role for the stability of the system.

A probabilistic model for P2P-VoD systems is introduced by Yipeng Zhou, Tom Fu, Z. Chiu and Ming Dah [235]. The scenario is quite simplified, assuming a closed system with peers wishing videos movies of identical sizes and poissonian number of requests incoming to all peers. Despite, they prove an amazing result under these circumstances. In order to minimize the server's load heterogeneous scenarios behave exactly as an equivalent homogeneous system. An Adaptive Random with Low Balancing is proposed and outperforms the PPLive-VoD design, as simulations show. However, it is not so realistic and needs global network information. The assumptions are similar to works from Di Wu, Chao Liangz, Yong Liuz and Keith Rossy [219, 220]. They Propose View-Upload Decoupling (VUD) to re-Design multiple channels. VUD strictly decouples peer downloading from uploading, trying to bring stability to multichannel systems and enabling cross-channel resource sharing. The authors propose peer-assignment using different substreams (swarms) in order to minimize the bandwidth overhead. This technique is though inspired in live streaming. In fact VoD shares challenges with live: sequential playback delivery, dynamic churn of heterogeneous nodes. However, VoD is a multiple-source system with retrieval and request diversity. Recall that the tit-for-tat policy is impractical, due to the asymmetric relation of exchange (an older peer is never benefited by a younger one). By means of combinatorics and simple probabilistic distributions, Nadim Parvez, Carey Williamson, Anirban Mahanti and Niklas Carlsson show that naive piece selection strategies (random and in-order selection) have either poor sequential progress or unacceptable expected download times [158]. The authors help the understanding with intuition: older peers receive too many requests from younger peers, but only can serve few of them, and with in-order selection the idle up connections of younger peers are wasted. Consequently, young peers consume scarce resources of seeds and senior peers, impeding the progress of middle-aged peers [158]. An in-order First-Come First-Served (FCFS) queuing model finds a trade-off, with lower start-up delay but similar download time compared with Rarest First. Valuable mathematical analysis is there presented, including simulations with close agreement. Several authors are based on a live-streaming result to extend the fact that the file-exchange efficiency is near the unit [206]. Some works are theoretically sound but far from practical in VoD systems. The authors use a layered coding network to maximize the streaming bandwidth [51]. They prove Heterogeneous-Rate Layer Allocation problem is NP-complete, and find an algorithmic resolution. However, they particularly focus on the streaming problem, but do not address caching techniques or interactivity, which are key elements in the design of VoD-P2P systems.

2.5 Conclusions

Nowadays, the VoD service is immensely popular, with YouTube, Google Videos, MSN Videos and PPLive-VoD. There are strong evidences to support the incipient efforts of the peer-to-peer design for VoD services are in the right way, both promoting scalability and offloading the server. However, the design of P2P-VoD imposes technical challenges. BitTorrent has been widely adopted for file sharing purposes, but it cannot be used in its primitive specification, mainly because the tit-for-tat and rarest first policies are not suitable. BitOS shows a trade-off between availability and urgency, combining the rarest first policy with greedy requests, as an alternative chunk scheduling policy. The authors of Tribler propose Give-To-Get (G2G), that nicely explores the asymmetric nature of peers to ranking potential downloaders using a gossip-protocol, polling all its grand-children.

Researchers from PPLive and the University of Hong Kong conducted the first practical design and measurement issues deployed by a high-scalable real-world P2P-VoD system (2008). The peer-to-peer introduction to the VoD world is practically in an experimental stage, and the literature reveals scarce mathematical foundation of these systems.

Chapter 4 proposes a mathematical framework to understand trades-off in the stability and capacity of peer-to-peer systems for Video on-Demand. There, we will confirm the strength of the peer-to-peer philosophy to address flash crowds and offload a cluster of servers. Additionally, a combinatorial optimization problem is proposed to determine an optimal Multiple Video Cache (MVC) solution.

Chapter 3

Live Video Streaming

3.1 Introduction

In live video streaming, the generation, dissemination and playback occur simultaneously. As a consequence, the system must comply with stringent timing requirements, and users should be synchronized in the playback, which is a characteristic difference with respect to on-demand video streaming. In live peer-to-peer systems, the core mechanisms are scheduling policies, specifically, the neighboring policies which determine the overlay structure and the chunk scheduling policy, which trades efficient bandwidth allocation, latency and overheads. A general rule is that the availability of information always helps to make better decisions (neighboring and scheduling policies). However, the knowledge and overlay awareness come at a cost, which is major overhead in the global system, and possibly the need of centralized entities. Successful commercial platforms like PPLive [163], PPStream [164], TVAnts [209], TVUNetworks [210] and SopCast [200], offer live video streaming to several thousands of users, with proprietary protocols not revealed in the literature. A different paradigm is offered by GoalBit, an open-source peer-to-peer streaming platform that widely delivers live streaming to end users [18].
The available literature related with live streaming in peer-to-peer networks is really vast. In this chapter we discuss the main features of scheduling and general aspects of live peer-to-peer systems. The design of resilient live-streaming services must face all those features together. However, in order to keep simplicity we will cover them one-by-one (in fact, the public literature rarely focuses on several challenges simultaneously).

This chapter is organized as follows. Section 3.2 presents a performance comparison of tree-based and mesh-based topologies. An analysis of pull and push schemes, their combinations, pros and cons, are presented in Section 3.3. The capacity and performance of live streaming systems is briefly discussed in Section 3.4. A revision of existing chunk scheduling policies is provided in Section 3.5. Peer neighboring policies are covered in Section 3.6, whereas an overview of incentive policies to encourage contribution are presented in Section 3.7. Section 3.8 describes two real platforms for live streaming: GoalBit and PPLive. GoalBit is a BitTorrent-based open-source platform that currently offers live streaming to end-users, and is

considered as a benchmark in this thesis. PPLive, on the contrary, is a commercial widely used platform with proprietary protocols. We describe briefly the Goalbit protocol with scope in its scheduling policy, and show hints of experiments from PPLive. Finally, Section 3.9 contains the main conclusions of this chapter.

3.2 Tree or Mesh?

In a tree-based peer-to-peer approach, participating peers logically organize in one or several trees. In streaming, the single-tree approach is discarded, because the delay increases linearly with its size and the bandwidth resources are not appropriately exploited. The multi-tree topology is just an extension of end system multicast [95]. Each tree is rooted, and the seed pushes video chunks towards the different branches. All peers are within diverse directed trees, and they participate as internal nodes only once, and as leaf nodes in the others. The content distribution mechanism is extremely simple: peers push messages to the children peers as soon as they receive it. However, the tree construction and repairing algorithms show to be challenging. As in file sharing, the diameter of the network should be reduced in order to keep latency under control. Additionally, the fan-out degree should be roughly balanced in order not to load excessively certain nodes and make the stability of the system vulnerable to node churn. On the other hand, the mesh-based approach is inspired in BitTorrent, where peers connect in a random fashion. The main idea is to inherit the stability and robustness of random networks [25]. Peers cooperate in an epidemic style, where the information is propagated sequentially, and chunks are intended to reach the whole peer population.
Tree-based systems are typified by SplitStream, which is proved to be optimal in an homogeneous scenario with no churn [31]. A joining node is a leaf of all trees, but one. If all trees are regular and peers were homogeneous, with $m+1$ directed trees every peer uploads a streaming rate of λ and recovers a rate of $m \times \frac{\lambda}{m} = \lambda$. The construction in SplitStream tries to build balanced trees. A worst case scenario occurs when a huge amount of internal nodes depart the system, which can be catastrophic, and is called *deadlock event* [132]. In those cases leaf nodes periodically try to reconnect to the tree, and the payload can be surprisingly high. An illustrative example of mesh-based live peer-to-peer platform is PRIME [133]. There, peers report their status with push messages and the children request for chunks pulling from their parents, in accordance with their ranking weighting bandwidth and availability. A comparative study of SplitStream and PRIME is given by Nazanin Magharei, Reza Rejaie and Yang Guo [132]. They point out a key difference: the mesh-based approach is able to give a dynamic shape to the different delivery trees, in a bandwidth-driven fashion. The authors conduct a fair comparison of both systems via simulations, and show that the mesh-based approach consistently outperforms trees, regarding different metrics as bandwidth utilization, average delay and resilience to node-churn.

3.3 Pull or Push?

Sanghavi, Hajek and Maussolié describe push, pull and hybrid schemes probabilistically, under different scenarios [190]. They assume soft or hard uploading constraints and one-sided

or two-sided algorithms (where the information is needed in either one or both terminals for each request or forward). In the soft constraint the upload capacity is unlimited, whereas in the hard constraint peers have unit upload bandwidth. Their results are remarkable, and suggest push schemes have fast dissemination at the beginning but the ending even may not come (for instance and intuitively, in a highly biased pushed selection). On the other hand, pull schemes start slowly but the time completion is accelerated at the end. The authors introduce *Interleave*, which basically applies a push scheme with nearest-to-the deadline first in even slots, and a pull scheme requesting the latest chunk not in the buffer, in odd slots. Additionally, The source pushes a new piece in every even slot, and it replies to pull in odd slots. The performance of Interleave is within a constant factor of 3,2 times the best completion time. In two-sided algorithms, the payload is increased at the cost of near-optimal performance. In [126], the authors relax synchronization and homogeneity assumptions from [126], and wonder whether Interleave can be extended or applied into a stringent real time application. They outstand the decentralized nature of Interleave, and show an extended version for a more realistic scenario, and point-out Interleave could be flavored with efficient requests with two sided protocols avoiding the waste of some time-slots, hence being a lower bound in the achievable performance. It is worth to notice that both papers focused on time completion, but do not measure losses with a playback delivery ratio, which is a fundamental impact factor of live streaming.
In this thesis we will stick to the two-sided pull-based approach, for its efficiency, its payload trade-off and the huge empirical evidences of applicability in real platforms confirm [18, 163, 164, 210].

3.4 Streaming Rates and Delays in Live Streaming

Yong Liu presents additional reasons to prefer mesh-based rather than tree-based topologies [123]. With a quite elementary systematic analysis, the author provides the minimum and worst-case achievable latency in peer-to-peer systems in an elegant way, in both homogeneous and heterogeneous environments. Actually, with a discrete-time model it is proved that the best branching process is achieved by means of a *Snow-ball* algorithm, in which the accumulation of the aggregated uploading bandwidth mimics the formation of a snow-ball, in a mesh-based scenario [123]. The dynamic delivery tree makes the mesh-based approach near from the applicability of this optimal delay algorithm. Remarkably, it is also proved that heterogeneous networks always achieve lower latencies than its corresponding average homogeneous network. Intuitively, the bandwidth of the most resourceful peers are available first, and they disseminate the message faster. Yong Liu also shows that a network with dominant propagation delay has lower performance but the bandwidth at disposal increases exponentially again, with lower rates. The bounds here obtained are suggested as reference for future design of distributed protocols. Jun Luo cites Snowball, a dissemination protocol that achieves the best delay, and points out a weakness: it is completely centralized, thus impractical [130]. The intuition is replication and relay to newest peer, so one message could be sent in $O(log_2(N))$ time slots to N peers. If we could *guess* which peers do not have the message, a selective push would achieve. However, that binary information needs global availability. First, an ideal centralized solution with multiple trees is proposed, and then it is extended to a distributed fashion with se-

quential relay with round-robin. The server also pushes messages to the roots of different trees in a round-robin fashion. In order to contrast its performance, the authors construct an underlying topology with GT-ITM [86], and consider not only the best Snow-Ball delay but PPlive delay, based on an insight of this proprietary protocol [93]. The new algorithm outperforms PPLive.

Other works consider separately the propagation delay (or path delay) and the chunk-scheduling delay (i.e. the request time), trying to reduce the propagation delay with a network-aware methodology [193]. Another common approach is to exploit network locality, picking neighbors from the same autonomous system whenever possible [42, 221]. For instance, the Proactive Provider Participation methodology (P4P) specifies a bilevel optimization problem to minimize the maximum link utilization, keeping the max-flow between peers within the network. The P4P problem has a fully polynomial time approximation scheme (FPTAS), and the corresponding solution figures-out tempting bandwidth savings and better global performance [14, 15]. David R. Choffnes and Fabián E. Bustamante present a latency-driven neighbor selection policy, that avoids costly cross-ISP traffic without sacrificing system performance [42]. The basic idea is to manage a set of CDNs, and count the number and time of DNS lookups for different peers. Via ICMP pings the RTT is measured, and a score vector is assigned for each peer, whose coordinates are different CDNs. The scores are based on the percentage of time connected to each CDN (in practice, different Class-C IP addresses). Two peers are encouraged to communicate if their vectors are similar (specifically, if the dot product is low). The authors developed Ono, a Java plug-in for Azureus, and deployed extensive experimental studies. They came-up with reduced cross-ISP traffic with insignificant overhead.
A complementary approach is to minimize the chunk scheduling delay. Zhengjun Chen, Kaiping Xue and Peilin Hong introduce a chunk scheduling policy pressed by delay and urgencies [38]. Using elementary primitives, peers request the nearest-to-deadline chunks first, and avoid retransmissions with decline messages. They theoretically prove the delay achieves the lower bound proposed in [123] for homogeneous scenarios, and via simulations they confirm near optimality for heterogeneous cases. The authors focus on delay, and do not study playback losses. However, they simulate the Rarest First policy, and show it has high delays.

Laurent Massoulié, Andy Twigg, Christos Gkantsidis and Pablo Rodríguez propose randomized decentralized broadcasting algorithms to maximize the streaming rate, and prove their optimality analytically [136]. Formidably, they prove a more general version of the classical Edmonds' theorem, which states that in a single-source edge-capacitated network, the maximum streaming rate is exactly the min-cut for the node source. Edmonds proved his result in a centralized manner using packing edge-disjoint arborescences, which is not suitable for peer-to-peer systems for instance. Massoulié et. al. developed a completely distributed algorithm in which nodes forward packets randomly without duplication. The unit transmissions conform a Markov Chain (all edge capacities are integer), and the authors prove this chain is ergodic whenever the transmission rate is below the min-cut. In the limit of equality, Edmonds' theorem is retrieved. They also prove that random forwarding to the most deprived peer achieves the optimal streaming rate in a node-capacitated fully-connected network. The results have theoretical importance by itself, and the authors correctly believe they give hints for the design of practical peer-to-peer systems as well. Additionally, they studied node capacitated systems

and propose a linear programming problem to maximize the streaming rate. In an optimistic viewpoint, distributed systems can achieve optimal performance.
Bonaldy, Massoulié, Mathieuy, Perino and Twigg developed a mathematically rigorous model to study the scheduling design of peer-to-peer live streaming in an epidemic fashion [26]. The system has a single server which injects video chunks at rate λ, and peers with limited uploading bandwidth s_i. With a push-scheme, peers choose a neighbor and send a chunk regarding different policies. They prove a surprising result: the random peer selection combined with the latest useful chunk sending achieves both optimal diffusion rates and the lowest possible source delay, up to an additive constant term, when peers are identical. The authors argue that epidemic (or gossip style) protocols achieve unbeatable rates and delay. A drawback to their analysis of chunk scheduling policies is that their metrics are not suitable for live streaming protocols. The source latency does not represent the time needed to configure the buffering times (the user could find poor playback continuity in this case). Instead, a useful measure is to consider the time a joining peer needs to attain a similar performance when compared with peers within the system [233].

3.5 Chunk Scheduling Policies

Aggelos Vlavianos, Marios Iliofotou and Michalis Faloutsos even support that BitTorrent is the *King of the protocols*, but the chunk scheduling policy should be adapted for streaming purposes. They propose a minimalistic design called BiTOS (Bit TOrrent for Streaming), which tries to keep all the advantages of file sharing in BitTorrent, including a priority-based chunk policy [214]. The buffer is partitioned in high-priority and remaining sets. Peers either receive the rarest piece from high-priority set with probability p, or the rarest from the remaining set. The operator can choose the importance-to-urgency parameter p and cardinal of priority set s. In this way, the authors trade availability and playback urgency. In practice, this simple protocol extension could work for on-demand video streaming, but in live streaming the tracker cannot identify current video chunks in the system beforehand, so its protocol deserves further modifications. A valuable result obtained from simulations is that the limited Rarest First has better continuity index than a purely sequential strategy, which might result counter-intuitive. The main reason is that Rarest First tries to maximize the global chunk availability, whereas a greedy sequential request will not replicate rare chunks, which will be finally missed.

Yipeng Zhou, Dah Ming Chiu and John C.S. Lui provide the first tractable analytical model to capture the characteristic buffering-playback trade-off from live streaming peer-to-peer system [233]. The discrete-time model assumes a single server that periodically pushes video chunks in playback order to a randomly picked peer of a fully connected swarm (with identical upload capacities and buffer sizes), and the playback is synchronized between all peers. Peers cooperate exchanging one video chunk in each time slot, following a pull-based scheme. The mathematical model sounds simplistic at the first sight. However, the authors prove an identical buffer size is the only Nash equilibrium in the corresponding game of trying the best individual performance in a more recent journal [234]. In particular, a peer with high buffer capabilities will not improve its playback if it decides to use higher buffer sizes than all other

peers (intuitively, the downloading will be limited by other peers). What is more, a lack of synchronization in neighboring peers degrades the global performance, which makes the cooperative model optimistic in both aspects, and valuable to study. The authors focus on the chunk scheduling policy, inspired in previous works [44, 57, 214]. They suggest a Greedy policy (or *nearest-to-deadline*) and the classical Rarest First widely adopted in BitTorrent, which is the most representative influential work in academic circles [72]. On one hand, they show via simulations that BitTorrent is scalable but does not comply real-time requirements proper from live streaming. On the other, Greedy presents good performance for low-scale scenarios, but miserably fails with massive populations, with poor playback continuity. Therefore, they propose a Mixture policy combining the best of both worlds, showing nice properties. The Mixture policy is quite similar to the BiTOS proposal. A chosen set of m rarest chunks are requested, and immediately after the nearest-to-the-deadline chunks are sequentially requested. BiTOS breaks the buffer into two disjoint areas (high or low priority) and then picks chunks in one of the areas using the rarest first policy. A further insight in the design of chunk policies is presented in [231]. There, Bridge Q. Zhao, John C.S. Lui and Dah-Ming Chiu propose a continuous version of the model from Zhou et. al. They conduct the research with a focus on playback continuity, and find exponentially large sets, called V-shaped policies, that include asymptotically optimal policies. However, they do not cover latency aspects, and the search in the optimal sets turns prohibitive for high-resource peers (with large buffer size). A valuable insight is that the optimal solution turns more greedy when the server capacity is increased.
In a similar fashion, Ilias Chatzidrossos, György Dán and Viktoria Fodor propose a discrete-time push-based model to design chunk scheduling policies [34]. They consider an open system with churn, in which nodes connect to d neighboring peers. At the beginning of every time-slot every peer makes a forwarding decision, i.e., chooses a neighbor and a packet, and sends the chosen packet to the chosen neighbor. They also include out-dated buffer maps, which gives additional realism to the model. Via simulations they conclude random packet policies outperform fresh-data-first policies. Additionally, the possession of imperfect information dramatically deteriorates the overall performance. The authors study four policies, and the model does not seem flexible enough to treat a rich number of scheduling policies. It is worth to mention there are alternative approaches to the design of chunk scheduling policies, with layered stream [124, 125]. The basic idea is to send multiple streams with different priority, and peers could display a video stream even with limited bandwidth resources. The technique is promising to achieve convergent multi-device service for cellular systems, set-top-boxes and personal computers, which clearly have different bandwidth requirements. Although it is useful to solve heterogeneity, it still waits to changes in the client-side. Though not covered in this thesis, layered techniques offer interesting perspectives for future work.

3.6 Neighboring Policies

In order to build a robust network overlay, several approaches are analyzed, from ISP-friendliness and locality, to content availability, latency-driven aspects and clustering, contribution awareness to incentive altruistic peers with higher degrees, among many others. Usually, the bandwidth awareness provides smart designs, based on special clustering attaining resilient network

topologies. However, the global awareness comes at the cost of high overheads, which are sometimes prohibitive in systems with scarce resources and real-time requirements.
The point of departure of Darshan Purandare and Ratan Guha is the paper [199], that suggests the interaction between peers within a similar bandwidth range lead to optimal resource utilization. They propose BEAM (Bit strEAMing), a BitTorrent-style protocol which seeds resourceful peers called *power nodes*, in order to speed-up the propagation and cope with the content bottleneck problem [166]. A tracker periodically ranks resourceful peers, to choose and potentially replace power nodes, based on a score considering volume of uploaded content, in relation with its downloaded content. Therefore, peers are encouraged to contribute, so as to be directly fed by the source server. Once a peer joins the system he connects to the tracker, who sends a list of 40 peers. Peers are grouped into sub-sets of small size h, called *alliances*, and peers can be either accepted or neglected to a certain alliance, regarding a maximum number k of simultaneous alliances. Fixing the pair (h, k) appropriately, the system turns to inherit properties of a Small World Network, which include a low diameter and dense clustering [215]. The authors argue the system is both robust to network failures like node churn, and provides a fast chunk dissemination. Simulations are carried-out in a network-level, using BRITE universal topology generator [141] to contrast the QoS between BEAM and CoolStreaming. The results show reduced amount of jitter and latency with respect to CoolStreaming. However, the authors do not include the chunk scheduling sensitivity in their analysis.

Purvi Shah and Jehan-François Pâris propose modifications to the original BitTorrent protocol to extend its success for live streaming services [194]. They explain the Rarest First policy is not suitable, given its non-sequential distribution by the server. At the same time, the optimistic unchoking policy should be replaced as well, given that resourceful peers achieve higher downloads, but peer with scarce resources suffer from isolation. They introduce a sliding window to collect video-chunks, and suggest to apply the Rarest First policy within the chunks of that window. Although they recognize BitTorrent's efficiency is reduced looking for less chunks, timing requirements are addressed now. The buffering technique is combined with randomization in the peer selection during the bootstrap, and tit-for-tat under regime. The authors measure playback-delivery ratio and bandwidth efficiency (the global upload volume related with the bandwidth availability), comparing via simulations three chunk scheduling policies: Sequential request, Rarest First in the buffer and the classical Local Rarest First but with all chunks. Both the sequential request and the raw Rarest First have poor delivery ratios, whereas Rarest First combined with sliding window achieves 83,3% of delivery ratio. With separate simulations they confirm the randomized-tit-for-tat policy achieves higher bandwidth efficiency than a raw tit-for-tat. The authors do not measure buffering times, which is precisely the main drawback of the Rarest First policy, even with a sliding window.

Zhenfeng Ren, Ju Liu, Fenglin Qin and Yanwei Wang discuss the impact of node-degree and buffer size in the cooperative model from Yipeng Zhou [233], via simulations. They confirm the buffer size is a more important parameter than node-degree [173]. They also find an expression for the expected number of peers owning a fixed chunk. Curiously, if they consider a static buffer with no playback and infinite buffer-capacity, they recover a file-sharing model, in which the issue is to completely distribute the whole video with minimum time, and the playback continuity immediately lacks of interest. This observation was not further exploited by the authors, who focused the study on live-streaming, concluding the number of neighbors

does little to the playback continuity, but can reduce latencies. This result is not in contradiction with the one from [63]. Indeed, Zhenfeng Ren et. al. study the system for low node-degrees, whereas Zhou Yipeng states there is a neglecting effect above a certain threshold. Both works are complementary, and confirm there is a threshold where the network behaves as fully connected.
Menasché et. al. suggest that swarm-based systems can attain higher throughput when publishers adopt the most deprived peer selection, combined with rarest first (or random useful) chunk scheduling policy [142]. They first find an upped bound for the throughput of the system, and observe via simulations that the bound can be achieved, being that bound higher than the one attained by random peer selection policies. The system does not capture timing constraints, and is more suitable for file sharing than live streaming. Simon Koo, George Lee and Karthik Kannan support content availability is the most important factor for BitTorrent-based networks [105]. A combinatorial problem is proposed whose objective is to maximize the sum of disjointness among all neighbor peers. They solve the problem with a Genetic Algorithm-based heuristic. The authors from [35] prove the maximum disjointness problem is NP-Complete, and show a naive random overlay can build near-optimal content availability, being well adapted to the real nature of Internet.

Sylvia Ratnasamy et. al. propose a *binning* scheme for network overlay, based on minimum latency [171]. The success of this technique is tested regarding overlay construction and server selection. The authors are focused on CAN, and the scope is not live-streaming. Hassan Barjini et. al. develop a rigorous mathematical analysis of flooding-based techniques for content searching in unstructured peer-to-peer networks [10]. It includes a neat classification and analysis of redundancy. They show a probabilistic TTL limited scheme can improve the performance of traditional flooding with high TTL. However, the authors do not treat real scenarios, and its applicability to live-streaming seems far from real, given the high level of redundancy (for instance, a naive flooding in Gnutella carries 70 percent of packet redundancy, as previous works show).

Nazanin Magharei and Reza Rejaie propose an Overlay Monitoring and Repair (OMR) mechanism as a distributed and scalable approach to maintain proper overlay connectivity in swarm-based peer-to-peer streaming systems [134]. The key idea is to use delivered quality to individual peers as an indicator of poor connectivity from other regions of the overlay. The point of departure is the inclusion of traffic localization in an ISP-friendly way suggested by some works [221]. Nevertheless, the delivered quality is not implied in the objective. Additionally, peer-to-peer swarming systems are naturally organized in clusters, possibly by locality reasons or content matching in a random mesh-overlay. Basically, the repairing algorithm studies the structure of delivery trees, and gives the opportunity to resourceful nodes to relocate in a higher position, whenever its available resources cannot be gracefully exploited, in a probabilistic manner. The authors show by simulations that the QoS perceived is consistently increased in each round, and the number of repairing is decreased with time, hence the system is stable. However, they do not measure the quality of experience (QoE) of end-users, nor include node-churn. Additionally, P4P is ISP-friendly but its QoE is indeed increased, as we could confirm [14, 15].
Ana Paula Couto da Silva, Emilio Leonardi, Marco Mellia and Michela Meo study large-scale

mesh-based P2P systems pressed by timing requirements, with deterministic fluid models [52]. They discuss the performance of Random Choice (RC), Locality Awareness (LA) and a Hybrid (Hy) peer selection policy, where peers are randomly distributed in a bi-dimensional torus. Different policies are translated into special families of random graphs. The authors study the performance of the three peer selection policies regarding diffusion rates and chunk latency (inversely proportional to the RTT, or its square root), and remark the importance of peers bandwidth and location awareness to increase the network overall efficiency. The authors provide valuable insights for the design of a robust neighboring policy, which is discriminated in scenarios. Specifically, they suggest that random topologies have excellent performance when the peer upload capacity is the transmission bottleneck, which is a usual scenario for mesh-based peer-to-peer streaming systems. Location aware topologies usually take the lead when the bottleneck is due to the congestion control mechanism (consequently, when the network imperfections cannot be neglected, a structural design based on P4P provides a useful alternative). Additionally, hybrid schemes take the best of both cases. While it is never the best one, it always presents performance very close to those of the best scheme. The creation of a cluster of large-bandwidth peers can be convenient when heterogeneous peer capacities are considered. A careful design should guarantee the connectivity of large-bandwidth peers with narrow-bandwidth peers.
Our results are independent but extremely aligned with the works from Ana Paula Couto et. al. [52]. In Chapter 5, a naive random overlay is considered together with special chunk scheduling policies, and the mesh-overlay attains excellent performance, for homogeneous scenarios. Under heterogeneous systems with free-riding, we could confirm the importance of contribution awareness, and the system is highly scalable whenever free-riders can be correctly recognized. Interestingly, the best neighboring schemes in heterogeneous scenarios can be obtained with clustering, as predicted in [52]. Last but not least, we could witness an improved quality of experience when regarding an ISP-friendly bandwidth allocation mechanism in the GoalBit system [14, 15]. We believe the effects of heterogeneity and free-riding are more aggressive than network bottleneck problems, so we stick to random building in homogeneous systems, and clustering-based overlay building in heterogeneous systems [186].

3.7 Incentive Policies

Free-riding and fairness are distinguished causes of concern in the design of peer-to-peer networks. Incentive policies encourage cooperation and peers not to free-ride. A possible classification of incentive policies is based on actions targeted to individual peers: punishing, payment or service differentiation [1]. Other works classify incentive policies in distributed systems regarding direct or indirect communication among peers, in order to take actions [232]. The design of incentives is extremely diverse, and is usually inspired in game theory and economical models, where a process tries to detect anomalies or poor contributions from certain peers and react. In this section we will sketch alternative approaches, in the lights of live-streaming distribution.

The mechanisms to encourage peers to contribute has been a cause of concern in peer-to-

peer systems even before the introduction of BitTorrent. C. Buragohain, D. Agrawal and S. Suri model a P2P system as an N-player competitive game, and study qualitatively properties of its Nash equilibrium, where users only wish to maximize their bandwidth usage [27]. Sabrina Lin, Vicky Zhao and Ray Liu study the Nash equilibrium of a competitive P2P system, where peers can even cheat in order to maximize their own benefits [119]. A proof-cheat is included, where peers are encouraged to cooperate in order to achieve their objectives. However, only 2 peers are considered, in a far from real scenario. Yang-hua Chu and Hui Zhang argue altruism must by quantified, in order to address the design and performance of peer-to-peer streaming [94]. They show via simulations a considerable impact in the system's performance when the level of altruism (ratio between contribution and reception) moves from $K = 1$ to $K = 2$, with substantial improvement in streaming rates, and no damage in the dynamism of path adaptation (which would occasionally cause instability).

Zhengye Liu et. al. propose a layered incentive mechanism for pull-based scenarios, in which altruistic peers receive more layers [124]. The system forces all peers receive lower layers which are more important, where altruistic peers receive additional layers. The authors show via simulations that the layered mechanism is more effective than multiple descriptor coding (MDC).

Yang-hua Chu, John Chuang and Hui Zhang propose an economical-guided taxation model, where resource-rich peers contribute more bandwidth to the system, and subsidize for the resource poor peers [43]. This redistribution of wealth improves social welfare. The issue is to address the ill-conditioning behavior of tit-for-tat in streaming environments. The authors observe taxation provides a direct mapping between contribution and benefit, in contrast to other incentive mechanisms based on currencies [7, 85, 213, 216] or reputation [27, 88], which provide indirect mappings between contribution and benefit. They assume a rationale peer's behavior, which try to maximize the individual utility of the system (the difference between benefit and cost). In contrast, the publisher wants to maximize the social welfare of the system, and defines a taxation policy. The authors use a linear taxation, which is both simple and robust [143, 202]. They defend it is effective, and can be easily implemented in a real P2P streaming environment. To address peer heterogeneity, MDC is used, where sub-streams are sent in a multi-tree approach like SplitStream, and each peer is an internal node only once [31]. In order to study the performance of a linear taxation scheme, a lower-bound is given by the Bit-for-Bit policy, whereas a completely altruistic behavior of peers is an upper-bound. The novel taxation scheme is proved to improve the social welfare, with reasonable overhead. However, the authors recognize a structural instability due to variable rates in different paths, and altruistic peers could change their behavior because of their new rules, elements which need further research. Shuang Yang and Xin Wang propose an incentive mechanism for tree-based live streaming system [223]. The basic idea is to rotate nodes, which is a smart way to swap the position of nodes in the tree, keeping the maximum number of previous connection to other children of rotating peers. They proved in a previous paper that the blocking probability of a certain user only depends on the number of free-riders and their height [224]. Therefore, the rotation is both effective and cheap. They do not include a discrimination mechanism between free-riders and normal peers, which are all identical.

Robbert van Renesse, Maya Haridasan and Ingrid Jansch-Porto present and evaluate an auditing model based on sampling the system and using the sampled information to build a global view of how the system is currently behaving [211]. Based on it, auditors employ strategies to identify the misbehaving nodes that should be punished. More specifically, untrusted local auditors request for information about previous rounds to local neighbors, and send the information to global auditors. The latter process accusations from several peers to a potential target free-rider, and determines whether it should be expurged from the system or not. Global auditors define an uploading threshold under which peers are punished, which is related to the live streaming rate and the information of local auditors. The authors assure the auditing system is scalable, mainly because the workload of local auditing does not increase with the size of the network, and the payload can be neglected, achieving at the same time a fixed streaming rate punishing free-riders. However, they do not include latency or playback delivery in a chunk level.

Fenglin Qin, Liansheng Ge, Qi Liu and Ju Liu develop a neat mathematical analysis of free-riding effects for live-streaming P2P networks [167]. Their motivation is supported by the massive presence of free-riding in certain P2P networks, but the lack of knowledge in its effect (although there are several works providing incentive-based mechanism to encourage contribution). For instance, in Gnutella the ratio of free riders increased from 66.7% to 85% and to 97% in 2000, 2005 and 2007 respectively [109]. Additionally, T. Silverston, F. Olivier and J. Crowcroft carried out traffic measurement and analysis on three popular P2P streaming applications: PPStream, PPLive, and SopCast, and concluded that fairness is not achieved in P2P streaming systems [198]. The authors find a rough expression for the network efficiency and chunk delivery ratio, as a function of the percentage of peers within the system. They show via simulations that the efficiency and delivery ratio falls down dramatically when the number of free-riders is increased. The model is in some way pessimistic. For example, they use the exchange efficiency from [169] and include the factor $(1 - \rho)$, meaning that peers independently select neighbors to cooperate, but this is in contradiction with the fact that peers will not choose free-riders following a tit-for-tat policy. Therefore, it is concluded that an incentive-based policy is mandatory for live-streaming services in P2P networks.

Bridge Zhao, John Lui and Dah-Ming Chiu propose a general mathematical framework to evaluate the stability and evolution of a family of shared history based incentive policies [232]. They recall the micro-payment system, which is suitable only for highly centralized systems with resourceful servers [85]. The tit-for-tat policy is included in a most general class of incentives, there called *private history-based mechanism*, where the resource is shared if certain historic conditions of generosity between two peers are met. They point-out that in a massive distributed system, two peers rarely contact many times, so that class of incentives has visible limitations. A second class of incentives is called *shared history based mechanism*, where a peer may infer the reputation of a requester based on the past experience of other peers. The mathematical model is both simple and tractable, with enough flexibility to test the stability and resilience of families of incentives working simultaneously in the system by different peers. It is assumed peers can periodically switch to another strategy, i.e. peers can *learn*, trying to imi-

tate the most cost-effective strategies. For each scenario with different set of incentive policies, a different differential equation governs the evolution of peers from different classes (which applies different policies). The system is composed by cooperative, reciprocated and defector peers. A cooperative peer always sends information when requested; a reciprocated helps whenever the requester does it; and defectors never contributes. The most interesting behavior is the one of reciprocators. The authors propose two behaviors of reciprocated peers, called Image and Proportional policies. In the Image policy, reciprocators probabilistically help requesters, with equal probability as if requesters help other peers (a probabilistic version of tit-for-tat); the Probabilistic policy was first presented for Electronic Commerce applications in [75], and reciprocators probabilistically help regarding the ratio between contribution and consumption of requesters. The authors finally study the stability and performance of the Image and Proportional policies. They show that the population of defectors rapidly increases following the Image policy, and the system is unstable. On the other hand, the proportional policy leads to a robust and scalable system. A possible reason is that the latter policy considers both consumption and contribution.

3.8 Two Paradigmatic Platforms

We will give two samples of real live streaming platforms that propose completely different paradigms. On one hand we have PPLive, a commercial platform that offers live streaming with proprietary protocols not available to academia. The following subsection describes the little known aspects from PPLive that could be learned from some experimental setups and reverse engineering. On the other hand we have GoalBit, which is the first peer-to-peer network that widely offers live streaming to end-users, and their codes and protocols are open, hence available to academia. We will briefly describe its main characteristics, focusing on its chunk scheduling policy.

3.8.1 GoalBit

GoalBit is the first free and open-source that offers live media streaming to end hosts. A source node (broadcaster) by means of special intermediate nodes called *super-peers* dynamically broadcasts multiple channels to peers, which are the *raison d'etre* of the whole system. The broadcaster is responsible to access the live media stream to be distributed and put it into the network, whereas super-peers have the dissemination task, together with normal peer. GoalBit works in a BitTorrent-based fashion, and includes super-peers, which are resourceful peers managed by the service provider (it is desired to dynamically promote helpful and stable peers as super-peers in future specifications). The peer excursion of a system is kept as simple as possible: it clicks on a .goalbit channel often available in the web Server. The tracker immediately offers a list of potential seeds, including super-peers. The chunk dissemination is done in a gossip-style, with buffer bitmap periodically exchanged and pull requests. It is worth to mention that GoalBit is compatible with multiple encoding formats and devices transparently to the end-user, which makes it suitable to broadcast user-generated streaming as well as operator-generated. The complete GoalBit suite for live media generation, encapsulation, distribution and playback is described in [18]. The GoalBit Packetized Stream (GBPS) takes care of the

encapsulation, whereas the distribution is provided by the GoalBit Transport Protocol (GBTP). Here we will focus on the core scheduling mechanism (i.e. neighboring and chunk scheduling policy), pointing out the main differences with BitTorrent. The primitives for communication between peers are quite simple though enough to have valuable control of the local swarm state. The reader is referred to [18] for more details.

The peer selection policy follows the traditional tit-for-tat principle from BitTorrent. The most generous neighboring peers are unchoked, hence enabled, to download from the local peer. Generosity is easily measured by counting the number of downloaded chunks from that neighboring peer. With regular intervals of T seconds, every peer decides to award unchoking the N most contributing neighboring peers. Additionally, a diversification mechanism is included optimistically unchoking a random peer within the network every MT seconds, being M a positive integer.
There is a common agreement in the scientific community that BitTorrent is appropriate for offline applications but does not comply timing requirements of live and on-demand systems [57, 214, 233]. The main reason is its chunk scheduling policy, namely *Rarest First*. In Rarest First, peers always request first the rarest chunk from its neighbors, trying to balance the availability of the system and converge to a uniform chunk distribution. However, it is both empirically and theoretically proved that Rarest First has unacceptable buffering times, which makes it unsuitable for live streaming. A practically opposite policy is the nearest-to-deadline rule, which represents a *Greedy* notion for the replication problem. This policy is intuitively economic, given that it needs no counting, and the buffering times can be reduced. Nevertheless, the delivery ratio is not satisfactory when Greedy is adopted in the whole live system. The intuition here is that the uniform chunk availability (from Rarest First) is completely broken with Greedy, because chunks far away from the playback are rare in the system. GoalBit proposes a hybrid policy, specified next. The buffer is categorized in three ranges: urgent, normal and future. If some urgent chunk is missing, the local peer requests the nearest-to-deadline chunk first. Otherwise, a missing chunk is picked sampling an exponentially distributed random variable, where the probability is monotonically decreasing from usual along to the to future buffer range. An illustration of the current GoalBit chunk scheduling policy is presented in Figure 3.1.

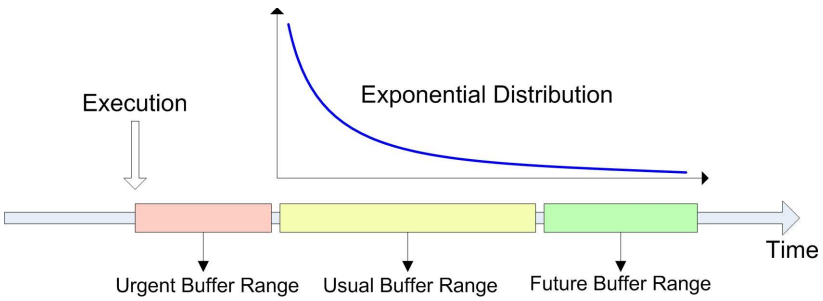


Figure 3.1: Chunk scheduling policy in GoalBit.

3.8.2 PPLive

The PPLive web site provides limited information about its proprietary technology [163]. With passive packet sniffing, Xiaojun Hei et. al. found PPLive has a not unusual bootstrap to get access to different channels, but the TV engine is sophisticated and the starting cooperation is aggressive [93]. The layout of the passive monitoring test was rustic: 2 home PCs with cable modem and 2 University campus with 100 Mbps Ethernet access. They used Ethereal for packet sniffing, carefully filtering cross traffic and counting TCP SYN connections with affirmative answer coming from exchanging between other IPs [71]. Two PCs (one residential and other form the campus) where connected to a 5-star top channel, whereas the other two were connected to a 3-star channel, in both cases for 2 hours. The main difficulty is that the TCP connections contain both signaling and chunk messages, and they cannot be easily discriminated. Hence, they propose a simple heuristic: only sessions that exchange more than 10 packets with sizes higher than 1200 bytes are considered session instead of signaling. The empirical distribution shows that the median length of a session is 20 seconds, so it is concluded that peers can exchange only few video chunks during a session. A TCP connection has many streaming variations, also the content availability of a certain peer. This imposes a variable streaming rate of the messages exchanges between peers. Nevertheless, with this statistical analysis it is also proved that peers have concurrent downloading from multiple peers. The TCP fluctuations are absorbed by both the aggregated downloading and double-buffering mechanism, which is similar to a token bucket. The result in PPLive is a quite smooth video-playback.

A performance analysis shows that the pop-up channel delay is 10-15 seconds, while the buffering delay additional 10-15 seconds. The potential upload from the campus' peers is highly exploited, offering 4,4 GBytes and 3,7 GBytes during the global sessions. Indeed, the campus peers have a complete mis-balance, uploading even ten times more that what they upload. The residential peer from the less popular channel practically does not contribute with the system, but plays the video channel correctly. These results show a main difference with BitTorrent, where users are encouraged to contribute via tit-for-tat, and should keep a balanced contribution with the system. What is more, this opposite behavior of upload/download with respect to the ADSL access might affect the ISPs business.

The buffer size is suspected to be dynamically adjusted, varying between 10 and 30 MBytes. Interestingly enough, the residential peer from the 3-star channel experimented a drop-out of 1 active peer in minute 33, and the PPLive system immediately recovered the state, by finding several active peers. The streaming rate consequently decreased in that minute but rapidly increased to the playback streaming rate. A primitive analysis of traffic locality was implemented, with a prefix matching of IP addresses. It suggests the traffic generated in U.S.A. mostly stays in U.S.A., whereas the majority of the download traffic comes from Asia. Nothing is here revealed about the gossiping protocol or chunk scheduling policy from PPLive.

The same authors publish valuable additional information about the behavior of users and scheduling policies of PPLive, by means of a sophisticated active crawler [93]. They discovered that the peak times of network utilization occur during 9 P.M. to 1 A.M. at China time, with scarce peaks between 9 P.M. and 2 P.M. in USA time. The number of active peers in dif-

ferent channels is intensified during weekends. The results confirm that people enjoy PPLive during their leisure time (other measures show VoIP is most used at office). The annual Spring Festival Gala on Chinese new year is one of the most popular programs in the Chinese community all over the world. The lifetime of users vary from seconds to 16 hours, but 90% users disconnect before 1,5 hours. The peer population comes from Asia in a high dominant proportion, following U.S.A. and other minorities. In January 28 2006, the Festival Gala on Chinese new year was broadcast by PPLive, maintaining 200.000 concurrent users during more than 4 hours! This fact confirms the peer-to-peer cooperative scheme can be highly scalable. The playback continuity was measured counting the number of freezings. The authors confirm the loss delivery ratio is in the order of 10^{-3}, and the PPLive system rapidly improves the storage when an incidental screen freezing occurs. It is suggested that PPLive adaptively builds buffer levels as a function of streaming quality degradation. The resourceless peers face more often freezings than resourceful peers, so peer heterogeneity is translated into different playback performance. In all cases, the playback losses are under acceptable margins. The authors conclude that lower start-up delays are needed to offer IPTV to achieve efficient channel switching, and the upload of resourceful peers will be even less-balanced with the tendency of higher streaming rates. Additionally, the lag behind different peers reach the order of minutes. Summing all, they feel live-streaming peer-to-peer applications are yet in the infancy.

3.9 Conclusions

We are all witnessing an explosive increment in television through the Internet. The most similar service from a traditional television is live streaming. It sounds disappointing that the most successful commercial live streaming peer-to-peer platforms are closed for academia and general public [163, 164, 210]. Consequently, the universal access of traditional television in multiple devices comes slowly. The most deployed topology for live streaming distribution is the mesh-overlay. It provides robustness under churn and dynamic tree diversification to exploit the bandwidth resource of the whole system. This robustness usually comes at the cost of high network overheads, in order to keep the necessary awareness of the available resources. Indeed, better scheduling decisions can be obtained with more information at disposal. There are two traditional modes of peer cooperation in live streaming, namely push or pull-based. In the first one, peers must pick a target peer (possibly at random) and then pushes one or several chunks. This method is one-sided, hence very cheap when regarding network overhead. Gracefully, it also provides a fast adaptation when a peer has an empty buffer. However, some works predict it has poor performance when the buffers are filled with abundant chunks, and push-systems suffer from unbounded duplication (the same peer receives a non-desired chunk from several sources). Pull-based systems are pressed by requests: receivers choose a (possible random) peer and then select a desired chunk to be downloaded from it. Most deployed systems are pull-mesh based [18, 163, 164, 210], and the information exchange occurs in an epidemic (gossip)-style, where chunks dynamically visit directed acyclic paths. Perhaps guided by intuition rather than mathematical foundation, because pull requests, though bidirectional, are driven by *needs* and easy to understand. The core resilience mechanism for pull-mesh based live systems is peer selection and chunk scheduling policies, where the neighboring is often

predetermined as random, exploiting the connectivity properties of random networks [25]. In 2007, the first tractable model for the analysis of chunk scheduling policies is proposed [233]. It is proved to be robust, and captures the playback-buffering trade-off, which is highly related with the two most important factors of the human behavior: a person feels extremely annoying to watch several video-cuts, and is not intended to wait minutes to connect a video channel. There is a common agreement in the scientific community that BitTorrent is appropriate for offline applications but does not comply timing requirements of live and on-demand systems [18, 57, 214, 233]. Moreover, the authors from the first mathematical model have recently recognized that the problem of meeting real-time streaming requirements while preserving scalability is a challenging problem still not well understood [234]. With a different approach and taking the source latency as a measure, Maussolié classifies open problems that born from epidemic models of chunk propagation [138]. Chapter 5 provides a modest contribution to the understanding of the playback-buffering trade-off, in the lights of the model [233]. At the same time, the open-source paradigm for peer-to-peer networks is here promoted, showing the performance of new chunk scheduling policies in GoalBit. We hope in the future the design of live-streaming peer-to-peer systems will be as clear as in file-sharing.

Part II

CONTRIBUTIONS

Chapter 4

Stability and Capacity of Peer-Assisted Video-On-Demand Networks

4.1 Introduction

Video on-demand distribution over the Internet has several challenges, as we discussed in Chapter 2. The video replication policy must cope with the asymmetric playback nature from different peers of these systems. Smart solutions were proposed during the short-life of existence of highly-scalable on-demand peer-to-peer systems, like grand-children gossiping [62], sophisticated Multiple Video Cache policy in PPLive-VoD [98], anchor-based solutions for VCR interactivity and caching policies to maximize global availability [96], among others.

In this chapter we focus on the design of a Multiple Video Cache (MVC) policy to maximize availability. Basically, we manage a set of heterogeneous cache-nodes (super-peers in the GoalBit system), and want to define the best caching policy (the video-types that should be stored in each cache node), given the statistics of the demand and peer population. For that purpose, we will consider a simple fluid model to understand the streaming rates and expected peer excursion times. Then we introduce a combinatorial optimization problem, trying to find the caching policy that minimizes the expected excursion time between peers. The statistical data is taken from a passive YouTube Crawler, specifically designed to understand the session dynamism and video popularity. The results show to be encouraging: peer-assisted systems outperform nearly ten times the throughput of a traditional Content Delivery Network (CDN). This suggests peer-assisted systems are cost-effective, and YouTube would have valuable benefits encouraging users to contribute.

This chapter is structured as follows. Section 4.2 presents a motivation of this work, whereas Section 4.3 contains a summary of related work. Section 4.4 introduces a general fluid model, in which peers can concurrently download several videos, and are classified according to the number of simultaneous downloads. Two special sub-models are discussed in-depth in Section 4.5. The first is a concurrent model for BitTorrent-based networks. The second derived from the most general is a sequential fluid model, in which each peer downloads video con-

tents sequentially. The sequential model is globally stable, and we can find closed expressions for the expected waiting times, regarding P2P and CDN systems separately, and prove that the performance of a P2P system is never worse than that of a CDN. A combinatorial optimization problem (COP) captures the Caching Problem in Section 4.6, trying to allocate video replicas in super-peers in order to minimize the expected waiting time. We prove the feasibility of the Caching Problem is an NP-Complete decision problem, and therefore we develop a greedy randomized heuristic to solve it. Real-life scenarios based on YouTube traces are analyzed in Section 4.7. Finally, Section 4.8 contains the main conclusions and discusses several aspects for future research.

4.2 Motivation

In the Client-Server architecture, servers offer a simple and predictable service to their users. However, this service can be easily collapsed by a mismatch between the download requirements and upload server capacity. Established in 2005, YouTube has become the most successful Internet site providing a new generation of short video sharing service, comprising approximately nearly 10% of all traffic on the Internet [40]. However, the network access is yet working with a client-server architecture, and the operator (Google corporation) must afford more than one million dollars per day just for bandwidth requirements, which is a clear motivation to add cooperation in YouTube's users [96]. Peer-to-peer networks represent an attractive paradigm for the deployment of high-scalable video-on demand services. They are self-organized communities developed at the application layer, in which users, called *peers*, offer their resources (bandwidth, memory and CPU-time) to others, basically because they share common interests. The cooperation between users reduce the operational costs, and promote scalability. Nevertheless, the design of these networks imposes many challenges. Peers join and leave the network when they wish in an unpredictable manner (known as node churn). Therefore, the global network resources is a function of time. Moreover, peers are heterogeneous, and the uplink/downlink capacities are not similar, in part due to the high penetration of ADSL services [21]. Some selfish peers exploit the resources of the whole system but do not contribute, called free-riders. The topological design and cooperative schemes are considered core-mechanisms in order to address those issues. A neat survey covering the design of resilient peer-to-peer video streaming can be found in [1].

Some commercial P2P networks for on-demand streaming are available. The most successful are VoD-PPlive [98], VoD-PPstream [164] and Spotify [106].

An inspirational system for replication and fast dissemination of files is BitTorrent, created by Bram Cohen [23, 44]. BitTorrent was originally designed for offline downloading. However, nowadays most of the P2P applications over the Internet are BitTorrent-based. One of such applications is the GoalBit Video Platform, the first free and open source peer-to-peer streaming Network [18]. In these systems, peers are either *downloaders* when they actively download content, or *seeders*, once they finished the download process but remain connected, sharing already downloaded items. GoalBit introduces a third node-type to the network named *super-peer*. These kind of nodes have higher bandwidth resources than a normal peer, and usually join the network with longer life-times (very stable peers). Super-peers are encouraged to store

and forward video streaming to end-users (with a short life-time). In the current GoalBit protocol specification, super-peers are nodes managed by the operator of the platform, hosted in the cloud, and implement a specific caching policy [18]. Therefore, super-peers is the GoalBit's name for the cache-servers. In GoalBit, there is also an entity or software named *tracker*, which knows all the peers that are seeding or downloading an item. The reader can find summary of GoalBit's design philosophy in Section 3.8.1, and a thorough description in [18]. The motivation of this chapter is to explore the best caching policy for on-demand video streaming, in order to minimize the expected download times experienced by end-users. Our framework is general enough to be applied either into GoalBit or other peer-assisted cooperative architectures as well.

4.3 Related Work

Usually, the development of experiments in real P2P systems is expensive, and a large deviation from customer expectations would be detrimental to their reputation [206]. As a consequence, the scientific community works to develop mathematical models in order to predict the behavior of the system. In [58], Yang and de Veciana justify mathematically the consistency of the service capacity of P2P file sharing services. They propose a branching process, and state that a P2P system highly outperforms a traditional CDN. A basic Markovian model is also introduced to describe peer evolution. Qiu and Srikant analyze BitTorrent-like systems under steady state and its variability, showing empirical validation as well [169]. A steady state analysis is first presented with a simple fluid model, in which the peer evolution is captured by exogenous poissonian arrivals and exponential departures in the system. They consider homogeneous peers, and find a closed expression for the expected waiting time. A sensitivity analysis of this performance measure with respect to different design parameters offers one of the first insights of the BitTorrent's correctness. A special treatment is included for the file sharing efficiency between peers, and states the robustness of random peer selection. They develop a fluid model whose steady state is partially characterized as locally stable, and it was conjectured that it is globally stable as well. The reader can find the fluid model and the mentioned results in Section 1.2 (Mathematical Foundations of File-Sharing Systems). The conjecture is true, and proved for the first time in [168]. The fluid model proposed by Qiu and Srikant is generalized in a first stage by [208], extending the model for several concurrent multi-torrents. The authors argue that most BitTorrent users download several files at the same time. A second generalization is proposed by Pablo Rodríguez-Bocca and Clauda Rostagnol [183], where the authors introduce the presence of super-peers in the network, and study the steady state of a video on-demand application.

Here, we propose a further generalization of [183]. A general fluid model is presented, adding node churn. We prove both mathematically and empirically that the peer-to-peer architecture outperforms traditional Content Delivery Networks (CDNs), in a general environment. Then we explore optimal caching policies in sequential systems (in which users can either download zero or one video file at a given time). We propose the Caching Problem, and prove its NP-Completeness, showing its performance via simulations regarding real-traces from YouTube.

4.4 General Fluid Model

Consider an open network which offers K video items with sizes $\{s_1, \ldots, s_K\}$. Peers join the network, download progressively one or possibly many concurrent video items and abort the system when they wish. Peers are then classified in exhaustive and mutually disjoint sets $C^1, \ldots, C^K$, where C^i is the set of peers that download i video items simultaneously. Denote $x_j^i(t)$ the number of peers from class C^i that are downloading video j in a certain instant t. They join the network following a poissonian process of respective rates λ_j^i, and abort the system with exponential law, and respective rates θ_j^i. The number of seeders owning exactly i video items, and seeding video j at instant t is denoted by $y_j^i(t)$, and depart the system exponentially with rates γ_j^i. We shall assume identical peers, with respective upload and download capacities denoted by μ and c. Peers also contribute with the system uploading video streaming. The exchange effectiveness between peers is a coefficient $\eta : 0 \leq \eta \leq 1$ that indicates the amount of uploaded bandwidth successfully exploited in the system[1]. All seeders from class C^i that own video j can decide a portion ρ_j of their available uploading capacity, in order to feed downloaders of video j. Super-peers behave like seeders, but they do not leave the system. The number of super-peers from class C^i seeding video j are denoted by z_j^i, and have upload capacity ρ. All this information can be summarized in a General Fluid Model (GFM), specified as follows:

$$\begin{cases} \dfrac{dx_j^i(t)}{dt} = \lambda_i^j - \theta_j^i x_j^i(t) - \min\{c_j^i x_j^i(t), \eta\mu_j^i x_j^i(t) + \displaystyle\sum_k (\mu_j^k y_j^k(t) + \rho_j^k z_j^k)\}, & \text{(4.1a)} \\ \dfrac{dy_j^i(t)}{dt} = \min\{c_j^i x_j^i(t), \eta\mu_j^i x_j^i(t) + \displaystyle\sum_k (\mu_j^k y_j^k(t) + \rho_j^k z_j^k)\} - \gamma_j^i y_j^i(t), & \text{(4.1b)} \end{cases}$$

where additionally:

1. $\mu_j^i = \frac{\mu_i}{s_j}$ is upload rate for video j from peers in class C^i , and $\sum_{k \in C^i} \mu_k = \mu$.
2. $c_j^i = \frac{c_i}{s_j}$ is the download rate for item j from peers in class C^i, and $\sum_{k \in C^i} c_k = c$.
3. $\rho_j^i = \frac{\rho_i}{s_j}$ is the streaming rate for item j from super-peers in class C^i, and $\sum_{k \in C^i} \rho_k = \rho$.
4. θ_j^i is the disconnection rate for item j from peers in class C^i.
5. γ_j^i is the disconnection rate for item j from seeders in class C^i.

Observe that the General Fluid Model is a non-linear switched system of $2K^2$ ordinary differential equations, where the terms are expressed in peers per second, in compatible units. Videos with higher size are normally slower to download, so the video sizes s_j appear in the denominator. The GFM is just a balance of entrance and departure of peers in the system, including

[1]Note that due to asymmetric reasons, this parameter is not near the unit as predicted by Tewari and Kleinrock for file sharing [206].

the presence of special components, like seeders and super-peers. As soon as peers completely download the video stream, they are promoted to seeders, explaining the additive term on the right hand of (4.1b). The minimum function means that the bottleneck is either in downloading or uploading. A peer (seeder) disconnection from a certain video-item does not imply the complete abortion of the system. We will prove an elementary result for the GFM which turns to be useful in particular sub-models of it. For simplicity we will denote $[K] = \{1, \ldots, K\}$ throughout the rest of the chapter.

Proposition 4.4.1 *The GFM is a positive system.*

Proof. We must prove that given an arbitrary starting condition with $x_j^i(0) \geq 0$ and $y_j^i(0) \geq 0$ for all $i, j \in [K]$, then necessarily $x_j^i(t) \geq 0$ and $y_j^i(t) \geq 0$ for all $t \geq 0$ and $i, j \in [K]$. Suppose in a certain variable from the set $\{x_j^i\}_{i,j\in[K]}$ gets null in some time t_0. By the corresponding Equation (4.1a) the minimum function is null in that instant, and hence $\frac{dx_j^i}{dt} = \lambda_j^i > 0$. Analogously, suppose now that $y_j^i(t_1) = 0$ for a certain $t_1 > 0$ and $i, j \in [K]$. The right side of Equation (4.1b) assumes in t_0 the value of the minimum function, which is non-negative, and $\frac{dy_j^i}{dt} \geq 0$ again. As a consequence, the peer and seeder populations under the GFM never take negative values.

Q.E.D.

4.5 Two Outstanding Sub-Models

4.5.1 Concurrent Fluid Model (CFM)

The number of variables involved in the GFM force us to assume further hypothesis in order to analyze the stationary state of the system and have an insight of the super-peers optimal operation, which are the only nodes that can be managed by the network operator. Inspired in BitTorrent-based systems, we will strictly stick to the following assumptions:

1. "Fair transmission": the resources are equally distributed in the different concurrent videos: $\mu_i = \frac{\mu}{i}$, $c_i = \frac{c}{i}$ and $\rho_i = \frac{\rho}{i}$.

2. "Tit-for-tat": Peers in class C^i that at time t are downloading video j receive from all other downloaders a streaming rate proportional to the upload bandwidth μ_j^i and their population:
$$\left(\frac{\mu_j^i x_j^i(t)}{\sum_k \mu_j^k x_j^k(t)}\right) \sum_k \eta \mu_j^k x_j^k(t) = \eta \mu_j^i x_j^i(t).$$

3. "Fair Seeders": Peers from class C^i that at time t are downloading video j receive from all the seeders a streaming rate proportional to the download bandwidth and their population:
$$\left(\frac{c_j^i x_j^i(t)}{\sum_k c_j^k x_j^k(t)}\right) \sum_k \mu_j^k y_j^k(t) = \alpha_j^i(t) \sum_k \mu_j^k y_j^k(t).$$

4. "Fair Super-peers": Analogously, peers from class C^i that at time t are downloading video j receive from all the super-peers a streaming rate proportional to the download bandwidth and their population: $\alpha_j^i(t) \sum_k \rho_j^k z_j^k$.

5. "Peers Departures": the peers and seeders departures follow the *Zipf law*, being linearly decreasing with respect to the number of concurrent video streaming: $\gamma_j^i = \gamma/i$, and $\theta_j^i = \theta/i$.

Let us briefly discuss the reasons for these assumptions. Under a homogeneous system, fairness is a natural hypothesis, in order to preserve the same opportunities and expected performance. Assumption 2 is an ideal approximation of the corresponding tit-for-tat concept from BitTorrent (in file-sharing [44]). Recall that in video on-demand the playback is not symmetric. Nevertheless, a well-design incentive policy will encourage peers to contribute. Thus, Assumption 2 represents an ideal and well-design incentive mechanism, and a complementary notion of fairness. Assumptions 3 and 4 are similar in spirit. Assumption 5 captures the user's behavior. Intuitively, the worth of the connection is increased when the user downloads more video streams. We model the interest-relation as linearly with the number of concurrent streaming, following the Zipf's law as in other works in peer-to-peer computing [226]. Figure 4.1 summarizes the symbolic notation.

K	available videos
s_j	size of video-item j
$x_j^i(t)$	downloaders in class C^i downloading video j at time t
$y_j^i(t)$	seeders in class C^i seeding video j at time t
$z_j^i(t)$	super-peers in class C^i seeding video j at time t
λ_j^i	arrival rate for peers in class C^i requesting video j
θ_j^i	departure rate of peers in class C^i requesting video j
γ_j^i	departure rate of seeders in class C^i seeding video j
c	total download bandwidth for each peer.
μ	total upload bandwidth for each peer.
ρ	total upload bandwidth for each super-peer.
η	exchange effectiveness between peers ($\eta \in [0,1]$).

Figure 4.1: Symbolic notation for the Concurrent Fluid Model.

After including these BitTorrent-based assumptions to the GFM we get the P2P Concurrent Fluid Model (P2P-CFM):

$$\begin{cases} \dfrac{dx_j^i}{dt} = \lambda_j^i - \dfrac{\theta}{i} x_j^i - \min\left\{ \dfrac{c}{is_j} x_j^i, \eta \dfrac{\mu}{is_j} x_j^i + \alpha_j^i \displaystyle\sum_k \left(\dfrac{\mu}{s_j} \dfrac{y_j^k}{k} + \dfrac{\rho}{s_j} \dfrac{z_j^k}{k} \right) \right\} & \text{(4.2a)} \\ \dfrac{dy_j^i}{dt} = \min\left\{ \dfrac{c}{is_j} x_j^i, \eta \dfrac{\mu}{is_j} x_j^i + \alpha_j^i \displaystyle\sum_k \left(\dfrac{\mu}{s_j} \dfrac{y_j^k}{k} + \dfrac{\rho}{s_j} \dfrac{z_j^k}{k} \right) \right\} - \dfrac{\gamma}{i} y_j^i, & \text{(4.2b)} \end{cases}$$

where $\alpha_j^i(t) = \frac{x_j^i(t)/i}{\sum_k x_j^k(t)/k}$, and the independent variable t was omitted for short.
A traditional CDN (with a client-server paradigm) can be viewed as a particular case of this analytical approach. Specifically, users do not cooperate ($\mu = 0$), seeders do not participate in the network ($y_j^i(t) = 0$) and the previously named super-peers are now static servers. Replacing these parameters in Expression (4.2a), the CDN Concurrent Fluid Model (CDN-CFM) is defined by the following system of ordinary differential equations:

$$\frac{dx_j^i(t)}{dt} = \lambda_j^i - \theta\frac{x_j^i}{i} - \min\left\{\frac{c}{s_j}\frac{x_j^i(t)}{i}, \alpha_j^i(t)\sum_k \rho_j^k z_j^k\right\} \tag{4.3}$$

4.5.1.1 Rest Point Analysis for CFM

A Corollary from Proposition 4.4.1 is that both the P2P-CFM and CDN-CFM are positive systems. If we find a time t such that simultaneously $\frac{dx_i^j(t)}{dt} = \frac{dy_i^j(t)}{dt} = 0$ for every pair $i, j \in [K]$, the system will rest indefinitely in the same constant vector state $(\overline{x_j^i}, \overline{y_j^i})$. This is called a stationary state. Now, we will find the stationary state for the P2P-CFM.

Proposition 4.5.1 *The rest point for the P2P-CFM is:*

$$\begin{cases} \overline{x_j^i} = \max\left\{\dfrac{i\lambda_j^i s_j}{\theta s_j + c}, \dfrac{i\lambda_j^i(\lambda_j - \overline{\rho_j} - \overline{\phi_j})}{\lambda_j(\theta + \frac{\eta\mu}{s_j} - \frac{\mu\theta}{\gamma s_j})}\right\} & \text{(4.4a)} \\ \overline{y_j^i} = \dfrac{i\lambda_j^i - \theta\overline{x_j^i}}{\gamma}, & \text{(4.4b)} \end{cases}$$

where $\overline{\rho_j} = \sum_k \rho_j^k z_j^k$, $\lambda_j = \sum_i \lambda_j^i$ *and* $\overline{\phi_j} = \frac{\mu\lambda_j}{\gamma s_j}$.

Proof. Expression (4.4b) is directly obtained summing Equations (4.2a) and (4.2b). When the download capacity is the system's bottleneck the steady state for the P2P-CFM can be found solving the following linear system of equations:

$$\begin{cases} \lambda_j^i - \dfrac{\theta}{i}\overline{x_j^i} - \dfrac{c}{is_j}\overline{x_j^i} = 0 & \text{(4.5a)} \\ \dfrac{c}{is_j}\overline{x_j^i} - \gamma_j^i\overline{y_j^i} = 0 & \text{(4.5b)} \end{cases}$$

From Equation (4.5a) we get immediately that:

$$\overline{x_j^i} = \frac{i\lambda_j^i s_j}{\theta s_j + c} \tag{4.6}$$

On the other hand, when the upload capacity is the system's bottleneck, we must solve the following non-linear system:

$$\begin{cases} \lambda_j^i - \frac{\theta}{i}\overline{x_j^i} - \eta\frac{\mu}{is_j}\overline{x_j^i} - \frac{\overline{x_j^i}/i}{\sum_k \overline{x_j^k}/k}\sum_k(\frac{\mu}{s_j}\frac{\overline{y_j^k}}{k} + \frac{\rho}{s_j}\frac{z_j^k}{k}) = 0 & \text{(4.7a)} \\ \eta\frac{\mu}{is_j}x_j^i(t) + \frac{\overline{x_j^i}/i}{\sum_k \overline{x_j^k}/k}\sum_k\left(\frac{\mu}{s_j}\frac{\overline{y_j^k}}{k} + \frac{\rho}{s_j}\frac{z_j^k}{k}\right) - \frac{\gamma}{i}\overline{y_j^i} = 0 & \text{(4.7b)} \end{cases}$$

Denote for short $\overline{\rho_j} = \sum_k \frac{\rho}{s_j}\frac{z_j^k}{k}$ and $\overline{\phi_j} = \frac{\mu\lambda_j}{\gamma s_j}$. Summing (4.7a) over all classes $i \in [K]$:

$$\sum_i \frac{\overline{x_j^i}}{i} = \frac{\lambda_j - \overline{\phi_j} - \overline{\rho_j}}{\theta + \frac{\eta\mu}{s_j} - \frac{\mu\theta}{\gamma s_j}}. \tag{4.8}$$

Additionally, Equation (4.7a) can be re-written:

$$\lambda_j^i = \left(\theta + \frac{\eta\mu}{s_j} - \frac{\mu\theta}{\gamma s_j} + \frac{\overline{\rho_j} + \overline{\phi_j}}{\sum_i \frac{\overline{x_j^i}}{i}}\right)\frac{\overline{x_j^i}}{i} \tag{4.9}$$

Replacing (4.8) into (4.9) and taking a common denominator, we obtain the desired result.

Q.E.D.

The expressions for the steady state in the CDN can be obtained either making $\mu = 0$ in Equation (4.4a) or solving Equation (4.3) with $\frac{dx_j^i}{dt} = 0$:

$$\overline{x_j^i}_{CDN} = \max\left\{\frac{i\lambda_j^i s_j}{\theta s_j + c}, \frac{i\lambda_j^i(\lambda_j - \overline{\rho_j})}{\theta\lambda_j}\right\} \tag{4.10}$$

Now we can compare the capacity of the P2P-CFM and CDN-CFM systems. Let us assume stability for a moment (we prove global stability of a special but important case in the following subsection). Denote T_{CFM}^{P2P} and T_{CFM}^{CDN} the expected waiting times under regime for the respective systems P2P-CFM and CDN-CFM. The following proposition is intuitive, and sounds:

Proposition 4.5.2 $T_{CFM}^{P2P} \leq T_{CFM}^{CDN}$.

Proof. Consider the random variable T_j^i that represents the waiting time for user-type (j, i) (a member of class x_j^i). By Little's law we relate the mean waiting time with the number of users under regime:

$$E(T_j^i) = \frac{\overline{x_j^i}}{\lambda_j^i}. \tag{4.11}$$

Equation (4.11) holds for both systems (P2P and CDN), where the number of users are $\overline{x^i_{jP2P}}$ and $\overline{x^i_{jCDN}}$ respectively. Let us denote X to the random variable that represents the class of certain arrival in the GFM. It has range $R_X = \{(j,i) : j,i \in [K]\}$, and $P(X = (j,i))$ is the probability that a certain new arrival is from class x^i_j. The poissonian arrivals with intensity rates λ^i_j imply that $P(X = (j,i)) = \frac{\lambda^i_j}{\lambda}$, being λ the global sum rate. The mean waiting time of a user in the P2P-CFM can be found via conditional expectation [78]:

$$\begin{aligned} E(T) &= E(E(T/X)) \\ &= \sum_i \sum_j P(X=(j,i))E(T/X=(j,i)) \\ &= \sum_i \sum_j \frac{\lambda^i_j}{\lambda} E(T^i_j) \\ &= \sum_i \sum_j \frac{\lambda^i_j}{\lambda} \frac{\overline{x^i_j}}{\lambda^i_j} \\ &= \frac{1}{\lambda} \sum_i \sum_j \overline{x^i_j}, \end{aligned}$$

where we used Little's law. Notice that equalities hold again for both systems:

$$\begin{aligned} T^{P2P}_{CFM} &= \frac{1}{\lambda} \sum_i \sum_j \overline{x^i_{jP2P}}, \\ T^{CDN}_{CFM} &= \frac{1}{\lambda} \sum_i \sum_j \overline{x^i_{jCDN}}. \end{aligned}$$

Hence, it suffices to prove that the number of downloaders in the $P2P$ fluid model is never greater than the one in the CDN model: $\overline{x^i_{jP2P}} \leq \overline{x^i_{jCDN}}$. We use Expressions (4.10), (4.4a) and elementary algebra. If the download is the system's bottleneck then the equality is obvious. Otherwise, the second argument of the maximum function in Expression (4.10) must dominate, and in particular by positivity $\lambda_j - \rho_j$ must be positive. Therefore, the following chain of inequalities holds:

$$\begin{aligned} \overline{x^i_{jP2P}} &= \frac{i\lambda^i_j(\lambda_j - \overline{\rho_j}) - i\lambda^i_j\overline{\phi_j}}{\lambda_j\left(\theta + \frac{\mu}{s_j}(\eta - \frac{\theta}{\gamma})\right)} \\ &< \frac{i\lambda^i_j(\lambda_j - \overline{\rho_j})}{\theta\lambda_j} \\ &= \overline{x^i_{jCDN}}. \end{aligned}$$

This inequality holds if and only if

$$(-i\lambda^i_j\overline{\phi_j})(\theta\lambda_j) < i\lambda^i_j(\lambda_j - \overline{\rho_j})(\lambda_j\frac{\mu}{s_j}(\eta - \frac{\theta}{\gamma})).$$

Using that $\overline{\phi_j} = \frac{\mu\lambda_j}{\gamma s_j}$ and canceling common factors, the inequality holds if and only if

$$\lambda_j \frac{\theta}{\gamma} + (\lambda_j - \overline{\rho_j})(\eta - \frac{\theta}{\gamma}) > 0.$$

Equivalently: $\eta\lambda_j > \overline{\rho_j}(\eta - \frac{\theta}{\gamma})$. But the latter inequality is obviously true, since $\lambda_j - \overline{\rho_j} > 0$.

Q.E.D.

Observe that when the upload is the bottleneck, the cooperative system always outperforms raw CDN technology. Naturally, when the download is the bottleneck the effect of peers contribution is null, and both systems are equivalent. The remaining of this chapter focuses on an outstanding case, defined by a single-class system, where each peer downloads exactly one video item at a time.

4.5.2 Sequential Fluid Model (SFM)

In this subsection we will study the GFM in the particular case in which peers download exactly one video item at a time (the single-class case - $i = 1$). We will call it P2P-Sequential Fluid Model (P2P-SFM):

$$\begin{cases} \dfrac{dx_j(t)}{dt} = \lambda_j - \theta x_j(t) - \min\left\{ \dfrac{c}{s_j} x_j(t), \eta \dfrac{\mu}{s_j} x_j(t) + \dfrac{\mu}{s_j} y_j(t) + \dfrac{\rho}{s_j} z_j \right\} & \text{(4.12a)} \\ \dfrac{dy_j(t)}{dt} = \min\left\{ \dfrac{c}{s_j} x_j(t), \eta \dfrac{\mu}{s_j} x_j(t) + \dfrac{\mu}{s_j} y_j(t) + \dfrac{\rho}{s_j} z_j \right\} - \gamma y_j(t). & \text{(4.12b)} \end{cases}$$

A direct calculation shows that:

$$\begin{cases} \overline{x_j}^{P2P}_{SFM} = \max\left\{ \dfrac{\lambda_j s_j}{\theta s_j + c}, \dfrac{\lambda_j(\gamma s_j - \mu) - \gamma\rho z_j}{\theta(\gamma s_j - \mu) + \eta\gamma\mu} \right\}, & \text{(4.13a)} \\ \overline{y_j}^{P2P}_{SFM} = \dfrac{\lambda_j - \theta \overline{x_j}^{P2P}_{SFM}}{\gamma}. & \text{(4.13b)} \end{cases}$$

Recall that an equilibrium point $\overline{x}$ of a dynamic system is locally stable if there exists a positive radius R such that $x(t) \to \overline{x}$ whenever $\|x(0) - \overline{x}\| < R$. An equilibrium point $\overline{x}$ is globally stable if the orbit $x(t)$ converges to $\overline{x}$ for every starting point. In linear systems of differential equations (i.e. $\dot{x} = Ax$), local stability is equivalent to global stability, and occurs if and only if all the eigenvalues of the characteristic matrix A have negative real parts. The reader can find further definitions and examples in [129].

The P2P-SFM is a linear-switched system, i.e. a special class of dynamic system. The global stability of a special sub-system of the P2P-SFM has been studied in [168]. More precisely, it is the P2P-SFM when a single video item is distributed and with no super-peer assistance (i.e. the unidimensional case with no terms $\rho_j z_j$). There, the global stability is proved, but the authors do not address any performance analysis. We will prove here that the

P2P-SFM is always globally stable. The only assumption is that $\gamma > 0$. Otherwise, seeders accumulate and never leave the system, leading to non-stability (this case is non-realistic as well). The proof will follow closely the ideas from Dongyu Qiu and Wei Qian Sang [168], although some subtle details are different (mainly the manipulations of inequalities). Here we prove a useful lemma which makes the difference with that work (the remainder of the proof is in the Appendix).

Lemma 4.5.3 *The P2P-SFM is locally stable if* $(c - \eta\mu)\overline{x_j}^{P2P}_{SFM} \neq \mu\overline{y_j}^{P2P}_{SFM} + \rho z_j$.

Remark 4.5.4 *In fact, the P2P-SFM is also locally (and globally) stable in general. The proofs are tricky because the behavior of the system is switched many times when the equality holds.*

Proof. The P2P-SFM is a set of K pairs of independent ordinary differential equations in the (x, y) plane. In the product topology of finite sets, point-wise convergence is equivalent to convergence coordinate-wise. Therefore, the P2P-SFM is locally (globally) stable if and only if each block is locally (globally) stable. Without loss of generality, we will study the following linear switched system:

$$\begin{cases} \dfrac{dx}{dt} = \lambda - \theta x(t) - \min\{cx(t), \eta\mu x(t) + \mu y(t) + \rho z\} & \text{(4.14a)} \\ \dfrac{dy}{dt} = \min\{cx(t), \eta\mu x(t) + \mu y(t) + \rho z\} - \gamma y(t), & \text{(4.14b)} \end{cases}$$

where $\overline{x}$ and $\overline{y}$ are respectively:

$$\begin{cases} \overline{x} = \max\left\{\dfrac{\lambda}{\theta + c}, \dfrac{(\gamma - \mu)\lambda - \gamma\rho z}{(\gamma - \mu)\theta + \eta\mu\gamma}\right\} & \text{(4.15a)} \\ \overline{y} = \dfrac{\lambda - \theta\overline{x}}{\gamma} & \text{(4.15b)} \end{cases}$$

The system switches between two linear systems. If $\frac{c-\eta\mu}{\mu}x(t) \leq y(t) + \frac{\rho z}{\mu}$ the dynamic system is governed by System I:

$$I \begin{cases} \dfrac{dx(t)}{dt} = \lambda - (\theta + c)x(t) & \text{(4.16a)} \\ \dfrac{dy(t)}{dt} = cx(t) - \gamma y(t). & \text{(4.16b)} \end{cases}$$

Otherwise, the dynamic system is governed by System II:

$$II \begin{cases} \dfrac{dx}{dt} = \lambda - (\theta + \eta\mu)x(t) - \mu y(t) - \rho z & \text{(4.17a)} \\ \dfrac{dy}{dt} = \eta\mu x(t) + (\mu - \gamma)y(t) + \rho z. & \text{(4.17b)} \end{cases}$$

First, we will study the stability of both linear systems separately. The solution of System I can be immediately obtained for an arbitrary starting point $(x(0), y(0))$:

$$\begin{aligned} x(t) &= k_0 e^{-(\theta+c)t} + k_1; \\ y(t) &= k_2 e^{-\gamma t} + k_3 e^{-(\theta+c)t} + k_4, \text{ if } \gamma \neq \theta + c \\ y(t) &= k_5 e^{-\gamma t} + k_6 t e^{-\gamma t} + k_7, \text{ if } \gamma = \theta + c, \end{aligned}$$

for certain real numbers k_i. The eigenvalues are $-\gamma < 0$ and $-(\theta + c) < 0$. Hence, System I is always globally stable. Moreover, if $c < \mu\eta$ then $cx(t) < \eta\mu x(t) + \mu y(t)$ for all $t \geq 0$ because $y(t) \geq 0$, and the P2P-SFM is governed by System I hence globally stable in this case.
Recall that a linear system is locally (or globally) stable if and only if the real part of all its eigenvalues are negative. By direct calculations the characteristic polynomial of System II is:

$$p(\lambda) = \lambda^2 + (\theta + \gamma + (1 - \eta)\mu)\lambda + \mu\eta\gamma + (\gamma - \mu)\theta$$

The stability of System II must be discussed regarding the network parameters. Note in particular that if $\gamma \geq \mu$, the quadratic polynomial $p(\lambda)$ has all real positive coefficients, and consequently both eigenvalues are negative. Therefore, System II is globally stable when $\gamma \geq \mu$.
Now we return to the P2P-SFM. Let us call $m = \frac{c-\eta\mu}{\mu}$ and $y_0 = \frac{\rho z}{\mu}$.
On one hand, if the rest point $(\overline{x}, \overline{y})$ is in Area I: $m\overline{x} < \overline{y} + y_0$ and we choose $(x(0), y(0))$ near the rest point, then the P2P-SFM is governed by System I, which is globally (hence locally) stable. In this case, the P2P-SFM is locally stable.
On the other hand, if the rest point is in Area II: $m\overline{x} > \overline{y} + y_0$, and we choose a starting condition near the rest point, the P2P-SFM is governed by System II, which is globally stable whenever $\gamma \geq \mu$. Suppose for a moment that $\gamma < \mu$. Without loss of generality we assume $c > \mu\eta$ (otherwise we will have that the P2P-SFM is globally stable). By Equation (4.15b), $\gamma\overline{y} = \lambda - \theta\overline{x}$, so the condition of Area II is equivalent to $(m\gamma + \theta)\overline{x} < \lambda + \gamma y_0$, or equivalently:

$$0 < \overline{x} = \frac{(\gamma - \mu)\lambda - \gamma\rho z}{(\gamma - \mu)\theta + \eta\mu\gamma} < \frac{\mu\lambda + \gamma\rho z}{\mu\theta + \gamma(c - \mu\eta)}.$$

Given that we assumed $\gamma < \mu$, the numerator of $\overline{x}$ is negative. Therefore, by positivity its denominator must be also negative. The following equivalent inequalities can be obtained canceling several common terms. All of them state that the rest point $(\overline{x}, \overline{y})$ is in Area II:

$$\begin{aligned} [(\gamma - \mu)\lambda - \gamma\rho z][\mu\theta + \gamma(c - \mu\eta)] &> [\mu\lambda + \gamma\rho z][(\gamma - \mu)\theta + \eta\mu\gamma], \\ \mu\theta(\lambda - \rho z) + (c - \mu\eta)(\lambda\gamma - \mu\lambda - \gamma\rho z) &> \mu\lambda\theta + \mu^2\eta\lambda + \rho z(\gamma - \mu)\theta + \rho z\eta\mu\gamma, \\ \gamma\lambda(c - \eta\mu) &> \mu\lambda c + \gamma\rho z(c + \theta), \\ \lambda c(\gamma - \mu) &> \gamma\lambda\mu\eta + \gamma\rho z(c + \theta), \end{aligned}$$

where the second inequality of the chain is obtained canceling first the common term $-\mu^2\lambda\theta$ and then the common factor $\gamma > 0$. But the latter inequality is impossible, since $\gamma < \mu$ and its right hand is positive. Then necessarily $\gamma \geq \mu$ whenever the rest point is in Area II, in which case System II is stable and hence, the P2P-SFM is again locally stable in this case.

Q.E.D.

The analysis of the special case $m\overline{x} = \overline{y} + y_0$ is not straightforward, and is tied to global stability. However, a more general result is true (recall that we only require $\gamma > 0$; otherwise, the P2P-SFM is trivially unstable, because the number of seeders will increase indefinitely).

Theorem 4.5.5 *The P2P-SFM is globally stable.*

Proof. The main idea is to prove that the evolution $(x(t), y(t))$ (for each block of the P2P-SFM) rests indefinitely after a certain t_0 either in Area I or Area II, and finally use Lemma 4.5.3 to conclude. The reader can find the complete proof in the Appendix. The proof is close to that of Dongyu Qiu and Wei Qian Sang [168], where the authors study a very similar system, but with a single file and no super-peer assistance (i.e. without the constant term ρz in the minimum argument).

Q.E.D.

The prove can be reproduced from [168], and is included in the Appendix for completeness and self-contained reasons. Lemma 4.5.3 can also be obtained from that previous work, but the most difficult case (local stability in Area II) has its own difficulties as could be appreciated in the end of its proof. The curious reader can find a careful analysis of Linear Switched Systems and a vast number of references in [116].

We briefly review the traditional CDN for this important single-class system (CDN Sequential Fluid Model, or CDN-SFM):

$$\frac{dx_j(t)}{dt} = \lambda_j - \theta x_j(t) - \min\left\{\frac{c}{s_j}x_j(t), \frac{\rho}{s_j}z_j\right\}$$

As a corollary of Theorem 4.5.5, the CDN-SFM is globally stable. The user population converges to the rest point $\overline{x_j}$:

$$\overline{x_j}_{SFM}^{CDN} = \max\left\{\frac{\lambda_j s_j}{\theta s_j + c}, \frac{\lambda_j s_j - \rho z_j}{\theta s_j}\right\}$$

4.5.3 Expected Waiting Times

The performance of the Peer-to-Peer Video on-demand sequential system is never worse than its equivalent CDN version:

Proposition 4.5.6 $T_{SFM}^{P2P} \leq T_{SFM}^{CDN}$.

Proof. By Theorem 4.5.5 the system converges to the rest point. Again, the equality holds when the download is the system's bottleneck. Otherwise, it is enough to show that $\overline{x_j}_{SFM}^{P2P} < \overline{x_j}_{SFM}^{CDN}$, and the result follows from Little's law.

We will construct an auxiliary inequality to conclude linearly the proof.
Observe that $\eta\mu\lambda_j > 0 > -\rho_j z_j \theta$. Adding the term $\theta s_j \lambda_j$ we have:

$$(\theta s_j + \eta\mu)\lambda_j > (\lambda_j s_j - \rho_j z_j)\theta.$$

Multiplying by the negative factor $-\gamma\mu$ on both sides: $-\gamma(\theta s_j+\eta\mu)\lambda_j\mu < -\gamma(\lambda_j s_j-\rho z_j)\mu\theta$. Now, we add the term $(\gamma\theta s_j + \eta\mu\gamma)(\gamma\lambda_j s_j - \gamma\rho z_j)$ on both sides, to get:

$$(\gamma\lambda_j s_j - \gamma\rho z_j)(\gamma\theta s_j + \eta\mu\gamma - \mu\theta) > (\gamma\theta s_j + \eta\mu\gamma)(\gamma\lambda_j s_j - \gamma\rho z_j - \lambda_j\mu)$$

We can rewrite the last inequality as follows:

$$\frac{\gamma\lambda_j s_j - \gamma\rho z_j}{\gamma\theta s_j + \eta\mu\gamma} > \frac{\lambda_j(\gamma s_j - \mu) - \gamma\rho z_j}{\gamma\theta s_j - \mu\theta + \eta\mu\gamma}$$

and the proof follows linearly:

$$\begin{aligned}
\overline{x_j}^{P2P}_{SFM} &= \frac{\lambda_j(\gamma s_j - \mu) - \gamma_j\rho_j z_j}{\theta\gamma s_j - \mu\theta + \eta\mu\gamma} \\
&< \frac{\gamma\lambda_j s_j - \gamma\rho z_j}{\gamma\theta s_j + \eta\mu\gamma} \\
&= \frac{\lambda_j s_j - \rho z_j}{\theta s_j + \eta\mu} \\
&< \frac{\lambda_j s_j - \rho z_j}{\theta s_j} = \overline{x_j}^{CDN}_{SFM}.
\end{aligned}$$

Q.E.D.

In order to understand the consistency of the obtained results we will analyze the sensibility of the expected time $T_{j\,SFM}$ for video j, with respect to the network parameters: entry rates, abortion rates, file sharing efficiency, sizes of the different video items and super-peers capacities. We would like to firstly remark that the number of peers under the rest point does not depend on the seeders aborting rate if it is large enough. In fact, when $\gamma s_j >> \mu$ we have that

$$\overline{x_j}_{SFM} \approx \frac{\gamma(\lambda_j s_j - \rho z_j)}{\gamma(\theta s_j + \eta\mu)} = \frac{\lambda_j s_j - \rho z_j}{\theta s_j + \eta\mu}. \tag{4.18}$$

In real-life networks, the seeders usually abort after complete downloading, and Expression (4.18) is indeed a good approximation. Additionally, the upload is the system's bottleneck, and $\gamma s_j >> \mu$ is a reasonable assumption. On the lights of Theorem 4.5.5 we conclude that sequential video on-demand networks are globally stable. Via Little's law we can find a rough approximation for the expected download times for video $j \in [K]$ in the SFM:

$$T_{j\,SFM} = \frac{\overline{x_j}^{P2P}_{SFM}}{\lambda_j} = \frac{1}{\lambda_j}\frac{\lambda_j s_j - \rho z_j}{\theta s_j + \eta\mu} \tag{4.19}$$

By direct derivation of Expression (4.19) with respect to the network parameters, it can be observed that:

1. The waiting times are monotonically increasing with respect to the videos sizes. This is consistent with our intuitive idea that bigger files will take more time.

2. If the entry rates increase, peers will wait more. This is a common element of waiting systems with limited resources.

3. When the abortion rates of peers θ is increased, the expected waiting times are consequently reduced. It is evident that peers that depart before downloading will experience lower time excursions, whereas the number of peers under steady state is hence reduced. The departure rates play the role of a *decay* in the entry rate.

4. Naturally, when the sharing efficiency η is increased, the throughput of the system is increased as well, and the mean waiting times are consequently reduced.

5. Finally, the throughput of the system increases with the super-peers capacity ρ, and the number of replicas for video j in the network, z_j.

4.6 Combinatorial Optimization Problem

4.6.1 Description

The main goal of this work is to minimize the mean waiting times in a progressive video on-demand system, assisted by super-peers managed by the operator. For the sake of simplicity and efficiency, we will focus on the P2P sequential fluid model (P2P-SFM), in which closed forms can be obtained for the mean waiting times.

Let us denote X to the random variable that represents the class of an entry peer in the SFM. The poissonian process with intensity rates λ_i imply that $P(X = i) = \frac{\lambda_i}{\lambda}$, being λ the sum rate. The mean waiting time of a user in the P2P-SFM can be found analogously to the GFM (by using Little's law and conditional expectation):

$$E(T) = \frac{1}{\lambda} \sum_{j=1}^{K} \overline{x_j}^{P2P}_{SFM}$$

Accordingly, the mean waiting time is proportional to the whole population size, so we will minimize the latter. We must decide the number of video replicas among P super-peers. The decision variable is a binary matrix E of size $P \times K$, whose entries are $E(p, j) = 1$ if and only if we store video item $j \in [K]$ in super-peer $p \in [P]$. We also impose that every video item must be duplicated, for availability and redundancy reasons. Let u_n be the unit column vector of n elements (all its entries are 1), $s = (s_1, \ldots, s_K)^t$ the video sizes and $S = (S_1, \ldots, S_P)^t$ the super-peers' storage capacity.

We define the Caching Problem in matrix form as follows:

$$\min_{E} \sum_{j=1}^{K} \max \left\{ \frac{\lambda_j s_j}{\theta s_j + c}, \frac{\lambda_j s_j - \rho z_j}{\theta s_j + \eta \mu} \right\}$$

$$s.t.$$

$$\begin{cases} E \times s \leq S & \text{(4.20a)} \\ E^t \times u_P = z & \text{(4.20b)} \\ z \geq 2u_K & \text{(4.20c)} \\ E(p,j) \in \{0,1\},\ \forall p \in [P], j \in [K]. & \text{(4.20d)} \end{cases}$$

The objective is to minimize the mean waiting times. Constraint (4.20a) states that super-peer's storage capacity cannot be exceeded. Constraint (4.20b) relates the number of replicas for video item j, named z_j, with the matrix E (summing its columns). Constraint (4.20c) imposes that each video item must be available in the network at least twice, whereas Constraint (4.20d) just states E is a binary matrix. We will prove that the feasibility problem is NP-Complete. As a corollary, we will have that the decision problem whether exists or not a solution that achieves a lower bound is also NP-Complete. The following decision problems will be useful.

Definition 4.6.1 *$PARTITION$: Given a set of positive integers $A = \{a_1, \ldots, a_n\}$. Is there a subset $B \subset A$ such that $\sum_B a_i = \sum_{B^C} a_i$?*

$PARTITION$ is an NP-Complete decision problem [80].

Definition 4.6.2 *$CACHING{-}FEASIBILITY$: Given an arbitrary instance of the Caching Problem. Does it accept a feasible solution?*

Definition 4.6.3 *$CACHING - QUALITY$: Given an arbitrary instance of the Caching Problem and a positive real c. Does it accept a feasible solution whose score is at least c?*

Theorem 4.6.4 *$CACHING - FEASIBILITY$ is NP-Complete.*

Proof. We will prove both that $CACHING{-}FEASIBILITY$ is in NP and find a reduction from $PARTITION$, hence proving its hardness. Given an arbitrary instance of the Caching Problem and a binary matrix E, we can determine in polynomial time by a non-deterministic Turing Machine whether the solution is feasible or not. In fact, we find $E \times s$ in $O\big(K^2\big)$ operations, so Constraint (4.20a) can be checked in polytime with the input K. The vector z can also be computed by a matrix times a vector, in $O\big(K^2\big)$ sums, and compared with $2u_K$ in $O(K)$ operations. Therefore, $CACHING - FEASIBILITY$ is in NP.
We will find a reduction from $PARTITION$ now. To close the proof, we will show that if we can determine feasibility of an arbitrary instance of the Caching Problem in polynomial time, then we would solve $PARTITION$ in polynomial time as well. Consider an arbitrary set of positive integers $A = \{a_1, \ldots, a_n\}$ and let us call $a_{sum} = \sum_{i=1}^{n} a_i$. We construct the following Caching Problem instance, with $P = 3$, $S_1 = S$, $S_2 = S_3 = a_{sum}/2$ and $s_j = a_j$ for all $j \in [n]$. It is clear that this transformation is polynomial. Given that $S_1 + S_2 + S_3 = 2\sum_{i=1}^{n} s_i$, Constraint (4.20c) forces the three super-peers to store video items at their full capacity. Therefore, a feasible solution complies that $S_2 = \sum_B a_i = \sum_{B^C} a_i = S_3 = a_{sum}/2$, for a certain $B \subset A$. As a consequence, if $CACHING - FEASIBILITY$ can be solved for every instance in polynomial time, then every instance of $PARTITION$ can be solved in polynomial time as well. This concludes that the decision problem $CACHING - FEASIBILITY$ is NP-Complete.

Q.E.D.

Corollary 4.6.5 *$CACHING - QUALITY$ is NP-Complete.*

Observe that the objective function depends on the variable $z_j = \sum_p E(p,j)$ if and only if the objective function of the Caching Problem is dominated by its second argument. This is true if and only if:

$$\frac{\lambda_j s_j - \rho z_j}{\theta s_j + \eta\mu} > \frac{\lambda_j s_j}{\theta s_j + c}. \tag{4.21}$$

Inequality (4.21) can be re-written to obtain:

$$2 \leq z_j < \frac{\lambda_j}{\rho} \frac{c - \eta\mu}{c + \theta s_j}, \tag{4.22}$$

where we added the constraint $z_j \geq 2$.

The following remarks are corollary of Expression (4.22), and help to understand the limits in the design of a P2P Video-on Demand assisted service:

1. Unless $c > \eta\mu$, the peers contribution to the system can be neglected without sacrificing performance. In practice the peers download capacity is always higher than its upload, so the inequality holds.

2. If the super-peers' capacity ρ (or server capacity in a CDN) is extremely high in relation with the peers needs λ_j, the cooperation can be neglected again.

3. The decision variable z_j is also upper-bounded. In fact, the streaming rate is divided in the down-link once a new video item is included in the super-peers' storage. Inequality (4.22) represents a threshold for z_j.

We will solve the Caching Problem under different scenarios in order to show the strength and limitations of a P2P Video-on-Demand assisted service.

4.6.2 Greedy Randomized Resolution

GRASP (Greedy Randomized Adaptive Search Procedure) is a well known metaheuristic, which has been applied for solving many hard combinatorial optimization problems with high-performance. A GRASP is an iterative process, in which each iteration consists of two phases: construction and local search. The construction phase builds a feasible solution, whose neighborhood is explored by local search. The best solution over all GRASP iterations is returned as the result. Details of this metaheuristic can be seen in the Appendix.

We present a GRASP resolution for the Caching Problem. The metaheuristic can be studied in two stages. The first one constructs a seed of our GRASP-heuristic, and it is named $GreedySeed$. The second stage is a classical local search improvement, named $LocalSearch$.

In $GreedySeed$ every video is greedily stored in the two *fattest* super-peers (i.e. the ones with the highest remaining storage capacity). Note that in this multi-knapsack flavored problem the costs are the item's sizes s_j, whereas the profits are the reduction of the population $\overline{x_j}$.

Then, we introduce the benefit-to-cost vector $W = (w_1, \ldots, w_K)$ such that

$$w_j = \frac{1}{s_j \overline{x}_{j\,SFM}^{P2P}}.$$

Note that an increment in the benefit-to-cost ratio represents cache or time savings. The population size $\overline{x}_{j\,SFM}^{P2P}$ depends on the number of super-peers z_j seeding video j, which *a priori* is unknown. For that reason, we compute first an approximation $W' = (w'_1, \ldots, w'_K)$ for the vector W:

$$w'_j = w_j|_{z_j=0} = \min\left\{\frac{\theta s_j + c}{\lambda_j s_j^2}, \frac{\theta s_j + \eta\mu}{\lambda_j s_j^2}\right\} \tag{4.23}$$

In practice the peer's upload capacity is always lower than its download, so $\eta\mu < c$, and:

$$w'_j = \frac{\theta s_j + \eta\mu}{\lambda_j s_j^2} \tag{4.24}$$

Without loss of generality, we will assume that $w'_1 > w'_2 > \ldots > w'_K$ (in other words, videos are numbered in decreasing benefit-to-cost ratio when $z_j = 0$).

$GreedySeed$ is specified in Algorithm 1. The vector W' is computed in Line 1, using Equation (4.24). The video items are sorted in decreasing cost-benefit ratio, in Line 2. In the iterative block (Lines 3 to 8) each video item is assigned in turns to the two fattest super-peers, named p_1 and p_2. Video item j is then stored in both super-peers: the decision variables $E(p_1, j)$ and $E(p_2, j)$ are turned-on in Lines 5 and 6 respectively. Finally, the super-peer resources $\{S_i\}_{i=1,\ldots,P}$ are updated in Line 7. Under optimistic scenarios (enough memory in cache nodes), Algorithm $GreedySeed$ returns a feasible solution, contained in the decision matrix E.

Algorithm 1 $E = GreedySeed(\lambda, \theta, \gamma.\eta, s, S, \mu, c, \rho)$

1: $W' \leftarrow \mathsf{FindW}(\lambda, \theta, \gamma.\eta, \mathsf{s}, \mu, \mathsf{c}, \rho)$
2: $SortVideos(W')$
3: **for** $j = 1$ TO K **do**
4: $\quad (p_1, p_2) \leftarrow \mathsf{TwoFattest}(S_1, \ldots, S_P)$
5: $\quad E(p_1, j) \leftarrow 1$
6: $\quad E(p_2, j) \leftarrow 1$
7: $\quad Update(S_1, \ldots, S_P)$
8: **end for**
9: **return** E

In order to improve the solution E returned by $GreedySeed$, a local search improvement is introduced in a second stage. The idea is very simple: in each step we first try to add, delete or swap video-items in super-peers. The pseudo-code for this local search stage is shown in Algorithm 2.

Algorithm 2 $E^* = LocalSearch(E)$

1: $(E, improve) \leftarrow \mathsf{Add}(Rand(SP, Video))$
2: IF $improve$ GO TO Line 1
3: $(E, improve) \leftarrow \mathsf{Delete}(Rand(SP, Video))$
4: IF $improve$ GO TO Line 1
5: $(E, improve) \leftarrow \mathsf{Swap}(Rand(SP1, V1), Rand(SP2, V2))$
6: IF $improve$ GO TO Line 1
7: $E^* \leftarrow E$
8: **return** E^*

Remark 4.6.6

1. *Functions Add, $Delete$ and $Swap$ work only if the new solution is feasible and better.*
2. *The effects of Functions Add and $Delete$ never cancel-out.*

4.7 Results in a Real-Life Scenario

Currently, GoalBit supports high-quality Live and on-Demand video streaming to end users. We wish to improve the performance of the VoD distribution by adding or removing video replicas in the system. In order to predict the behavior of our new storage-scheduling technique, we picked up real-life traces taken from YouTube. A crawler script was designed to collect some useful information as follows:

(1) We take a video URL to start.

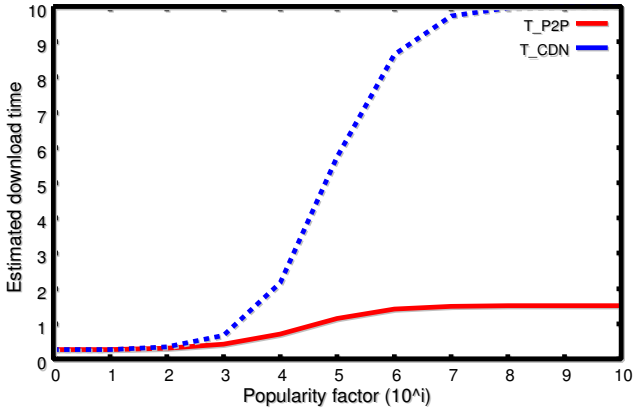
(2) From this URL we get useful video data (size, time online, number of views, and others).

(3) We save this data in a database.

(4) We collect all the related videos URLs, and go back to Step (1) with a new video URL.

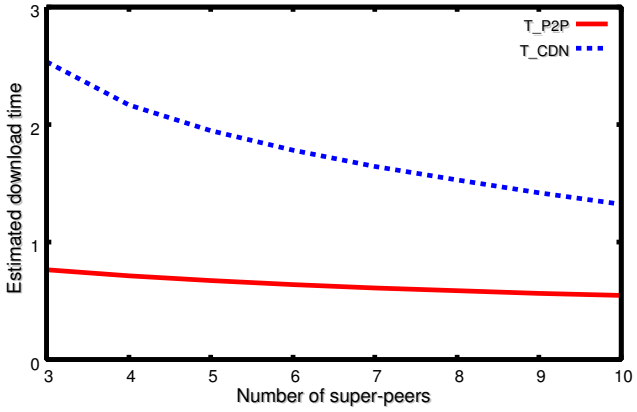
This process was executed during 3 days, allowing us to have useful information of more than 59.000 YouTube videos. With this information we estimated *videos' popularities* λ_j based on the number of views and on-line time. We stress the system introducing a popularity factor β to the vector $(\lambda_1, \ldots, \lambda_K)$. In this way, we can contrast the performance of a CDN vs P2P deployment in flash-crowded, low-populated and intermediate scenarios. We use an abortion rate of $\theta = 0,1$ peer per second, file sharing efficiency of $\eta = 0,5$, download rate of $c = 1$ Megabytes per second, $d = c/4$, and a system with $P = 4$ super-peers (or servers) with a capacity of $\rho = 10$ Megabytes per second, storing $K = 59000$ video items. Figure 4.2 shows the estimated download time for the P2P and CDN models (with solid and dashed lines respectively) versus i, where the stress factor β takes values 10^i, $i = 1, \ldots, 15$. Figure 4.2 underlines two essential features. First, the expected time for a P2P sequential system is never worse than the one of a traditional CDN system, as can be predicted by Proposition 4.5.6. Second, the performance of both systems is quite similar for low-populated scenarios, whereas the time savings for peers are remarkable in high-populated scenarios.

Figure 4.2: Download time for CDN and P2P when increasing popularity



A second experiment was conducted to figure-out how the system's performance can be affected in terms of scalability. For a fixed popularity factor we want to find the mean waiting time for different number of super-peers (servers). Figure 4.3 illustrates the average waiting time for both P2P and CDN systems (with solid and dashed lines respectively) versus P, where P is the number of super-peers (servers) in the system. We fixed the popularity factor $\beta = 10^3$, but a similar behavior can be appreciated for other popularities. From this experiment, we can conclude that P2P system can work similarly with less resources, while CDN has a very important variability in its performance when increasing the number of servers. The results suggest that peers can download the desired video item nearly ten times faster than users in a traditional CDN, in massive scenarios. This suggests that the average waiting time in the P2P system is consistently low, whereas the CDN performance is effectively improved distributing the load to more servers. All tests where executed in a home-PC (Intel Core i7, 8 GB RAM), getting more than 300.000 modifications during the $LocalSearch$ phase, with a running-time of 14 hours for each experiment.

Figure 4.3: Download time for CDN and P2P versus the number of cache-servers



4.8 Conclusions and Future Work

In this chapter a general framework for the analysis and design of concurrent and sequential video on-demand assisted services is provided. Under this framework of expected evolution, the sequential system is always globally stable, converging to a known rest point. We found closed expressions for the expected waiting times in both CDN and P2P approaches, and theoretically confirmed the peer-to-peer philosophy always outperforms traditional CDNs, showing the first hint of effectiveness in the assisted cooperative network.

An experimental validation of the P2P and CDN systems and their performance is presented regarding real-traces passively taken from a YouTube Crawler. The results are encouraging, showing that a P2P assisted platform preserves its resilience against adverse environments like flash crowds. However, an end-user is predicted to wait even five times longer in a pure CDN when a flash crowd is encountered.

There are several aspects and open problems that deserve further research. We are interested in the peer-assisted performance in concurrent scenarios, when users enjoy (or better progressively download) more than one video content simultaneously. The multiple-video application could be useful for the monitoring of security systems, or to better exploit peer resources, while they play one movie and completely download other video items. We proved the cooperative philosophy outperforms the traditional client-server architecture, but we could not determine the conditions to state global stability for the concurrent fluid model. Additionally, we address a static Multiple Caching Problem, where video items are stored before-hand. However, a challenging design would dynamically adapt the cache contents by monitoring or predicting changes in a controlled system, regarding the video popularity and transient number of peers. Our trends for future work include stability and capacity analysis in concurrent video on-demand assisted scenarios, and the implementation of concurrent services in the GoalBit platform.

Chapter 5

A Pull-Mesh Model for Live Streaming P2P Networks

5.1 Introduction

In Chapter 3 we have discussed the main challenges in the design of both effective and efficient peer-to-peer networks to offer live video streaming. Mesh-based overlays show to provide flexibility enough to address a tree delivery in a dynamic way, hence better exploiting the bandwidth capabilities of the overall network [123, 132]. Peers in the system can either communicate in a one-sided way with reduced overheads, or two-sided way, trading overhead for information availability and better opportunities in the design of scheduling protocols. Push systems are suitable for one-sided communication, in which the sender chooses both the the chunks to send and target peer. However, there are rigorous works suggesting that one-sided push protocols suffer in live streaming a problem similar to starvation in file sharing (the last chunks are hard to be found), whereas one-sided pull protocols are slow at the beginning of the dissemination process [190]. Additionally, in the real-world there are already successful commercial platforms such as PPLive [163], PPStream [164], SopCast [200], TVAnts [209] and TVUNetworks [210], offering live video streaming to hundreds of thousands of concurrent users. Unfortunately, all of them have proprietary protocols not available for academia. Nevertheless, by reversal engineering there are strong evidences to support that all of them use a gossip-style two-sided communication protocol, in a mesh-based fashion. They are all considered BitTorrent-based networks, as a reference to the many similarities they share with the BitTorrent's philosophy [44].

However, there is a common agreement in the scientific community that BitTorrent is near-optimal for file-sharing, but does not comply real-time requirements for live-streaming services. Several works confirm its main drawback for live streaming is its chunk scheduling policy: Rarest First [34, 57, 214, 229, 233]. As a consequence, a great research effort has been focused on the best design of chunk scheduling policies, given a certain peer selection policy (usually, approaching a random overlay). The first tractable chunk scheduling model for live streaming is presented by Yipeng Zhou, Dah Ming Chiu and J.C.S. Lui [233]. The system there

proposed takes into account foundational characteristics of live streaming, such as cooperation, synchronism, real time constraints, playback continuity and chunk scheduling policy. Specifically, it is a two-sided pull based cooperative process where peers have scarce resources, and the metrology includes the two most important video factors, namely start-up latency and playback continuity [182]. Different chunk scheduling policies determine different network states under regime, hence the model can discriminate and contrast the performance of different policies. Other models similar in spirit are [26, 34, 137, 190, 214]. Bonald, Massoulié, Mathieu, Perino and Twigg study the performance of peer selection and chunk scheduling policies in an epidemic-style [26]. They measure the playback continuity counting chunk losses, but the delay is measured from source to end-user, not capturing the needed time for a peer to reach the state of others when joining the system. Massoulié and Vojnović study the stability of flat and layered systems (i.e. in which the interaction takes effect between peers with the same number of coupons). They provide valuable theoretical results, with focus on file-sharing rather than live services. Furthermore, peers are welcome with one coupon from the server, which turns the system non-scalable [137]. Sanghavi, Hajek and Massoulié study a gossip-based peer communication [190]. They show pros and cons of push and pull-based schemes for mesh overlays and one or two-sided protocols, but the chunk policies are not structural but stochastic or descriptive (random useful, blind chunk, most deprived peer, etc.). Their work is complementary to the original from [233]. BitOS is a BitTorrent-based system were new chunk policies are provided, that outperform Rarest First only via simulations, but without introducing a mathematical model [214]. The most similar approach is given by Chatzidrossos, Dan and Fodor, with a slight more complex system [33, 34]. They also introduce node churn and study chunk policies in a buffer-level. However, only 4 policies are there designed and compared, and the model does not seem flexible enough to measure the performance of a wide variety of chunk policies.

In this chapter an in-depth analysis of the cooperative model from [233] is performed. Sections 5.2 and 5.3 are essentially adapted from [233], introducing respectively model description and classical scheduling policies. Section 5.4 summarizes a robustness analysis of the model, detailed in Zhou Yipeng's thesis [63]. The main novelties of this chapter are presented in Sections 5.5 to 5.9, and were disseminated in proceedings [16, 17, 186, 187] and journals [184, 185].

Specifically, Section 5.5 contains a preliminary analysis of the simple model. Bounds in the playback continuity and start-up latency are there provided. An Ideal policy is introduced, and provides a universal bound for playback continuity not achievable by any feasible policy. A stochastic policy is introduced for didactic reasons, where peers can skip useful chunks trying to maximize information availability. We mathematically show its performance is poor, and give an insight that a forced pull is mandatory once a useful chunk is found. As a consequence, a deterministic family of Permutation-based policies is our universe to search for high-performing policies, which contains previous classical policies (named Rarest First, Greedy and a Mixture). A Follower System is introduced, trying to reflect the performance achieved by desired buffer states under regime (the buffer states statistically determines the performance of the system). This first *ideal approach* works as a *dirty mirror*. However, via

experimental setups the results outstand a special subfamily of policies, called the Subfamily of W-shaped policies, which has polynomial cardinality. Section 5.6 presents a feasible and more sophisticated approach to address the playback-buffering trade-off. First, a single-score is defined for chunk scheduling policies which captures this trade-off, in an intrinsic-system way, counting the number of steps during a peer request. A combinatorial optimization problem is then proposed, which is suitably translated into an Asymmetric Traveling Salesman Problem (ATSP). The latter problem is finally solved heuristically, following an Ant Colony Optimization-inspired approach. This resolution returns chunk scheduling policies with better performance than classical ones, both in a theoretical and empirical way. Indeed, the metrics from the model confirm this theoretical breakthrough, and the new policy is introduced in a real live streaming platform called GoalBit in Section 5.7. The cooperative model in its purest form has identical nodes. Section 5.8 extends the model, giving basic insights of heterogeneity and hints to deal with free riders. The Extended Model outstands the importance of full knowledge, showing that the system is scalable even under presence of free-riders, whenever servers can recognize (and punish) them. The results highlight trade-offs between contribution awareness and overhead (in order to monitor peers resources and level of altruism). The section concludes with simulations regarding networks in which normal peers interact with double-peers (who doubles the resources of the formers). Finally, Section 5.9 contains concluding remarks.

5.2 Model Description

Consider a closed system with a single source-node (the server), and a fully-connected mesh with $M > 1$ identical peers, with buffer capacity N. All peers are connected to the server, and wish to display the same live video channel. The server iteratively organizes the unlimited video channel in chunks of equal size, and picks a peer uniformly at random to push the current chunk. The time is slotted, and it is assumed the server periodically disseminates one video chunk in playback order, and the period (i.e. one time slot) lasts exactly one chunk consumption in the media player. At the end of a time slot, all chunks are shifted up one step closer to the deadline, and the last chunk being played is removed from the buffer. All media players are assumed to be synchronized with time. The user will experience a *video-cut* whenever the chunk at position N is missing, hence skipped by the media player. Figure 5.1 illustrates the buffer structure. Note that if the chunks are numbered $1, 2, \ldots$ and the time slot is $t \geq N - 1$, then the chunk being played has number $t - N + 1$.

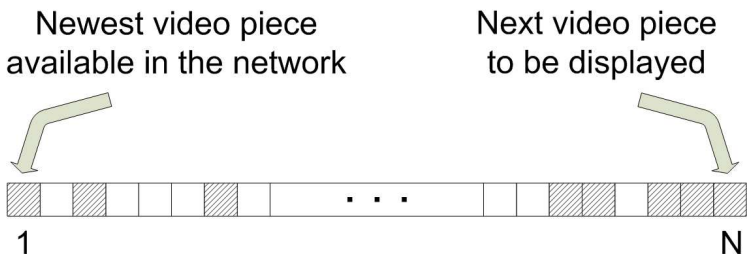


Figure 5.1: Buffer structure for each peer. Buffer-cell 1 has the newest video chunk of the system, whereas buffer-cell N contains the chunk currently being played on the screen.

The buffer state can be represented by a binary word of N bits, where the symbol 1 means that the corresponding buffer-cell is filled with a chunk, whereas the symbol 0 means an empty buffer-cell. For example, the bitmap 00101 means that the buffer size is $N = 5$, the local peer is currently playing a correct video chunk, will certainly watch another correct video chunk two time slots later but will have a video cut in the next time slot, unless it requests in this time slot from another peer. Naturally, all peers possibly have different buffer states, but if the cooperation is symmetric, the states will be statistically similar. Peers want to fill buffer cell N as many times as possible, in order to have a continuous playback with no cuts. The following definitions provide means to measure the overall network performance, under regime and symmetric peer conditions.

Definition 5.2.1 *Let us denote with p_i the occupancy probability of buffer-cell $i \in [N] = \{1, \ldots, N\}$, where the buffer set is denoted by $[N]$ for short. The vector $p = (p_1, p_2, \ldots, p_N)$ will be called* bitmap probability*, or just bitmap when there is no danger of confusion.*

Definition 5.2.2 *The* playback continuity *or playback-delivery ratio is measured by $c = p_N$, and represents the ratio between chunks correctly played and chunks delivered by the server, when the number of chunks tends to infinity.*

Suppose for a moment that peers do not cooperate. This is the case of a server with scarce resources, which can feed at most one peer. It is clear that under this scenario, each buffer cell will be filled one time slot out of M, and $p_i = \frac{1}{M}$ for all $i \in [N]$ and all peers. In particular, the playback continuity will be $p_N = \frac{1}{M}$. Therefore, the performance of the client-server architecture is miserable with scarce upload resources, and peers will eventually experience several video cuts. Therefore, peers are forced to cooperate in order to enjoy the video stream reasonably. In this model peers cooperate following a pull-based scheme, basically, pressed by requests. At the beginning of each time slot, all peers choose a peer uniformly at random to request for chunks. Peers can either finish the time slot with no additional chunk (a failed slot) or with one chunk (a successful slot). A request works as follows. Suppose peer A picked a random partner, which we call B, to pull one chunk. Following a specified chunk scheduling policy, peer A can check en empty buffer-cell, and ask peer B whether it has the corresponding chunk or not (recall peers are synchronized with time). If peer B does not own that chunk, A can try other buffer position, and the process is repeated. If peer A is lucky, it will download one peer from B in that time slot. On the contrary, if A could not find a chunk after an exhaustive revision of the buffer, that time slot was not useful for him (peer A cannot connect to other peer different from B in the same time slot).

Assume all peers follow the same chunk scheduling policy, and the system reaches a regime with bitmap p. If an exogenous peer C joins the network, we would like to know how many time slots will be necessary for that peer to reach the same bitmap. The key observation here is that peers within the network have an expected number of $L = \sum_{i=1}^{N} p_i$ chunks in the whole buffer, and C will have all successful slots at the beginning. A rough approximation is to consider that C will in fact have L successful time slots.

Definition 5.2.3 *The* latency *or buffering time, is the expected time for an exogenous peer to reach the bitmap of the system under regime, and will be measured by $L = \sum_{i=1}^{N} p_i$*

Now we will measure the playback continuity and latency for any given chunk scheduling policy. Under a cooperative regime, peers can get chunk at position p_{i+1} either by promotion with time (i.e the buffer cell at position i was already filled in the previous time slot) or pulling that chunk in the previous slot, with probability q_i:

$$p_{i+1} = p_i + q_i, \forall i \in [N-1]. \tag{5.1}$$

We need to find an explicit expression for the cooperative terms q_i. Consider the following events:

- $WANT(A, i)$: peer A has no chunk at buffer-cell i.
- $HAVE(B, i)$: peer B owns the chunk at buffer-cell i.
- $SELECT(A, B, i)$: peer A pulls chunk at buffer-cell i from B.

We shall use the first capital letters for short. Therefore, peer A pulls chunk i exactly when the three simultaneous conditions are met, i.e. $q_i = P(W(A, i) \cap H(B, i) \cap S(A, B, i))$, or using conditional events:

$$q_i = P(W(A, i))P(H(B, i)/W(A, i))P(S(A, B, i)/(W(A, i) \cap H(B, i))). \tag{5.2}$$

Clearly, under regime and fair network conditions we have $P(W(A, i)) = 1 - p_i$ for all peers, and with high number of peers it is reasonable to consider that $P(H(B, i)/W(A, i)) = P(H(B, i)) = p_i$. We will further assume that the chunks are independently distributed in the network, so: $P(S(A, B, i)/(W(A, i) \cap H(B, i))) = P(S(A, B, i))$.
Equation (5.2) can be re-written:

$$q_i = (1 - p_i)p_i s_i, \forall i \in [N-1]. \tag{5.3}$$

Replacing Equation (5.3) in (5.1), we get a recursive formula for the bitmap:

$$\begin{cases} p_1 & = \frac{1}{M} \\ p_{i+1} & = p_i + (1 - p_i)p_i s_i, \forall i \in [N-1] \end{cases} \tag{5.4}$$

Definition 5.2.4 *Given a chunk scheduling policy, the* strategic sequence s_i *represents the probability to select chunk at buffer-cell i.*

The recursive bitmap formula (5.4) will be used several times in this chapter. We encourage the reader to keep in mind the following interpretation for it: buffer-cell at position $i + 1$ can be filled either by promotion with time (with probability p_i) or pulling from one peer (with probability q_i). The latter event occurs exactly when the local peer does not own buffer-cell i (event with probability $1 - p_i$), the requested peer does (event with probability p_i) and there is not better chunk to be downloaded (event with probability s_i).
Given a scheduling policy, we will have a different strategic sequence s_i. The aim of this chapter is to find chunk policies which achieve high playback continuity p_N and at the same time low buffering times, given by $L = \sum_{i=1}^{N} p_i$. The reader might feel the model has an excessive number of assumptions. However, Section 5.4 shows the robustness of the model. First, we will get some familiarity with the model presenting classical policies and finding explicit expressions for the strategic sequence s_i in each case.

5.3 Classical Policies and a Mixture

This section presents two classical scheduling policies, named Rarest First and Greedy, and a mixture of them (named simply Mixture). Rarest First enjoys a prestigious place nowadays, being highly deployed in currently file-sharing systems like BitTorrent [44]. In the Rarest First policy, all peers try to pull the rarest chunk among the neighboring peers, in this case, the rarest chunks in the global network (the network is fully connected). Given that the cooperative term q_i is a probability, the bitmap increments when we get closer to the playback deadline. More specifically, $p_{i+1} \geq p_i$ for all $i \in [N-1]$. Therefore, a peer following the Rarest First policy tries to pull first chunk at buffer-cell $i = 1$. If it fails, the next rarest chunk is $i = 2$, and so on, until getting a chunk or loosing the time slot. The strategic sequence for Rarest First is then:

$$s_i = (1 - \frac{1}{M}) \prod_{j=1}^{i-1} [1 - (1 - p_j)p_j], \forall i \in [N-1]. \tag{5.5}$$

Observe that a success occurs at position j only if the local peer does not own that chunk and the requesting peer does, with probability $(1-p_j)p_j$. Expression (5.5) has a clear interpretation. In order to reach the buffer-cell i during a request, the local peer should not be chosen by the server (event with probability $1 - 1/M$) and a fail must occur in all previous buffer-cells. It is subtle but important to notice that a peer cannot download more than one chunk during a slot, hence a request is not suitable for the peer chosen by the server.
An alternative selfish-nature policy is the Greedy notion for this problem. As soon as we know we want to minimize losses, a greedy solution would always ask the nearest-to-deadline chunk first. In this case, Greedy is the opposite policy to Rarest First, and the request starts at the closest-to-deadline position $i = N - 1$ (buffer-cell N is actually being played and cannot be requested, cause it will not be useful in the next time slot). The strategic sequence for Greedy is:

$$s_i = (1 - \frac{1}{M}) \prod_{j=i+1}^{N-1} [1 - (1 - p_j)p_j], \forall i \in [N-1]. \tag{5.6}$$

The interpretation is analogous to that of Rarest First. Figure 5.3 shows a structural buffer representation of both classical strategies.

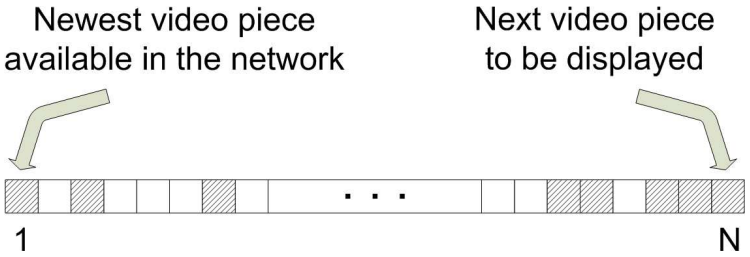


Figure 5.2: Request order following classical policies. Greedy requests the nearest-to-deadline first (buffer-cell $N - 1$), whereas Rarest First applies an opposite rule (starting from position 1).

Expressions (5.5) and (5.6) can be substantially simplified:

Proposition 5.3.1 *For Rarest First we have that:*

$$s_i = 1 - p_i, \forall i \in [N-1]. \tag{5.7}$$

Proof. By induction over the finite set $[N-1]$. The product from Expression (5.5) has no factors hence equals 1 when $i = 1$. Therefore $s_1 = 1 - \frac{1}{M} = 1 - p_1$, and the base step holds. If we assume the result holds for some positive integer $h < N - 1$, then using again Expression (5.5) we get that:

$$\begin{aligned} s_{h+1} &= s_h[1 - p_h(1-p_h)] = (1-p_h)[1-p_h(1-p_h)] \\ &= 1 - [p_h + p_h(1-p_h)^2] = 1 - p_{h+1}. \end{aligned}$$

Q.E.D.

An analogous result holds for Greedy.

Proposition 5.3.2 *In Greedy:*

$$s_i = 1 - (p_N - p_{i+1}) - p_1, \forall i \in [N-1]. \tag{5.8}$$

Proof. By induction over the finite set $[N-1]$, starting in the base step $i = N-1$ down-to $i = 1$. By evaluation from Expression (5.6) we get that $s_{N-1} = 1 - 1/M = 1 - p_1$, so the base step holds. Assume that the result is correct for certain h such that $1 < h < N$, i.e. $s_h = 1 - (p_N - p_{h+1}) - p_1$ Combining the general bitmap recursion (5.4) and (5.6):

$$\begin{aligned} s_{h-1} &= s_h[1 - p_h(1-p_h)] = s_h - p_h(1-p_h)s_h \\ &= (1 - (p_N - p_{h+1}) - p_1) - (p_{h+1} - p_h) = 1 - (p_N - p_h) - p_1). \end{aligned}$$

Q.E.D.

Alternative proofs for Propositions 5.3.1 and 5.3.2 can be found in [233]. It is well-known that Greedy achieves low buffering times, but is not as scalable as Rarest First [217, 233]. Intuitively, Greedy pulls urgent chunks, conditioning the buffer fast, but rarest chunks are difficult to get. On the other hand, Rarest First presents good playback at the cost of higher buffering times, not suitable for live streaming purposes. A Mixture of both classical policies is feasible, cutting the buffer into two parts, applying Rarest First in buffer cells $[m]$ for a certain $m \in [N-1]$, and finally applying Greedy in the buffer-set $[N-1] - [m]$. This policy offers a lower start-up latency than Rarest First, and simultaneously a good playback. It is suggested to see [233] for more details of the three policies.

Figure 5.3 shows graphically the bitmap for both classical policies and the Mixture, with common network values of $M = 1000$ peers with buffer capacities $N = 40$.

5.4 Model Robustness

At first, the model sounds extremely simplistic. In Yipeng Zhou's thesis the robustness of this model is thoroughly covered [63]. In this section, we will sketch the major concerns there addressed, to better understand the strength of this simple cooperative model. Specifically, we would like to answer the following questions:

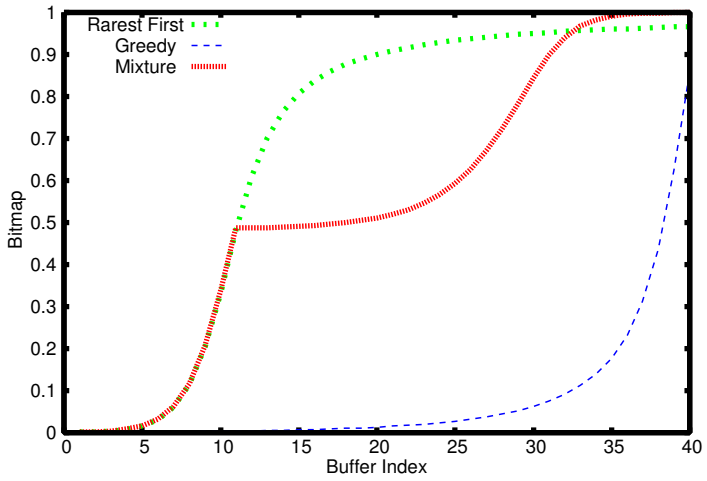


Figure 5.3: Bitmap for classical policies and Mixture, with $M = 1000$ peers and buffer size $N = 40$.

1) Does the system reach a stationary state?

2) The model assumes infinite upload bandwidth. Is it realistic?

3) Is synchronism the best option for peers?

4) Can we assume without loss of generality that all peers own an identical buffer?

5) Are the independence assumptions good enough?

6) Is the expression $L = \sum_{i=1}^{N} p_i$ a fair approximation for the start-up latency?

All the answers are affirmative, when the peer population M is big. For the first one, a great deal of validations are provided, while the rigorous mathematical prove remains open.

For the second one, fix an arbitrary peer A in the network. A random peer selection policy implies the random variable X_A counting the times A is selected to upload is binomially distributed, specifically $X_A \sim Bin(M-1, \frac{1}{M-1})$. Hence, the expected number of peers picking peer A to upload is 1. At the same time, A could be picked more than three times with a vanishing probability when M is large enough. Indeed, for M large enough the Poisson approximation holds, and $X_A \approx P(1)$ with $P(X_A \leq 3) > 0,98$. This gives evidence that the model works similarly in an upload-unconstrained version. Validations further strength this probabilistic facts, regarding unit bandwidth peers and letting the system evolve. This property is exploited in an extended version of this model in Section 5.8.

The third and fourth questions are deep, and related. A fair comparison between an heterogeneous buffer system and homogeneous equivalent one shows advances to the latter. Indeed, when the swarm is synchronized, peers have no incentives to change their offsets (in this way, keeping the greatest buffer overlap to cooperate) and therefore the synchronized cluster is a

Nash Equilibrium (in fact, the only one), so the answer for the third question is affirmative. If peers are synchronized in the playback and peers could choose to have different buffer sizes, it is interesting to compare an heterogeneous buffer system with its corresponding homogeneous average-buffer size. Assuming a *rationale peer selection* condition in the heterogeneous environment (peers always send request to useful potential uploaders), the system is equivalent to the homogeneous case (the recursive buffer-map holds again). However, there are disadvantages when an heterogeneous buffer system is considered: the average playback continuity is lower than its equivalent homogeneous system, at least, on the lights of the Rarest First policy. The independence assumption and rough approximation for start-up latency are quite realistic for large population sizes, as validations show.

In order not to extend the list, we just point-out that the connectivity of the system can be relaxed, and simulations also show that over a lower-bound degree the bitmap gaps are negligible. The model is not strongly affected under resourceless peers who cannot download the streaming rate but a factor $f \in (0, 1)$ of it.

The reader can find more precise statements to support the robustness of the model in Yipeng's thesis [63]. In Section 5.8 an extended model is introduced, providing some insights of peer heterogeneity and free-riding effects.

5.5 Ideal Approach

In this section an ideal search for high-performance policies is proposed. Two performance endpoints are given in Subsections 5.5.1 and 5.5.2. The former an Ideal policy, whose playback continuity is not achievable and provides a universal bound. The latter a Weighted Greedy policy which is theoretically proved to have poor playback continuity. In Subsection 5.5.3 the performance of the Ideal policy is then compared with Rarest First, showing their playback capabilities when the peer's buffer size increases without bound. In the lights of the poor performance of stochastic policies, a deterministic Family of permutation-based policies is defined in Subsection 5.5.4. There, we show properties that evidence the richness of this family. Relying on these properties, we provide a Follower System in Subsection 5.5.5, whose main characteristic is trying to reflect the behavior of target bitmaps. We carried-out some tests to measure the capacity of the Follower System to provide outstanding policies. We will conclude the result is negative: the Follower System behaves like a *dirty mirror*. However, these tests suggest the high performance of a special Subfamily of chunk policies, named W-shaped because of the request order which takes into account for closest to deadline chunks first (the $/$ part of the W), then the least urgent chunks (the $\setminus$ part of the W) and finally a $\wedge$-zig-zag started in the middle of the buffer. For example, if $N = 7$ and the priority of request is given by the order 651342, then $I = 2$ represents the $/$ part (buffer positions 6 and 5) of the W-member, $J = 1$ represents the $\setminus$ part of the W-member and 342 is the $\wedge$-zig-zag, picking the remaining positions in the middle of the buffer.

5.5.1 Universal Bound

Although the model does not permit to pull more than one chunk in one time slot, relaxing this assumption we can find a universal bound for the playback continuity. As we will see, this bound is below the unit, hence strengthening the idea that real systems cannot obtain perfect continuity.

Definition 5.5.1 *In the Ideal policy, whenever peer A contacts peer B, it pulls all possible chunks at a go. Mathematically, the Ideal policy is defined by: $s_i = 1$, $\forall i \in [N-1]$.*

The Ideal policy is not achievable by normal peers, who have unit download capacity. Nevertheless, a *special system* whose peers have N times the capacity of normal peers provides us a mean to find universal bounds.

Definition 5.5.2 *A super-peer is a peer that has both infinite download and upload bandwidth capacities.*

Super-peers are entities able to perform the Ideal policy. Replace the normal peers from the cooperative model with super-peers, with unit strategic sequence. We have two useful results, that together state a universal bound for the playback continuity.

Proposition 1 *The Ideal policy outperforms all other scheduling policies.*

Proof. Introduce only super-peers in the cooperative model, and let them apply the Ideal policy, with $s_i = 1$. The bitmap recursion is then:

$$\begin{cases} p_1^* = \frac{1}{M}, \\ p_{i+1}^* = p_i^* + (1 - p_i^*)p_i^*, \ \forall i \in [N-1]. \end{cases} \tag{5.9}$$

The performance of the normal network is governed by the bitmap recursion given in (5.4). If peers do not follow the Ideal sequence, there exists some integer $j \in [N-1]$ such that $s_j < 1$. A direct induction then shows that $p_i < p_i^*$ for all $i > j$.

Q.E.D.

Proposition (1) is not surprising: just confirms the Ideal policy is the best. In order to find a universal bound for the playback continuity, we must solve the non-linear recursion (5.9) explicitly for any given positive integers $M > 1$ and N, and then evaluate to get p_N^*. Then by Proposition 1 we are sure that no other policy would achieve that bound.
Non-linear recursions appear in several combinatorial problems and dynamic systems, and finding explicit solutions is more an exception that a rule. In this opportunity we will include a trick, which identifies the recursion with probabilities of constructed simple events.

Theorem 5.5.3 *The universal bound for playback continuity is $p_N^* = 1 - (1 - \frac{1}{M})^{2^N}$*

Proof. We will solve explicitly the non-linear recursion (5.9), and then evaluate at $i = N$ to get the result. The main trick is to rewrite (5.9) in the following way:

$$\begin{cases} p_1^* = \frac{1}{M}, \\ p_{i+1}^* = p_i^* + p_i^* - (p_i^*)^2, \forall i \in [N-1]. \end{cases}$$

We can observe that $p_{i+1}^* = P(A_1 \cup A_2)$, being A_1 and A_2 independent events in a certain probability space, both with probability p_i^*. By induction, we have that $p_i^* = P(\cup_{j=1}^{2^i} A_j)$, being $\{A_j\}_{1 \leq j \leq 2^i}$ a set of independent events with probabilities $\frac{1}{M}$. The prove does not need to construct such a probability space and events: this way leads to the desired sequence. By the Inclusion-Exclusion principle we get that:

$$\begin{aligned} p_i^* &= P(\cup_{j=1}^{2^i} A_j) = \sum_{j=1}^{2^i} P(A_j) - \sum_{1 \leq j_1 < j_2 < 2^i} P(A_{j_1} \cap A_{j_2}) + \ldots + (-1)^{2^i} P(\cap_{j=1}^{2^i} A_j) \\ &= \binom{2^i}{1} p_1^1 - \binom{2^i}{2} p_1^2 + \ldots + (-1)^{2^i} \binom{2^i}{2^i} p_1^{2^i} \\ &= 1 - \sum_{j=0}^{2^i} \binom{2^i}{j} (-1)^{2^i - j} p_1^j = 1 - (1 - \frac{1}{M})^{2^i}. \end{aligned}$$

Replacing $i = N$ we get the result.

Q.E.D.

A trivial upper-bound for the start-up latency in normal peers is N, given that the latency $L = \sum_i^N p_i$ is a sum of N probabilities. However, the Ideal system with super-peers provides a more rigid upper-bound:

Corollary 5.5.4 *The universal bound for start-up latency is* $L = N - \sum_{i=1}^{N} (1 - \frac{1}{M})^{2^i}$

Proof. A direct induction shows that $p_i \leq p_i^*$ for any given feasible chunk policy and index $i \in [N]$. Summing all over i, we get the desired upper-bound.

Q.E.D.

Observe also that super-peers achieve the smallest buffering times as well, because a joining super-peer reaches the state of another just in one time slot (pulling all their chunks).

5.5.2 An Ill-Designed Stochastic Policy

To illustrate the design complexity and gain intuition with the problem, let us study the performance of a Weighted Greedy policy. With this stochastic weighted policy the peer looks for nearest-to-deadline chunks first, but can skip sometimes one chunk even if it is useful, trying to exploit both Greedy and Rarest First advantages. Consider a weight factor $q : 0 < q < 1$ and the discrete power-law probability mass $r = (r_1, \ldots, r_{N-1})$, being $r_i = \frac{q^i}{\sum_{j=1}^{N-1} q^j}$. The

Weighted Greedy policy works as follows: peer A tries to pull chunk $N-1$. If it is feasible, it tosses a coin with probability of success r_{N-1}. If it succeeded, the pull takes place and the request finishes. Otherwise, the same process is applied, tossing a coin with probability success r_{N-2}, and so on. The strategic sequence for the Weighted Greedy is:

$$s_i = (1 - \frac{1}{M})r_i \prod_{j=i+1}^{N-1} [1 - (1 - p_j)p_j r_j], \forall i \in [N-1] \tag{5.10}$$

The reasoning of Expression (5.10) is analogous for the ones of the classical policies, where we add the Bernoulli condition in each step during the request. The following proposition discards all intention to use the Weighted Greedy policy for practical purposes.

Proposition 5.5.5 *Weighted Greedy never achieves a playback continuity higher than* $3/4$.

Proof. By Expression (5.10) we have for all $i < N-1$ that $s_i = \frac{s_{i+1}}{q}[1-(1-p_{i+1})p_{i+1}r_{i+1}] < \frac{s_{i+1}}{q}$. A direct induction over the finite set $[N-2]$ shows that $s_i < \frac{s_{N-1}}{q^{N-i-1}}$, for all $i < N-1$. Observe that $s_{N-1} = (1 - \frac{1}{M})r_{N-1} < r_{N-1} = \frac{q^{N-1}}{\sum_{i=1}^{N-1} q^i} < q^{N-2}$.
Therefore $s_i < q^{i-1}$ for all $i \in [N-2]$. Replacing in the recursive bitmap yields:

$$p_{i+1} - p_i = (1 - p_i)p_i s_i < \frac{s_i}{4} < \frac{s_{N-1}}{4q^{N-i-1}}, \forall i \in [N-1].$$

We can then construct a summation to express the playback continuity:

$$\begin{aligned} p_N &= p_1 + \sum_{i=1}^{N-1}(p_{i+1} - p_i) < \frac{1}{M} + \frac{s_{N-1}}{4q^{N-1}} \sum_{i=1}^{N-1} q^i \\ &< \frac{1}{2} + \frac{1}{4}\frac{r_{N-1}}{q^{N-1}} \sum_{i=1}^{N-1} q^i = \frac{3}{4}. \end{aligned}$$

Q.E.D.

As a consequence, the playback-buffering trade-off cannot be attained with an elementary stochastic sampling, like a Weighted Greedy. Observe that the final bound $3/4$ is far from rigid in massive networks (we used that $M > 1$ to find the first term of $1/2$). The intuition suggests to pull one chunk whenever it is feasible; otherwise there is a non-negligible probability to have a failed-slot. This is the main reason why we will restrict our search world to deterministic policies.

5.5.3 Convergence to Perfect Playback

By technological reasons, it is natural to ask what happens in the case of unlimited storage, when the buffer size N tends to infinity. Although we know real policies cannot achieve perfect continuity, we have the optimistic result for Rarest First:

Proposition 5.5.6 *Following Rarest First, peers tend to have perfect continuity when the buffer tends to infinity. Moreover, the convergence order is linear.*

Proof. In Rarest First we know from Equation (5.7) that the bitmap complies:

$$p_{i+1} = p_i + (1 - p_i)^2 p_i, \, \forall i \in [N] \tag{5.11}$$

The extended sequence $\hat{p}_1 = \frac{1}{M}$ and $\hat{p}_{i+1} = \hat{p}_i + (1-\hat{p}_i)^2\hat{p}_i$ to the natural domain is increasing and bounded by 1, hence has a limit α. For any fixed buffer size N we define a sequence extending the bitmap, such that $p_i^N = p_i$ for all $i \leq N$ but $p_i^N = p_N$ whenever $i > N$. All the sequences $\{p^N\}_{N\geq 1}$ are increasing and bounded (each one by a different the universal bound). The sequence $\hat{p}$ is Cauchy. Therefore, by elementary topology the sequence of sequences $\{p^N\}_{N\geq 1}$ inherits the Cauchy property in the uniform topology for the space sequences of real numbers, and by completeness converges to the sequence $\hat{p}$. Formally, the sequence of continuities is captured by $\hat{p}$. Taking limits on both sides from (5.11) yields $\alpha = \alpha + (1-\alpha)^2\alpha$, so either $\alpha = 0$ or $\alpha = 1$. But $\hat{p}$ is increasing and $\hat{p} > 0$. As a consequence, $\alpha = 1$. Finally, the convergence order is linear, given that:

$$\lim_{n\to\infty} \frac{1 - \hat{p}_n}{1 - \hat{p}_{n-1}} =$$
$$\lim_{n\to\infty} \frac{1 - \hat{p}_{n-1} - \hat{p}_{n-1}(1 - \hat{p}_{n-1})^2}{1 - \hat{p}_{n-1}} = 1.$$

Q.E.D.

The previous topological argument is a technical engine to define a notion of convergence starting from finite sets [102]. We will omit this process and treat directly the sequence of playback continuity, taking the limit with N, when necessary.

Theorem 5.5.7 *Super-peers tend to have perfect continuity when the buffer tends to infinity. Moreover, the convergence order is quadratic.*

Proof. The first part is obvious (the Ideal Policy dominates Rarest First, and the latter converges to perfect continuity). Super-peers can apply the Ideal policy, characterized by $s_i = 1$. Finally, its convergence order can be found easily by using Expression 5.9 for the Ideal policy:

$$lim_{N\to\infty} \frac{1 - p_N}{(1 - p_{N-1})^2} =$$
$$lim_{N\to\infty} \frac{1 - p_{N-1}(2 - p_{N-1})}{(1 - p_{N-1})^2} = 1.$$

Hence, its convergence order is 2, and the result holds.

Q.E.D.

Proposition 5.5.7 can also be obtained with the explicit expression for the Ideal policy, found in the proof of Theorem 5.5.3. As a consequence, the scheduling policies will always work with convergence order p between linear and quadratic when the buffer increases (i.e. such that $1 < p < 2$). In fact, there are better policies than Rarest First, and the Ideal policy is the best (pull the whole buffer at a go).

5.5.4 A Family of Permutation-based policies

A natural way to obtain diversity is to use an arbitrary permutation to decide the order of a request. Let us consider the set of permutations Π_{N-1} of the first $N-1$ buffer positions:

$$\Pi_{N-1} = \{\pi : \{1,\ldots,N-1\} \mapsto \{1,\ldots,N-1\},\, \pi(i) \neq \pi(j)\, \forall i \neq j\}.$$

For each permutation $\pi \in \Pi_{N-1}$ we can associate the following chunk scheduling policy. In the first step, A examines its buffer at position $\pi(1)$. If that chunk is missing and B owns it, the download is performed. Otherwise (either because peer A owns that chunk or B does not), A shifts to position $\pi(2)$ and the process is repeated (until there is either success or failure, the two only possible final results). This scheme leads to $(N-1)!$ different chunk scheduling policies. We call them *permutation-based* policies. For the chunk policy associated with permutation $\pi \in \Pi_{N-1}$, the corresponding strategic sequence is related with the bitmap:

$$s_{\pi(i)} = (1 - \frac{1}{M}) \prod_{j=1}^{i-1} \Big(1 - p_{\pi(j)}(1 - p_{\pi(j)})\Big). \tag{5.12}$$

Equation (5.12) can be interpreted in this way: peer A will ask for position $\pi(i)$ only if it was not selected by the server and every buffer position $\mathcal{B}_i$ such that $j < i$ meets one of the following two conditions: peer A already owns the chunk at position $\pi(i)$, or neither A nor B owns it. It is interesting to notice that other policies could be considered, for example using random variables to decide which position of the buffer to ask for next. In this thesis we will restrict the attention to deterministic policies, inspired in the poor performance attained by the Weighted Greedy policy. We suspect that under this model, permutation-based policies capture all the deterministic possibilities. Moreover, some permutation-based policies outperform classical ones, as we will confirm in the final results of our proposal. Indeed, the family of permutation-based policies Π_{N-1} enjoys many useful properties, which guide us to define algorithms to find efficient ones. The first is that they are a super-set that includes previous classical policies (and their mixture). Observe that the identity permutation $\pi(i) = i$ and the reverse one ($\pi(i) = N - i$) define the Rarest First and Greedy policies, respectively. For any given index $m : 1 \leq m \leq N-1$, there is a Mixture between Greedy and Rarest first, captured with the following permutation π_m:

$$\begin{aligned} &\pi_m(i) = i, \quad i = 1,\ldots,m; \\ &\pi_m(i) = N - (i - m), \quad i = m+1,\ldots,N-1. \end{aligned}$$

Clearly, all mixtures of the classical policies are permutation-based policies. This trivial fact confirms that the quality of the best permutation policy will not decay. In fact, better policies can be found.

Lemma 5.5.8 *Degradation in the Selection*
The sequence $s_{\pi(i)}$ is strictly monotone decreasing.

Proof. By (5.12) we have that $s_{\pi(i+1)} = s_{\pi(i)}[1 - p_{\pi(i)}(1 - p_{\pi(i)})] < s_{\pi(i)}$.

Q.E.D.

Definition 5.5.9 *The Cayley distance $d(\pi_1, \pi_2)$ between permutations π_1 and π_2 is the minimum number of transpositions needed to obtain π_2 from π_1.*

Let $x = (x_1, \ldots, x_{N-1})$ and and $y = (y_1, \ldots, y_{N-1})$ be injective real-valued vectors. There are unique permutations π_x and π_y such that $x_{\pi_x(1)} > x_{\pi_x(2)} > \ldots > x_{\pi_x(N-1)}$ and $y_{\pi_y(1)} > y_{\pi_y(2)} > \ldots > y_{\pi_y(N-1)}$

Definition 5.5.10 *The Cayley pseudo-distance between the vectors x and y is $d(x, y) = d(\pi_x, \pi_y)$.*

Corollary 5.5.11 *Approximation Strategy Property*
For every injective real-valued sequence $(x_1, \ldots, x_{N-1})$, there is only one permutation π whose strategic sequence s verifies that $d(x, s) = 0$.

Proof. By Lemma 5.5.8 the evidence is the only permutation π that complies:

$$x_{\pi(1)} > x_{\pi(2)} > \cdots > x_{\pi(N-1)}$$

Q.E.D.

So far, we know how to approximate a given injective vector choosing an appropriate permutation policy. However, we want to have a full comprehension of the relation between our permutation π and the bitmap $p = (p_1, \ldots, p_N)$, which determines both the delivery ratio p_N and buffering times by $L = \sum_{i=1}^{N} p_i$. Additionally, we know a pessimistic result of universal playback bound, and have an insight of the convergence order to perfect playback continuity when the buffer size is increased. A simple link between the bitmap and strategic sequence is offered by the bitmap recursion (5.4).

Definition 5.5.12 *Given a desired bitmap p, the corresponding* ideal strategic sequence s^{id} *can easily be obtained with a restatement of* (5.4)*:*

$$s_i^{id} = \frac{p_{i+1} - p_i}{(1 - p_i)p_i}, \forall i \in \{1, \ldots, N-1\} \tag{5.13}$$

This reverse-viewpoint of the problem will be exploited to introduce a Follower System, trying to reflect the behavior of a desired bitmap.

5.5.5 The Follower System

Let us recall that we search for permutations which achieve high continuity and low latency. The Approximation Strategy Property leads us to invert the search order, following these stages:

1. Choose an ideal bitmap $p = (p_1, \ldots, p_{N-1}, p_N)$.
2. Find the corresponding ideal strategic sequence s^{id} with Equation (5.13).
3. Find the only permutation π such that $s^{id}_{\pi(1)} > s^{id}_{\pi(2)} > \ldots > s^{id}_{\pi(N-1)}$.

Thanks to the Approximation Strategy Property, we have that the strategic sequence s associated with permutation π verifies that $d(s, s^{id}) = 0$. Hence, we are able to "imitate" the ideal vector s^{id} with a feasible strategic sequence s. Nevertheless, this imitation does not assure that the respective bitmaps are similar. In order to solve the puzzle of searching for high quality permutations, it is necessary to *evaluate the continuity and latency of a given permutation*:

Definition 5.5.13 *For every permutation π, the Non-Linear System $NLS(\pi)$ consists of the recursive bitmap (5.4) and recursive expression for the strategic sequence given by (5.12), with unknowns $\{p_i\}_{i=2,\dots,N} \cup (\{s_i\}_{i=1\dots,N-1} - \{s_{\pi(1)}\})$:*

$$NLS(\pi): \begin{cases} p_1 & = \frac{1}{M} \\ p_{i+1} & = p_i + (1-p_i)p_i s_i, \quad \forall i = 1,\dots,N-1 \\ s_{\pi(1)} & = 1 - \frac{1}{M} \\ s_{\pi(i+1)} & = s_{\pi(i)}(p_{\pi(i)} + (1-p_{\pi(i)})^2) \quad \forall i = 1,\dots,N-2 \end{cases}$$

Solving the non-linear system $NLS(\pi)$ (for example with the Newton-Raphson method), it is possible to evaluate the performance for any particular permutation.

In a first attempt to find an optimal permutation, we consider the control system illustrated in Figure 5.4. This system shows the *reverse design* recently stated: the input is the desired bitmap p_i, and the output is its best approximation p_i^*. We call this serial blocks the *Follower System*.

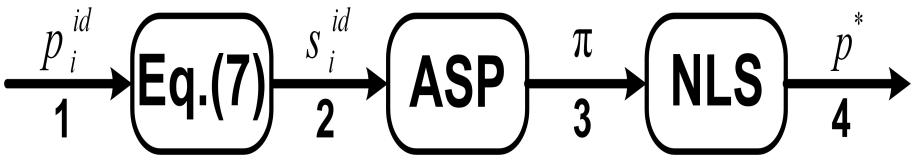


Figure 5.4: Follower System: receives a desired bitmap p and returns a feasible bitmap p^*, with the nearest result to the input. Observe that it permits to obtain the permutation policy π that achieves p^* (in Step 3).

5.5.6 A Subfamily of W-Shaped policies

Several inputs were injected into the Follower System to test its performance. An ambitious design of the input vector p is an exponential sequence with unit playback continuity, because its regularity and low values of latency:

$$p_i^\epsilon = M^{\frac{N-i}{1-N}}, \quad i = 1,\dots,N. \tag{5.14}$$

The output can be analytically found in this case. Note that $p_{i+1}^\epsilon = M^{\frac{1}{N-1}} p_i^\epsilon > p_i^\epsilon$, and the

corresponding ideal strategic sequence s^{id} respects the following identity:

$$\begin{aligned}\frac{s^{id}_{i+1}}{s^{id}_{i}} &= \frac{p^{\epsilon}_{i+2} - p^{\epsilon}_{i+1}}{(1-p^{\epsilon}_{i+1})p^{\epsilon}_{i+1}} \frac{(1-p^{\epsilon}_{i})p^{\epsilon}_{i}}{p^{\epsilon}_{i+1} - p^{\epsilon}_{i}} \\ &= \frac{1-p^{\epsilon}_{i}}{1-p^{\epsilon}_{i+1}} > 1, \quad \forall i \in \{1,\ldots,N-2\}\end{aligned}$$

This means that the strategic sequence to simulate is increasing, so the output permutation is $\pi(i) = N - i$. The output falls into the Greedy policy, something not desirable.
Then we input the Ramp Vector shown in Figure 5.5(a) for the case $M = 1000$ and $N = 40$. The corresponding ideal strategic sequence is not feasible, because its magnitude exceeds the unit (it is not a probability). However, the output shows a key element of this system. The two pairs of functions p, p^*, s and s^{id}, are contrasted in Figure 5.5(a) and Figure 5.5(b) respectively.

The first observation from Figures 5.5(a) and 5.5(b) is that this Follower System fails again when trying to follow the segmented bitmap, and shows that the first idea does not work as desired. However, the experience with the Follower System shows us how a direct peak in the permutation policy (see Figure 5.5(b)) generates an abrupt change in the bitmap (change in slopes of Figure 5.5(a)). The position of the peak plays a critical role, because if it is next to the playback the continuity will be poor (as in the case of Greedy). On the other hand, a high latency will be carried out whenever the peak is chosen far away from the playback (for instance, with the Rarest First policy). Absolute maximums are avoided in order not to get high latencies. These observations motivate us to introduce the following:

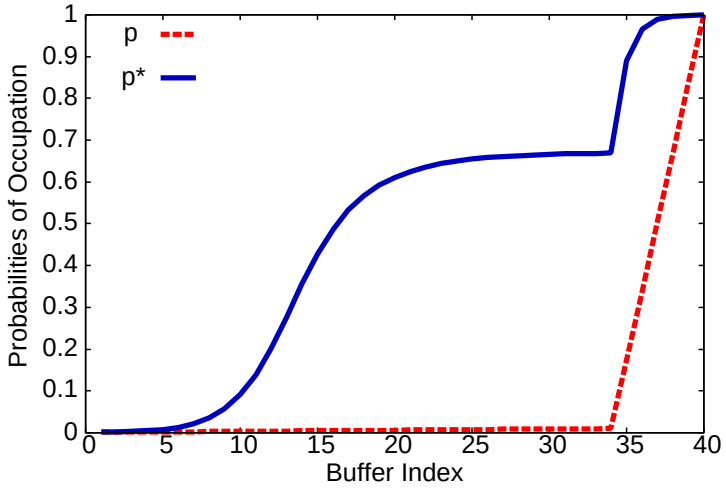
Definition 5.5.14 *For each pair of naturals $(I, J) : I + J \leq N - 1$, there is one permutation of the W-*Shaped Policies*, that can be expressed as follows:*

$$\begin{aligned}\pi(i) &= N - i, \quad i = 1,\ldots,I, \\ \pi(I+j) &= j, \quad j = 1,\ldots,J \\ \pi(I+J+k) &= \left\lfloor \frac{N+J-I}{2} \right\rfloor + \left\lceil \frac{k}{2} \right\rceil (-1)^{k+1}, \\ k &= 1,\ldots,N-I-J-1.\end{aligned}$$

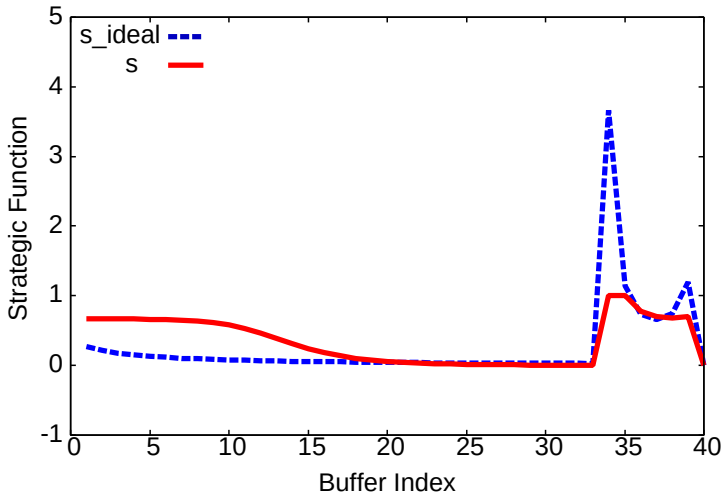
The reader can check that these permutation-based policies present a W-shaped buffer-priority. The I buffer-cells nearest-to-the-deadline have the highest priority, whereas the J far-away follow in priority. Then, a zig-zag priority is defined in the middle of the buffer (the $\wedge$-part of the W priority). Curiously, in [231] the authors analyzed the same model via Markov-Chains in a continuity-driven fashion, and suggest another sub-family of V-Shaped policies simultaneously and independently:

Definition 5.5.15 *Let $k \in [N-1]$ be the buffer-cell with the lowest priority (i.e. $\pi(N-1) = k$). A permutation member is V-Shaped if the priority increases as the position moves away from k.*

The authors prove that the V-Shaped policies contain the asymptotically optimal policy when the buffer size tends to infinity. The number of V-Shaped policies is exponential with the buffer



(a) Segmented bitmap and its output.



(b) Ideal and feasible strategic sequences s_{id} and s.

Figure 5.5: Ideal vs feasible bitmaps and strategic sequences for the case $M = 1000$, $N = 40$.

capacity N. Hence, an exhaustive search among the V-Shaped policies is computationally prohibitive for large buffer sizes. Note that the number of W-Shaped members is the cardinal $\{(I, J) : I + J \leq N - 1\}$, or the cardinal of natural solutions to the equality $I + J + K = N - 1$. By elementary combinatorics, this number is $\binom{N+1}{2} = \frac{N(N+1)}{2}$, polynomial in the buffer capacity N.

5.6 Feasible Approach

5.6.1 A Single-objective Combinatorial Problem

The cooperative system is fully characterized by a scheduling policy π, which must be chosen regarding playback and buffering times. Equation (5.4) assures that the bitmap is monotonically increasing. Then, there is a trade-off between the continuity p_N and the buffering time $L = \sum_{i=1}^{N} p_i$. The following analytical result will determine a natural single-objective measure for this cooperative system:

Proposition 2 *Let π be a permutation of the natural set $\{1, \ldots, N-1\}$, and X_π the random variable that represents the number of steps in a successful request. Then, its expected value $E(X_\pi)$ is:*

$$E(X_\pi) = \frac{M}{M-1} \sum_{i=1}^{N-1} \pi(i)\big(p_{i+1} - p_i\big).$$

Proof. Let α_i be the probability of having a successful request in step i. Then:

$$\begin{aligned} E(X_\pi) &= \sum_{i=1}^{N-1} i\alpha_i \prod_{j=1}^{i-1}(1-\alpha_j) = \sum_{i=1}^{N-1} ip_{\pi(i)}(1-p_{\pi(i)}) \prod_{j=1}^{i-1}[1-p_{\pi(j)}(1-p_{\pi(j)})] \\ &= \frac{1}{1-\frac{1}{M}} \sum_{i=1}^{N-1} ip_{\pi(i)}(1-p_{\pi(i)})s_{\pi(i)} = \frac{M}{M-1} \sum_{i=1}^{N-1} i(p_{\pi(i)+1} - p_{\pi(i)}) \\ &= \frac{M}{M-1} \sum_{i=1}^{N-1} \pi(i)(p_{i+1} - p_i). \end{aligned}$$

Q.E.D.

Then, $Q(\pi) = E(X_\pi)$ is a linear combination of the jumps $p_{i+1} - p_i$. Moreover, it is monotonically increasing with the continuity p_N, for its derivative with respect to p_N is $\pi(N-1) > 0$. Interestingly enough, this measure takes a natural trade-off when the Rarest First policy is considered. In Rarest First we have $\pi(i) = i$ for all $i \in [N-1]$, and the single score is:

$$E(X_\pi) = \frac{M}{M-1}\Big(\sum_{i=1}^{N-1} ip_{i+1} - \sum_{i=1}^{N-1} ip_i\Big) = \frac{M}{M-1}(Np_N - L) \tag{5.15}$$

As a consequence, the number of steps in a successful request is highly related with our original performance metrics. Let us recall that all requests take place in one time slot, so long requests do not affect the time response of the system. Then, it is convenient to maximize the quality $Q(\pi)$, which defines the score of a combinatorial optimization problem, where the chunk scheduling policy represents the decision variable.

Definition 5.6.1 *The Score for the Cooperative Network Game (CNG) with a permutation π is the expected number of steps in a successful request $Q(\pi) = E(X_\pi)$.*

In order to find the best chunk scheduling policy, the following Cooperative Network Game (CNG) must be solved:

$$\text{CNG}: \quad \max_{\pi \in \Pi_{N-1}} E(X_\pi) \tag{5.16}$$

$$s.t. \tag{5.17}$$

$$\begin{cases} p_1 = \frac{1}{M} \\ p_{i+1} = p_i + p_i(1-p_i)s_i, \quad \forall i \in \{1, \ldots, N-1\} \\ s_{\pi(1)} = 1 - \frac{1}{M} \\ s_{\pi(i+1)} = s_{\pi(i)} + p_{\pi(i)} - p_{\pi(i)+1}, \quad \forall i \in \{1, \ldots, N-2\} \end{cases} \tag{5.18}$$

5.6.2 Problem Translation

From now on, we will solve the CNG stated in (5.16), trying to find the best chunk scheduling policy π. In this section we will translate the CNG into a suitable Asymmetric Traveling Salesman Problem (ATSP). Recall that a tour in a complete graph is a closed simple walk, where all nodes are visited exactly once (except, naturally, the first node).

Definition 5.6.2 *$HAMILTONIAN - TOUR$: Does an arbitrary graph G contain a Hamiltonian tour?*

Definition 5.6.3 *$ATSP$: Find the cheapest tour of a fully-connected directed weighted graph $G = (V, V \times V)$, with positive costs $w_{i,j}$ in every edge (v_i, v_j), $v_i \neq v_j$.*

The Traveling Salesman Problem is the undirected version of the ATSP, and is NP-Complete, given that it is in NP and it can be seen as a restriction from the Hamiltonian-Path decision problem, which is an NP-Complete decision problem [80]. Clearly, taking $w_{i,j} = w_{j,i}$ for every $i \neq j$ we see the ATSP is at least as hard as the TSP, and it is an NP-Complete optimization problem as well. The ATSP can be solved heuristically following an Ant Colony Optimization (ACO) approach, which is inspired in the way ants find the shortest path between their nests and their food [12]. The reader can find a deep analysis of this nature-inspired metaheuristic in [24, 65–67]. A complete graph of N nodes allows to get a bijection between a hamiltonian tour and a given permutation. We translate the CNG into an instance of the Asymmetric Traveling Salesman Problem (ATSP [153]) using the following bijection.

Proposition 3 *There is a bijection between the space of permutations Π_{N-1} and the set of directed hamiltonian tours of an N-Clique.*

Proof. Let K_N be an $N-$clique with labeled nodes $\{1, \ldots, N\}$, and N an auxiliary node, from which all directed hamiltonian tours T start. Both sets are finite with cardinal $(N-1)!$. By counting, it suffices to find an injection $\varphi : T \mapsto P$. Let us define for every directed tour $t = \{N, v_1, v_2, \ldots, v_N - 1, N\}$ the permutation $\pi(i) = v_i$, $i = 1, \ldots, N-1$. Function $\varphi(t) = \pi$ is one-to-one, and the result holds.

Q.E.D.

Consider the composition law in the permutation space Π_{N-1}. Some trivial facts on the group of permutations $(\Pi_{N-1}, \cdot)$ will be very useful in order to define a neighborhood structure in the space Π_{N-1}, that will be introduced in a metaheuristic resolution. All permutations are generated by a product of a finite number of transpositions. There are several metrics adopted for permutation spaces. For an overview of distances on permutation groups we refer the reader to [8]. Recall that the Cayley distance $d(\pi_1, \pi_2)$ between permutations π_1 and π_2 is the minimum number of transpositions needed to obtain π_2 from π_1.

Definition 5.6.4 *A distance d on Π_{N-1} is* graphic *if $d(\pi_1, \pi_2)$ is the length of a shortest path joining π_1 and π_2 in the simple graph with vertex set Π_{N-1} and edges $(\pi_1, \pi_2) : d(\pi_1, \pi_2) = 1$.*

Lemma 5.6.5 *The Cayley distance is graphic.*

Proof. Take two arbitrary permutations $\pi_x \neq \pi_y$, and call $l = d(\pi_1, \pi_2) > 0$. By its definition, there exists transpositions $t_1, \ldots, t_l$ such that $\pi_y = t_l t_{l-1} \ldots t_1 \pi_x$. Define recursively the family of adjacent intermediate permutations $\pi_0 = \pi_x$ and $\pi_{i+1} = t_{i+1}\pi_i$, such that $\pi_y = \pi_l$. Then clearly $d(\pi_i, \pi_{i+1}) = 1$, and there exists a path of length l between the nodes π_x and π_y. If there were a shorter path of length $r < l$, the r-path from π_x to π_y would provide intermediate permutations that differ in only one transposition, obtaining that $r \geq d(\pi_1, \pi_2) = l$, a contradiction. Hence, $\pi_x = \pi_0, \pi_1, \ldots, \pi_l = \pi_y$ is the shortest path between π_x and π_y, and the Cayley distance is graphic.

Q.E.D.

The previous lemma can be strengthened. In fact, every integer-valued distance on any set X is graphic if and only if $d(a, b) > 1$ implies $d(a, c) + d(c, b) = d(a, b)$ for some c (see [59]).

Definition 5.6.6 *A neighborhood structure for the CNG with feasible set Π_{N-1} is a collection $\{\mathcal{N}_\pi\}_{\pi \in \Pi_{N-1}}$ of subsets of Π_{N-1} such that:*

1. *The resulting hypergraph is connected: given $\pi_x, \pi_y \in \Pi_{N-1}$, there exists a sequence of solutions $\pi_x = \pi_0, \pi_1, \ldots, \pi_l = \pi_y$ such that $\pi_i \in \mathcal{N}_{\pi_{i-1}}$ for all $i = 1, \ldots, l$.*

2. *For every $\pi \in \Pi_{N-1}$ we can decide in polytime whether there exists a better neighbor $\pi' \in \mathcal{N}_\pi$ such that $Q(\pi') > Q(\pi)$, whenever it exists.*

The previous algebraic objects are useful to define a neighborhood structure in our space of permutations Π_{N-1}. Consider for each permutation π the unit ball $\mathcal{N}_\pi = \{\pi' \in \Pi_{N-1} : d(\pi, \pi') = 1\}$.

Proposition 5.6.7 *The set of unit balls $\mathcal{B} = \{\mathcal{N}_\pi, \pi \in \Pi_{N-1}\}$ is a neighborhood structure for the CNG*

Proof. By Lemma 5.6.5 the Cayley distance is graphic, and consequently $\mathcal{B}$ generates a connected hypergraph. Suppose that we reach a solution π for the CNG $\max_{\Pi_{N-1}} Q(\pi)$. There are exactly $|\mathcal{N}_\pi| = C_2^{N-1}$ neighbors, a quadratic polynomial in N. In order to evaluate the quality of a neighbor the non-linear system $NLS(\pi)$ must be solved. We can decide if a neighboring solution is better than π in polytime, because the Newton-Raphson method is a polynomial approximation scheme, with quadratic order of convergence to the solution. Hence, $\mathcal{B}$ is a neighborhood structure for the maximization problem $\max_{\Pi_{N-1}} Q(\pi)$.

Q.E.D.

All these tools are used to define an ACO-based Algorithm, which finds high competitive policies for the CNG.

5.6.3 An Ant-Colony Resolution

We propose and Ant-based algorithm that returns high-quality chunk scheduling policies. The Main Algorithm can be studied in four blocks (see Algorithm 3). In the first block (Line 1), a weighted network is defined, translating in this way the original CNG into a suitable ATSP. A non-negative cost is assigned to each edge when Function $Edges$ is called, with a learning mechanism based on ant exploration. The second block prepares the Ant-Colony application calling Function $Pheromones$, which will allow to trace high quality tours. The Subfamily of W-Shaped policies from Definition 5.5.14 will be considered here as a seed for the pheromone trails. The third block (Line 3) is the $AntWorkers$ application itself, which returns a permutation π. Finally, a local improvement is introduced exploiting the neighborhood structure of the permutation group by means of Function $LocalSearch$. Note that two neighboring permutations are translated into two tours that visit all nodes exactly in the same order, but two of them [48].

Algorithm 3 $\pi = Main\ Algorithm(M, N, d, \tau, \alpha, \beta, \rho, ants, iterations)$

1: $d(E) = Edges(M, N, ants)$
2: $\tau(E) = Pheromones(M, N, ants, SubFamily)$
3: $\pi = AntWorkers(M, N, d, \tau, \alpha, \beta, \rho, ants)$
4: $\pi_{out} = LocalSearch(\pi, M, N, iterations)$
5: **return** π_{out}

Functions $Edges$, $Pheromones$, $AntWorkers$ and $LocalSearch$ will be presented in the following subsections.

5.6.3.1 Edges

The whole solution is designed following an ant-worker's philosophy. The basic idea is that several artificial ants start a tour in the auxiliary node N, and measure distances (or leave traces) according to the quality of the recently visited tour.

During $Edges$, the translation from the CNG to an ATSP takes place. It returns a non-negative matrix $d(E)$ whose entries $d(i, j)$ contain the edge-costs of a directed N-clique. A

unit-cost is assigned to all edges in Line 1, and the Greedy policy ($\pi(i) = N - i$) is considered as a reference score in Line 2. In Lines 3 to 6 several ant-tours are built in order to update costs for edges. Each ant stochastically chooses the next node to visit with Tabu-nodes (in order not to visit the same node twice). During Function $VisitCycle$, ants choose the next step according to the following probability distribution:

$$p(x_{j+1}) = \frac{d(x_j, x_{j+1})^{-1}}{\sum_{i \in NoCycle} d(x_j, x_i)^{-1}} \tag{5.19}$$

This means that shorter tours are desirable. In Line 5 the matrix d is updated. $UpdateCost$ finds the best policy so far. Then, all edges inside the tour π are updated according to its scores:

$$d(\pi(j), \pi(j+1)) = 10(N - j) \times \frac{Q_{max}}{Q(\pi)},$$

where Q_{max} is the best score obtained so far. The additional ladder-factors $N - j$ avoids re-visiting a cycle several times.

Algorithm 4 $d(E) = Edges(M, N, ants)$

1: $d(E) = 1$
2: $Quality = Greedy$
3: **for** $i = 1$ **to** $ants$ **do**
4: $\quad \pi = VisitCycle(d(E))$
5: $\quad d(E) = UpdateCost(\pi(1), \ldots, \pi(i))$
6: **end for**
7: **return** $d(E)$

5.6.3.2 Pheromones

The main ingredient of Function $Pheromones$ is that tours are deterministic, given by the Subfamily of W-Shaped policies from Definition 5.5.14, exploiting the *smell* of high quality that provides this SubFamily. In this way, the preliminary analysis is incorporated as a seed in this sophisticated algorithm. It returns a matrix τ with the trail of pheromones for each edge. The notion of pheromones helps ants in the main block called $AntWorkers$ to build tours with better quality. See Algorithm 5 for details.

5.6.3.3 AntWorkers

Function $AntWorkers$ has several similarities with Ant-System, originally proposed by Marco Dorigo and Luca Maria Gambardella in 1997 [66]. There, the authors distribute m ants into n cities of a TSP instance, and each ant builds a tour using a Tabu list, in order not to visit twice the same node. Ants leave trails of pheromones, and select stochastic tours weighting both shorter paths and pheromones. The diversification of the metaheuristic depends on a set of parameters:

Algorithm 5 $d(E) = Pheromones(M, N, ants, SubFamily)$

1: $d(E) = 1$
2: $Quality = Greedy$
3: **for each** $\pi \in SubFamily$ **do**
4: $\quad Q = Quality(\pi)$
5: $\quad \tau = UpdatePheromones(\pi(1), \ldots, \pi(i))$
6: **end for**
7: **return** τ

1. The visibility of the path $\beta \geq 0$.
2. The relative importance to the trail $\alpha \geq 0$.
3. The trail persistence $\rho : 0 \leq \rho \leq 1$ or *evaporation factor* $1 - \rho$, and
4. A constant Q related to the quantity of trail laid by ants.

Let $x_1, x_2, \ldots, x_j$ be the first j nodes visited by an ant. Each ant builds a biased tour according to the following jump-probability distribution:

$$p(x_j, x_{j+1}) = \frac{\tau(x_j, x_{j+1})^\alpha d(x_j, x_{j+1})^{-\beta}}{\sum_{i>j} \tau(x_j, x_i)^\alpha d(x_j, x_i)^{-\beta}}, \tag{5.20}$$

where $\tau(x_i, x_j)$ is the trail of pheromone for edge (x_i, x_j), and the constraint $i > j$ over the sum is the *Tabu List* of nodes, in order not to visit twice the same node. It is interesting to notice that under perfect visibility and discarding the trail (i.e. $\beta \to \infty$ and $\alpha = 0$), Ant-System is exactly the *Greedy* heuristic for the TSP. In this way, the parameters α and β impose a trade-off between greediness and the *smell* of ants, based on pheromones. The updating of pheromones in its classical implementation is strictly based on the evaporation factor $\rho : 0 \leq \rho \leq 1$ and the quality of each tour. More specifically, each visited edge receives, per ant, a trail proportional to the trail persistence and quality of the tour. Given that the TSP is a minimization problem:

$$\tau(x_i, x_j)^{t+n} = \rho\tau(x_i, x_j)^t + \sum_{k=1}^{m} \frac{Q}{L_k} 1_{(x_i,x_j)\in Cyclek}, \tag{5.21}$$

where L_K is the length of the tour visited by ant k and $\tau(x_i, x_j)^t$ denotes the pheromone trail for edge (x_i, x_j) at time t. Note that only the edges visited by ants contribute to the sum (the indicator 1_X is one only if X is true), and the trails of the others receive a *trail evaporation*, by the factor $1 - \rho$. We refer the reader to [66] for an overview of the Ant-System design, and its performance with respect to other classical metaheuristics for the TSP. The book [67] contains an in-depth analysis of the properties of this population-based metaheuristics. Now, we will show our particular implementation of Function $AntWorkers$, paying particular attention to the differences with the original Ant-System. Basically, our implementation introduces four differences, in order to address the nature of our problem and its translation:

1. *Attractor Nest*: the network has an auxiliary node N, that plays the role of an attractor nest. All ants start and finish the tour at this node. In fact, every tour has a corresponding permutation-based policy, in accordance with Proposition 3.

2. *Serial walk*: in the original Ant-System implementation, all ants walk simultaneously. In this design, ants explore the network serially, and each ant updates the trails only after finishing its tour.

3. *Weighted Tours*: in order to increase the diversification of the biased tours, we avoid ants to visit the same edges in their first step. This is an artificial guide to ants, so as to visit all edges of the network at least once. In this way, the trail of every edge is updated at least twice.

4. *Massive Population*: the authors of the original implementation suggest to use n ants (i.e. one ant per node). Here, we will consider at least three times the number of nodes. In fact, the network has $N(N-1)/2$ edges, but each ant visits exactly N edges. The probability of visiting all edges is increasing with the population of serial ants in this single-run system.

$AntWorkers$ is presented in Algorithm 6.

Algorithm 6 $\pi = AntWorkers(M, N, d, \tau, \alpha, \beta, \rho, ants)$

1: $Quality = Greedy(M, N)$
2: **for** $i = 1$ **to** $ants$ **do**
3: $\quad \pi_i = AntCycle(d, \tau, \alpha, \beta)$
4: $\quad \tau = NewPheromones(\rho, \tau, Q, Q_{max})$
5: **end for**
6: **return** $\pi = MostVisitedCycle(\pi_1, \ldots, \pi_{ants})$

A reference score is the Greedy policy (Line 1). The serial implementation covers Lines 2 to 5, where the nest-node is an attractor (all ants start and finish in this node). During Function $AntCycle$ (Line 3), each ant builds a stochastic tour according to the probability vector defined in Equation (5.20). A technical difference with respect to the original Ants-System is defined in Line 4, where the trail update takes place. Specifically, only the trails of those edges visited by ant i are modified in Function $NewPheromones$. The new trail for the edge (x_j, x_{j+1}) is:

$$\tau(x_j, x_{j+1}) = (1-\rho)\tau(x_j, x_{j+1}) + \rho \times \frac{10(N-j)Q(\pi)}{Q_{max}}, \tag{5.22}$$

where the factor $10(N-j)$ provides a similarity of magnitudes between pheromones and distances. Notice that the ladder-factors $10(N-j)$ were fixed during the network construction in Function $Edge$ of the Main Algorithm. In this way we do not give priority to pheromones or distances. Additionally, we are mainly interested in the shortest hamiltonian path (the order in which the $N-1$ nodes are visited).

5.6.3.4 LocalSearch

The output permutation of $AntWorkers$ is locally improved via a simple Local Search phase. It looks for the best permutation among their neighbors. If the maximum number of iterations is not reached, a local optimum policy is returned. $LocalSearch$ is very simple, and exploits the properties of the Cayley distance. See details in Algorithm 7.

Algorithm 7 $\pi_{out} = LocalSearch(\pi, M, N, iterations)$

1: $Quality = Greedy$
2: **while** $iterations$ not reached **and improves do**
3: $\quad \pi = BestNeighbor(\pi, M, N)$
4: **end while**
5: **return** π

5.6.3.5 Computational Effort

In order to understand the trade-off between the computational effort and the quality of the solutions that offers the Main Algorithm, let us count the quantity of operations, taking the number of score evaluations as the basic operation. The following result summarizes the block that imposes the biggest computational effort, and the convergence order with respect to N.

Proposition 4 *Let N be the buffer size and $T(N)$ the mean computing-time for the quality $E(X_\pi)$. If $ants$ and $iterations$ have order $O(N)$, then the mean total time for running the Main Algorithm is $\mathcal{T} = O(N^3 T(N))$.*

Proof. The Local Search phase imposes the largest computational effort. The number of neighbors of a given permutation is $\binom{N-1}{2} = \frac{(N-1)(N-2)}{2}$. In order to find the best neighbor, it is necessary to evaluate all those neighbors in each iteration. If the number of iterations is linear with N , then this block imposes the biggest computational effort, and is cubic in N. If the mean computing-time for a score evaluation is $T(N)$, the total running time is $\mathcal{T} = O(N^3 T(N))$.

Q.E.D.

5.6.4 Discussion of Chunk Scheduling Policies

It is well known the Local Rarest First policy introduced by BitTorrent is capable of providing high performance for file sharing, but is not suitable for networks with stringent timing requirements, as live video streaming. A very basic solution was proposed in BiTOS, splitting the buffer into two parts: an urgent, which captures a set of k chunks that are close to the deadline, and the remaining buffer set. Peers either receive the rarest chunk from the high-priority set with probability p, or the rarest from the remaining set (with probability $1 - p$). In this way, BiTOS trades real-time urgency for availability, which is known a strength when running Rarest First. The parameters p and k can be tuned as desired, even adjusted dynamically as a function of monitoring losses. A similar idea is proposed by Sanghavi, Hajek and Mausolié, called *Interleave*. It basically interleaves a push scheme with nearest-to-the deadline first

in even slots, with a pull scheme requesting the latest chunk not in the buffer, in odd slots. Observe the trade is now given deterministically, by interleaving two techniques in time slots. The playback-buffering trade-off is also captured by Yipeng Zhou, Dah Ming Chiu and John C.S. Lui, which provided the model we are studying here [233]. They identify the nearest-to-deadline with the Greedy policy, and predict poor playback performance of this policy under massive peer populations. On the other hand, Rarest First is proved again to fail with real time urgencies. They associated the solution with a *Mixture* of both policies, running the Rarest First policy at the beginning, until a fixed buffer-cell m, and complete the remaining buffer following the Greedy policy. The effort is then focused on the scalability of a sequence of Mixture policies adjusting m as a function of the network parameters (M, N), finding asymptotically optimal solutions [63]. Bridge Zhao, together with John Lui and Dah-Ming Chiu, provided a continuous version of the cooperative model [231], when the server broadcasts a channel with a required streaming rate f, possibly sent to multiple peers concurrently, according to the server upload bandwidth and peers downloading capacity. The authors notice the best solutions for finite sets becomes more greedy when the server turns more resourceful, with an exhaustive list of permutation-based policies for a limited buffer capacity N. Their approach exploits properties of Density Dependent Jump Markov Processes (DDJMP), and the objective has only playback continuity as target. Their results suggest a subfamily of V-shaped scheduling policies, where the name is justified to the priority of a request: the first element is a certain buffer-cell $x < N$, and the priority is decreased when we move away from x along both sides. Again, they show optimality when $M \to \infty$ for playback continuity.

However, we know that even Rarest First has optimal playback continuity when $M \to \infty$, so, the start-up latency must necessary be weighted in the design. We stick to the discrete model, and an ideal approach (Section 5.5) suggests a polynomial size subfamily of W-shaped policies, in order to search for a playback-delay trade-off. The name is justified because Greedy is runned at $I < N$ chunks, followed by Rarest First in J chunks, and finally applying a deterministic zig-zag in the middle of the buffer, along its sides.
The V-shaped policies and our family of W-shaped policies share certain similarities as the reader can appreciate. An evident similarity is the presence of a fixed element $x < N$ of the buffer-cells, from which the priority is decreased along both sides. We also give condition to the buffer taking some chunks on both sides before a deterministic zig-zag scheme (whereas in the V-shaped policies, the priority can be decreased taking more than one chunk at left and then others at right along the fixed element $x < N$). An important difference is that an exhaustive search among V-shaped members is computationally prohibitive (the size is exponential with the buffer), whereas our subfamily of W-shaped policies is polynomial in the buffer capacity.

Curiously, it is worth to mention that both works introducing the subfamilies were first presented practically in simultaneous conferences in 2009 [16, 231], and both teams were not aware the advances of the other[1]. Nevertheless, we will show the performance of classical policies can be outperformed theoretically with our Ant-Colony inspired resolution. However, a self-critic is valid here: the structure of the ACO-based output permutation cannot be deter-

[1]The writer recently exchanged mails with Dr. Yipeng Zhou, and our publishings were news for his team.

mined in advance (the permutation policy has no clear pattern), and the computational effort of the Main Algorithm increases as the cubic of N to return an answer. This means that in order to tune a high-performing policy dynamically with the peer population M (or buffer size N), a high amount of computational effort should be performed off-line (and the system should be capable to absorb the variable behavior of the permutations when N or M are shifted). The subfamily of W-shaped policies is quadratic with N, hence an exhaustive search among them can trade quality for CPU time.
In order to figure-out these concerns and study the performance of chunk policies so far encountered, we will contrast policies theoretically (in the lights of the cooperative model), and also empirically, introducing different chunk policies in the real GoalBit platform, in the following section.

5.7 Results in a Real Platform

5.7.1 Comparison with Historical Policies

In P2P networks, two classical chunk scheduling policies are *Rarest First*: $\pi(i) = i$, and *Greedy*: $\pi(i) = N - i$. The former works properly in downloading, but not for streaming purposes. The same authors [233] of the original model propose in [231] a slight modification of the model, in which the server can offer video chunks to a randomly chosen fraction f of the M peers in the network. There, they find an asymptotic approximation to the optimal chunk scheduling policy when the buffer size increases. They measure quality only regarding playback continuity, and conjecture that the optimal policy is inside a subfamily of V-shaped policies, and becomes more greedy when f increases. Let k be the buffer-cell with the lowest priority. Then a permutation member is V-Shaped if the priority is increased as the position moves away from k. The exhaustive search among the V-shaped subfamily of policies is still computationally prohibitive (it has exponential size with N).

We tuned the parameters of the Main Algorithm inspired in [68], and adapted to our particular problem. Our final implementation used the Main Algorithm with $\alpha = 0.4, \beta = 1.5, \rho = 0.5$ and 100 ants, for the common-network parameters $N = 30$ and $M = 100$. Figure 5.6 presents the bitmaps for different chunk scheduling policies. Table 5.1 shows that the obtained permutation achieves an excellent continuity and at the same time a latency comparable to the one reached by Greedy, outperforming classical policies as well as an average of randomly chosen V-shaped policies.

We include in Table 5.1 the mixture with the highest continuity. The Average V-Shaped refers to the average performance over one-hundred randomly chosen V-Shaped policies. Over those one-hundred samples, our permutation-policy has better playback continuity than 88 samples, and achieves lower latencies than 86 samples. Additionally, we could not find even one V-Shaped sample with both better continuity and latency than our permutation-based policy. This results confirm a highly competitive trade-off between playback continuity and buffering-times of our proposal. The V-Shaped members define both high playback policies but high buffering times as well. Recall that the authors of [231] focus the design on playback continuity only.

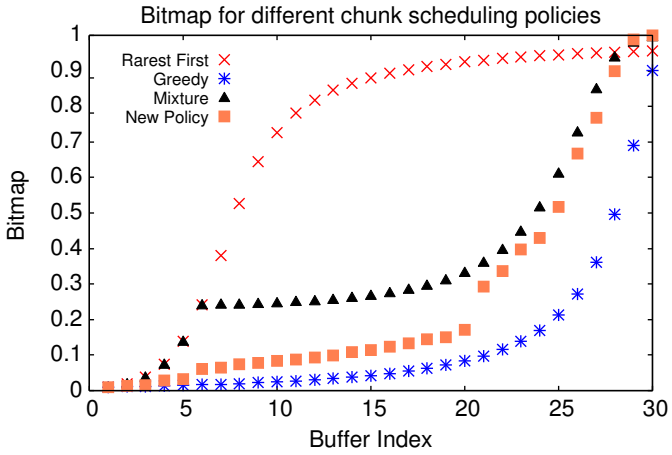


Figure 5.6: Bitmaps for different chunk scheduling policies.

Table 5.1: Performance of different chunk scheduling policies.

Policy	Continuity	Latency
Rarest First	0.9571	21.0011
Greedy	0.9020	4.1094
Mixture	0.9970	14.4798
Average V-Shaped	0.9670	17.6683
Main Algorithm	**0.9998**	**7.9821**

5.7.2 Results in a Real-Life Scenario

It is well known the BitTorrent's success for content downloading. However, it does not comply with the requirements of video streaming applications. GoalBit maintains the BitTorrent's philosophy mixing the tit-for-tat strategy with optimistic unchoking, extending the success in the peer selection process, that is a key element in the design of protocols for cooperation [44]. The clear weakness of BitTorrent for streaming applications is its chunk scheduling policy: Rarest First. The analysis of this section shows its unacceptable latencies. We carried-out real-life experiments based on a GoalBit emulator, to figure-out the main characteristics of different chunk-scheduling policies. First, we took real traces from a previous GoalBit distribution of a football match. Therefore, we completely reproduce the real distribution (even with the identical protocol specification), but with a different chunk scheduling policy. The emulator reproduces all but network failures. Recall from Section 3.8.1 that GoalBit keeps three buffer categories: urgent, normal and future. If some urgent chunk is missing, the local peer requests the nearest-to-deadline chunk first. Otherwise, a missing chunk is picked sampling an exponentially distributed random variable, where the probability is monotonically decreasing from usual along to the future buffer range. The main idea of the test is to keep the structure of the GoalBit buffer, and compare the classical policies with a chosen member

of our design, easy to include in the GoalBit protocol. Observe the W-shaped policies have a structure in three-phases, hence is suitable to introduce in the GoalBit protocol with minor changes. By a polynomial search among the set of W-shaped policies, we could determine the one with highest score when $N = 40$, which is defined by $(I, J) = (16, 1)$. Therefore, three different deterministic chunk scheduling policies were considered: Rarest First, Greedy and the W-Shaped member defined by $(I, J) = (16, 1)$. This test case considers a buffer capacity of $N = 40$ and 45 peers joining the network. In GoalBit, the video player halts when a video chunk is missing, and the player will skip frames. Therefore, users will have a re-buffering whenever a chunk is lost. Figures 5.7, 5.8 and 5.9 show respectively the first buffering time (or start-up latency), number of re-bufferings and their corresponding duration (in seconds), for each user.

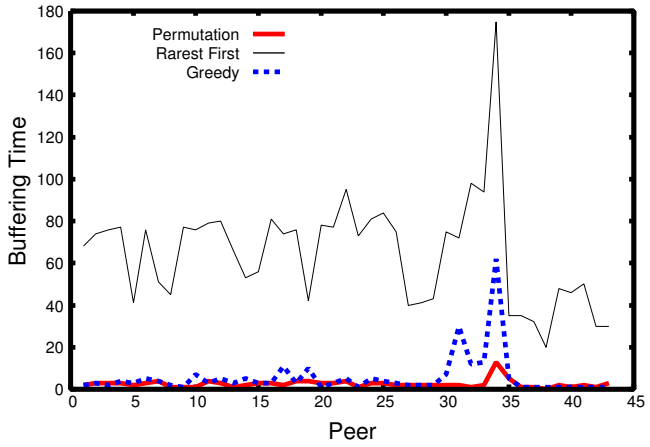


Figure 5.7: Buffering-time for 45 peers using different chunk scheduling policies.

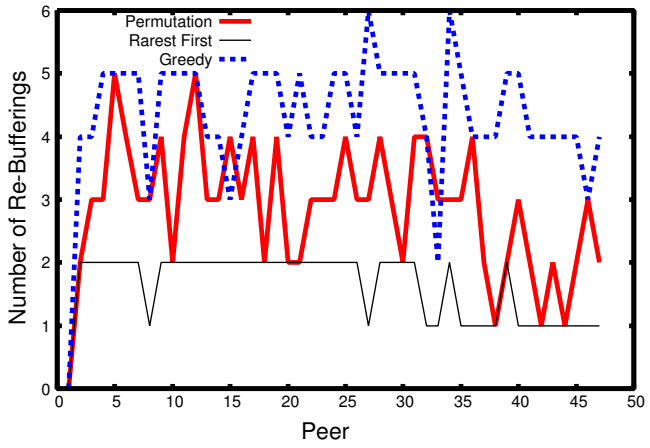


Figure 5.8: Number of re-bufferings (measure of playback continuity) for 45 peers using different chunk scheduling policies.

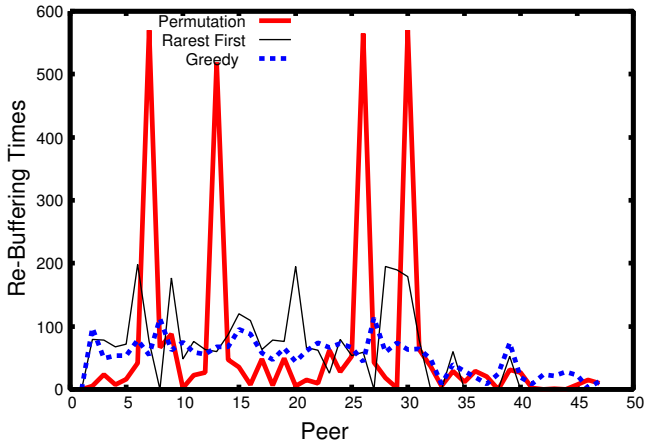


Figure 5.9: Average re-bufferings (in seconds) for 45 peers using different chunk scheduling policies.

Figure 5.7 shows clearly that the Rarest First policy has unacceptable start-up latencies for streaming purposes. In fact, users should wait more than one minute in average to start playing the video content following the Rarest First policy. The new policy is competitive in relation with the Greedy policy, having more reduced start-up latencies than Greedy for most of the peers. These latencies last no more than five seconds, which is a reasonable waiting time for users. Figure 5.8 illustrates the interruption of the video signal. The Greedy policy clearly presents interruptions more often. Most of the peers experience between four and six video interruptions when the Greedy Policy is introduced. Rarest first trades-off video cuts with buffering times. However, this policy is discarded for live streaming purposes regarding start-up latencies in the order of minutes. When GoalBit follows the new permutation-based policy, peers experience an intermediate number of video cuts, practically always lower than the Greedy policy (more specifically, only Peers 15 and 35 had just one cut higher than Greedy following our Permutation policy). Finally, Figure 5.9 shows that four peers experienced longer cuts when our permutation policy is introduced. However, the performance of the Permutation policy is higher in the rest of the peers, with respect to both classical policies.

5.8 Extended Model

5.8.1 Introduction

Suppose a static network that has M peers of Class X, M' peers of Class Y and a server that has the original video content (where $X, Y \in \{0, 1, 2, 3\}$). The server cuts the video into small chunks, and shares them in turns. In each time slot, the server chooses one peer at random from Class X with probability α, or one peer at random from Class Y with probability $1 - \alpha$, and sends one chunk to that peer. As in the simple model, peers can cooperate. More precisely, one peer from Class X either chooses with probability β another peer at random from its own class or a peer from Class Y at random with probability $1 - \beta$. Symmetrically, peers from

Class Y can request other peers from their own class (chosen at random) with probability β', or from Class X (with probability $1-\beta'$). Every peer tries to download the highest number of chunks during each time slot, and that number will depend on the uploading bandwidth of the contacted class. For example, if a peer requests a double-peer (with double bandwidth), it will be able to download two chunks during the same time slot. The request process is identical to that of the simple model, but it may continue after one chunk is obtained.

Definition 5.8.1 *A* free-rider *is a peer that has infinite downloading bandwidth, but no uploading bandwidth. When a peer requests a free rider, it will get no chunk on that time slot.*

In other words, it is a selfish peer, that asks for chunks but does not share them.

Definition 5.8.2 *A* normal peer *has infinite downloading bandwidth and unit uploading bandwidth. When a peer requests a normal peer, the time slot works as in the simple model.*

Definition 5.8.3 *A* double-bandwidth peer *has infinite downloading bandwidth and double uploading bandwidth. When a peer requests a double-bandwidth peer, it can get zero, one or two chunks.*

For example, if one peer follows the Rarest First policy and requests a double-bandwidth peer, then the request works as in the simple model. However, if a download occurs, the peer goes on asking for the next chunks, until downloading another one or reaching position $N-1$ of its buffer. An analogous request occurs when the chunk scheduling policy is identified with an arbitrary permutation.

Definition 5.8.4 *A* super-peer *has both infinite downloading and uploading bandwidth. When a peer requests a super-peer, it will take all chunks in only one time slot.*

5.8.2 Definition of the Extended Model

The optimization problem is specified as follows. The two classes X and Y, the number of peers M and M' and the buffer size N are given. We want to plan the network by choosing the parameters α, β and β' as well as the permutation π, in order to maximize the average playback continuity in the network. More specifically, the Extended Model (from now on the EM) is captured by the following optimization problem:

$$\max f(\pi, M, N, \alpha, \beta) = \frac{Mp_N + M'p'_N}{M + M'} \tag{5.23}$$

s.t.

$$\begin{cases} p_1 = \frac{\alpha}{M} \\ p'_1 = \frac{1-\alpha}{M'} \\ p_{i+1} = p_i + (1-p_i)[\beta p_i s_i^{(X,X,\pi)} + (1-\beta)p'_i s_i^{(X,Y,\pi)}] \\ p'_{i+1} = p'_i + (1-p'_i)[\beta' p'_i s_i^{(Y,Y,\pi)} + (1-\beta')p_i s_i^{(Y,X,\pi)}] \\ \alpha, \beta, \beta' \in [0,1] \end{cases}$$

where $s_i^{(X,Y,\pi)}$ is the strategic sequence for a peer from class X using permutation π requesting a peer from class Y. This expression will be found for each possible pair of classes (X, Y). The objective is to maximize the average quality of experience of all peers in the network (identifying quality with playback continuity). If we recall that the server sends with probability α one peer from Class X at random, then obviously $p_1 = \alpha/M$ and $p'_1 = (1-\alpha)/M'$ hold. The following equations are correct under steady state, and take into account the fact that the requested peer can be from their own class or the foreign class. We shall fix the parameters β and β' according to random peer selection (i.e. $\beta = M/(M+M')$ and $\beta' = M'/(M+M')$). In fact, we will show that under a full knowledge assumption, the network can work in optimal conditions and the combinatorial problem is reduced to the simple model, which has been extensively analyzed in previous works [16, 17, 181, 187]. The intuition here is that if the server as well as the peers can discover which peers have the highest bandwidth, then the server will send chunks to them, and all peers will direct requests to this powerful peers (which play the role of intermediate nodes of a tree-like structure).

There are exactly $4^2 - \binom{4}{2} = 10$ different interaction of pairs of the four classes (we are considering only once the cases of interaction between classes X and Y, when $X \neq Y$). Moreover, the cases of self-interaction can be reduced to the simple model. More precisely, the self-interaction between free-riders is strictly inadmissible, and does not deserve our attention. The interaction between normal peers behaves exactly as in the simple model, and between double-bandwidth peers translates proportionally to the case of the simple model (in fact, cut the time slot into two half). There is something to say for the case of self-interaction between super-peers. As a consequence, we will focus on 7 scenarios: the six different pairs of classes, and the simple model with infinite bandwidth.

5.8.3 Extended Model under Full Knowledge

From now on, we study the EM (Extended Model) when different classes interact (i.e. $X \neq Y$).

Definition 5.8.5 *The network in the EM has* full knowledge *when the server can recognize the different classes of peers in the network, and peers can deduce the best class-request (if it is better to ask one peer from its own class or the foreign class).*

Definition 5.8.6 *A peer-class has* higher level *than other when it has higher uploading bandwidth.*

Definition 5.8.7 *The server is* fair *when each peer in the network has the same probability of getting a chunk from it.*

Definition 5.8.8 *The network is* balanced *when the peer selection policy is at random.*

Theorem 5.8.9 *The EM is computationally more complex than the Simple Model.*

Proof. We will prove that the EM is trivially reduced to the simple model under full knowledge and fairness. Without loss of generality, suppose X has higher class than Y. Then clearly a peer

has more chances to download chunk at position i requesting peers from class X rather than from class Y, and $s_i^{(X,X,\pi)} \geq s_i^{(X,Y,\pi)}$. Given that peers can recognize the highest class, they will always choose peers from class X to ask for chunks, so $\beta = 1$ and $\beta' = 0$. By symmetry, observe that $s_i^{(Y,X,\pi)} = s_i^{(X,X,\pi)}$. Denote this number with s_i^{π} for brevity. Replacing in the EM we have that:

$$\begin{cases} p_1 & = \frac{\alpha}{M} \\ p'_1 & = \frac{1-\alpha}{M'} \\ p_{i+1} & = p_i + (1-p_i)[p_i s_i^{\pi}] \\ p'_{i+1} & = p'_i + (1-p'_i)[p_i s_i^{\pi}] \\ \alpha & \in [0,1] \end{cases}$$

Assuming fairness, the server will send chunks with probability $\alpha = M/(M+M')$. As a consequence, $p_1 = p'_1 = 1/(M+M')$. Hence, both recursive expressions are the same, and the sequences p_i and p'_i coincide. Moreover, the problem was reduced to:

$$\begin{cases} p_1 & = \frac{\alpha}{M} \\ p_{i+1} & = p_i + (1-p_i) p_i s_i^{\pi} \end{cases} \tag{5.24}$$

being π a permutation, which is exactly the simple model with $M + M'$ peers.

Q.E.D.

So far, we know that the peers with higher class perform better under the simple model, and super-peers achieve the best performance, with unit strategic sequence ($s_i = 1$).

5.8.4 Dealing with Free Riders

As we said before, the self-interaction of free-riders is not admissible (it is evident that without cooperation the network does not work). The reader can check that if all peers are free-riders then $p_i = p_1 = \beta/M < 1/M, \forall i$, and this performance is not acceptable since the network normally works with hundreds or thousands of peers. Similar results are obtained for the second class: $p'_i = (1-\beta)/M'$ is constant.

The interaction between free-riders and other classes has a special treatment. Particularly, suppose that $X = 0$ (free-rider class) and $Y \neq 0$. Under full knowledge, the server will always choose to send chunks to peers from class Y, so $\alpha = 0$. Moreover, free-riders will choose to complete requests considering peers from Class Y, which will prefer to do self-requests, so $\beta = 0$ and $\beta' = 1$. Replacing in the EM:

$$\begin{cases} p_1 & = 0 \\ p'_1 & = \frac{1}{M'} \\ p_{i+1} & = p_i + (1-p_i) p'_i s_i^{(X,Y,\pi)} \\ p'_{i+1} & = p'_i + (1-p'_i) p'_i s_i^{(Y,Y,\pi)} \end{cases}$$

As a consequence, the quality of all non-free-riders in the network is equivalent to that of the simple model. Note that $p_1 = 0$ but $p_2 > 0$. For example, if the Rarest First policy is applied, then the sequence $\{p_i\}_{1 \leq i \leq N}$ converges to 1 as N tends to infinity, and behaves exactly the same as $\{p'_i\}_{1 \leq i \leq N}$ but with a shift. In this way, the free-riders *follow* the performance of the other class, and the network scales.

The previous discussion shows that under full knowledge, the planning of the network is reduced to choose a chunk scheduling policy, or a permutation π, as in the case of the simple model (which has been extensively analyzed already). However, if the server cannot identify classes, it will tune $\alpha \neq 0$, and the performance of the network dramatically decreases, because chunks given to free-riders will be missing for all but only one peer. Hence, the network scales if and only if $\alpha = 0$. This results outstand the importance of the recognition of free-riders, under this new extension of the simple model. The full-knowledge hypothesis is strictly necessary in this case. This is an evidence of the empirical complexity of designing a scalable streaming network: normally the broadcaster does *not* have full knowledge, and peers neither.

5.8.5 The Presence of Super-Peers

Naturally, when one of the classes working in the network are super-peers, the cooperation is easier. Under full knowledge of the network (i.e. the server as well as peers can recognize classes of different peers), the server will always send chunks to super-peers, and the other class will be pleased to complete full requests to them, making the network scalable. The quality of experience of every peer in the network follows, under these circumstances, the one of super-peers (as if there were no other class) in the simple model. As a consequence, all peers will have (discarding the small initial shift) the following bitmap:

$$\begin{cases} p_1 & = 1/M \\ p_{i+1} & = p_i(2 - p_i), \forall i \in [N-1], \end{cases} \tag{5.25}$$

being M the number of super-peers in the network and p_N the playback continuity for each peer. When free-riders or super-peers are present inside the network, the analysis of the EM is trivial (because the strategic sequence is reduced to 0 or 1 respectively). In the following section we analyze the most complex interaction.

5.8.6 Interaction Between Normal and Double-Peers

This case is clearly the most complex to analyze. Intuitively, the server should send chunks to the double-bandwidth peers, and the request always directed to them. Under full knowledge this will happen, and normal-peers will tend to follow the quality of double-bandwidth peers. Let us focus on a more realistic scenario. Choose X as normal peers and Y double-bandwidth peers. Now, we will find an expression for the strategic sequences $s_i^{(X,Y,\pi)}$ and $s_i^{(Y,Y,\pi)}$ (the other two cases are self-requests, and expressed as in the simple model). For brevity, s_i denotes the probability that normal-peers have to take *the first chunk* from a double-bandwidth peer. If

k is such that $\pi_{k+1} = i$ then:

$$s_i = (1 - \alpha/M) \prod_{j=1}^{k} [1 - (1 - p_{\pi_j}) p'_{\pi_j}] \tag{5.26}$$

Expression (5.26) deserves an explanation. One peer from class X will download the first chunk at position i from class Y following permutation π whenever it fails in all previous positions (and success at position i) and is not chosen by the server (with probability $p_1 = 1 - \alpha/M$). Hence, a fail at all positions $\pi_j, j = 1, \ldots, k-1$ such that $\pi_k = i$ must occur. Moreover, a fail at position π_j occurs when it is not the case that the requesting peer does not have that chunk (with probability $1 - p_{\pi_j}$) and the requested peer does (event with probability p'_{π_j}). Then, a fail at position π_j has probability $1 - (1 - p_{\pi_j}) p'_{\pi_j}$.
Now, we are ready to express the sequence $s_i^{(X,Y,\pi)}$:

$$s_i^{(X,Y,\pi)} = s_i + s_i \sum_{j=1}^{k-1} \frac{(1 - p_{\pi_j}) p'_{\pi_j}}{1 - (1 - p_{\pi_j}) p'_{\pi_j}} \tag{5.27}$$

When asking a double-bandwidth peer, we can download chunk at position i in the first chance (the first term) or we downloaded a previous position $\pi_j, j = 1, \ldots, k-1$ with success. The factor

$$(1 - p_{\pi_j}) p'_{\pi_j} / [1 - (1 - p_{\pi_j}) p'_{\pi_j}],$$

represents a *replace* of a success instead of a fail at position π_j in the expression s_i.
In a similar way, the strategic sequence $s_i^{(Y,Y,\pi)}$ is:

$$s_i^{(Y,Y,\pi)} = s_i^* + s_i^* \sum_{j=1}^{k-1} \frac{(1 - p'_{\pi_j}) p'_{\pi_j}}{1 - (1 - p'_{\pi_j}) p'_{\pi_j}}, \tag{5.28}$$

where

$$s_i^* = (1 - (1 - \alpha)/M) \prod_{j=1}^{k-1:\pi_k = i} [1 - (1 - p'_{\pi_j}) p'_{\pi_j}]. \tag{5.29}$$

The EM can be obtained for this interaction by substitution.

5.8.7 Empirical Results

We will concentrate on a *worst case* scenario, by taking the Rarest First policy (i.e. $\pi_i = i$), and analyzing the scalability of the network under different mass of double-bandwidth peers, with no knowledge of the network, which implies that the peer selection is balanced: $\beta = M/(M + M')$ and $\beta' = M'/(M+M')$. Consider the common-network values $M+M' = 1000$ and $N = 40$. Table 5.2 presents the objective function $f(\alpha, M) = (Mp_N + M'p'_N)/(M + M')$ when the mass of double-bandwidth peers is variable accordingly with $M' \in \{350, 250, 150, 100, 0\}$ double-bandwidth peers and correspondingly $M = 1000 - M'$ normal peers. Table 5.3 contains the function $p_N - p'_N$ taking the same set for M and probability α.

Table 5.2: Expected continuity $f(\alpha, M)$ for a balanced network with different number of double-bandwidth peers.

α	350	250	150	100	0
0.0	1.0000	0.9998	0.9979	0.9941	0.9666
0.1	1.0000	0.9998	0.9979	0.9940	0.9665
0.2	1.0000	0.9998	0.9978	0.9939	0.9663
0.3	1.0000	0.9998	0.9978	0.9938	0.9661
0.4	1.0000	0.9998	0.9977	0.9937	0.9658
0.5	1.0000	0.9998	0.9977	0.9936	0.9655
0.6	1.0000	0.9998	0.9976	0.9934	0.9651
0.7	1.0000	0.9997	0.9975	0.9932	0.9646
0.8	1.0000	0.9997	0.9974	0.9929	0.9638
0.9	1.0000	0.9997	0.9972	0.9925	0.9625
1.0	1.0000	0.9997	0.9970	0.9918	impossible

Table 5.3: Difference in continuity $p_N - p'_N$ between double-bandwidth peers and normal peers, with different number of normal peers.

α	350	250	150	100	0
0.0	0.0002	0.0013	0.0067	0.0117	0.0006
0.1	0.0001	0.0010	0.0055	0.0094	0.0005
0.2	0.0001	0.0008	0.0041	0.0071	0.0004
0.3	0.0001	0.0005	0.0028	0.0048	0.0002
0.4	0.0000	0.0003	0.0014	0.0024	0.0001
0.5	0	0	0	0	0
0.6	-0.0000	-0.0003	-0.0015	-0.0025	-0.0001
0.7	-0.0001	-0.0006	-0.0030	-0.0051	-0.0003
0.8	-0.0001	-0.0009	-0.0047	-0.0077	-0.0004
0.9	-0.0002	-0.0013	-0.0065	-0.0106	-0.0006
1.0	-0.0002	-0.0017	-0.0086	-0.0140	impossible

It can be appreciated from Table 5.2 that the network always scales, although the server cannot recognize peers and tunes incorrectly the parameter α. Certainly, the performance is the best when $\alpha = 0$ (that is, to choose always double-bandwidth peers to send chunks). It can be noticed that the average continuity is higher than 96% in all instances, so the video quality is high. It is interesting to analyze if the video quality of normal peers is similar to double-peers or not. Table 5.3 contains the difference of continuity $p_N - p'_N$. It is obvious that when the parameter α is increased, the quality of normal peers is increased as well. Moreover, in the case $\alpha = 0.5$ both classes of peers experience the same video quality, and there is a symmetry in the instances $\alpha = i/10$ and $\alpha = (10 - i)/10$. It is evident that peers can follow double-bandwidth peers, and peers have better continuity than super-peers when $\alpha > 0.5$. This empirical analysis shows that the network scales when peers and double-peers interact, even under pessimistic scenarios. A further experiment with the balanced case of $\alpha = 0.5$ shows the scalability property of this network when the storage size increases. Figure 5.10 reveals the average continuity of normal (and double-bandwidth) peers as a function of the buffer size N,

considering again different amounts of double-bandwidth peers. It can be appreciated that the average continuity is higher than 90% when the storage capacity is higher than 25, even when the number of double-bandwidth peers is small.

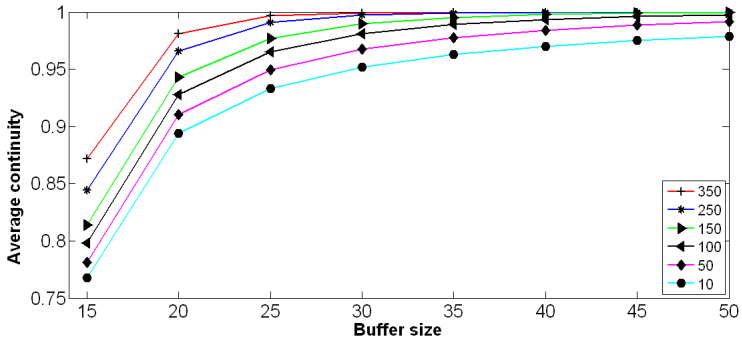


Figure 5.10: Evolution of the average continuity of peers as a function of the buffer storage capacity N.

5.9 Conclusions

In this chapter an in-depth analysis of the mathematical model [233] is detailed. The system addressed is pull-mesh based, as many successful commercial peer-to-peer live video streaming services. The model is simple yet robust, and captures the design of chunk policies, showing explicitly the buffering-playback trade-off, which represent the most shocking parameters to measure quality of experience [182]. The most valuable results are performance bounds, the best chunk scheduling policies so far in the lights of the model, nice performance results in the GoalBit platform and an overview of major causes of concern in live peer-to-peer systems not included in the original model, as free-riding and heterogeneity.

First it is theoretically proved a pessimistic result, that states the impossibility to play a perfect video, without cuts. However, future breakthroughs related with storage-technology will provide us near-perfect playback continuity. On the other hand, higher buffer capacities necessary increment the start-up latency. A brief visit to stochastic-based chunk policies (a Weighted Greedy policy) suggests a pull should occur whenever possible during a request, otherwise peers could examine the whole buffer having fails in several time slots. Therefore, our whole world is the set of deterministic Permutation-based policies, that exploit the structural viewpoint of the buffer model. A preliminary study outstands a subset of W-shaped chunk scheduling policies, which are then used as a trail in a more sophisticated heuristic resolution. The problem suffers several translations, by means of a key element which is the expected extension of a request. Exploiting the similarity between permutations and tours in a weighted graph, an alternative ATSP was faced to find effective chunk scheduling policies. The results are encouraging, confirming that the ATSP formulation was suitable, and the combinatorial problem useful to decode the playback-delivery trade-off.
An Extended Model is also introduced, in order to have an insight of the effects of free-riding

and heterogeneity. This model assumes a swarming policy based on uploading bandwidth, classifying peers as free-riders, normal, double-bandwidth and super-peers with infinite bandwidth (which achieve the playback universal bound). A primitive analysis demonstrates the strength of the full knowledge hypothesis in the network. In fact, the scalability of the network is guaranteed whenever the server as well as peers can recognize different classes. When free-riders interact with other classes, peers will always experience a non-negligible number of cuts in the video content, unless the server sends chunks to non-free-riders. On the other hand, when super-peers take part of it, the network always scales.

The performance of the Rarest First policy was contrasted with the one of super-peers. Particularly, the convergence to perfect playback continuity is faster for super-peers. Indeed, there is no feasible policy that can achieve quadratic convergence to the perfect continuity, even with high buffer size. Finally, the most complex scenario considered the interaction between normal and double-bandwidth peers, and was analyzed via simulations. The results outstand the importance of full knowledge, and further analysis is needed to have a comprehension of the possibility to achieve a full-knowledge, and the compromise with controlled overheads. A major challenge is to dynamically adjust the buffer capacity to adapt the population's needs, as seems to occur during transient time in some real networks like PPLive. The subset of W-shaped policies is suitable to include in a real-time dynamic chunk scheduling policy, given that they are polynomial in cardinality, and a fast search for optimality is feasible.

Part III

CONCLUSIONS AND FUTURE WORK

Chapter 6

Concluding Remarks and Trends for Future Research

The original concept of Internet was suitable for the client-server architecture, that cannot cope with massive flash crowds, or bandwidth sensitive systems as video streaming. Concretely, their resources do not scale with the size of the network. The introduction of peer-to-peer networks is a promising solution, where peers self-organize in a network overlay topology, being both clients and servers.

There are three video streaming modes suitable for Internet distribution. All of them share common concerns like node churn and variable asymmetric downlink/uplink resources. File sharing represents the most rustic mode, in which the video file is stored by a set of node servers, and should be completely downloaded before its playback. The usual performance metric is the makespan or time completion of peers (average or worst case time completion). New challenges are added when different streaming modes are considered (in some literature file sharing is not included as a streaming mode). On-demand video streaming copes with asymmetric nature of peers, who progressively download the video stream and watch it online. Finally in live video streaming the video is simultaneously generated, distributed and played by all users, with stringent real time requirements.

In this thesis, a mathematical analysis of the two most challenging streaming modes was provided, namely on-demand and live video streaming. For video on-demand, a deterministic fluid model is introduced, to understand the advantages of peer-assistance in relation with a raw Content Delivery Network (CDN). The attention is focused on the stability and capacity of both systems, under different levels of complexity. In the concurrent model (where users even can watch multiple video contents at the same time), the peer-assistance is proved to outperform the CDN always, which is an intuitive result (it is better to exploit idle users capacity). However, we could not fully characterize the stability of concurrent systems. In the sequential level (where users do not watch more than one content simultaneously), the system turns to be always globally stable, and again CDNs are outperformed by the peer-to-peer philosophy. By means of Little's law, a combinatorial optimization problem is here provided, which tries

to minimize the peer excursion times, deciding which video-items should be stored in cache nodes (super-peers in the GoalBit system). The problem is NP-Complete, and is heuristically solved with a GRASP methodology. Interestingly enough, the results suggest even a 90% of savings in server bandwidth, showing the peer-assistance in VoD services are a highly promising alternative, and should be integrally included as a solution to cope massive populations, specially in the distribution of popular files.

In live streaming, the chore resilience mechanism is scheduling, i.e. neighboring and chunk scheduling policies. Following the BitTorrent's philosophy, live streaming networks tend to use a random overlay topology. As a consequence, a key element in the design of live video streaming is its chunk scheduling policy. A pull-mesh cooperative system is deeply analyzed here. The mathematical model is robust, and gives a tractable characterization of the network in steady state under different chunk scheduling models. Additionally, it includes playback continuity and start-up latency as performance metrics, which represent the most shocking parameters to measure quality of experience for video distribution. An ideal approach helps to find universal bounds for both metrics, and a Follower System, which suggests a subfamily of W-shaped policies. They were used as a trail in a more sophisticated ad-hoc Ant-Colony resolution, which returns high-performing policies, when compared with previous classical approaches.

The production of this thesis has been disseminated in several proceedings as well as journals. The point of departure of the research in caching policies for P2P-VoD systems is [183], where my colleagues Pablo Rodríguez-Bocca and Claudia Rostagnol propose a combinatorial optimization problem to minimize the expected excursion times for end-users. Some inconsistencies were detected in the model, specially in the treatment of bandwidth bottlenecks (upload or download) and network stability, which was used there with no proof. The model suffered improvements, and the peer-assistance is proved to outperform the traditional CDN architecture under quite general scenarios in a short paper [177]. However, the stability of the fluid model was, up to that moment, an open problem. The peer-to-peer philosophy is proved to always outperform CDN systems, and the first stability results were included in a full paper [178]. The Sequential Fluid Model is finally proved to be globally stable, and the combinatorial optimization problem (the Caching Problem) is inside the class of NP-Complete problems, also proved for the first time in the journal [188].

A preliminary understanding of the cooperative chunk scheduling model for live streaming, including the Follower System, was first presented in [16]. The analytical expression for the expected extension of a request and combinatorial optimization problem, including an Ant-Colony-based resolution was introduced in [17]. The main ingredients and problem translation to an ATSP is summarized in [187]. Two compilation works, which include a historical revision of scheduling policies and results in the real GoalBit platform are disseminated in journals [184, 185]. Specifically, a carefully detailed design of the ant-colony exploration is presented in [184], whereas the compilation work [185] is more suitable for an operational research audience. An extension of the cooperative model is introduced for the first time in [186], where the goal is to study the impact of heterogeneity and free-riding effects. The system turns

to be highly scalable under full knowledge, even when free-riders take part of the network. The overlay is naturally organized in a tree-based structure, where resourceful peers are parents of normal peer (or free-riders). It is worth to point-out that the network performance is dramatically deteriorated when the server cannot recognize the different entities in the system (normal peers, double peers, free-riders and super-peers). The GoalBit architecture is first presented to the research community in [18], which represents the benchmark that supports the experimental results of this thesis.

In the whole research process, accuracy was compromised to gain simplicity. This is a natural fact inherent to the art of mathematical modeling, where the results give an overview of the system's behavior, and suggest hints of the system design. As main conclusions, this thesis promotes the inclusion of peer-assistance in VoD CDN architectures like YouTube, who spends millions of dollars per month in bandwidth from ISPs. Additionally, the Rarest First policy from BitTorrent is not suitable for streaming, and we just give hints of new chunk scheduling policies. However, the playback-delivery trade-off is not well understood yet. Indeed, here we addressed a static and structural analysis, but a dynamic buffer adaptation was not provided. We believe a closed-look at successful proprietary networks as PPLive would give additional hints, which could be mixed with the mathematical results here provided. Additionally, we highlight valuable benefits of contribution and bandwidth awareness under heterogeneous networks under presence of potential free-riders. Indeed, the cooperative system is highly scalable when the server is able to discriminate different entities in the network. However, if free-riders are frequently *gifted peers* (i.e. the server chooses them to send video-chunks several times), the performance drops dramatically.

As trends for future work, we would like to perform a rigorous analysis of dynamic chunk scheduling policies with buffer adaptation, tuned in accordance with peer populations and their needs. Additionally, the theoretical proof of global stability in the General (and Concurrent) Fluid Model would reinforce the peer-assistance in VoD systems, and open new way towards the deployment of multiple video sessions in a single home-PC, which seems to be common practice for near future, guiding by users behavior.

Chapter 7

Open Problems

Several problems remain unsolved, and are issued from this thesis. We will enumerate in Sections 7.1 and 7.2 open mathematical problems related with Chapters 4 and 5 respectively. A brief discussion of open technological concerns is included in Section 7.3

7.1 Simple Fluid Model for On-Demand Video Streaming

1. Fully characterize the conditions for global stability, for the General Fluid Model.
2. Fully characterize the conditions for global stability, for the Concurrent Fluid Model.
3. Provide an analysis of variance in a neighborhood of the expected system-evolution.
4. Find a polynomial-time algorithm to compute feasibility for an arbitrary instance of the Caching Problem, or prove this is impossible. This challenging problem would solve the millennium dilemma problem for classical complexity theory.

7.2 Cooperative Model for Live Video Streaming

1. Does the cooperative model running with a deterministic Permutation-based policy eventually converges to a regime state, for all permutations? We believe this is true, but a formal proof is missing, and it remains as a conjecture.
2. Find an exact closed expression for the start-up latency.
3. Find all the coefficients of the polynomial p_N of degree 3^{N-1} such that $p_{i+1} = p_i + p_i(1-p_i)^2$, as a function of $p_1 = \frac{1}{M}$.
4. Find all the coefficients of the polynomial p_N such that $p_{i+1} = p_i + p_i(1-p_i)(1-p_N+p_{i+1}-p_1)$, as a function of $p_1 = \frac{1}{M}$.
5. Find the ordinary generating function of the sequence $c_n = (1-\frac{1}{M})^{2^n}$, being M a positive integer (this could be useful to provide an alternative expression for the universal upper-bound for start-up latency).

6. Find the pattern of permutation-based policy with best playback-continuity (start-up latency) in terms of the positive integers M and N.

7. Solve the non-linear system $NLS(\pi)$ exactly, given an arbitrary permutation π.

8. Solve the Combinatorial Problem $\max_{\pi \in \Pi_{N-1}} E(X_\pi)$, being X_π the random variable that represents the number of steps during a request.

7.3 Discussion of Technological Concerns

A deep understanding of successful proprietary live streaming networks would possibly facilitate the peer-to-peer computing field. There are works that strongly suggest PPLive uses a dynamic buffer policy, with a variable buffer size starting from $N = 10$ and reaching $N = 110$ chunks. In this thesis only static solutions are provided for fixed buffer sizes. However, the number of W-shaped scheduling policies is polynomial with the buffer size. A time-consuming but feasible solution is to find the best W-shaped policy off-line, for every N over a finite set, and dynamically (on-line) adjust the policies when moving N. The structure of the W-shaped policy is simple, and has similarities with previous structures. There is a small subset dedicated to chunk urgencies, controlled by a positive integer I. Immediately, the requests proceeds looking for the J chunks farthest from the playback deadline, trading urgency for chunk availability. Until this moment, the policy is quite similar to the Mixture, only reversing the order (the Mixture policy starts the request running Rarest First and then Greedy; all W-shaped policies run Greedy first and then Rarest First). Nevertheless, a great difference with respect to other solutions is the "zig-zag" priority used in the rest of the buffers. The implementation of these policies should first find a function $\phi : (M, N) \to (I, J)$ which would determine a complete-and-dynamic chunk scheduling policy.
On the other hand, the adaptation of a Dynamic Multiple Video Cache (MVC) sounds an ambitious task. Until the best of our knowledge, so far the MVC techniques are all based on heuristics (usually considering the least used video item, or last used through a historic database, etc.). Here we gave a first step, with a formal state of a combinatorial optimization problem (the Caching Problem from Chapter 5). However, our solution is static, in the lights of a stationary state. A dynamic approach for the caching problem in video-streaming design starting from the transient state is an unknown task yet.

Chapter 8

Appendix

8.1 Proof of Global Stability of the P2P-SFM

Here we will conclude the proof of the statement:

Theorem 8.1.1 *The P2P-SFM is globally stable.*

As commented before, we skip the case in which $\gamma_j = 0$ for some $j \in [K]$, given that in this case the number of seeds increases without bound, and is clearly unrealistic. Hence, we will prove the result when $\gamma_j > 0$ for all $j \in [K]$. By convergence in the product topology, it suffices to prove that the following linear switched system is globally stable:

$$\begin{cases} \dfrac{dx}{dt} = \lambda - \theta x(t) - \min\{cx(t), \eta\mu x(t) + \mu y(t) + \rho z\} & \text{(8.1a)} \\ \dfrac{dy}{dt} = \min\{cx(t), \eta\mu x(t) + \mu y(t) + \rho z\} - \gamma y(t), & \text{(8.1b)} \end{cases}$$

We recall that if $c < \mu\eta$ the system is linear and globally stable. Hence, without loss of generality we mas consider $c \geq \eta\mu$. Consider The following Areas of the (x, y)-plane:

$$I = \{(x, y) \in \mathbb{R}^2 : mx < y + y_0\},$$

$$II = \{(x, y) \in \mathbb{R}^2 : mx > y + y_0\},$$

being $m = \frac{c-\eta\mu}{\mu}$ and $y_0 = \frac{\rho z}{\mu}$. When (x, y) is in Area I the system is governed by System I:

$$(I) \begin{cases} \dfrac{dx(t)}{dt} = \lambda - (\theta + c)x(t) & \text{(8.2a)} \\ \dfrac{dy(t)}{dt} = cx(t) - \gamma y(t). & \text{(8.2b)} \end{cases}$$

Otherwise, it is governed by System II:

$$(II) \begin{cases} \dfrac{dx}{dt} = \lambda - (\theta + \eta\mu)x(t) - \mu y(t) - \rho z & \text{(8.3a)} \\ \dfrac{dy}{dt} = \eta\mu x(t) + (\mu - \gamma)y(t) + \rho z. & \text{(8.3b)} \end{cases}$$

We already proved that System I is globally stable, whereas System II is globally stable whenever $\gamma > \mu$. Additionally, we proved that the P2P-SFM is locally stable either in Area I or Area II, but when the equality holds for the rest point $(\overline{x}, \overline{y})$ (i.e. $m\overline{x} = \overline{y} + y_0$), even local stability needs further analysis, and is linked with the global stability of the system.
In order to prove global stability of the P2P-SFM, the main idea is to prove that the orbit $(x(t), y(t))$ governed by Equations (8.1a) and (8.1b) rests either in Area I or Area II indefinitely after a certain finite time t^*. As a consequence, the local stability of both areas conclude the global stability of the linear switched system. Let us consider ψ such that $tg(\psi) = m$, and the following linear transformation:

$$\begin{cases} x' = xcos(\psi) + (y + y_0)sin(\psi) & \text{(8.4a)} \\ y' = -xsin(\psi) + (y + y_0)cos(\psi) & \text{(8.4b)} \end{cases}$$

The linear transformation maps the line $mx = y + y_0$ onto the x' axis, and Area I (II) onto the semi-plane $y' > 0$ ($y' < 0$). In this way, the switchings can be studied easily. Let us find the derivative of y', when $y' = 0$:

$$\begin{aligned} \dot{y'} &= -(\lambda - (\theta + c)x)sin(\psi) + (cx - \gamma y)cos(\psi) \\ &= (-m\lambda + (\theta + c)mx + cx - \gamma y)\frac{sin(\psi)}{m} \\ &= (-m\lambda + (\theta + c)mx + cx - \gamma mx + \gamma y_0)\frac{sin(\psi)}{m} \\ &= ((\frac{c}{m} + \theta + c - \gamma)x - \lambda + \frac{\gamma y_0}{m})sin(\psi) \end{aligned}$$

When the system switches from Area I to Area II we must have $\dot{y'} > 0$, and as a consequence $x > \frac{m\lambda - \gamma y_0}{c + m(\theta + c - \gamma)}$. When the system switches from Area II to Area I, the opposite equality holds. Therefore, the switches occur only in special parts of the plane.
The following lemmas are just re-written, and in some parts adapted, from the journal of Dongyu Qiu and Wei Qian Sang [168]. All the merits are for them.

Lemma 8.1.2 *If the rest point $(\overline{x}, \overline{y})$ is in Area I, the P2P-SFM is globally stable.*

Proof. If $(x(0), y(0))$ is in Area I and the evolution rests in that area, the global stability of System I guarantees convergence to the rest point. Otherwise, if $(x(0), y(0))$ is in Area II (or in Area I but switched to Area II in a certain instant), we can divide the analysis whether $\gamma > \mu$ or not. If $\gamma > \mu$ System II is stable, but the rest point is in Area I, so necessarily the orbit must return to Area I. Otherwise ($\gamma < \mu$), System II is unstable, and $y(t)$ increases exponentially whereas $x(t)$ vanishes to 0, so the return to Area I is guaranteed. Let us call t_0 the instant where the system is switched from Area II to Area I. We know that $y'(t_0) = 0$ and $\dot{y'}(t_0) > 0$. If $\gamma \neq \theta + c$, we have that $y'(t) = \overline{y'} + k_1 e^{-(\theta+c)t} + k_2 e^{-\gamma t}$, which together with the initial conditions assures that it has no more than one extremal point, so we cannot find another instant $t_1 > t_0$ such that $y'(t_1) < 0$, and the orbit stays in Area I forever. The situation is analogous when $\gamma \neq \theta + c$ and $y'(t) = \overline{y'} + k_1 e^{-\gamma t} + k_2 t e^{-\gamma t}$.

Q.E.D.

Lemma 8.1.3 *If $(\overline{x}, \overline{y})$ is in the line $mx = y + y_0$, the P2P-SFM is globally stable.*

Proof. In this case $\gamma > \mu$, and both linear systems are globally stable. The previous reasoning leads again to observe that the orbit rests indefinitely in Area I, unless it never enters Area II (in which case it will rest in Area II).

Q.E.D.

The most tricky case is where the rest point is in Area II. If the eigenvalues of System II are real negative, the same previous reasoning holds to prove global stability. Otherwise, we must play with a Lyapunov function and trigonometry. Without loss of generality, we will assume that $(x(0), y(0))$ is in Area II (otherwise, given that Area I is stable, we would have a certain instant where the system is switched to Area II). We will use even indices t_{2i} to denote the instants where the system is switched from Area II to Area I, and odd indices t_{2i+1} for the return instants. From now on, we will assume the rest point $(\overline{x}, \overline{y})$ is in Area II (i.e. $m\overline{x} > \overline{y} + y_0$).

Lemma 8.1.4

$$\left|y'(\dot{t_{2i+1}})\right| \leq \left|\dot{y'}(t_{2i})\right|, \forall i. \tag{8.5}$$

Proof. Without loss of generality we will prove that $\left|\dot{y'}(t_1)\right| \leq \left|\dot{y'}(t_2)\right|$. We will choose two positive reals λ_1 and λ_2 such that the function $V(t) = \lambda_1(y'(t) - \overline{y'})^2 + \lambda_2(\dot{y'})^2$ is a Lyapunov function. The result then follows because $V(t_1) \leq V(t_2)$, and as a consequence $\left|\dot{y'}(t_1)\right| \leq \left|\dot{y'}(t_2)\right|$.
Consider the vector $z(t) = [y'(t) - \overline{y'}, \dot{y'}(t)]^t$. Given that System I is stable, in Area I we can consider the system

$$\dot{z} = Az, \tag{8.6}$$

where A has non-negative real values, and for some real numbers a_1 and a_2

$$A = \begin{pmatrix} 0 & 1 \\ a_1 & a_2 \end{pmatrix}$$

We know that the characteristic polynomial $p(\lambda) = \lambda^2 - a_2\lambda - a_1$ must have eigenvalues with negative real parts, hence both a_1 and a_2 must be negative. Now we can find the derivative of $V(t)$ in terms of the coefficients a_1 and a_2:

$$\begin{aligned}
\dot{V} &= 2\lambda_1\dot{y'}(y' - \overline{y'}) + 2\lambda_2\dot{y'}\ddot{y'} \\
&= 2\lambda_1\dot{y'}(y' - \overline{y'}) + 2\lambda_2\dot{y'}(a_1(y' - \overline{y'}) + a_2\dot{y'}) \\
&= 2\dot{y'}(y' - \overline{y'})(\lambda_1 + a_1\lambda_2) + 2\lambda_2 a_2\left(\dot{y'}\right)^2
\end{aligned}$$

Choosing $\lambda_1 = -a_1\lambda_2 > 0$ we finally get that

$$\dot{V}(t) = 2\lambda_2 a_2(\dot{y'})^2(t) \leq 0.$$

Q.E.D.

Recall that the case left for study is when the eigenvalues of System II are conjugated. As a consequence, the expression for $y'(t)$ is known beforehand:

$$y'(t) = -\overline{y'} + Ae^{-\alpha(t-t_1)}cos(w(t-t_1)+\phi)$$

We know that its derivative at $y'(t_1) = 0$, so $cos(\phi) = \frac{y'}{A} > 0$, and $\phi \in [\frac{-\pi}{2}, \frac{\pi}{2}]$.

Lemma 8.1.5

- *If* $\left|\dot{y'}(t_{2i+1})\right| < \alpha\overline{y'}$*, the system will rest indefinitely in Area II.*
- $\left|\dot{y'}(t_{2i})\right| < e^{-\frac{\alpha\pi}{w}}\left|\dot{y'}(t_{2i-1})\right|$

Proof. Let $\hat{t}$ be the first relative maximum for $y'(t)$ after t_{2i+1}. Then $2\pi = w(\hat{t} - t_{2i+1} + \phi)$ and $y'(t_{2i+1}) = -\overline{y'} + Ae^{-\alpha(\hat{t}-t_{2i+1})}$. By its definition we know that $y(t_{2i+1}) = 0$, so $\overline{y'} = Acos(\phi)$. Additionally by derivation we have that $\dot{y'}(t_{2i+1}) = -Aasin(\phi+\beta)$. Let us assume that $\left|\dot{y'}(t_{2i+1})\right| < Aasin(\beta)$. Then $sin(\phi+\beta) < sin(\beta)$, so $0 < \phi+\beta < \beta$, or $-\beta < \phi < 0$. As a consequence we get that $A = \frac{\overline{y'}}{cos(\phi)} \leq \frac{\overline{y'}}{cos(\beta)}$, or $Acos(\beta) \leq \overline{y'}$. Replacing we get that:

$$y'(\hat{t}) \leq -\overline{y'}(1 - e^{-\alpha(\hat{t}-t_{2i+1})}) < 0, \tag{8.7}$$

and the evolution $(x'(t), y'(t))$ keeps under the axis x', or equivalently the system keeps in Area II forever. Observe that:

$$\begin{aligned}\left|y'(\dot{t_{2i+1}})\right| &= Aasin(\phi+\beta)\\ &= Aa[sin(\phi)cos(\beta) + cos(\phi)sin(\beta)\\ &\leq Aasin(\beta)cos(\phi) = Acos(\phi)asin(\beta)\\ &\leq \overline{y'}asin(\beta)\\ &= \epsilon \leq Asin(\beta),\end{aligned}$$

where we observe that $a^2sin^2(\beta) = (\alpha^2+w^2)\frac{tg^2(\beta)}{1+tg^2(\beta)} = \alpha^2$, and hence $\epsilon = \overline{y'}asin(\beta)$. Therefore, our assumption $\dot{y'}(t_{2i+1}) < \epsilon$ is weaker than our hypothesis $\left|\dot{y'}(t_{2i+1})\right| < \alpha\overline{y'}$, so the first statement holds.

Suppose now that t_{2i+2} is finite, so $y'(t_{2i+1}) = y'(t_{2i+2}) = 0$. Replacing we get that:

$$\overline{y'} = Acos(\phi) = Ae^{-\alpha(t_{2i+2}-t_{2i+1})}cos(w(t_{2i+2}-t_{2i+1})+\phi), \tag{8.8}$$

and necessarily $0 \leq \phi \leq \frac{\pi}{2}$ (otherwise, t_{2i+2} would not be finite). By trigonometric relations we get that $|sin(w(t_{2i+2}-t_{2i+1})+\phi+\beta)| \leq |sin(\pi+\phi+\beta)|$ and finally,

$$\begin{aligned}\left|\frac{\dot{y'}(t_{2i+2})}{\dot{y'}(t_{2i+1})}\right| &\leq \left|\frac{Aae^{-\alpha(t_{2i+2}-t_{2i+1})}sin(w(t_{2i+2}-t_{2i+1})+\phi+\beta)}{Aasin(\phi+\beta)}\right|\\ &\leq e^{-\alpha(t_{2i+2}-t_{2i+1})}\\ &\leq e^{-\alpha\frac{\pi}{w}}\\ &< 1.\end{aligned}$$

Q.E.D.

Corollary 8.1.6 *If the rest point is in Area II, the P2P-SFM is globally stable.*

Proof. Combining Lemmas 8.1.4 and 8.1.5, we get that:

$$\begin{aligned}\left|\dot{y'}(t_{2i+1})\right| &\leq \left|\dot{y'}(t_{2i})\right| \\ &< e^{-i\frac{\alpha\pi}{w}} \left|\dot{y'}(t_1)\right|\end{aligned}$$

As a consequence, if we choose $i > \frac{w}{2\pi} ln(\frac{\dot{y'}(t_1)}{\epsilon})$, we find that

$$\left|\dot{y'}(t_{2i+1})\right| < \epsilon,$$

so the orbit rests after t_{2i+1} in Area II, and the system is globally stable.

Q.E.D.

Combining Lemmas 8.1.2, 8.1.3 and 8.1.6, we obtain the desired conclusion:

Theorem 8.1.7 *The P2P-SFM is globally stable.*

8.2 GRASP

Combinatorial optimization problems arise in several real-world problems (economics, telecommunication, transport, politics, industry), were humans-beings have the opportunity to choose among several options. Usually, that number of options cannot be exhaustively analyzed, mainly because its number increases exponentially with an input size of the system. Much work has been done over the last six decades to develop optimal seeking methods that do not explicitly require an examination of each alternative, giving shape to the field of *Combinatorial Optimization* [157]. Usually, the problem is computationally intractable (i.e. NP-Hard), or sufficiently large to admit an exact algorithm, and a smart search technique should be considered exploiting the real structure of the problem, via heuristics. Optimality is not guaranteed, but compromised at the cost of computational efficiency.
Metaheuristics are an abstraction of search methodologies which are widely applicable to optimization problems. The most promising are Simulated Annealing [104], Tabu Search [83], Genetic Algorithms [84], Variable Neighborhood Search [91], GRASP [76], Ant Colony Optimization [64] and Particle Swarm Optimization [103], among others. The interested reader can find a complete list and details in the Handbook of Metaheuristics [82].

Greedy Randomized Adaptive Search Procedure (GRASP) is a multi-start or iterative process [118], where feasible solutions are produced in a first phase, and neighbor solutions are explored in a second phase. The best overall solution is returned as the result. The first implementation is due to Tomas Feo and Mauricio Resende, were the authors address a hard set covering problem arising for Steiner triple systems [76]. They introduce adaptation and randomness to the classical Greedy heuristic for the set covering problem (where $P_1, \ldots, P_n$ cover the set $J = \{1, \ldots, m\}$ and the objective is to find the minimum cardinality set $I \subset \{1, \ldots, n\}$ such that $\cup_{i \in I} P_i = J$).

GRASP is a powerful methodology to address hard combinatorial optimization problems, and has been successfully implemented in particular to several telecommunications problems, such as Internet Telephony [201], Cellular Systems [5, 6], Connectivity [29] and Wide Area Network design [179]. Here we will sketch the GRASP metaheuristic based on the work from Mauricio Resende and Celso Ribeiro, which is useful as template to solve a wide family of combinatorial problems [174]. Consider a ground set $E = \{1, \ldots, n\}$, a feasible set $F \subseteq 2^E$ for the optimization problem $\min_{A \subseteq E} f(A)$, and an objective function $f : 2^E \rightarrow \mathbb{R}$. The Pseudo-code 8 illustrates the main blocks of a GRASP procedure for minimization, where $Max\_Iterations$ iterations are performed, $\alpha \in [0, 1]$ is the quantity of randomness in the process and $\mathcal{N}$ is a neighborhood structure of solutions (basically, a rule that defines a neighbor of a certain solution). The cycle includes Lines $1 - 5$, and the best solution encountered during the cycle is finally returned in Line 6. Lines 2 and 3 represent respectively the Construction and Local Search phases, whereas the partially best solution is updated in Line 4.

Algorithm 8 $S = GRASP(MaxIterations, \mathcal{N})$

1: **for** $k = 1$ **to** $Max\_Iterations$ **do**
2: $S \leftarrow Greedy\_Randomized\_Construction(\alpha)$
3: $S \leftarrow Local\_Search(S, \mathcal{N})$
4: $Update\_Solution(S)$
5: **end for**
6: **return** S

A general approach for the Greedy Randomized Construction is specified in Pseudo-code 9. Solution S is empty at the beginning, in Line 1, and an auxiliary set C has the potential elements to be added to S. A carefully chosen element from C is picked up during each iteration of the While loop (Lines $3-9$), which is finished once a feasible solution is met. A Greedy construction would choose c^{min}, which is the element with the lowest cost to be added to the partial non-feasible solution (Line 4). On the other hand, c^{max} is the most expensive element to be added (Line 5). The *Restricted Candidate List* RCL is defined in Line 6, and has all the elements whose cost are below a certain threshold (see Line 6). In Line 7, a random element from the RCL is picked and added to the solution S. The process is repeated until a feasible solution S is found. It is worth to notice the effect of the input parameter $\alpha \in [0, 1]$. When $\alpha = 0$, the Greedy construction is retrieved. On the contrary, $\alpha = 1$ means a completely random construction. Therefore, the parameter α imposes a trade-off between diversification and greediness.

Algorithm 9 $S = Greedy\_Randomized\_Construction(\alpha)$

1: $S \leftarrow \emptyset$
2: $C \leftarrow E$
3: **while** $C \neq \emptyset$ **do**
4: $c^{min} \leftarrow \min_{c \in C} f(S \cup \{c\})$
5: $c^{max} \leftarrow \max_{c \in C} f(S \cup \{c\})$
6: $RCL \leftarrow \{c \in C : f(S \cup \{c\}) \leq f(S \cup \{c^{min}\}) + \alpha(f(S \cup \{c^{max}\}) - f(S \cup \{c^{min}\}))\}$
7: $S \leftarrow S \cup Random(RCL)$
8: $Update(C)$
9: **end while**
10: **return** S

The Greedy Randomized Construction does not provide guarantee of local optimality. For that reason, a Local Search phase is finally introduced, in order to return a locally optimal solution (which could be incidentally globally optimal). In order to define this phase, a rule to define neighbors of a certain solution is mandatory, called a *neighborhood structure*. A better neighbor solution is iteratively picked until no improvement is possible. A general local search phase is presented in pseudo-code 10.

Algorithm 10 $S = Local\_Search(S, \mathcal{N})$

1: $H(S) = \{X \in \mathcal{N}(S) : f(X) < f(S)\}$
2: **while** $H(S) \neq \emptyset$ **do**
3: $S \leftarrow ChooseIn(H)$
4: $H(S) = \{X \in \mathcal{N}(S) : f(X) < f(S)\}$
5: **end while**
6: **return** S

The success of the local search phase strongly depends on the quality of the starting solution, the computational cost for finding a better local solution, and naturally, on the richness of the neighborhood structure. The interested reader can find valuable literature and GRASP enhancements in [77, 174] and references therein [1].

[1] The reader can find a generous number of references in the web-site http://www2.research.att.com/ mgcr/grasp/

Bibliography

[1] Osama Abboud, Konstantin Pussep, Aleksandra Kovacevic, Katharina Mohr, Sebastian Kaune, and Ralf Steinmetz. Enabling Resilient P2P Video Streaming: Survey and Analysis. *Multimedia Systems*, 17(3):177–197, Jun 2011.

[2] Eytan Adar. Drawing crowds and bit welfare. *SIGecom Exch.*, 5(4):31–40, jul 2005.

[3] Eytan Adar and Bernardo A. Huberman. Free riding on gnutella. *First Monday*, 5(10), sep 2000.

[4] Akamai website. http://www.akamai.com/, 2007.

[5] E. Amaldi, A. Capone, and F. Malucelli. Planning UMTS base station location: Optimization models with power control and algorithms. *IEEE Transactions on Wireless Communications*, 2(5):939–952, 2003.

[6] E. Amaldi, A. Capone, F. Malucelli, and F. Signori. Optimization models and algorithms for downlink UMTS radio planning. In *Wireless Communications and Networking, (WCNC'03)*, volume 2, pages 827–831, March 2003.

[7] P. Antoniadis and C. Courcoubetis. Market models for p2p content distribution. In *Proceedings of the 1st international conference on Agents and peer-to-peer computing (AP2PC'02)*, pages 138–143, Berlin, Heidelberg, 2003. Springer-Verlag.

[8] Vikraman Arvind and Pushkar S. Joglekar. Algorithmic Problems for Metrics on Permutation Groups. In *Proceedings of the 34th Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM '08)*, volume 4910 of *Lecture Notes in Computer Science*, pages 136–147. Springer, 2008.

[9] Robert Axelrod. *The Evolution of Cooperation*. Basic Books, New York, NY, USA, 1984.

[10] Hassan Barjini, Mohamed Othman, Hamidah Ibrahim, and Nur Izura Udzir. Shortcoming, problems and analytical comparison for flooding-based search techniques in unstructured P2P networks. *Peer-to-Peer Networking and Applications*, 5(1):1–13, 2012.

[11] Andrés Barrios, Matías Barrios, Daniel De Vera, Pablo Rodríguez-Bocca, and Claudia Rostagnol. GoalBit: A Free and Open Source Peer-to-Peer Streaming Network.

In *Proceedings of the 19th International Conference on Multimedia (ACM MULTIMEDIA 2011), Open Source Software Competition*, pages 727–730, New York, NY, USA, November 2011. ACM.

[12] R. Beckers, J.L. Deneubourg, and S. Goss. Trails and U-turns in the selection of the shortest path by the ant Lasius Niger. *Journal of Theoretical Biology*, 159:397–415, 1992.

[13] Tim Berners-lee, Robert Cailliau, and Bernd Pollermann. World-Wide Web: The Information Universe. *Electronic Networking: Research, Applications and Policy*, 2(1):52–58, 1992.

[14] María Elisa Bertinat, Darío Padula, Franco Robledo, Pablo Rodríguez-Bocca, and Pablo Romero. A Simple Proactive Provider Participation Technique in a Mesh-Based Peer-to-Peer Streaming Service. In *Proceedings of the 6th International Conference on Hybrid Artificial Intelligence Systems (HAIS'11)*, volume 6679, pages 42–50, London, UK, 23-25 May 2011. Springer, Lecture Notes in Computer Science.

[15] María Elisa Bertinat, Darío Padula, Franco Robledo, Pablo Rodríguez-Bocca, and Pablo Romero. Optimal Bandwidth Allocation in mesh-based Peer-to-Peer Streaming Networks. In *Proceedings of the International Network Optimization Conference (INOC'11)*, volume 6701, pages 529–534, London, UK, 13-16 June 2011. Springer, Lecture Notes in Computer Science.

[16] María Elisa Bertinat, Daniel De Vera, Darío Padula, Franco Robledo, Pablo Rodríguez-Bocca, and Pablo Romero. Systematic Procedure for Improving Continuity and Latency on a P2P Streaming Protocol. In *Proceedings of the 1th IEEE Latin-American Conference on Communications (LatinCom'09)*, pages 8–11, Washington, DC, USA, September 2009. IEEE Computer Society.

[17] María Elisa Bertinat, Daniel De Vera, Darío Padula, Franco Robledo, Pablo Rodríguez-Bocca, Pablo Romero, and Gerardo Rubino. A COP for Cooperation in a P2P Streaming Protocol. In *Proceedings of the IEEE International Conference on Ultra Modern Telecommunications (ICUMT'09)*, pages 1–7, Washington, DC, USA, 12-14 October 2009. IEEE Computer Society.

[18] María Elisa Bertinat, Daniel De Vera, Darío Padula, Franco Robledo, Pablo Rodríguez-Bocca, Pablo Romero, and Gerardo Rubino. GoalBit: The First Free and Open Source Peer-to-Peer Streaming Network. In *Proceedings of the 5th international IFIP/ACM Latin American conference on Networking (LANC'09)*, pages 83–93, New York, USA, September 2009. ACM Digital Library.

[19] A. R. Bharambe, C. Herley, and V. N. Padmanabhan. Analyzing and Improving BitTorrent Performance. Technical Report MSR-TR-2005-03, Microsoft Research, 2006.

[20] Ashwin R. Bharambe, Cormac Herley, and Venkata N. Padmanabhan. Analyzing and Improving a BitTorrent Network's Performance Mechanisms. In *INFOCOM*, 2006.

[21] John A. Bingham. *ADSL, VDSL, and Multicarrier Modulation: Wiley Series in Telecommunications and Signal Processing*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 2000.

[22] BitThief home page. http://dcg.ethz.ch/projects/bitthief/, 2006.

[23] Bittorrent Protocol Specification v1.0. http://wiki.theory.org/BitTorrentSpecification, 2010.

[24] Christian Blum. Ant colony optimization: Introduction and recent trends. *Physics of life Reviews*, 2:353–373, october 2005.

[25] B. Bollobás. *Random Graphs*. Academic Press, London, 1985.

[26] Thomas Bonald, Laurent Massoulie, Fabien Mathieu, Diego Perino, and Andrew Twigg. Epidemic live streaming: optimal performance trade-offs. In *Proceedings of the 2008 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 325–336, New York, NY, USA, 2008. ACM.

[27] C. Buragohain, D. Agrawal, and S. Suri. A game theoretic framework for incentives in P2P systems. In *Proceedings of the Third International Conference on Peer-to-Peer Computing, (P2P'03).*, pages 48–56, 2003.

[28] R. W. Burns. Television: An International History of the Formative Years. *Technology and Culture*, 40(2):443–444, 1999.

[29] S.A. Canuto, M.G.C. Resende, and C.C. Ribeiro. Local search with perturbation for the prize-collecting Steiner tree problems in graphs. *Networks*, 38:50–58, 2001.

[30] Bengt Carlsson and Rune Gustavsson. The Rise and Fall of Napster - An Evolutionary Approach. In *AMT '01: Proceedings of the 6th International Computer Science Conference on Active Media Technology*, pages 347–354, London, UK, 2001. Springer-Verlag.

[31] Miguel Castro, Peter Druschel, Anne-Marie Kermarrec, Animesh Nandi, Antony Rowstron, and Atul Singh. SplitStream: high-bandwidth multicast in cooperative environments. In *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 298–313, New York, NY, USA, 2003. ACM Press.

[32] I. M. Chakravarti, R. G. LAHA, and J. Roy. *Handbook of Methods of Applied Statistic*. Wiley Series in Probability and Statistics: Applied Probability and Statistics Section Series. John Wiley & Sons Canada, 1967.

[33] Ilias Chatzidrossos. *Live Streaming Performance of Peer-to-Peer Systems*. PhD thesis, Royal Institute of Technology (KTH), feb 2012.

[34] Ilias Chatzidrossos, György Dan, and Viktoria Fodor. Delay and playout probability trade-off in mesh-based peer-to-peer streaming with delayed buffer map updates. *Peer-to-Peer Networking and Applications*, 3(3):208–221, 2010.

[35] Wu Chehai, Lu Xianliang, and Duan Hancong. Analysis of Content Availability Optimization in BitTorrent. In *Proceedings of the 2006 International Conference on Hybrid Information Technology - Volume 01*, ICHIT '06, pages 525–529, Washington, DC, USA, 2006. IEEE Computer Society.

[36] Minghua Chen, Mung Chiang, Phil Chou, Jin Li, Shao Liu, and Sudipta Sengupta. P2P streaming capacity: survey and recent results. In *Allerton'09: Proceedings of the 47th annual Allerton conference on Communication, control, and computing*, pages 378–387, Piscataway, NJ, USA, 2009. IEEE Press.

[37] Y.F. Chen, Y. Huang, R. Jana, H. Jiang, M. Rabinovich, B. Wei, and Z. Xiao. When is P2P Technology Beneficial for IPTV Services? In *Proceedings of the 17th International Workshop on Network and Operating Systems Support for Digital Audio and Video*. ACM Press, may 2007.

[38] Zhengjun Chen, Kaiping Xue, and Peilin Hong. A Study on Reducing Chunk Scheduling Delay for Mesh-Based P2P Live Streaming. In *Proceedings of the 2008 Seventh International Conference on Grid and Cooperative Computing*, GCC '08, pages 356–361, Washington, DC, USA, 2008. IEEE Computer Society.

[39] Bin Cheng, Hai Jin, and Xiaofei Liao. Rindy: A ring based overlay network for peer-to-peer on-demand streaming. In *Proceedings of the 3rd International Conference on Ubiquitous Intelligence and Computing (UIC '06)*, pages 1048–1058, 2006.

[40] Xu Cheng, Cameron Dale, and Jiangchuan Liu. Understanding the Characteristics of Internet Short Video Sharing: YouTube as a Case Study. Technical report, Cornell University, july 2007.

[41] Yung R. Choe, Derek L. Schuff, Jagadeesh M. Dyaberi, and Vijay S. Pai. Improving VoD server efficiency with bittorrent. In *MULTIMEDIA '07: Proceedings of the 15th international conference on Multimedia*, pages 117–126, New York, NY, USA, 2007. ACM Press.

[42] David R. Choffnes and Fabián E. Bustamante. Taming the Torrent: A Practical Approach to Reducing Cross-ISP Traffic in Peer-to-Peer Systems. *ACM SIGCOMM Computer Communication Review*, 38(4):363–374, 2008.

[43] Yang-hua Chu, John Chuang, and Hui Zhang. A case for taxation in peer-to-peer streaming broadcast. In *Proceedings of the ACM SIGCOMM workshop on Practice and theory of incentives in networked systems*, PINS '04, pages 205–212, New York, NY, USA, 2004. ACM.

[44] Bram Cohen. Incentives Build Robustness in BitTorrent. *www.bramcohen.com*, 1:1–5, May 2003.

[45] Reuven Cohen, Keren Erez, Daniel Ben-Avraham, and Shlomo Havlin. Resilience of the Internet to Random Breakdowns. *Physical Review Letters*, 85(21):4626–4628, November 2000.

[46] Thomas M. Cover and Joy A. Thomas. *Elements of information theory*. Wiley-Interscience, New York, NY, USA, 1991.

[47] Albert Creus-Mir, Ramon Casadesus-Masanell, and Andres Hervas-Drane. Bandwidth allocation in peer-to-peer file sharing networks. *Comput. Commun.*, 31(2):257–265, feb 2008.

[48] G. A. Croes. A Method for Solving Traveling-Salesman Problems. *Operations Research*, 6:791–812, 1958.

[49] Scott A. Crosby and Dan S. Wallach. An Analysis of BitTorrent's Two Kademlia-Based DHTs. Technical Report TR-07-04, Department of Computer Science, Rice University, June 2007.

[50] Yi Cui, Baochun Li, and Klara Nahrstedt. oStream: Asynchronous Streaming Multicast in Application-Layer Overlay Networks. *IEEE Journal on Selected Areas in Communications*, 22(1):91–106, 2003.

[51] Yi Cui and Klara Nahrstedt. Layered Peer-to-Peer Streaming. In *Proceedings of the 13th international workshop on Network and operating systems support for digital audio and video (NOSSDAV '03)*, pages 162 – 171, New York, NY, USA, 2003. ACM Press.

[52] Ana Paula Couto da Silva, Emilio Leonardi, Marco Mellia, and Michela Meo. Chunk Distribution in Mesh-Based Large-Scale P2P Streaming Systems: A Fluid Approach. *IEEE Transactions on Parallel and Distributed Systems*, 22:451–463, 2011.

[53] Lucia D'Acunto, Nazareno Andrade, Johan Pouwelse, and Henk Sips. Peer Selection Strategies for Improved QoS in Heterogeneous BitTorrent-Like VoD Systems. In *Proceedings of the 2010 IEEE International Symposium on Multimedia*, pages 89–96, Washington, DC, USA, 2010. IEEE Computer Society.

[54] Ralph B D'Agostino and Michael A Stephens. *Goodness-of-fit techniques*. Marcel Dekker, Inc., New York, NY, USA, 1986.

[55] Cameron Dale and Jiangchuan Liu. A Measurement Study of Piece Population in BitTorrent. In *GLOBECOM'07*, pages 405–410, 2007.

[56] Asit Dan, Dinkar Sitaram, and Perwez Shahabuddin. Scheduling Policies for an On-Demand Video Server with Batching. In *Proceedings of ACM Multimedia*, pages 15–23, 1994.

[57] C. Dana, Danjue Li, D. Harrison, and Chen-Nee Chuah. BASS: BitTorrent Assisted Streaming System for Video-on-Demand. In *IEEE 7th Workshop on Multimedia Signal Processing*, pages 1–4, 2005.

[58] G. de Veciana and X. Yang. Fairness, incentives and performance in peer-to-peer networks. In *Proceedings of the Forty-First Annual Allerton Conference on Control, Communications and Computing*. IEEE, October 2003.

[59] M Deza and T. Huang. Metrics on permutations, a survey. *Journal of Combinatorics, Information and System Sciences*, 23:173–185, 1998.

[60] Prithula Dhungel, Xiaojun Hei, Keith W. Ross, and Nitesh Saxena. The pollution attack in P2P live video streaming: measurement results and defenses. In *Proceedings of the 2007 workshop on Peer-to-peer streaming and IP-TV*, P2P-TV '07, pages 323–328, New York, NY, USA, 2007. ACM.

[61] John Dilley, Bruce Maggs, Jay Parikh, Harald Prokop, Ramesh Sitaraman, and Bill Weihl. Globally Distributed Content Delivery. *IEEE Internet Computing*, 6(5):50–58, 2002.

[62] Tai T. Do, Kien A. Hua, and Mounir A. Tantaoui. P2VoD: Providing fault tolerant video-on-demand streaming in peer-to-peer environment. In *Proceedings of the IEEE International Conference on Communications (ICC 2004*, pages 1467–1472, 2004.

[63] Yves Dony. Modeling and Analysis of P2P Streaming. Memoria, Master's Thesis. The Chinese University of Hong Kong, Namur, Faculty of Computer Science, June 2008.

[64] M. Dorigo. *Optimization, Learning and Natural Algorithms*. PhD thesis, DEI Politecnico de Milano, Italia, 1992.

[65] Marco Dorigo, Mauro Birattari, and Thomas Stützle. Artificial Ants as a Computational Intelligence Technique. Technical Report 23, Institut de Recherches Interdisciplinaires, Universite Libre de Bruxelles, september 2006.

[66] Marco Dorigo and Luca Maria Gambardella. Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66, 1997.

[67] Marco Dorigo and Thomas Stutzle. *Ant Colony Optimization*. MIT Press, 2004.

[68] Haibin Duan, Guanjun Ma, and Senqi Liu. Experimental Study of the Adjustable Parameters in Basic Ant Colony Optimization Algorithm. *IEEE Congress on Evolutionary Computation*, 1(1):149–156, 2007.

[69] D. Erman. Extending bittorrent for streaming applications. In *4th Euro-FGI Workshop on New Trends in Modelling, Quantitative Methods and Measurements*, 2007.

[70] David Erman, Dragos Ilie, and Adrian Popescu. Bittorrent session characteristics and models. In *Proceedings of the 3rd International Working Conference on Performance Modelling and Evaluation of Heterogeneous Networks*, pages 1–10, jul 2005.

[71] Ethereal home page. http://www.ethereal.com/, 2007.

[72] Bin Fan, Dah-Ming Chiu, and J.C.S. Lui. The Delicate Tradeoffs in BitTorrent-like File Sharing Protocol Design. In *Proceedings of the 14th IEEE International Conference on Network Protocols, 2006. ICNP '06*, pages 239–248, nov 2006.

[73] Bin Fan, John C. S. Lui, and Dah-Ming Chiu. The design trade-offs of BitTorrent-like file sharing protocols. *IEEE/ACM Trans. Netw.*, 17(2):365–376, 2009.

[74] A. Farley. Broadcast Time in Communication Networks. *SIAM Journal on Applied Mathematics*, 39(2):385–390, 1980.

[75] Michal Feldman, Kevin Lai, Ion Stoica, and John Chuang. Robust incentive techniques for peer-to-peer networks. In *Proceedings of the 5th ACM conference on Electronic commerce*, EC '04, pages 102–111, New York, NY, USA, 2004. ACM.

[76] Thomas A. Feo and Mauricio G. C. Resende. A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, 8(2):67 – 71, 1989.

[77] P. Festa and M. G. C. Resende. GRASP: An annotated bibliography. In *Essays and Surveys in Metaheuristics*, pages 325–367. Kluwer Academic Publishers, 2002.

[78] Geral Folland. *Real Analysis: Modern Techniques and Their Applications*. Wiley Series of Pure and Applied Mathematics, 1999.

[79] Anh-Tuan Gai, Fabien Mathieu, Fabien de Montgolfier, and Julien Reynier. Stratification in P2P Networks: Application to BitTorrent. In *Proceedings of the 27th International Conference on Distributed Computing Systems ICDCS' 07*, page 30. IEEE Computer Society, June 25-27 2007.

[80] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Company, New York, NY, USA, 1979.

[81] Z. Ge, D. R. Figueiredo, Sharad Jaiswal, J. Kurose, and D. Towsley. Modeling peer-peer file sharing systems. In *Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies. INFOCOM 2003. IEEE*, volume 3, pages 2188–2198, March 2003.

[82] Michel Gendreau and Jean-Yves Potvin. *Handbook of Metaheuristics*. Springer Publishing Company, Incorporated, 2nd edition, 2010.

[83] Fred Glover. Tabu search - part i. *INFORMS Journal on Computing*, 1(3):190–206, 1989.

[84] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1989.

[85] Philippe Golle, Kevin Leyton-Brown, and Ilya Mironov. Incentives for sharing in peer-to-peer networks. In *Proceedings of the 3rd ACM conference on Electronic Commerce*, EC '01, pages 264–267, New York, NY, USA, 2001. ACM.

[86] GT-ITM home page. http://www.cc.gatech.edu/projects/gtitm/, 2000.

[87] Yang Guo, Kyoungwon Suh, Jim Kurose, and Don Towsley. P2Cast: peer-to-peer patching scheme for VoD service. In *in Proc. WWW'03*, pages 301–309, 2003.

[88] Ahsan Habib and John Chuang. Incentive Mechanism for Peer-to-Peer Media Streaming. In *In 12th IEEE International Workshop on Quality of Service (IWQoS' 04)*. IEEE, June 7-9 2004.

[89] Bruce Hajek and Ji Zhu. The Missing Piece Syndrome in Peer-to-Peer Communication. In *Proceedings of the IEEE International Symposium on Information Theory*, pages 1748 –1752, June 2010.

[90] Anwar Al Hamra, Ernst W. Biersack, and Guillaume Urvoy-Keller. A Pull-Based Approach for a VoD Service in P2P Networks. In *HSNMC'04*, pages 995–1006, 2004.

[91] Pierre Hansen and Nenad Mladenovic. Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130(3):449–467, May, 1 2001.

[92] Mohamed Hefeeda, Bharat K. Bhargava, and David K. Y. Yau. A hybrid architecture for cost-effective on-demand media streaming. *Computer Networks*, 44(3):353–382, 2004.

[93] X. Hei, C. Liang, J. Liang, Y. Liu, and K. W. Ross. Insights into PPLive: A Measurement Study of a Large-Scale P2P IPTV System. In *Proc. of IPTV Workshop, International World Wide Web Conference*, 2006.

[94] Yang hua Chu. Considering altruism in peer-to-peer internet streaming broadcast. In *Proceedings of ACM NOSSDAV*, pages 10–15. ACM Press, 2004.

[95] Yang hua Chu, Sanjay G. Rao, and Hui Zhang. A case for end system multicast. In *Proceedings of the 2000 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 1–12, New York, NY, USA, 2000. ACM Press.

[96] Cheng Huang, Jin Li, and Keith W. Ross. Can internet video-on-demand be profitable? *SIGCOMM Computer Communication Review*, 37(4):133–144, aug 2007.

[97] Cheng Huang, Angela Wang, Jin Li, and Keith W. Ross. Understanding hybrid cdn-p2p: why limelight needs its own red swoosh. In *Proceedings of the 18th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, NOSSDAV '08, pages 75–80, New York, NY, USA, 2008. ACM Press.

[98] Yan Huang, Tom Z.J. Fu, Dah-Ming Chiu, John C.S. Lui, and Cheng Huang. Challenges, Design and Analysis of a Large-Scale P2P-VoD System. *SIGCOMM Computer Communication Review*, 38(4):375–388, aug 2008.

[99] B. A. Huberman and R. M. Lukose. Social dilemmas and Internet congestion. *Science*, 277:535–537, 1997.

[100] M. Izal, Guillaume Urvoy-Keller, Ernst W. Biersack, P. A. Felber, A. Al Hamra, and L. Garcés-Erice. Dissecting BitTorrent: Five Months in a Torrent's Lifetime. In *Proceedings of the 5th Annual Passive & Active Measurement Workshop*, pages 1–11. Lecture Notes in Computer Science, April 2004.

[101] Thomas Karagiannis, Pablo Rodriguez, and Konstantina Papagiannaki. Should internet service providers fear peer-assisted content distribution? In *Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement (IMC '05)*, pages 6–6, Berkeley, CA, USA, 2005. USENIX Association.

[102] J.L Kelley. *General topology*. Springer-Verlag, New York, USA, 1975.

[103] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of the IEEE International Conference on Neural Networks*, volume 4, pages 1942–1948 vol.4. IEEE, November 1995.

[104] Scott Kirkpatrick. Optimization by simulated annealing: Quantitative studies. *Journal of Statistical Physics*, 34(5):975–986, March 1984.

[105] Simon G. M. Koo, George C. S. Lee, and Karthik Kannan. A Genetic-Algorithm-Based Neighbor-Selection Strategy for Hybrid Peer-to-Peer Networks. In *Proceedings of the International Conference On Computer Communications and Networks (ICCCN'04)*, pages 469–474, 2004.

[106] G. Kreitz and F. Niemela. Spotify – Large Scale, Low Latency, P2P Music-on-Demand Streaming. In *Proceedings of the Tenth IEEE International Conference on Peer-to-Peer Computing (P2P)*, pages 1–10, aug 2010.

[107] R. Kumar, Yong Liu, and K. Ross. Stochastic Fluid Theory for P2P Streaming Systems. In *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, pages 919–927, 2007.

[108] R. Kumar and K.W. Ross. Peer-Assisted File Distribution: The Minimum Distribution Time. *Hot Topics in Web Systems and Technologies*, 0:1–11, 2006.

[109] Kevin Lee, Danny Hughes, and James Walkerdine. On the penetration of business networks by p2p file sharing. In *Proceedings of the Second International Conference on Internet Monitoring and Protection (ICIMP'07)*, ICIMP '07, pages 23–28, Washington, DC, USA, 2007. IEEE Computer Society.

[110] A. Legout, G. Urvoy-Keller, and P. Michiardi. Understanding BitTorrent: An Experimental Perspective. Technical Report inria-00000156, INRIA, Sophia Antipolis, November 2005.

[111] Arnaud Legout, Nikitas Liogkas, Eddie Kohler, and Lixia Zhang. Clustering and sharing incentives in BitTorrent systems. *SIGMETRICS Perform. Eval. Rev.*, 35(1):301–312, jun 2007.

[112] Arnaud Legout, G. Urvoy-Keller, and P. Michiardi. Rarest first and choke algorithms are enough. In *IMC '06: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 203–216, New York, NY, USA, 2006. ACM.

[113] Songqing Chen Lei Guo and Xiaodong Zhang. Design and evaluation of a scalable and reliable P2P assisted proxy for on-demand streaming media delivery. *IEEE Transactions on Knowledge and Data Engineering*, 18(5):669 – 682, May 2006.

[114] Daniel M. Lewin. Consistent hashing and random trees: algorithms for caching in distributed networks. Memoria, Dept. of Electrical Engineering and Computer Science, 77 Massachusetts Avenue, Setiembre 2008.

[115] Jin Li, Philip A. Chou, and Cha Zhang. Mutualcast: An Efficient Mechanism for Content Distribution in a P2P Network. In *Proceedings of Acm Sigcomm Asia Workshop*, April 2005.

[116] D. Liberzon and M. Morse. Basic problems in stability and design of switched systems. *IEEE Control System Magazine*, pages 59–70, 1999.

[117] Limelight website. http://www.limelight.com/, 2007.

[118] S. Lin and B. W. Kernighan. An Effective Heuristic Algorithm for the Traveling-Salesman Problem. *Operations Research*, 21(2):498–516, March 1973.

[119] W. Sabrina Lin, H. Vicky Zhao, and K. J. Ray Liu. A game theoretic framework for incentive-based peer-to-peer live-streaming social networks. In *ICASSP*, pages 2141–2144. IEEE, 2008.

[120] Ma Lingjun, Pui-Sze Tsang, and King-Shan Lui. Improving file distribution performance by grouping in peer-to-peer networks. *IEEE Trans. on Netw. and Serv. Manag.*, 6(3):149–162, sep 2009.

[121] Nikitas Liogkas, Robert Nelson, Eddie Kohler, and Lixia Zhang. Exploiting BitTorrent for Fun (But Not Profit). In *Proceedings of the 5th International workshop on Peer-To-Peer Systems, IPTPS'06*, February, 27-28 2006.

[122] Shao Liu, Minghua Chen, Sudipta Sengupta, Mung Chiang, Jin Li, and Philip A. Chou. P2P Streaming Capacity under Node Degree Bound. In *ICDCS*, pages 587–598, 2010.

[123] Yong Liu. On the minimum delay peer-to-peer video streaming: how realtime can it be? In *Proceedings of the 15th international conference on Multimedia*, MULTIMEDIA '07, pages 127–136, New York, NY, USA, 2007. ACM.

[124] Zhengye Liu, Yanming Shen, Shivendra S. Panwar, Keith W. Ross, and Yao Wang. Using layered video to provide incentives in P2P live streaming. In *Proceedings of the 2007 workshop on Peer-to-peer streaming and IP-TV*, P2P-TV '07, pages 311–316, New York, NY, USA, 2007. ACM.

[125] Zhengye Liu, Yanming Shen, Keith W. Ross, Shivendra S. Panwar, and Yao Wang. LayerP2P: Using Layered Video Chunks in P2P Live Streaming. *IEEE Transactions on Multimedia*, 11(7):1340–1352, nov 2009.

[126] R. Lo Cigno, A. Russo, and D. Carra. On some fundamental properties of P2P push/pull protocols. In *Second International Conference on Communications and Electronics, 2008. ICCE 2008*, pages 67–73, 2008.

[127] Thomas Locher, Patrick Moor, Stefan Schmid, and Roger Wattenhofer. Free Riding in BitTorrent is Cheap. In *5th Workshop on Hot Topics in Networks (HotNets), Irvine, California, USA*, November 2006.

[128] Yue Lu, Jan David Mol, Fernando Kuipers, and Piet Van Mieghem. Analytical Model for Mesh-Based P2PVoD. In *Proceedings of the 2008 Tenth IEEE International Symposium on Multimedia*, ISM '08, pages 364–371, Washington, DC, USA, 2008. IEEE Computer Society.

[129] D.G. Luenberger. *Introduction to dynamic systems: theory, models, and applications*. Wiley, 1979.

[130] Jun Luo. Practical algorithm for minimum delay peer-to-peer media streaming. In *IEEE International Conference on Multimedia and Expo. (ICME'10)*, pages 986–991, 2010.

[131] Thibaut Lust and Jacques Teghem. The multiobjective multidimensional knapsack problem: a survey and a new approach. *Journals on Computers and Operations Research*, abs/1007.4063, 2010.

[132] Nazanin Magharei and Reza Rejaie. Mesh or multiple-tree: A comparative study of live P2P streaming approaches. In *in Proceedings of IEEE INFOCOM*, pages 1424–1432, 2007.

[133] Nazanin Magharei and Reza Rejaie. PRIME: Peer-to-Peer Receiver-drIven MEsh-Based Streaming. In *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, pages 1415–1423, may 2007.

[134] Nazanin Magharei and Reza Rejaie. Overlay monitoring and repair in swarm-based peer-to-peer streaming. In *Proceedings of the 18th international workshop on Network and operating systems support for digital audio and video*, NOSSDAV '09, pages 25–30, New York, NY, USA, 2009. ACM.

[135] PawełMarciniak, Nikitas Liogkas, Arnaud Legout, and Eddie Kohler. Small is not always beautiful. In *Proceedings of the 7th international conference on Peer-to-peer systems (IPTPS'08)*, pages 9–9, Berkeley, CA, USA, February 25-26 2008. USENIX Association.

[136] Laurent Massoulié, Andrew Twigg, Christos Gkantsidis, and Pablo Rodriguez. Randomized Decentralized Broadcasting Algorithms. In *INFOCOM*, pages 1073–1081, 2007.

[137] Laurent Massoulié and Milan Vojnović. Coupon replication systems. *SIGMETRICS Perform. Eval. Rev.*, 33(1):2–13, jun 2005.

[138] Laurent Massoulié. Peer-to-peer live streaming: Optimality results and open problems. In *CISS'08*, pages 313–315, 2008.

[139] F. Mathieu and J. Reynier. Missing Piece Issue and Upload Strategies in Flashcrowds and P2P-assisted Filesharing. In *Advanced Int'l Conference on Telecommunications and Int'l Conference on Internet and Web Applications and Services (AICT-ICIW'06)*, page 112. IEEE, February 19-25 2006.

[140] Petar Maymounkov and David Mazières. Kademlia: A Peer-to-Peer Information System Based on the XOR Metric. In *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*, pages 53–65, London, UK, 2002. Springer-Verlag.

[141] Alberto Medina, Anukool Lakhina, Ibrahim Matta, and John Byers. Brite: Universal topology generation from a user"s perspective. Technical report, Boston University, Boston, MA, USA, 2001.

[142] Daniel Sadoc Menasché, Antonio A. de A. Rocha, Edmundo A. de Souza e Silva, Don Towsley, and Rosa M. Meri Leäo. Implications of peer selection strategies by publishers on the performance of P2P swarming systems. *SIGMETRICS Perform. Eval. Rev.*, 39(3):55–57, dec 2011.

[143] James A. Mirrlees. An exploration in the theory of optimum income taxation. *Review of Economic Studies*, 38(114):175–208, 1971.

[144] J. J. D. Mol, J. A. Pouwelse, M. Meulpolder, D. H. J. Epema, and H. J. Sips. Give-to-Get: Free-riding-resilient Video-on-Demand in P2P Systems. In *Proceedings of the Fifteenth Annual Multimedia Computing and Networking Conference (MMCN'08)*, January 30-31 2008.

[145] J.J.D. Mol. *Free-riding, Resilient Video Streaming in Peer-to-Peer Networks*. PhD thesis, Delft University of Technology, Netherlands, January 2010.

[146] Igor M. Moraes and Otto Carlos Muniz Bandeira Duarte. A lifetime-based peer selection mechanism for peer-to-peer video-on-demand systems. In *Fourth International Conference on Communications and Electronics (ICC'10)*, pages 1–5, 2010.

[147] Jochen Mundinger, Richard Weber, and Gideon Weiss. Analysis of peer-to-peer file dissemination. *SIGMETRICS Perform. Eval. Rev.*, 34(3):12–14, dec 2006.

[148] Jochen Mundinger, Richard Weber, and Gideon Weiss. Optimal scheduling of peer-to-peer file dissemination. *J. of Scheduling*, 11(2):105–120, apr 2008.

[149] Napster home page. http://www.napster.com/, 2005.

[150] Joseph Needham. *Science and civilisation in China*. Cambridge University Press, 1965.

[151] E. Nicholas and C. Himmelberg. Critical Mass and Network Evolution in Telecommunications. In *Telecommunications Policy Research Conference, Gerard Brock (ed.)*, 1995.

[152] Ilkka Norros, Hannu Reittu, and Timo Eirola. On the stability of two-chunk file-sharing systems. *Queueing Syst. Theory Appl.*, 67(3):183–206, mar 2011.

[153] Temel Oncan, Y. Kuban Altynel, and Gilbert Laporte. A comparative analysis of several asymmetric traveling salesman problem formulations. *Computers & Operations Research.*, 36(3):637–654, march 2009.

[154] Venkata N. Padmanabhan, Helen J. Wang, and Philip A. Chou. Resilient Peer-to-Peer Streaming. In *Proceedings of the 11th IEEE International Conference on Network Protocols (ICNP'03)*, pages 16–27, Washington, DC, USA, November 4-7 2003. IEEE Computer Society.

[155] Venkata N. Padmanabhan, Helen J. Wang, Philip A. Chou, and Kunwadee Sripanidkulchai. Distributing streaming media content using cooperative networking. In *NOSSDAV '02*, pages 177–186, New York, NY, USA, 2002. ACM Press.

[156] D. Pakkala and J. Latvakoski. Towards a peer-to-peer extended content delivery network. In *Proceedings of the 14th IST Mobile & Wireless Communications*, June 2005.

[157] Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial optimization: algorithms and complexity*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1982.

[158] Nadim Parvez, Carey Williamson, Anirban Mahanti, and Niklas Carlsson. Analysis of bittorrent-like protocols for on-demand stored media streaming. *SIGMETRICS Perform. Eval. Rev.*, 36(1):301–312, jun 2008.

[159] Piatek, Michael, Krishnamurthy, Arvind, Venkataramani, Arun, Yang, Richard, Zhang, David, Jaffe, and Alexander. Contracts: practical contribution incentives for P2P live streaming. In *Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation (NSDI'10)*, pages 1–14, Berkeley, CA, USA, April 2010. USENIX Association.

[160] Michael Piatek, Tomas Isdal, Thomas Anderson, Arvind Krishnamurthy, and Arun Venkataramani. Do incentives build robustness in BitTorrent? In *Proceedings of 4th USENIX Symposium on Networked Systems Design & Implementation (NSDI'07)*. USENIX Association, April 2007.

[161] J. A. Pouwelse, P. Garbacki, J. Wang, A. Bakker, J. Yang, A. Iosup, D. H. J. Epema, M. Reinders, M. R. van Steen, and H. J. Sips. TRIBLER: a social-based peer-to-peer system. *Concurr. Comput. : Pract. Exper.*, pages 127–138, feb 2008.

[162] Johan Pouwelse, Paweł Garbacki, Dick Epema, and Henk Sips. The Bittorrent P2P File-Sharing System: Measurements and Analysis. In Miguel Castro and Robbert van Renesse, editors, *Peer-to-Peer Systems IV*, volume 3640, chapter 19, pages 205–216. Springer Lecture Notes in Computer Science, Berlin, Heidelberg, 2005.

[163] PPLive Home page. http://www.pplive.com, 2007.

[164] PPStream home page. http://www.ppstream.com/, 2007.

[165] Anna Puig-centelles, Oscar Ripolles, and Miguel Chover. P2P Traffic Measurements on the Emule System. In *Proceedings of the International Conference on Telecommunications, Networks and Systems 2007 (MCCSIS 2007)*, pages 51–58, London, UK, 2007. IADIS Digital Library.

[166] Darshan Purandare and Ratan Guha. An alliance based peering scheme for peer-to-peer live media streaming. In *Proceedings of the 2007 workshop on Peer-to-peer streaming and IP-TV*, P2P-TV '07, pages 340–345, New York, NY, USA, 2007. ACM.

[167] Fenglin Qin, Liansheng Ge, Qi Liu, and Ju Liu. Free Riding Analysis of Peer-to-Peer Streaming Systems. *Computational Information Systems*, 7(3):721–728, June 2011.

[168] Dongyu Qiu and Weiqian Sang. Global stability of Peer-to-Peer file sharing systems. *Comput. Commun.*, 31(2):212–219, feb 2008.

[169] Dongyu Qiu and R. Srikant. Modeling and Performance Analysis of BitTorrent-Like Peer-to-Peer Networks. In *Proceedings of SIGCOMM'04*, pages 367–378, New York, NY, USA, September 2004. ACM.

[170] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Schenker. A scalable content-addressable network. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 161–172, New York, NY, USA, 2001. ACM Press.

[171] Sylvia Ratnasamy, Mark Handley, Richard M. Karp, and Scott Shenker. Topologically-Aware Overlay Coruction and Server Selection. In *INFOCOM*, 2002.

[172] Reza Rejaie and Antonio Ortega. PALS: peer-to-peer adaptive layered streaming. In *Proceedings of the 13th international workshop on Network and operating systems support for digital audio and video*, pages 153–161, New York, NY, USA, 2003. ACM.

[173] Zhenfeng Ren, Ju Liu, Fenglin Qin, and Yanwei Wang. A Survey on Peer-to-Peer Streaming System's Neighbor Number. In *Proceedings of the 2009 Second International Workshop on Computer Science and Engineering - Volume 01*, IWCSE '09, pages 421–424, Washington, DC, USA, 2009. IEEE Computer Society.

[174] M. G. C. Resende and C. C. Ribeiro. *Greedy Randomized Adaptive Search Procedures*. In F. Glover and G. Kochenberger, editors, Handbook of Methaheuristics, Kluwer Academic Publishers, 2003.

[175] M. Ripeanu. Peer-to-peer architecture case study: Gnutella network. In *Proceedings of the First International Conference on Peer-to-Peer Computing*, pages 99–100, aug 2001.

[176] Larry Roberts. The Arpanet and computer networks. In *Proceedings of the ACM Conference on The history of personal workstations*, HPW '86, pages 51–58, New York, NY, USA, 1986. ACM.

[177] Franco Robledo, Pablo Rodríguez-Bocca, Pablo Romero, and Claudia Rostagnol. A new caching policy for cloud assisted Peer-to-Peer video-on-demand services. In *Proceedings of the IEEE International Conference on Peer-to-Peer Computing (P2PCONF'12)*, Washington, DC, USA, 3-5 September 2012. IEEEXplore digital library.

[178] Franco Robledo, Pablo Rodríguez-Bocca, Pablo Romero, and Claudia Rostagnol. Stability and Capacity of Peer-to-Peer Assisted Video-on-Demand Applications. In *Proceedings of the fourth Iternational Congress on Ultra Modern Telecommunications and Control Systems (ICUMT'12)*, Washington, DC, USA, 3-5 October 2012. IEEE Computer Society.

[179] Franco Robledo Amoza. *GRASP heuristics for Wide Area Network design*. PhD thesis, INRIA/IRISA, Université de Rennes I, Rennes, France, 2005.

[180] Pablo Rodriguez, See-Mong Tan, and Christos Gkantsidis. On the feasibility of commercial, legal P2P content distribution. *SIGCOMM Computer Communication Review*, 36(1):75–78, January 2006.

[181] Pablo Romero Rodríguez. Optimización de la Estrategia de Selección de Piezas de Video en Redes P2P. Master's Thesis. Universidad de la República, Facultad de Ingeniería. Montevideo, Uruguay, November 2009.

[182] Pablo Rodríguez-Bocca. *Quality-centric design of Peer-to-Peer systems for live-video broadcasting*. PhD thesis, INRIA/IRISA, Université de Rennes I, Rennes, France, April 2008.

[183] Pablo Rodríguez-Bocca and Claudia Rostagnol. Optimal Download Time in a cloud-assisted Peer-to-Peer Video on Demand service. In *Proceedings of the 5th International Network Optimization Conference (INOC'11)*, pages 283–288, London, UK, 13-16 June 2011. Springer, Lecture Notes in Computer Science.

[184] P. Romero, F. Robledo, and P. Rodríguez-Bocca. An Ant-Colony approach for the design of optimal Chunk Scheduling Policies in live Peer-to-peer networks. *International Journal of Metaheuristics*, To appear, 2012.

[185] P. Romero, F. Robledo, and P. Rodríguez-Bocca. Optimum Piece Selection Strategies for A Peer-to-Peer Video Streaming Platform. *Computers & Operations Research*, 2012. To appear.

[186] Pablo Romero, María Elisa Bertinat, Darío Padula, Franco Robledo, and Pablo Rodríguez-Bocca. A Cooperative Model for Multi-Class Peer-to-Peer Straming Networks. In *Proceedings of the 1st International Conference on Operations Research and Enterprise Systems (ICORES'12)*, pages 274–282, 4-6 February 2012.

[187] Pablo Romero, Franco Robledo, Pablo Rodríguez-Bocca, Darío Padula, and María Elisa Bertinat. A Cooperative Network Game Efficiently Solved via an Ant Colony Optimization Approach. In *Proceedings of the Seventh International Conference of Swarm*

Intelligence (ANTS'10), volume 6234, pages 336–343, London, UK, 8-10 September 2010. Springer, Lecture Notes in Computer Science.

[188] Pablo Romero, Franco Robledo, Pablo Rodríguez-Bocca, and Claudia Rostagnol. Analysis and Design of Peer-Assisted Video On-Demand Services. *International Transactions in Operational Research*, 2012.

[189] Antony Rowstron and Peter Druschel. Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems. *Lecture Notes in Computer Science*, 2218:329–347, 2001.

[190] Sujay Sanghavi, Bruce Hajek, and Laurent Massoulié. Gossiping with Multiple Messages. In *INFOCOM 2007. 26th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 6-12 May 2007, Anchorage, Alaska, USA*, pages 2135–2143. IEEE, 2007.

[191] Stefan Saroiu, Krishna P. Gummadi, Richard J. Dunn, Steven D. Gribble, and Henry M. Levy. An analysis of internet content delivery systems. *SIGOPS Oper. Syst. Rev.*, 36:315–327, December 2002.

[192] Stefan Saroiu, P. Krishna Gummadi, and Steven D. Gribble. A Measurement Study of Peer-to-Peer File Sharing Systems. In *Proceedings of the 9th International Conference in Multimedia Computing and Networking (MMCN'02)*, San Jose Convention Center, San Jose, California, USA, January 30-31 2002.

[193] Simon Schubert, Frank Uyeda, Nedeljko Vasić, Naveen Cherukuri, and Dejan Kostić. Bandwidth adaptation in streaming overlays. In *Proceedings of the 2nd international conference on COMmunication Systems and NETworks*, COMSNETS'10, pages 89–98, Piscataway, NJ, USA, 2010. IEEE Press.

[194] Purvi Shah and Jean françois Pâris. Peer-to-Peer Multimedia Streaming Using BitTorrent. In *Proceedings of the 26th IEEE International Performance Computing and Communications Conference (IPCCC'07)*, 2007.

[195] C. E. Shannon. A mathematical theory of communication. *Bell system technical journal*, 27:379–423, July 1948.

[196] S. Sheu, K. Hua, and W. Tavanapong. Chaining: a Generalized Batching Technique for Video-on-Demand Systems. In *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, pages 110–117, June 3-6 1997.

[197] Annapureddy Siddhartha, Saikat Guha, Christos Gkantsidis, Dinan Gunawardena, and Pablo Rodriguez. Is high-quality vod feasible using P2P swarming? In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, pages 903–912, New York, NY, USA, 2007. ACM.

[198] Thomas Silverston, Olivier Fourmaux, and Jon Crowcroft. Towards an incentive mechanism for peer-to-peer multimedia live streaming systems. In *Proceedings of the 2008*

Eighth International Conference on Peer-to-Peer Computing (P2P'08), pages 125–128, Washington, DC, USA, 2008. IEEE Computer Society.

[199] Tara Small, Ben Liang, and Baochun Li. Scaling laws and tradeoffs in peer-to-peer live multimedia streaming. In *Proceedings of the 14th annual ACM international conference on Multimedia*, MULTIMEDIA '06, pages 539–548, New York, NY, USA, 2006. ACM.

[200] SopCast - Free P2P internet TV. http://www.sopcast.org, 2007.

[201] A. Srinivasan, K.G. Ramakrishnan, K. Kumaram, M. Aravamudam, and S. Naqvi. Optimal design of signaling networks for Internet telephony. In *Proceedgins of the IEEE INFOCOM 2000*, March 2000.

[202] N. H. Stern. On the specification of models of optimum income taxation. *Journal of Public Economics*, 6(1-2):123–162, 1976.

[203] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 149–160, New York, NY, USA, 2001. ACM Press.

[204] Daniel Stutzbach and Reza Rejaie. Understanding churn in peer-to-peer networks. In *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, IMC '06, pages 189–202, New York, NY, USA, 2006. ACM.

[205] Andrew Tanenbaum. *Computer Networks*. Prentice Hall Professional Technical Reference, 4th edition, 2002.

[206] S. Tewari and L. Kleinrock. Analytical Model for BitTorrent-based Live Video Streaming. In *4th IEEE Consumer Communications and Networking Conference (CCNC'07)*, pages 976–980, Las Vegas, NV, January 2007.

[207] Y. Tian, D. Wu, and K. W. Ng. Modeling, Analysis and Improvement for BitTorrent-Like File Sharing Networks. In *Proceedings of 25th IEEE International Conference on Computer Communications (INFOCOM 2006)*, pages 1–11, apr 2006.

[208] Ye Tian, Di Wu, and Kam-Wing Ng. Analyzing Multiple File Downloading in BitTorrent. In *Proceedings of ICPP'06*, pages 297–306. IEEE, August 2006.

[209] TVAnts home page. http://cache.tvants.com/, 2007.

[210] TVUnetworks home page. http://tvunetworks.com/, 2007.

[211] Robbert van Renesse, Maya Haridasan, and Ingrid Jansch-Porto. Enforcing fairness in a live-streaming system. In *Multimedia Computing and Networking (MMCN'08)*, January 30-31 2008.

[212] Maarten van Steen, Franz J. Hauck, Gerco Ballintijn, and Andrew S. Tanenbaum. Algorithmic Design of the Globe Wide-Area Location Service. *Comput. J.*, 41(5):297–310, 1998.

[213] V. Vishnumurthy, S. Chandrakumar, and E. Sirer. KARMA: A Secure Economic Framework for Peer-to-Peer Resource Sharing. In *First Workshop on the Economics of Peer-to-Peer Systems*, June 2003.

[214] A. Vlavianos, M. Iliofotou, and M. Faloutsos. BiTOS: Enhancing BitTorrent for Supporting Streaming Applications. In *Proceedings of the 25th IEEE International Conference on Computer Communications INFOCOM'06*, pages 1–6, April 2006.

[215] Duncan J. Watts and Steven H. Strogatz. Collective dynamics of Small World Network. *Nature*, 393(6684):440 – 442, April, 6 1998.

[216] Bryce Wilcox-O'Hearn. Experiences deploying a large-scale emergent network. In *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, International workshop on Peer-To-Peer Systems (IPTPS '01), pages 104–110, London, UK, UK, 2002. Springer-Verlag.

[217] Chi-Jen Wu, Cheng-Ying Li, and Jan-Ming Ho. Improving the Download Time of BitTorrent-like Systems. In *IEEE International Conference on Communications 2007 (ICC 2007)*, pages 1125–1129, Glasgow, Scotland, June 2007.

[218] Di Wu. Fairness Analysis of Peer-to-Peer Streaming Systems. In *Proc. of International Symposium on Parallel Architectures, Algorithms and Programming (PAAP'09)*, December 2009. Invited Paper.

[219] Di Wu, Chao Liang, Yong Liu, and Keith Ross. View-Upload Decoupling: A Redesign of Multi-Channel P2P Video Systems. In *Proceedings of the 28th IEEE International Conference on Computer Communications INFOCOM'09*, pages 2726–2730, April 19-25 2009.

[220] Di Wu, Yong Liu, and Keith Ross. Queuing Network Models for Multi-Channel P2P Live Streaming Systems. In *Proceedings of the 28th IEEE International Conference on Computer Communications INFOCOM'09*, pages 73–81, April 19-25 2009.

[221] Haiyong Xie, Yang Richard Yang, Arvind Krishnamurthy, Yanbin Liu, and Avi Silberschatz. P4P: Portal for P2P Applications. *ACM SIGCOMM Computer Communication Review*, 38(4):351–362, October 2008.

[222] Dongyan Xu, Heung keung Chai, Catherine Rosenberg, and Sunil Kulkarni. Analysis of a hybrid architecture for cost-effective streaming media distribution. In *Proceedings of SPIE/ACM Conference on Multimedia Computing and Networking (MMCN'03)*, 2003.

[223] Shuang Yang and Xin Wang 0002. An Incentive Mechanism for Tree-based Live Media Streaming Service. *JNW*, 5(1):57–64, 2010.

[224] Shuang Yang and Xin Wang. A Probabilistic Approach to Analyze the Effect of Free-Riders in P2P Streaming Services. In *International Conference on Network and Parallel Computing (NPC'08)*, volume 5245, pages 387–391. Springer Lecture Notes in Computer Science, October, 18-20 2008.

[225] Xiangying Yang and Gustavo de Veciana. Service Capacity of Peer to Peer Networks. In *Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies (INFOCOM'04)*, volume 4, pages 2242–2252, 2004.

[226] Hongliang Yu, Dongdong Zheng, Ben Y. Zhao, and Weimin Zheng. Understanding user behavior in large-scale video-on-demand systems. *SIGOPS Oper. Syst. Rev.*, 40(4):333–344, apr 2006.

[227] Jiadi Yu, Minglu Li, Feng Hong, and Guangtao Xue. Free-Riding Analysis of BitTorrent-Like Peer-to-Peer Networks. In *Proceedings of the Asia-Pacific Conference on Service Computing, 2006 (APSCC'06)*, pages 534–538, 2006.

[228] Demetris Zeinalipour-yazti and Theodoros Folias. A Quantitative Analysis of the Gnutella Network Traffic. Technical report, Department of Computer Science University of California, 2002.

[229] Xinyan Zhang, Jiangchuan Liu, Bo Li, and Tak-Shing Peter Yum. CoolStreaming/DONet: A Data-driven Overlay Network for Peer-to-Peer Live Media Streaming. In *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'05)*, volume 3, pages 2102–2111, Washington, DC, USA, 2005. IEEE Computer Society.

[230] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph. Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing. Technical Report UCB/CSD-01-1141, UC Berkeley, apr 2001.

[231] Bridge Q. Zhao, John C. S. Lui, and Dah-Ming Chiu. Exploring the optimal chunk selection policy for data-driven P2P streaming systems. In *P2P '09: IEEE International Conference Peer-to-Peer Computing*, pages 271–280, Washington, sep 2009. IEEE Computer Society.

[232] Bridge Qiao Zhao, John C. S. Lui, and Dah-Ming Chiu. Mathematical modeling of incentive policies in p2p systems. In Joan Feigenbaum and Yang Richard Yang, editors, *NetEcon*, pages 97–102. ACM, 2008.

[233] Yipeng Zhou, Dah Ming Chiu, and J.C.S. Lui. A Simple Model for Analyzing P2P Streaming Protocols. In *Proceeding of the IEEE International Conference on Network Protocols (ICNP'07)*, pages 226–235, Beijing, China, October 2007.

[234] Yipeng Zhou, Dah-Ming Chiu, and John C. S. Lui. A simple model for chunk-scheduling strategies in p2p streaming. *IEEE/ACM Transactions on Networking (TON)*, 19(1):42–54, February 2011.

[235] Yipeng Zhou, Tom Z. J. Fu, and Dah Ming Chiu. Statistical modeling and analysis of P2P replication to support VoD service. In *INFOCOM*, pages 945–953. IEEE, 2011.

List of Figures