

Rapport de Thèse Professionnelle

Varvello Matteo

09/2006

« Theoretical Analysis of a Push-to-Peer System »

Société : Thomson Paris Research Lab

Encadrant dans l'entreprise : Christophe Diot

Encadrant académique : Ernst Biersack

Confidential

«Networking»

Institut Eurécom

Contents

1	General description of the Push-to-Peer system	1
1.1	Introduction	1
1.2	Problems and motivations	3
1.3	Definition of the Push-to-Peer	4
2	Related Work	7
2.1	Content distribution	7
2.2	Streaming p2p	8
2.3	Streaming/VoD proxy based	9
3	Analytical model of the Push-to-Peer system	10
3.1	Assumptions and system parameters	10
3.2	Basic model for the push phase - Zero Latency - $D_{up} = 1 - N_w = 1$	13
3.3	Push strategies - $D_{up} = 1 - N_w = 1$	14
3.3.1	The Deterministic Push	15
3.3.2	The Source Coding push	21
3.4	Trick-mode models for the push phase - $D_{up} = 1 - N_w > 1$	24
3.4.1	Zero Latency solution per window	24
3.4.2	Zero Initial Latency solution	27
3.4.3	Zero Latency solution per N_j windows	30
4	Analysis of the results	32
4.1	The Deterministic Push - $D_{up} = 1 - N_w = 1$	33
4.1.1	Balanced arrival pattern	33
4.1.2	Worst arrival pattern	36
4.1.3	Best arrival pattern	37
4.1.4	Comparison: Balanced - Best - Worst arrival pattern	39
4.2	The Source Coding Push - $D_{up} = 1 - N_w = 1$	41
4.3	Comparison: Deterministic Push - Source Coding Push	43

4.4	Trick-mode models for the push phase	
	- $D_{up} = 1 - N_w > 1$	45
4.4.1	Zero Latency solution per window	47
4.4.2	Zero Initial Latency solution	48
4.4.3	Zero latency solution per N_j windows	51
4.4.4	Comparison: Trick-mode models for the push	52
4.5	Centralized vs Push-to-Peer - $D_{up} = 1$	57
5	Conclusion and Future Work	61

Abstract

In this work we study the feasibility and design space of a p2p architecture for a VoD service. The first VoD services relied on complete centralized solutions. These traditional client-server architectures can be overwhelmed by the volume of requests from its clients. With the increasing demand, providers started distributing servers at the edges of the network. The evolution of VoD service toward a distributed architecture is the first step toward p2p.

We propose an innovative architecture for a VoD service called Push-to-Peer, in the context of classical ADSL connections. We focus on a structured architecture, i.e. end-users are peers within a DSLAM. Users store a fraction of the content pushed by the content-server in their local hard-disk. When a peer wants to watch a movie he gets served by other peers in the network. The goal is to offload the content-server by adding capabilities in the end-users.

We define a mathematical model to describe the design space of the system. We consider both cases, the distribution of non-coded and coded blocks. The metric of comparison between the two push techniques is the volume of data to push in each peer.

We show the benefits of a Push-to-Peer architecture compared to a centralized solution. We show we can reduce the load and the cost of the central server by adding some storage capabilities on the users.

Index Terms – Content distribution, Peer-to-Peer, VoD, upload, trick-mode, demand, source-coding.

Résumé

Dans ce travail nous étudions la faisabilité et le "design space" d'une architecture pair-à-pair (p2p) pour un service VoD.

Les premières architectures proposant un service VoD étaient des solutions centralisées. Ces architectures ont la caractéristique de pouvoir être surchargé par un grand nombre de clients demandant du contenu. Pour cela, et suite à la demande croissante de services VoD, les fournisseurs de contenu ont commencé à déployer des serveurs en périphérie du réseau. Cette évolution vers une architecture distribuée était le premier pas vers une architecture complètement p2p.

Dans ce travail, nous proposons une architecture pour un service VoD s'appelant Push-to-Peer, dans le contexte d'utilisateur ayant une connexion ADSL classique. Nous présentons une architecture structurée dans laquelle chaque utilisateur d'un même DSLAM est un pair d'un réseau p2p dédié au service de VoD. Les utilisateurs stockent sur leur disque dur de leur modem une fraction du contenu. Ces fractions de contenu ont été poussées par le serveur de contenu dans une première phase appelé phase "push". Dans une deuxième phase distincte appelé phase "pull", ces fractions de contenus sont servi en mode p2p à des pairs qui veulent regarder un film donné. Ainsi notre approche permet d'alléger la charge du serveur de contenu en ajoutant des capacités aux utilisateurs.

Dans ce document, nous définissons un modèle mathématique décrivant le "design space" du système. Nous nous intéressons au deux cas, 1) la distribution de contenus non-codés et 2) la distribution de contenus codés. La métrique de comparaison entre ces deux techniques est le volume de données à pousser à chaque pair.

Finalement, nous démontrons les avantages de l'architecture push to peer comparée à une solution centralisée. Nous montrons qu'on peut réduire la charge et le prix d'un serveur central en ajoutant de la capacité de stockage sur les utilisateurs.

Chapter 1

General description of the Push-to-Peer system

1.1 Introduction

Internet was born in the sixties as a military project to allow all the members to be connected together in a simple and robust way. The early Internet was used by computer experts, there was nothing friendly about it. There were no personal computers and end-users resources were limited. The network architecture was defined as client-server. Servers are powerful computers, clients are PCs or workstations on which users run applications. Clients rely on servers for resources, such as files, devices, and even processing power.

With the introduction of the World Wide Web and the growth of high speed connections, commercial traffic and users started to appear. The evolution from 56K modems to broadband access intrinsically changed users' interest: the data that could be found in the Web was not enough anymore. This evolution had an impact on the Internet traffic; multimedia content started to be popular.

In 1999-2000 a new application that really matched new users interests was released: Napster [15]. Napster was the original P2P application that popularized the concept to millions of people. Napster hosted a central server which indexed all the files that each Napster user had. In this sense Napster is a hybrid solution. The search of content is done via central server, the content-transfer is done via p2p.

End-users start to have an active role in the Internet. They are both client and server, i.e. "peers". The p2p concept starts to be extremely popular. End-users are attracted by the possibility to have free illegal content, researchers are attracted by the challenges of a new bright architecture.

A p2p architecture is an alternative to a centralized solution. It can be useful to either reduce the load on the central server or to improve the performance of a

system without updating the server. The great strength of a p2p architecture is that it is self-scalable: increasing the number of peers increases also the number of resources. In a centralized scheme the only solution to react to an increase in the demand is to upgrade or to add servers.

Figure 1.1 shows a comparison between a client-server and a p2p architecture. In a client-server architecture all requests are accomplished by the server. According to server capacity a certain demand can be managed. A communication between end-users is allowed only if relayed by the central server. In a p2p architecture each request evolves in a resource for the system. The end-users can directly communicate each other.

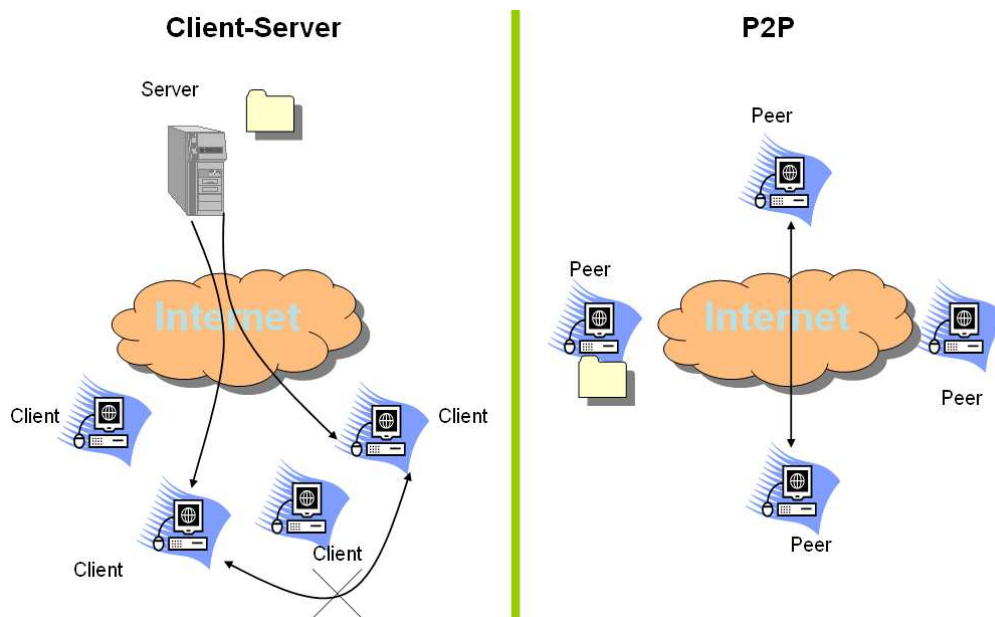


Figure 1.1: Client-server and p2p architecture

Because p2p applications are generally used for the download of illegal content, p2p is seen as an illegal way to communicate in the Internet. However p2p by itself is not illegal, it is a cheap and efficient way to replicate content in a network. Moreover it can also be used to share computation, to detect anomaly in a fast and efficient way. P2P is the most efficient way to exploit the massive resources available in the Internet, giving to all end systems an identical role.

In our work we are interested in how to use a p2p architecture for the distribution of VoD. We propose a solution, called "Push-to-Peer", that reduces the costs when compared to a classical centralized solution leveraging on a p2p architecture.

1.2 Problems and motivations

With the advent of high bandwidth DSL and tripleplay, VoD is becoming a really attractive service.

VoD gives the user the possibility to watch a video or a movie he/she wants when he/she wants. A VoD service enables a user to view the chosen movie while he/she is still downloading it and to watch it at any point in time, i.e. to operate the trick mode.

The initial VoD services relied on centralized solutions where servers and storage equipment were maintained in multiple location. A movie is brought to the customer using satellite or DSL for example. With the increasing demand, providers have to choose between expanding centralized servers or distributing servers at the edges of the network. The state of the art for the VoD is to use distributed servers in distributed VoD networks: in a cluster configuration, the edge servers cache only the most often-requested content for local distribution, while the central servers and library supply other titles, and also distribute new titles to the edge servers according to users behavior and global interest.

This evolution toward a more distributed architecture is the first step toward p2p. It reduces the load on the central server and offloads content on edge servers: this both saves money (load and bandwidth on the central server) and provides a better quality video service.

Anyway this architecture is still costly and has limited scalability because it relies on servers. For this reason we believe that p2p architecture can be an efficient and scalable alternative for the distribution of VoD.

In the last years most of the traffic in p2p systems such as Gnutella, KaZaA, eDonkey and BitTorrent is generated by video files [2]. Historically p2p has been designed to support the distribution of music files. With the evolution in users' connectivity the global interest moved to larger contents, such as games and video [2]. Unfortunately, in such systems the user needs to download the entire content before being able to play the video since they are not supporting VoD capabilities; this means that there can be a big latency before being able to play a video.

The rationale for Push-to-Peer is the following. Most network are unused during night time, this makes push technology attractive to bring content closer to customers. Instead of a server with great connection capability to sustain a large instantaneous demand, we can spread the volume of data to serve in a larger time and when the network is lightly loaded. The goal is both to reduce the costs for the content-provider and to achieve a better network utilization for the ISP.

Memory is generally cheap, so we could easily add some storage capability to users' set-top boxes or gateway: this memory can be used to store the data we push at night. In addition internet gateways are controlled by ISPs, which means they

are more secure than users' PCs.

Our idea is to mix a centralized solution with a p2p architecture: a content-server pushes parts of the content to the users, then the users rely on a p2p architecture to retrieve the missing parts when they want to see a specific movie.

If compared to a centralized solution, we can reduce the load and the cost of the central part by adding some storage capabilities on the users. Moreover we exploit the upload resources of each peer optimizing the bandwidth usage between a DSLAM and the customer when all customers decide to watch a different movie.

1.3 Definition of the Push-to-Peer

The "Push-to-Peer" is a system for distribution of VoD. The system is made of a content-server, a control-server and users, represented by the Internet gateways they have in their apartment, i.e. DSL modems or triple play boxes. The system has two different phases: push and peer.

In the push phase the content-server push the content on several peers. Peers store these data into a local hard disk and they are responsible to serve these data to other users. One single peer does not necessarily have all the content, however if we consider all end-devices together, the entire content is available. i.e. the content is stored in a distributed manner over all end-devices. Peers are located on the same DSLAM, which means we are considering a regular and controlled topology. We can assume that peers are more or less *always on* (as DSL modems are).

The content-server serves several DSLAMs and communication among DSLAMs is allowed. We simply replicate the same movies in several DSLAMs or try to create "clusters" according to users' preferences. In this work we focus on the distribution of a single movie to a single DSLAM. In Figure 1.2 we show the content-server role and the push phase.

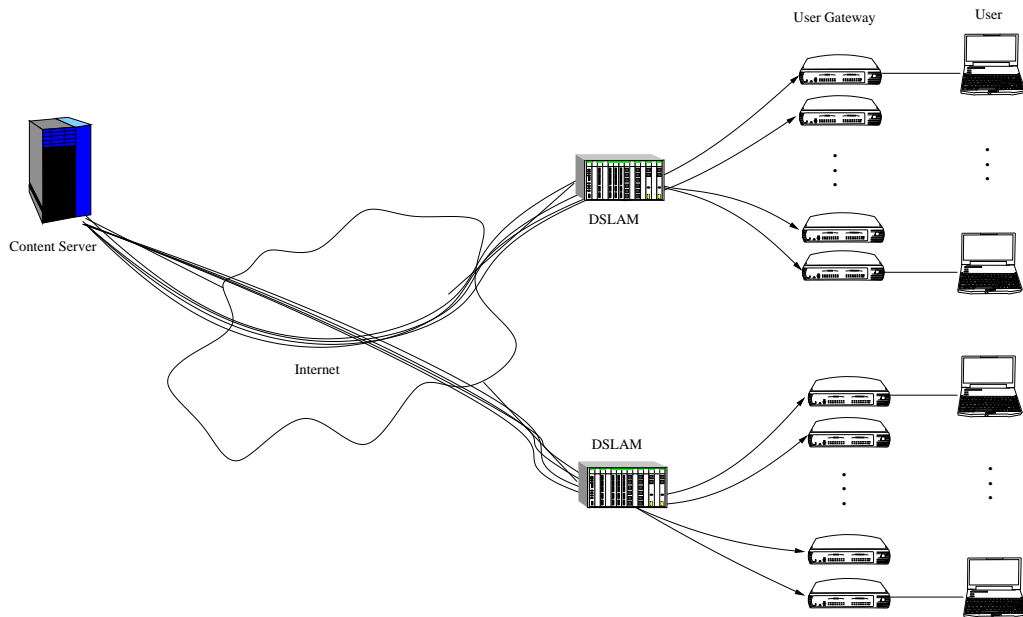


Figure 1.2: The push phase

When a user wants to watch a movie the system works as a p2p system among the gateways. We call this phase the peer phase (see Figure 1.3). The two phases, push and peer are completely distinct phases, i.e. we assume that users request a content only when the push phase for that content is done.

In Push-to-Peer each gateway is equipped with a hard disk to store the content pushed during the push phase. In each DSLAM there are 500-2000 citizens, which means a lot of potential distributed memory. In this work we study how to dimension this local hard disk and how much data we need to push to make the system work. Another issue that we investigate is to define how to realize the push phase in order to achieve the best possible performance during the peer phase.

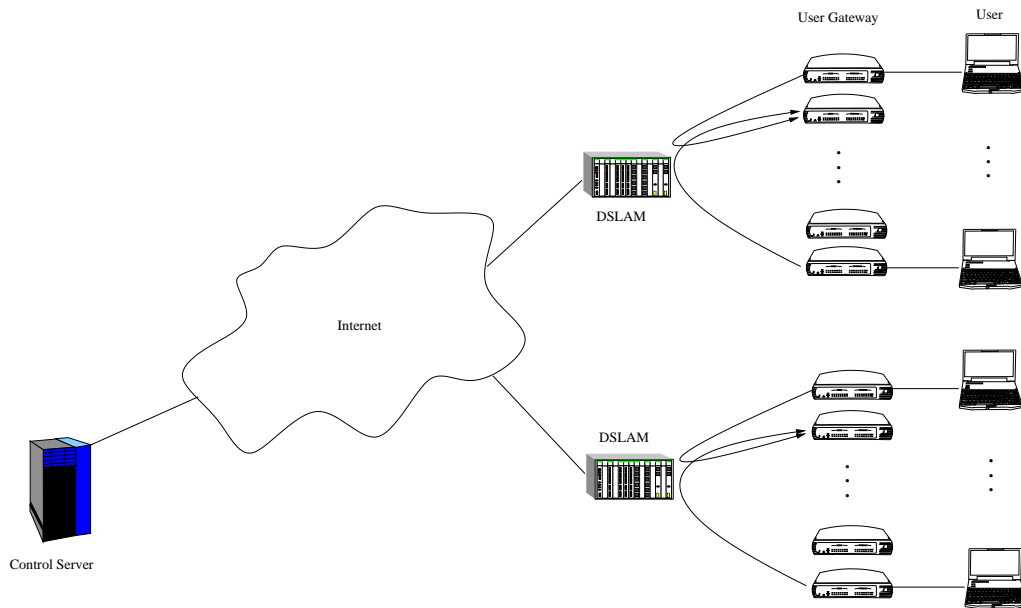


Figure 1.3: The peer phase

The motivations of the Push-to-Peer approach are as follows:

1. Bandwidth for the push phase can be used when it is available or cheap
 2. We can benefit from network locality in the pull-phase
 3. End-devices within the same network will communicate with each other without generating additional traffic on the link that connect the ISP to the Internet.
 4. Distributed storage increase the amount of available content compared to a pure push-solution where all the content is pushed to all end-devices. We also reduce the amount of traffic going from the server to the end-devices
 5. By relying on all end-devices bandwidth and storage, we considerably reduce the content server load compared to a pure server-based solution
-

Chapter 2

Related Work

2.1 Content distribution

Efficient content distribution is an active area of research. Historically content distribution was carried out by single server architectures, which have drawbacks like robustness to failures, bandwidth limitations and processing bottlenecks. To address these limitations the usage of mirrors was proposed: the content is replicated at several servers and the clients have to choose the most appropriate one, e.g. according to the client location or the load of the mirrors.

Content Distribution Networks (CDNs) like Akamai [5] are also based on the idea of replicating the content at several places in the network. More precisely, a CDN is a set of content servers, mirrors and caches placed in the network in order to reduce the clients download time.

Since server/mirror selection and placement is not a trivial task, and since the download performance can highly vary according to the chosen mirror, Rodriguez et al. [12] proposed to download from several mirrors in parallel.

One solution that is becoming popular is to use the unused space and bandwidth of people who download the content from a server to serve more users. Here, apart from the original server, the users requesting the content will also be able to distribute it [3]. This technology is known as P2P-based content distribution. It allows a user or peer to download parts of a file from other peer downloaders. It enables the original content provider to minimize the usage of his resources. BitTorrent is an example of a service that has implemented this technology [4]. Many interesting studies try to describe how BitTorrent works and to identify its strengths and weaknesses [1].

Gkantsidis et al. [7] propose to use network coding with a peer-to-peer content distribution protocol similar to BitTorrent. Each peer is able to produce redundant content based on the content he already downloaded. The authors of [7] claim

that this technique considerably reduces download times compared to the standard BitTorrent protocol.

CoralCDN [13] [6] is a peer-to-peer content distribution network that allows a user to run a web site that offers high performance and meets huge demand. Volunteer sites that run CoralCDN automatically replicate content as a side effect of users accessing it. A peer-to-peer DNS layer transparently redirects browsers to nearby participating cache nodes, which in turn cooperate to minimize load on the origin web server.

2.2 Streaming p2p

With the widespread of broadband accesses, multimedia services are getting increasingly popular among users and have contributed to a significant amount of today's Internet traffic. The first multimedia services relied on centralized solution: a web server answering to end-users requests. Unfortunately internet servers and internet channels often get overloaded, resulting in users experiencing poor performance. Moreover the growth of interests in multimedia service makes the situation always more critical every day.

Many multimedia applications involve live media streaming from a source to a large population of users. For these applications, IP Multicast is probably the most efficient vehicle; its deployment however remains confined due to many practical and political issues. Researchers thus have resorted to application-level solutions advocating a tree structure for data delivering. Anyway this model poorly matches an application level overlay with dynamic nodes. In [8] a data-driven overlay network is proposed. It is the availability of data that guides the flow directions, while not a specific overlay structure that restricts the flow directions.

PPStream [9] is an example of a P2PTV¹. The principle of the p2p Streaming Internet TV is similar to BitTorrent. The more users are online, the faster the programs can be loaded. What was considered an issue for a centralized solution becomes a strength for a p2p architecture.

While having great potential this technology has still some issues:

- No QoS. Compared to unicast no one can guarantee a reliable stream, since every user is a rebroadcaster. Each viewer is a part of a chain who can have negative influence on the reliability of the stream.
- To bypass the QoS issue, usual P2PTV applications rely on multiple peers sending multiple traffic to other peers, introducing extra data overhead for

¹The term P2PTV refers to peer-to-peer software applications designed to redistribute video streams on a p2p network, typically TV stations across the world

retransmits, communication and redundancy (up to 40 percent).

2.3 Streaming/VoD proxy based

Video on demand (VOD) systems allow users to select and watch video content over a network as part of an interactive television system. VOD systems either "stream" content, allowing viewing while the video is being downloaded, or "download" it in which the program is brought in its entirety to a set-top box before viewing starts.

All download and some streaming video on demand systems provide the user with a large subset of VCR functionality including pause, fast forward, fast rewind, slow forward, slow rewind, jump to previous/future frame etc (know as trick-mode). For streaming systems this requires more effort on the part of the server, and may also require greater network bandwidth.

The first VoD services were relying on a complete centralized solution, but in order to have a scalable solution content-provider started to distribute servers at the edges of the network. Providing a scalable and efficient VoD service to a wide client population has been extensively investigated in several works; the basic idea to achieve scalability is to serve multiple clients via multicast. An interesting cost-model for different VoD architecture can be found in [12].

Proxy caching is an interesting solution for web systems. The information they need to cache is typically pretty small (a size of 1K to 100K), so the decision at the proxy is easy: either caching an object in its entirety or not caching. Streaming media are big size object (a one-hour standard MPEG-1 video has a volume of about 675 MB); caching it entirely at a web proxy is clearly impractical, as several such large streams would exhaust the capacity of the cache. An interesting analysis on streaming media caching can be found in [11].

Chapter 3

Analytical model of the Push-to-Peer system

3.1 Assumptions and system parameters

We make the assumptions for the design of the Push-to-Peer system:

- Peers are *always* on line. It is very unlikely for a peer to disconnect his gateway (modem, triple play box) from the network, although there is some failure either of the power line or of the gateway itself.
- Each peer dedicates a fixed percentage of his upload bandwidth to the VoD service (Push-to-Peer). It is defined as B_{up} .
- Each peer has D_{up} upload slots and D_{down} download slots. We consider $D_{up} = 1$, which means a remote peer can only serve one peer at a time. In the analytical model we don't consider any limitations on D_{down} deriving from the TCP/IP stack implementation.
- There are N gateways connected to a DSLAM. The size of DSLAMs varies today between 200 up to 2000 clients. A typical value for a DSLAM is $N = 800$ users.
- A control-server knows exactly where all the chunks are located and the state of all peers, i.e. from which peers a peer is downloading from and to whom he is uploading to. Moreover we assume an a priori knowledge of the popularity of movies.
- During the peer phase each active peer gets a percentage of service $A = \frac{M}{m}$. So since each peer dedicates B_{up} for the VoD service the expected download rate is: $R_t = B_{up} * A$

- We consider the distribution of a single movie of size L consisting of L non-overlapping blocks, such that $\bigcup_{i=1}^L C_i = L$. We call a block “chunk“. Each chunk is unique and of size V_{ch} . Typical parameters for an MPEG-4 good quality video are:
 - Rate: $R_v = 1.5Mbps$
 - Size: $L = 1GB$
 - Length: $T \cong 90mins$
- The content L is divided in N_w windows. Over a window we define a specific push strategy. Each window has size equal to V_w and it is composed by $\frac{L}{N_w}$ non-overlapping blocks.

In table 3.1 and 3.2 we summarize all parameters which define the analytical model.

Table 3.1: Table of parameters - first half

NAME	DESCRIPTION
A	percentage of service each peer can receive
B_{down}	available download bandwidth for a peer
B_{up}	available upload bandwidth for a peer
D_{up}	maximum number of upload slots
D_{down}	maximum number of download slots
L	size of a movie in byte
N	number of gateways per DSLAM
N_s	number of sets of chunks
N_j	number of points a user can jump to
N_w	number of windows in a movie of size L
N_w^o	optimum number of windows in a movie of size L
n_a	number of active peers in the system at the same time
\widehat{n}_a	global demand during a time T_{ss}
R_d	aggregate download rate achieved among sets
R_s	download rate achieved per set
R_S	server rate
R_S^{P2P}	server rate within a centralized architecture
R_S^C	server rate within a Push-to-Peer architecture
R_t	target download rate of n_a active peers
R_v	video rate

Table 3.2: Table of parameters - second half

T	duration of a movie
T_{ch}	viewing time of a chunk
T_d	total available download time
T_j	length of a back or forward jump in time
T_l	average latency of a movie in the trick-mode
T_p	viewing time of chunks pushed as prefix
T_s	viewing time of chunks a peer serves
T_{ss}	time in which the global demand is \widehat{n}_a
T_{sd}	duration of the push phase
T_{sd}^C	duration of the push phase with a centralized architecture
T_{sd}^{p2p}	duration of the push phase with a Push-to-Peer architecture
T_s'	serving time of a set at a rate R_s
T_w	viewing time of a window
T_{se}^w	time to serve V_s^w within a window
V_c	volume of coded data
V_{ch}	size of a chunk
V_d	volume of data a peer download
V_m	total volume of data missed by a peer
V_p	size of the prefix pushed to a peer when $N_w = 1$
V_s	total volume of data served by a peer
V_w	size of a window
V^P	total volume of data to push
V_m^w	volume of data missed by a peer per window
V_p^w	length of the prefix per window
V_s^w	volume of data served by a peer per window
$V_m^{w'}$	volume of data missed by a peer in the first window
$V_p^{w'}$	length of the prefix in the first window
$V_s^{w'}$	volume of data served by a peer in the first window
X_r	chunks replication factor

3.2 Basic model for the push phase - Zero Latency

$$- D_{up} = 1 - N_w = 1$$

We present here the basic model for the push phase. We introduce several parameters to describe the content L . We are considering a video-content distribution, which means that chunks increase their priority according to the viewing time. We dimension the system for a zero-latency solution: a user can start to watch the movie as soon as he wants. We consider $N_w = 1$, i.e. a single window of size $V_w = L$.

In Figure 3.1 we see how chunks that made up a content can be organized during the push

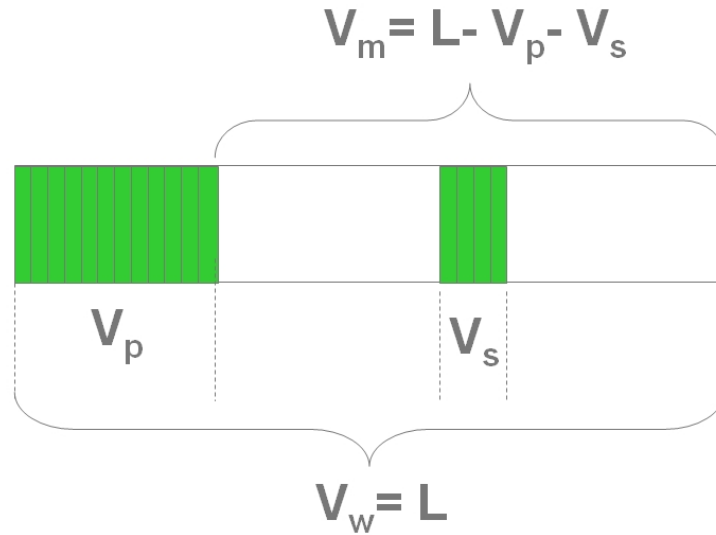


Figure 3.1: Basic chunks organization

Figure 3.1 shows how chunks are organized within a window:

- *Prefix*: we call V_p the volume of data the content-server pushes as a prefix per window. It is made up of V_p chunks that are pushed in the same way among all the peers.
- *Serving*: we call V_s the volume of data each peer serves per window. Size and organization of V_s depend on the push strategy we adopt.
- *Missing*: we call V_m the volume of data each peer misses per window.

3.3 Push strategies - $D_{up} = 1$ - $N_w = 1$

We present an analysis of two different push techniques considering a chunks organization as in Figure 3.1. The impact of the push strategy is in the amount of data each peer serves V_s and in the performance of the peer phase. We derive the analytical formulation of the system parameters V_p , V_s and V_m for each specific push technique adopted.

The first push strategy we discuss is the ‘‘Deterministic Push’’. The content-server spreads the original $\bigcup_{i=1}^L C_i = L$ chunks among peers deciding a priori where to push. When a peer $peer_x$ wants to watch a video, he gets V_m chunks from a limited number of peers. These peers are assigned by the control-server (peer-selection) considering which chunks $peer_x$ misses (piece-selection). These peers will serve $peer_x$ the entire V_m . The consequence is that an active peer needs a specific subset of k peers that serve him according to the piece-selection. We call this push strategy ‘‘deterministic push’’ because the content-server needs to choose where to push, i.e. to select peers in a deterministic way.

The second strategy we investigate is the ‘‘source coding push’’. The content-server doesn’t push the original $\bigcup_{i=1}^L C_i = L$ chunks of a movie, but some coded data. An active peer needs a random subset of k peers to reconstruct the content. The goal is to reduce the effect of the piece selection on the peer selection, increasing the flexibility in the choose of the peers we upload from. The drawback is the overhead introduced by the code, both in physical data to push and in the complexity of coding-decoding operations.

The system is dimensioned starting from a given value of the demand n_a . Starting from the demand we can define another parameter A , defined as the maximum percentage of service an active peer can get. Generally $A(n_a, N)$ assumes values in a continuous field and can be defined as the ratio between available resources and service requests:

$$A = \frac{N}{n_a} \quad (3.1)$$

Since we consider $D_{up} = 1$, the parameter A is also equal to the number of peers an active peer can get service from. To be more precise $\lfloor A \rfloor$ is the number of ‘‘sources’’ an active peer can have. Anyway in the analysis we consider A as an integer, so $A = \lfloor A \rfloor$. We introduce the parameter R_t defined as the maximum download rate an active peer can achieve:

$$R_t = A * B_{up} \quad (3.2)$$

We define T_d as the available download time. T_d is the time an active peer has

during the peer phase to retrieve his missing blocks. It depends on the prefix size V_p and on the push strategy adopted.

3.3.1 The Deterministic Push

The content-server divides the N peers in the system in $(A + 1)$ sets. In each set i there are about $\frac{N}{(A+1)}$ peers who are responsible to serve V_s chunks. At the same time the replication factor of the above V_s chunks is $X_r = \frac{N}{(A+1)}$. This division in sets allow to say that each peer in a set has A peers as neighbors in the remaining A sets. Moreover we can define in a general way the value of V_s :

$$V_s = \frac{(L - V_p)}{A + 1} \quad (3.3)$$

We introduce also a general expression for V_m :

$$V_m = L - V_p - V_s = \frac{(L - V_p)}{(A + 1)} * A \quad (3.4)$$

In Figure 3.2 we see the organization of chunks after a deterministic push where $N = 12$, $n_a = 4$ and so $A = 3$.

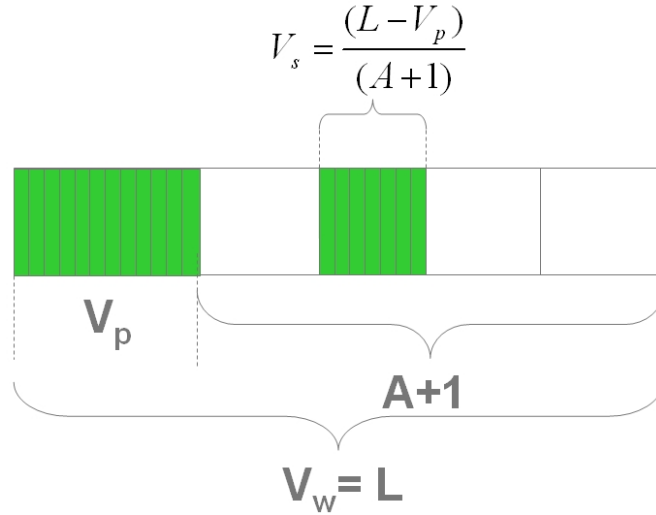


Figure 3.2: The Deterministic Push - $D_{up} = 1$ ($N = 12$, $n_a = 4$, $A = 3$)

We compute V_p , V_s and V_m considering the following arrival patterns:

- *Balanced arrival pattern:* the n_a active peers access the system at the same time, but requests are well spread among the $(A + 1)$ sets.
- *Worst arrival pattern:* the n_a active peers access the system at the same time and all requests come from the same sets.
- *Best arrival pattern:* the n_a active peers don't access the system at the same time. We define a global demand \widehat{n}_a the system sees during a time interval T_{ss} . We define a specific arrival sequence that either maximizes the system utilization and minimize $V_p + V_s$.

Deterministic Push - balanced arrival pattern

We derive the results for V_p , V_s and V_m considering a balanced arrival pattern. The value for the instantaneous demand in the system is equal to n_a and requests of service are coming from the maximum number of sets. The maximum number of requests per set is $\lceil \frac{n_a}{(A+1)} \rceil$. As before we remove any issues of the integer problem and we simply consider $\frac{n_a}{(A+1)}$.

According to the arrival we have defined we can state that during the pull phase each requesting peer achieves an average download rate $R_d = A * B_{up}$ among the $(A + 1)$ groups; this means a rate equal to B_{up} per group.

We give here an expression of the available download time T_d :

$$T_d = T_p + T_s - T_{ch} \cong T_p + T_s = \frac{V_p}{R_v} + \frac{V_s}{R_v} \quad (3.5)$$

In formula 3.5 we see three components:

- T_p is the viewing time of the prefix. During T_p an active peer can start the p2p phase while he is watching the first part (prefix) of the movie.
- T_s is the viewing time of a set. We need to ensure that an active peer finishes to download all the missing blocks of the first set before to play the last chunk of the first set. Since an active peer retrieves the missing blocks in parallel from all the $(A + 1)$ groups, the linear order is exploited only within chunks of the first set. The target rate $R_t = A * B_{up}$ is achieved only as an aggregate rate ($R_d = A * B_{up}$) among sets. The download rate per set results equal to $R_s = B_{up}$.
- T_{ch} is the viewing time of a chunk. We consider to exploit the linear order on the chunks within a set. While the user plays V_s he is also downloading

these chunks. The little unit of data we can show is a chunk. We cannot allow to download the last chunk while we view it. In formula 3.5 we simplify this value, since it is small compared with the other terms.

The time to retrieve the first set among the $(A + 1)$ sets (T'_s) must be equal to the available download time T_d defined in formula 3.5. So we introduce formula 3.6:

$$T'_s = T_p + T_s \Rightarrow \frac{V_s}{B_{up}} = T_p + T_s \quad (3.6)$$

In the left side of equation 3.6, we have the time a remote peer needs to upload his V_s at a speed B_{up} . We can substitute V_s with the expression defined in 3.3:

$$\frac{V_s}{B_{up}} = T_p + T_s \Rightarrow \frac{(L-V_p)}{B_{up}} = T_p + T_s \quad (3.7)$$

Finally we obtain an expression for V_p :

$$V_p = L * \frac{(R_v - B_{up})}{R_v + A * B_{up}} \quad (3.8)$$

In formula 3.8 we see that in case $B_{up} \geq R_v$ we can get $V_p \leq 0$; we need to add the condition that if $B_{up} \geq R_v$ $V_p = 0$. So we have that:

$$V_p = \begin{cases} L * \frac{(R_v - B_{up})}{R_v + A * B_{up}} & \text{if } B_{up} \leq R_v \\ 0 & \text{otherwise} \end{cases} \quad (3.9)$$

Starting from 3.9 and from 3.3 we can compute the expression of V_s :

$$V_s = \begin{cases} L * \frac{B_{up}}{R_v + A * B_{up}} & \text{if } B_{up} \leq R_v \\ \frac{L}{(A+1)} & \text{otherwise} \end{cases} \quad (3.10)$$

Now that we have an expression for V_p and V_s we can derive an expression for V_m considering equation 3.4:

$$V_m = \begin{cases} L * \frac{B_{up} * A}{R_v + A * B_{up}} & \text{if } B_{up} \leq R_v \\ L * \left(\frac{A}{(A+1)}\right) & \text{otherwise} \end{cases} \quad (3.11)$$

The Deterministic Push - worst arrival pattern

In the worst arrival pattern the value for the instantaneous demand in the system is equal to n_a and requests of service come from the minimum number of sets. In each set there are $\frac{N}{(A+1)}$ peers, so requests are coming from $\lceil \frac{n_a}{\frac{N}{(A+1)}} \rceil$ sets.

We explain here why this arrival pattern can be defined as the worst one. According to what we said the first $\frac{N}{A+1} \leq n_a$ active peers are all members of the same set i . As a consequence they monopolize all the resources from the remaining A sets. So the remaining $(n_a - \frac{N}{A+1})$ requests can be satisfied only by the members of set i getting worst performance respect to the target rate $R_t = A * B_{up}$.

Considering this arrival pattern the maximum number of requests we can serve at a rate R_t is reduced to $\frac{N}{A+1}$. Since our goal is to serve the worst arrival pattern made up of $\frac{N}{A}$ peers we need to consider as the demand is $\frac{N}{A-1}$. The consequence is that we need to increase the size of the prefix V_p respect to 3.9.

Starting from these considerations we can define the formula for a deterministic push with the worst arrival pattern:

$$V_p = \begin{cases} L * \frac{(R_v - B_{up})}{R_v + (A-1) * B_{up}} & \text{if } B_{up} \leq R_v \\ 0 & \text{otherwise} \end{cases} \quad (3.12)$$

$$V_s = \begin{cases} L * \frac{B_{up}}{R_v + (A-1) * B_{up}} & \text{if } B_{up} \leq R_v \\ \frac{L}{A} & \text{otherwise} \end{cases} \quad (3.13)$$

$$V_m = \begin{cases} L * \frac{B_{up} * A}{R_v + (A-1) * B_{up}} & \text{if } B_{up} \leq R_v \\ L * (\frac{A-1}{A}) & \text{otherwise} \end{cases} \quad (3.14)$$

The Deterministic Push - best arrival pattern

The deterministic push is based on the creation of $(A + 1)$ sets in order to serve a peak demand of n_a users. The control server assigns to each requesting peer A peers, one for each remaining set. According to this scheme an user achieves an aggregate download rate $R_d = \sum_{i=1}^A R_s = A * B_{up}$.

If the value of the instantaneous demand is smaller than n_a the control-server can assign A servers from the same set. This behavior allows to exploit the linear order both among chunks and groups. The consequence is that the aggregate download rate stays $R_d = A * B_{up}$, but the rate per set becomes equal to $R_s = A * B_{up}$.

We introduce the parameter \widehat{n}_a representing the demand the system sees during a session-time T_{se} . The content-server divides the peers in $(A + 1)$ sets, this means that there are $\frac{N}{A+1}$ servers for each set. According to a parallel download equal to $A = \frac{N}{n_a}$ we can define the number of requests n_a that can be served in parallel per group ensuring a target rate $R_t = A * B_{up}$:

$$n_a = \frac{N}{\frac{A+1}{A}} \quad (3.15)$$

After a time $T_s = \frac{V_s}{A * B_{up}}$ a group of n_a peers has finished to download the missing chunks for set i , so they can move to be served by peers in set $i + 1$ exploiting the linear order among groups/chunks. Which means that other n_a peers can enter the system and retrieve the missing chunks for set i .

We can define the arrival rate λ_s of the requesting peers:

$$\lambda_s = n_a * T_s = \frac{N}{\frac{A+1}{A}} * \frac{V_s}{A * B_{up}} \quad (3.16)$$

From 3.16 we can get also the value of T_{ss} to serve a demand \widehat{n}_a :

$$T_{ss} = \frac{\widehat{n}_a}{\lambda_s} \quad (3.17)$$

We notice that if we consider an instantaneous demand equal to n_a , a solution like this is really bad performing: the n_a requesting peers would collide on the same chunks achieving very poor performance.

According to this pattern we can achieve a download rate per set equal to $R_s = A * B_{up}$. We give here an expression of the available download time T_d :

$$T_d = T_p + T_m = T - T_s \quad (3.18)$$

The length of a movie T can be decomposed into three different times: T_p , T_s , T_m . The viewing time of the prefix T_p as usual is used to retrieve missing chunks. Since we completely exploit the linear order also T_m must be considered as an useful time: as soon as an active peer gets a chunk he can play it. The viewing time of the serving chunks T_s cannot be considered to contribute to the increase of T_d , since each peer serves chunks in a different position within L .

The time to retrieve the A missing sets has to be equal to the available download time T_d of 3.18. So we introduce formula 3.19:

$$\frac{V_s * A}{A * B_{up}} = T - T_s \quad (3.19)$$

Finally we obtain an expression for V_p :

$$V_p = L * \frac{(R_v - A * B_{up})}{R_v + B_{up}} \quad (3.20)$$

In formula 3.20 we see that in case $A * B_{up} \geq R_v$ we can get $V_p \leq 0$; we need to add the condition that if $A * B_{up} \geq R_v$ $V_p = 0$. So we have that:

$$V_p = \begin{cases} L * \frac{(R_v - A * B_{up})}{R_v + B_{up}} & \text{if } A * B_{up} \leq R_v \\ 0 & \text{otherwise} \end{cases} \quad (3.21)$$

Starting from 3.20 and from 3.3 we can compute the expression of V_s :

$$V_s = \begin{cases} L * \frac{B_{up}}{R_v + B_{up}} & \text{if } A * B_{up} \leq R_v \\ \frac{L}{(A+1)} & \text{otherwise} \end{cases} \quad (3.22)$$

Now that we have an expression for V_p and V_s we can derive an expression for V_m considering equation 3.4:

$$V_m = \begin{cases} L * \frac{B_{up} * A}{R_v + B_{up}} & \text{if } A * B_{up} \leq R_v \\ L * \frac{A}{(A+1)} & \text{otherwise} \end{cases} \quad (3.23)$$

3.3.2 The Source Coding push

The content-server push a prefix V_p in the same way among the N peers. Then he generates coded blocks by the $(L - V_p)$ remaining chunks. These blocks are distributed among peers and constitute V_s .

Many papers [16] [17] show that if the data from which we generate the coded blocks is large enough it is feasible to generate about an infinite number of different chunks. These codes are therefore called rateless codes.

In order to be sure to serve a maximum demand equal to n_a with a target rate $R_t = A * B_{up}$, the coded blocks are spread in $(A + 1)$ sets. This means each active peer needs at least A distinct peers to reconstruct V_m . In Figure 3.3 we see how we organize the chunks of a content if we consider a source coding push:

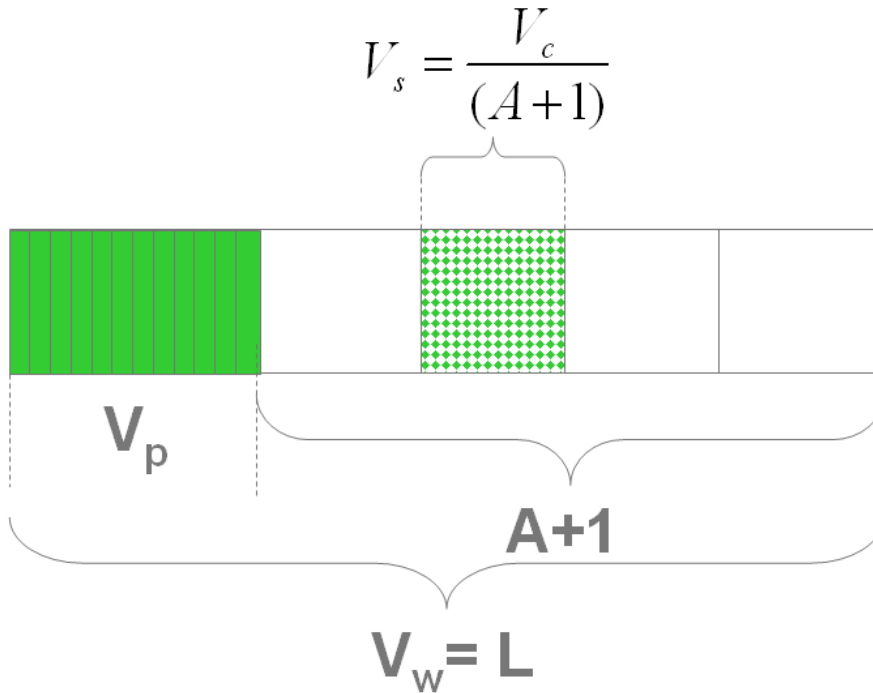


Figure 3.3: The Source Coding Push - $D_{up} = 1$ ($N = 12$, $n_a = 4$, $A = 3$)

The content-server pushes a prefix V_p in the same way among all the peers. On the remaining $(L - V_p)$ chunks the content-server adopts code. When coding is used an overhead is introduced. This overhead can be quantified by a parameter called inefficiency ratio, ε . The ratio means that each client needs $(1 + \textit{epsilon}) * (L - V_p)$ to reconstruct the original content $(L - V_p)$.

We can say that starting from $(L - V_p)$ the content server can generate a total volume of coded blocks equal to:

$$V_c = (L - V_p) * N * (1 + \varepsilon) \quad (3.24)$$

The addition of source coding allows to remove any issue related to the piece selection: a peer is only interested in finding A peers who can serve him. The notion of set is no more related to a selection of chunks, but it is only used to define the minimum amount of data each peer needs to serve such that the service provided by A remote peers is enough to reconstruct the original data.

After the expansion introduced by the code (see 3.24) we define in 3.25 V_s and V_m in function of V_p :

$$\begin{cases} V_s = \frac{V_c}{(A+1)} \Rightarrow V_s = \frac{(L-V_p)*(1+\varepsilon)}{(A+1)} \\ V_m = V_s * A \Rightarrow V_m = A * \frac{(L-V_p)*(1+\varepsilon)}{(A+1)} \end{cases} \quad (3.25)$$

The drawback of the source coding push is that an active peer needs to download all the missing blocks within a window before he is able to reconstruct and view the original content. The consequence is that we cannot exploit at all the linear order among chunks. In 3.26 we see the value for the available download time:

$$T_d = T_p \Rightarrow T_d = \frac{V_p}{R_v} \quad (3.26)$$

Putting together the available download time (T_d) and the volume of data we need to download (V_m) we can define the equation:

$$T_d = \frac{V_m}{B_{up} * A} \Rightarrow \frac{V_p}{R_v} = \frac{(L - V_p) * (1 + \varepsilon)}{A + 1} * A * \frac{1}{B_{up} * A} \quad (3.27)$$

The previous equation says that during T_p we get in parallel from A peers at the dedicated speed B_{up} the volume of coded data each peer serves and that we need in order to reconstruct the original content. This equation allows to compute V_p :

$$V_p = \frac{R_v * L * (1 + \varepsilon)}{(A + 1) * B_{up} + R_v(1 + \varepsilon)} \quad (3.28)$$

Starting from V_p we derive the formula for V_s and V_m considering 3.25:

$$V_s = L * \frac{(1 + \varepsilon) * B_{up}}{(A + 1) * B_{up} + R_v * (1 + \varepsilon)} \quad (3.29)$$

$$V_m = A * L * \frac{(1 + \varepsilon) * B_{up}}{(A + 1) * B_{up} + R_v * (1 + \varepsilon)} \quad (3.30)$$

3.4 Trick-mode models for the push phase

$$- D_{up} = 1 - N_w > 1$$

An important feature of a VoD service is the trick-mode. The user must be given the possibility to pause, rewind and fast forward the movie he is watching. Users use to sample movies by scanning through them, much like the intro-sampling mode supported by portable CD-player [14]. This means that the linear priority of chunks can be modified in a discontinuous linear priority.

These idea leads us to think about an organization of chunks in windows. Each window represents a different point in time of the movie. The number of windows (N_w) depends on the flexibility we want to give to the users to jump to several points in a movie.

We analyze several models for the push when $N_w > 1$. The interesting feature of the multiple windows is the possibility to better exploit the linear priority among chunks. As soon as an active peer has a window, he can play it. This results is independent on the push strategy we consider. So according to the previous results we focus on a Deterministic Push strategy. Moreover we consider the hypothesis of a balanced arrival pattern.

3.4.1 Zero Latency solution per window

We start analyzing the basic model for the push when $N_w > 1$. In Section 3.2 we defined the values of V_p , V_s , V_m to have a continuous view of the movie with an initial latency equal to zero. This scheme relayed on $N_w = 1$.

The logical extension of this basic schemes considering $N_w > 1$ foresees to share V_p , V_s , V_m among N_w windows. We introduce and define the parameters:

$$\left\{ \begin{array}{l} V_p^w = \frac{V_p}{N_w} \\ V_s^w = \frac{V_s}{N_w} \\ V_m^w = \frac{V_m}{N_w} \end{array} \right. \quad (3.31)$$

Figure 3.4 shows how we can map a Deterministic Push with $N_w = 1$ into a Deterministic Push with $N_w = 3$ using the depicted scheme.

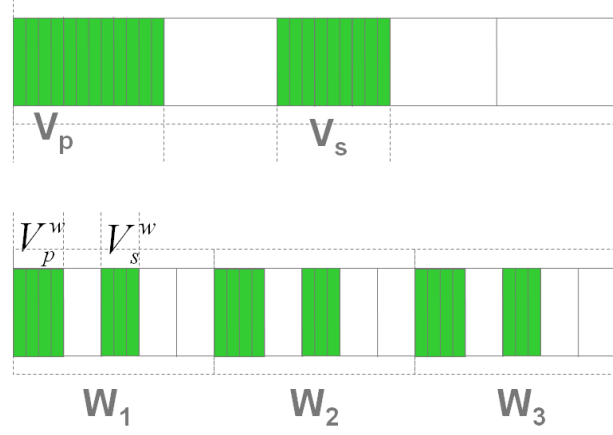


Figure 3.4: Deterministic Push $N_w = 1$ - Deterministic Push $N_w = 3$ - Zero latency

The scheme of Figure 3.4 is a zero-latency solution per window. An user can jump in N_w points in time of the movie and play the movie without any waiting.

We propose the formula for V_p^w , V_s^w , V_m^w for a Deterministic Push with balanced arrival pattern, simply derived by section 3.3.1:

$$V_p^w = \begin{cases} \frac{L}{N_w} * \frac{(R_v - B_{up})}{R_v + A * B_{up}} & \text{if } B_{up} \leq R_v \\ 0 & \text{otherwise} \end{cases} \quad (3.32)$$

$$V_s^w = \begin{cases} \frac{L}{N_w} * \frac{B_{up}}{R_v + A * B_{up}} & \text{if } B_{up} \leq R_v \\ \frac{L}{(A+1)} & \text{otherwise} \end{cases} \quad (3.33)$$

$$V_m^w = \begin{cases} \frac{L}{N_w} * \frac{B_{up}}{R_v + A * B_{up}} * A & \text{if } B_{up} \leq R_v \\ \frac{L}{N_w} * \frac{A}{(A+1)} & \text{otherwise} \end{cases} \quad (3.34)$$

The number of windows N_w is always limited. Each peer needs to serve at least one chunk per window, i.e. we need to respect: $\frac{V_s}{V_{ch}} \geq N_w$. As a consequence if we consider an input value for N_w , V_s can't never be smaller then $V_s = N_w * V_{ch}$.

Starting from the scheme of Figure 3.4 we derive a general expression for the total available download time T_d as a function of N_w (balanced arrival):

$$\frac{T_p}{N_w} + \frac{T_s}{N_w} - T_{ch} + \sum_{i=2}^{N_w} \left(\frac{T_m}{N_w} + \frac{T_p}{N_w} + \frac{T_s}{N_w} - T_{ch} \right) \quad (3.35)$$

Computing the summation we obtain:

$$T_d = T_p + T_s - N_w * T_{ch} + (N_w - 1) * \frac{T_m}{N_w} \quad (3.36)$$

If in formula 3.36 we set $N_w = 1$ we find again the result of formula 3.5. We compute T'_d and find the value of N_w which maximizes T_d :

$$\begin{cases} T'_d = -T_{ch} + \frac{T_m}{N_w} - \frac{T_m}{N_w^2} * (N_w - 1) \\ T'_d = 0 \end{cases} \Rightarrow N_w^o = \sqrt{\frac{V_m}{V_{ch}}} \quad (3.37)$$

In the base scheme we derived a formula per T_d considering an approximation on the term T_{ch} (see formula 3.5). In formula 3.37 we computed N_w^o without any approximation. Simplifying the value of T_{ch} , equation 3.37 has no more a maximum. In the next analysis we will do this simplification in order to obtain a clearer description of system's parameters. So we don't expect N_w to affect our results.

3.4.2 Zero Initial Latency solution

The model described in section 3.4.1 allows us to understand which improvements we can have thanks to a multiple-windows scheme. In the model described above, the available download time T_d for a user in a window i is equal to:

$$T_d = \begin{cases} T_p^w + T_s^w - T_{ch} + T_m^w \cong T_p^w + T_s^w + T_m^w & \text{for } i = 2 \dots N_w \\ T_p^{w'} + T_s^{w'} - T_{ch} \cong T_p^{w'} + T_s^{w'} & \text{if } i = 1 \end{cases} \quad (3.38)$$

We simplify the term T_{ch} in formula 3.38, since it is small respect to the other terms.

Here we define a push model where all windows i with $i = 2 \dots N_w$ are defined by V_p^w , V_s^w and V_m^w . For $i = 1$ we introduce the parameters $V_p^{w'}$, $V_s^{w'}$ and $V_m^{w'}$. Figure 3.5 illustrates the model.

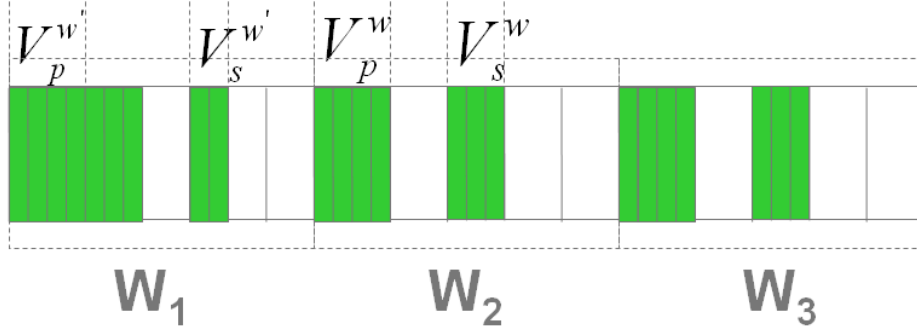


Figure 3.5: Deterministic Push - $N_w = 3$ - Zero Initial latency

We first compute the equations describing a window i for $i = 2 \dots M$. Then we obtain the parameters describing the first window. We are considering a Deterministic Push with balanced arrival.

Within a window i for $i = 2 \dots M$ each active peer gets V_s^w from A peers. As usual for the Deterministic Push $V_s^w = \frac{L}{(A+1)}$. Since we consider a balanced arrival pattern the download rate per set achieved is $R_s = B_{up}$.

Equation 3.39 shows an expression of T_{se}^w :

$$T_{se}^w = \frac{V_s^w}{R_s} \Rightarrow \frac{(\frac{L}{N_w} - V_p^w)}{(A+1) * B_{up}} \quad (3.39)$$

Putting together 3.38 for $i = 2 \dots N_w$ and 3.39 we define equation 3.40:

$$T_{se}^w = T_p^w + T_s^w + T_m^w \Rightarrow \frac{(\frac{L}{N_w} - V_p^w)}{(A+1) * B_{up}} = \frac{V_p^w}{R_v} + \frac{(\frac{L}{N_w} - V_p^w)}{(A+1) * R_v} + \frac{A * (\frac{L}{N_w} - V_p^w)}{(A+1) * R_v} \quad (3.40)$$

Finally we obtain an expression for V_p^w :

$$V_p^w = \frac{L}{N_w} * \frac{(R_v - B_{up} * (A+1))}{R_v} \quad (3.41)$$

In formula 3.41 we see that if $(A+1) * B_{up} \geq R_v$ we can get $V_p \leq 0$. So we add the condition $(A+1) * B_{up} \geq R_v$, $V_p = 0$. Formula 3.42 shows the expression for V_p^w :

$$V_p^w = \begin{cases} \frac{L}{N_w} * \frac{(R_v - B_{up} * (A+1))}{R_v} & \text{if } (A+1) * B_{up} \leq R_v \\ 0 & \text{otherwise} \end{cases} \quad (3.42)$$

Starting from 3.42 and 3.3 we can compute the expression of V_s^w :

$$V_s^w = \begin{cases} \frac{L}{N_w} * \frac{B_{up}}{R_v} & \text{if } (A+1) * B_{up} \leq R_v \\ \frac{L}{N_w} * \frac{1}{(A+1)} & \text{otherwise} \end{cases} \quad (3.43)$$

Now that we have an expression for V_p^w and V_s^w we can derive an expression for V_m^w considering equation 3.4:

$$V_m^w = \begin{cases} \frac{L}{N_w} * \frac{A * B_{up}}{R_v} & \text{if } (A+1) * B_{up} \leq R_v \\ \frac{L}{N_w} * \frac{A}{(A+1)} & \text{otherwise} \end{cases} \quad (3.44)$$

All windows i with $i = 2 \dots N_w$ are defined by the previous parameters. The innovative point for the multiple windows scheme is that each window can rely on the time T_m^w of the previous window. Of course the first window cannot rely on a previous window. As a consequence the previous parameters are not enough for the first window, but we need to define a specific push.

In the first window we need to ensure that the download of the missing blocks for window 2 leverages on T_m^w seconds in which we show data from the first window. So the value of T_d for the first window results equal to $T_d = T_w - T_m^w$. We can introduce formula 3.45:

$$T_{se}^{w'} = T_w - T_m^w \Rightarrow \frac{(\frac{L}{N_w} - V_p^{w'})}{N_w * (A + 1) * B_{up}} = \frac{\frac{L}{N_w}}{N_w * R_v} - \frac{\frac{L}{N_w} * \frac{A * B_{up}}{R_v}}{R_v} \quad (3.45)$$

Finally we obtain an expression for $V_p^{w'}$:

$$V_p^{w'} = \frac{L}{N_w} * \frac{R_v^2 + (A + 1) * (R_v + A * B_{up}^2 - B_{up} * R_v)}{R_v^2} \quad (3.46)$$

We notice that $V_p^{w'}$ in 3.46 can be zero in case that B_{up} assumes an extremely high value. We consider $V_p^{w'}$ always larger than zero, assuming a limitation on B_{up} .

Starting from 3.46 we can derive $V_s^{w'}$ considering 3.3 and $V_m^{w'}$ considering equation 3.4. Since the expression of $V_p^{w'}$ is a bit complex we don't introduce the explicit formulation of $V_s^{w'}$ and $V_m^{w'}$.

This scheme allows us to minimize the total amount of data to push if compared to the basic scheme of section 3.4.1. The drawback is that we have a zero initial latency solution, but not a zero-latency solution per window as the base scheme was. In formula 3.47 we define an expression for the average latency:

$$T_l = \frac{(N_w - 1) * T_m^w}{N_w} \quad (3.47)$$

3.4.3 Zero Latency solution per N_j windows

We introduce now a push scheme which generalizes the model of section 3.4.2. We define a new parameter called N_j . N_j represents the number of points in time of a movie we allow the user to jump to. In the scheme there are N_j equispaced windows where we have a push defined by $V_p^{w'}$, $V_s^{w'}$ and $V_m^{w'}$. The remaining $(N_w - N_j)$ windows are defined by V_p^w , V_s^w and V_m^w . Figure 3.6 shows the depicted model for $N_w = 4$ and $N_j = 2$

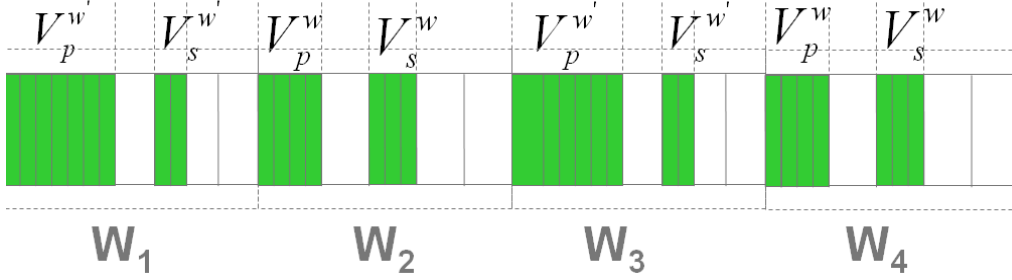


Figure 3.6: Multiple windows - Zero Latency solution per N_j windows - Deterministic Push - Balanced arrival pattern - $N_w = 4$, $N_j = 2$

The depicted push scheme can rely on the formula obtained in section 3.4.2. In fact that scheme is equal to this new push scheme when $N_j = 1$.

The parameter N_j can be consider as an input parameter. It depends on the jumping in time availability we want to give to users. The N_j windows with a different push are equispaced, this means we have a point in time where the user can jump each $\frac{N_w}{N_j}$ windows. We define $T_j = \frac{N_w}{N_j}$ as the length of a back or forward jump in time.

Increasing the value of N_j we have an increase in the storage occupation and a decrease in the average latency. In formula 3.48 we define an expression for the average latency:

$$T_l = \frac{(N_w - N_j) * T_m^w}{N_w} \quad (3.48)$$

Considering $N_j = 1$ allows minimizes the amount of data to push, e.g scheme of section 3.4.2. Increasing the value of N_j means to increase the amount of data to push. In formula 3.49 we introduce the total volume of data to push as a function of N_j :

$$V^P = N_j * (V_p^w + V_s^w) + (N_w - N_j) * (V_p^{w'} + V_s^{w'}) \quad (3.49)$$

The scheme of Figure 3.6 has sense only when V^P is smaller than the total volume of data to push for the scheme described in section 3.4.1. Otherwise it is always better to rely on the previous scheme, since it is a zero latency solution per N_w windows. In formula 3.49 we define a limitation on the parameter N_j :

$$N_j < N_w \frac{\left(\frac{V_p}{N_w} + \frac{V_s}{N_w}\right) - (V_p^w + V_s^w)}{(V_p^{w'} + V_s^{w'}) - (V_p^w + V_s^w)} \quad (3.50)$$

Chapter 4

Analysis of the results

In this chapter we present a quantitative analysis of the analytical models for the Push-to-Peer system. Then we propose a comparison among the several possible solutions.

The parameters and assumptions throughout this section are as follow:

- The content distribution system involved is that of a single high definition movie. The parameters to describe the movie are:
 - $L = 1 \text{ GB}$
 - $R_v = 1.5 \text{ Mbps}$
 - $T = 90 \text{ mns}$
- We consider a DSLAM with a population of 800 users, so $N = 800$.
- We consider $D_{up} = 1$ and $D_{down} \leq N$.

The term of comparison among the several solutions is the fraction of L the content-server needs to push in each gateway in order to have a system that can serve a target demand n_a . For each scenario we plot the volume of data to push for $B_{up} = 128 \text{ Kbps}$, $B_{up} = 640 \text{ Kbps}$, $B_{up} = 1 \text{ Mbps}$ and $B_{up} = 1.5 \text{ Mbps}$, which are typical upload values for classical ADSL.

Finally we compare the Centralized and Push-to-Peer architecture in terms of costs for the content provider.

4.1 The Deterministic Push - $D_{up} = 1$ - $N_w = 1$

The Deterministic Push relies on a distribution of the non-coded chunks C_i that made up a content L . The content-server decides a priori in which peers replicate a chunk. In section 3.3.1 we computed formula for parameters V_p , V_s and V_m according to some specific arrival patterns.

We analyze here the behavior of the Deterministic Push for each depicted arrival pattern. The metric of comparison is the fraction of L the content-server pushes in each gateway, i.e. $\frac{(V_p+V_s)}{L}$. We remind that we are considering zero-latency solutions.

4.1.1 Balanced arrival pattern

We start by considering a balanced arrival pattern, $D_{up} = 1$ and $N_w = 1$. Figure 4.1 shows the ratio between the volume of data to push per gateway and the length of the movie L as a function of the demand.

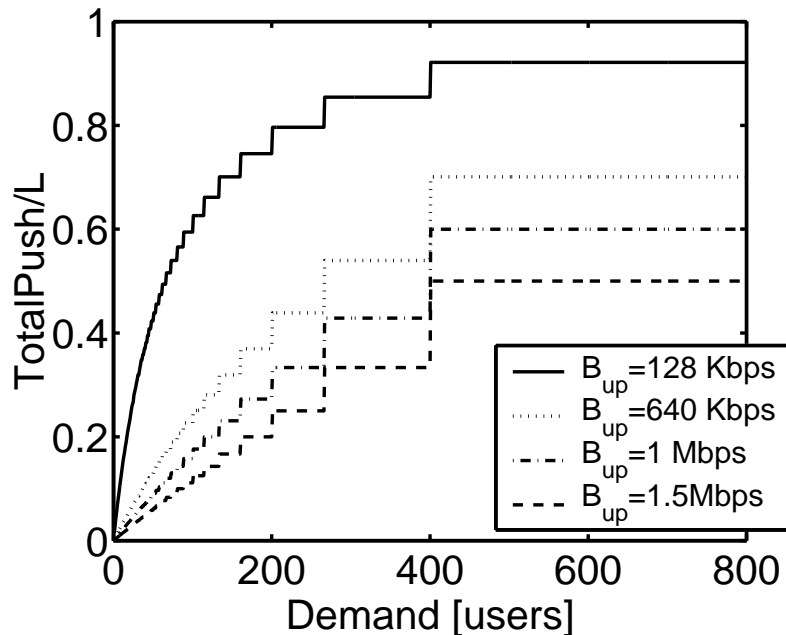


Figure 4.1: Fraction of L to push per gateway as a function of the demand - Deterministic Push - Balanced arrival pattern - $L = 1$ GB, $R_v = 1.5$ Mbps, $N = 800$ - $D_{up} = 1$, $N_w = 1$

Increasing the value of B_{up} reduces the volume of total data to push. Giving more upload capability to the peers allows the system to leverage more on the peer phase than on the push phase. For instance if we look at the curve with $B_{up} = 1$ Mbps, we see that also when the demand is very large the content-server

has to push only about half of the content per peer. The consequence of increasing B_{up} is both a reduction on the storage requirements on gateways and a save for the content-server.

According to formula 3.9 we see that when $B_{up} \geq R_v$ the value of V_p is 0. This means that the last curve in Figure 4.1 simply represents V_s , so the amount of data each peer serves. In Figure 4.2 we see a comparison between $\frac{V_p}{L}$ and $\frac{V_s}{L}$ for several values of B_{up} .

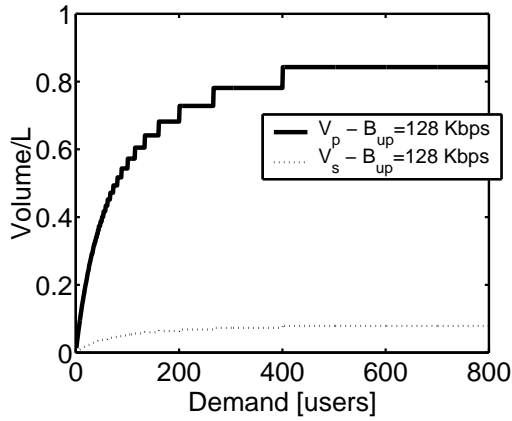
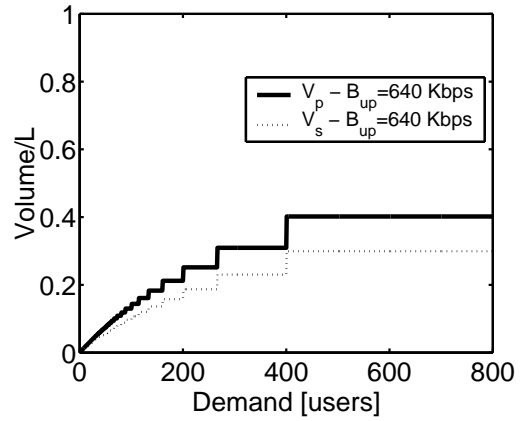
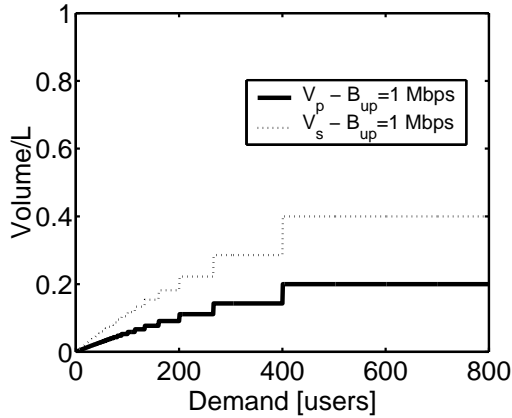
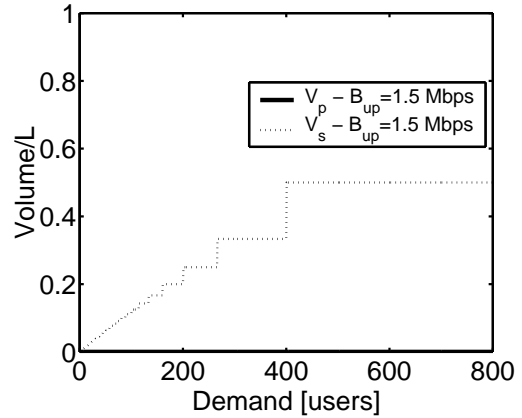
(a) $B_{up} = 128 \text{ Kbps}$ (b) $B_{up} = 640 \text{ Kbps}$ (c) $B_{up} = 1 \text{ Mbps}$ (d) $B_{up} = 1.5 \text{ Mbps}$

Figure 4.2: $\frac{V_p}{L}$ and $\frac{V_s}{L}$ as a function of the demand - Deterministic Push - Balanced arrival pattern - $L = 1 \text{ GB}$, $R_v = 1.5 \text{ Mbps}$, $N = 800$ - $D_{up} = 1$, $N_w = 1$

Considering small values of B_{up} (as in Figures 4.2(a) and 4.2(b)) means the system can't rely too much on the peer phase. As a consequence $V_p > V_s$.

In contrast by increasing B_{up} , V_s also increases. Therefore an active peer can get more during the peer phase. As a consequence the content-server can serve a smaller V_p . This means each peer needs to serve more, so the value of V_s increases.

In Figure 4.2(d) we consider the limit scenario. Since $B_{up} = R_v$ each peer doesn't need any prefix, $V_p = 0$. Each active peer simply gets the missing blocks in streaming from the serving peers.

Considering $B_{up} > R_v$ doesn't give any improvements in storage occupation. This is also expressed by formula 3.10: when $B_{up} \geq R_v$, V_s only depends on the number of sets we create, so implicitly on n_a . The reason is that we don't want to modify the replication factor X_r and the base scheme. Of course increasing B_{up} we decrease the time to retrieve the content.

Finally in Figure 4.3 we compare the size of $\frac{V_s}{L}$ for different values of B_{up} .

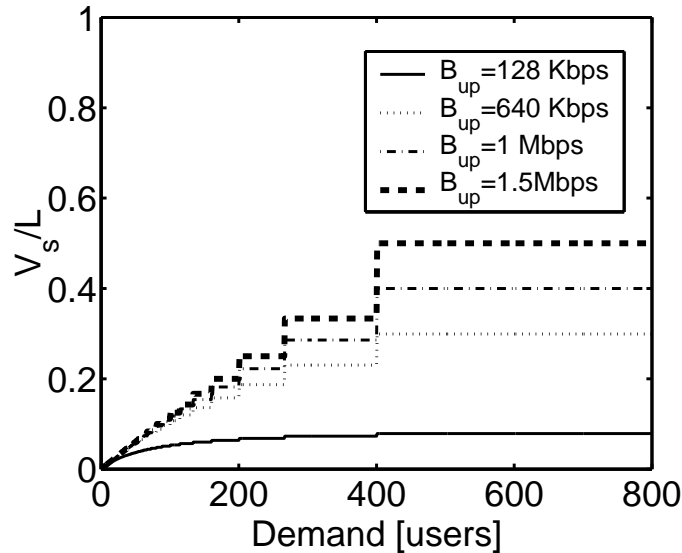


Figure 4.3: $\frac{V_s}{L}$ as a function of the demand - Deterministic Push - Balanced arrival pattern - $L = 1GB$, $R_v = 1.5 Mbps$, $N = 800$ - $D_{up} = 1$, $N_w = 1$

Figure 4.3 confirms the previous results. Increasing the value of B_{up} allows the system to leverage more on the peer phase. Each peer can serve and get more and therefore the value of V_s increases.

4.1.2 Worst arrival pattern

We consider now the worst arrival pattern, $D_{up} = 1$ and $N_w = 1$. Figure 4.4 shows the ratio between the volume to push per gateway and the length of the movie L as a function of the demand.

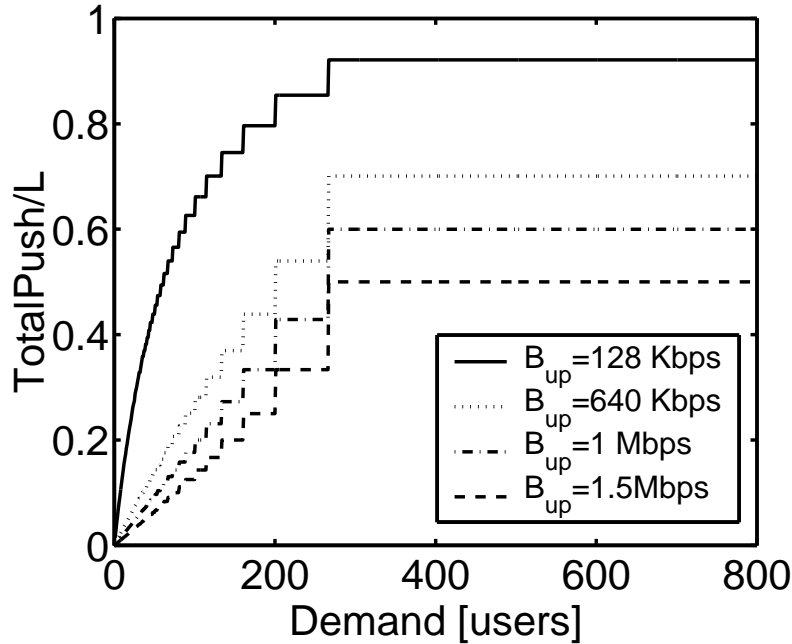


Figure 4.4: Fraction of L to push per gateway as a function of the demand - Deterministic Push - Worst arrival pattern - $L = 1$ GB, $R_v = 1.5$ Gbps, $N = 800$ - $D_{up} = 1$, $N_w = 1$

As expected in average we get a larger storage occupation respect to a balanced arrival pattern (see Figure 4.1). Anyway in Figure 4.4 when the demand is larger than $\frac{N}{2}$ ($A = 1$) we don't notice any difference with Figure 4.1. According to the deterministic scheme since $A = 1$ we create $A + 1 = 2$ sets, set_a and set_b . All members of set_a will be served by members of set_b and viceversa. The a priori decision of the content-server about the group creation has no effect when the demand is very high. As a consequence the gap between a balanced and a worst arrival pattern is reduced when n_a assumes the larger values.

This consideration allows us to say that increasing the value of the demand reduces the effect of a bad arrival pattern on the average system performance. Moreover increasing the number of requests decreases the probability to have all requests coming from the same group (worst arrival). We can conclude that the hypothesis to have a balanced arrival pattern has sense and it is not so strong.

4.1.3 Best arrival pattern

We consider now the best arrival pattern, $D_{up} = 1$ and $N_w = 1$. Figure 4.5 shows the ratio between the volume to push per gateway and the length of the movie L as a function of the demand.

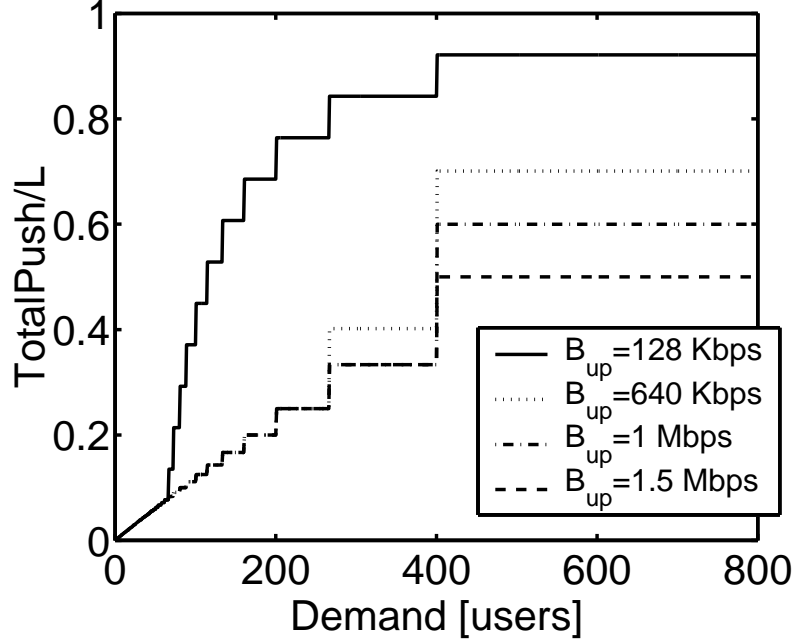


Figure 4.5: Fraction of L to push per gateway as a function of the demand - Deterministic Push - Best arrival pattern - $L = 1 GB$, $R_v = 1.5 Mbps$, $N = 800$ - $D_{up} = 1$, $N_w = 1$

The main feature of the best arrival pattern is that the download rate per set is equal to $R_s = A * B_{up}$. This means that the points where $V_p = 0$ depends on A and B_{up} . For this reason when the demand is very low the curves have the same shape, since $V_p = 0$ for all the curves and V_s depends only on A and L . Increasing n_a we start to have $V_p \neq 0$ for $B_{up} = 128 kbps$, then for $B_{up} = 640 kbps$ and so on.

In Figure 4.6 we plot $\frac{V_p}{L}$ and $\frac{V_s}{L}$ for different values of B_{up} . We can see how the value of n_a where V_p becomes $\neq 0$ is shifted on the right according to the value of B_{up} considered.

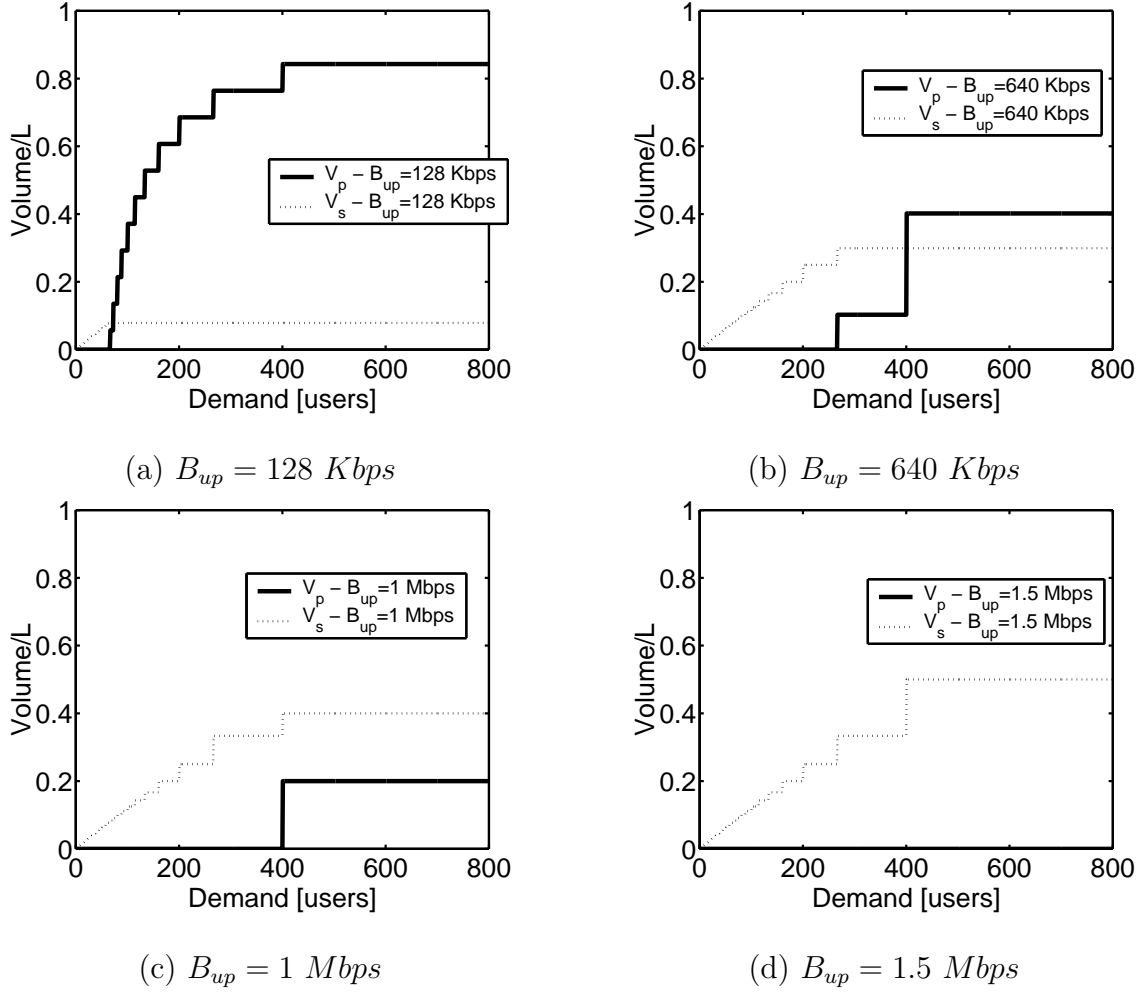


Figure 4.6: $\frac{V_p}{L}$ and $\frac{V_s}{L}$ as a function of the demand - Deterministic Push - Best arrival pattern - $L = 1 \text{ GB}$, $R_v = 1.5 \text{ Mbps}$, $N = 800$ - $D_{up} = 1$, $N_w = 1$

In Figure 4.6(d) we plot the extreme scenario with $B_{up} = R_v$. Also when $n_a = N$ the size of the prefix V_p is 0. Each active peer is simply streaming the chunks he downloads at a speed $R_d = R_v$. Each peer needs to serve half of the content, $V_s = \frac{L}{2}$.

Finally in Figure 4.7 we compare $\frac{V_s}{L}$ for different values of B_{up} . We confirm what we say above. The curves have the same behavior where V_p is equal to zero, and this value of n_a depends on the size of B_{up} . We notice also that when $A * B_{up} < R_v$ the value of V_s is constant, since it only depends from B_{up} and R_v (see equation 3.22).

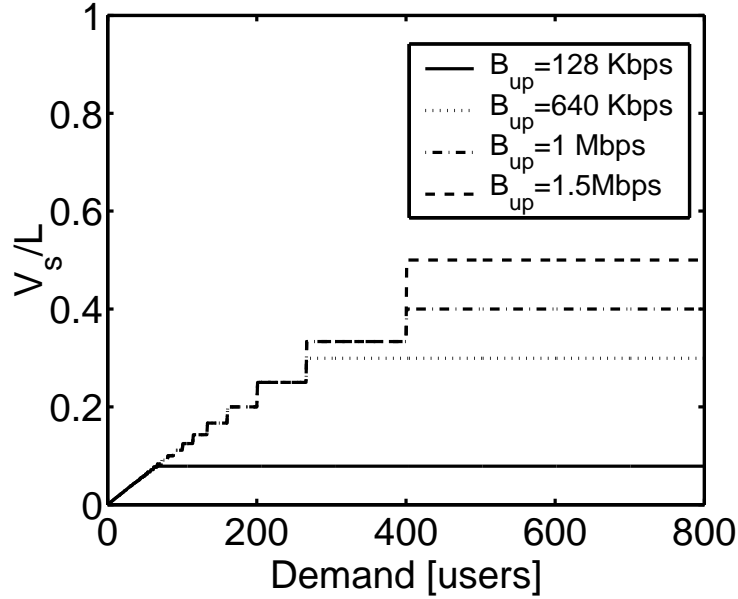


Figure 4.7: $\frac{V_s}{L}$ as a function of the demand - Deterministic push - Best arrival pattern - $L = 1 \text{ GB}$, $R_v = 1.5 \text{ Mbps}$, $N = 800$ - $D_{up} = 1$, $N_w = 1$

4.1.4 Comparison: Balanced - Best - Worst arrival pattern

We compare here the storage occupation for a Deterministic Push in case of worst, balanced and best arrival pattern. We refer to the results presented above.

Figure 4.8 shows the ratio between the total volume to serve and the length of the movie L as a function of the demand for the three different arrival patterns. The curves for worst and best arrival pattern represent the maximum and minimum bound for the storage occupation. The curve for a balanced arrival pattern is always bounded by the two previous curves for each value of $B_{up} < R_v$.

Figure 4.8(d) shows what happens when $B_{up} = R_v$. The curves of balanced and best arrival pattern overlap. The reason is that in both case $V_p = 0$ and so $V_s = \frac{L}{(A+1)}$ (see 3.10 and 3.22). In the equations B_{up} doesn't affect the volume of data to push. The reason is that we don't want to modify the base scheme made up of $(A + 1)$ sets and of a replication factor $X_r = \frac{N}{(A+1)}$. The effect of B_{up} and of the best arrival pattern affects only R_s .

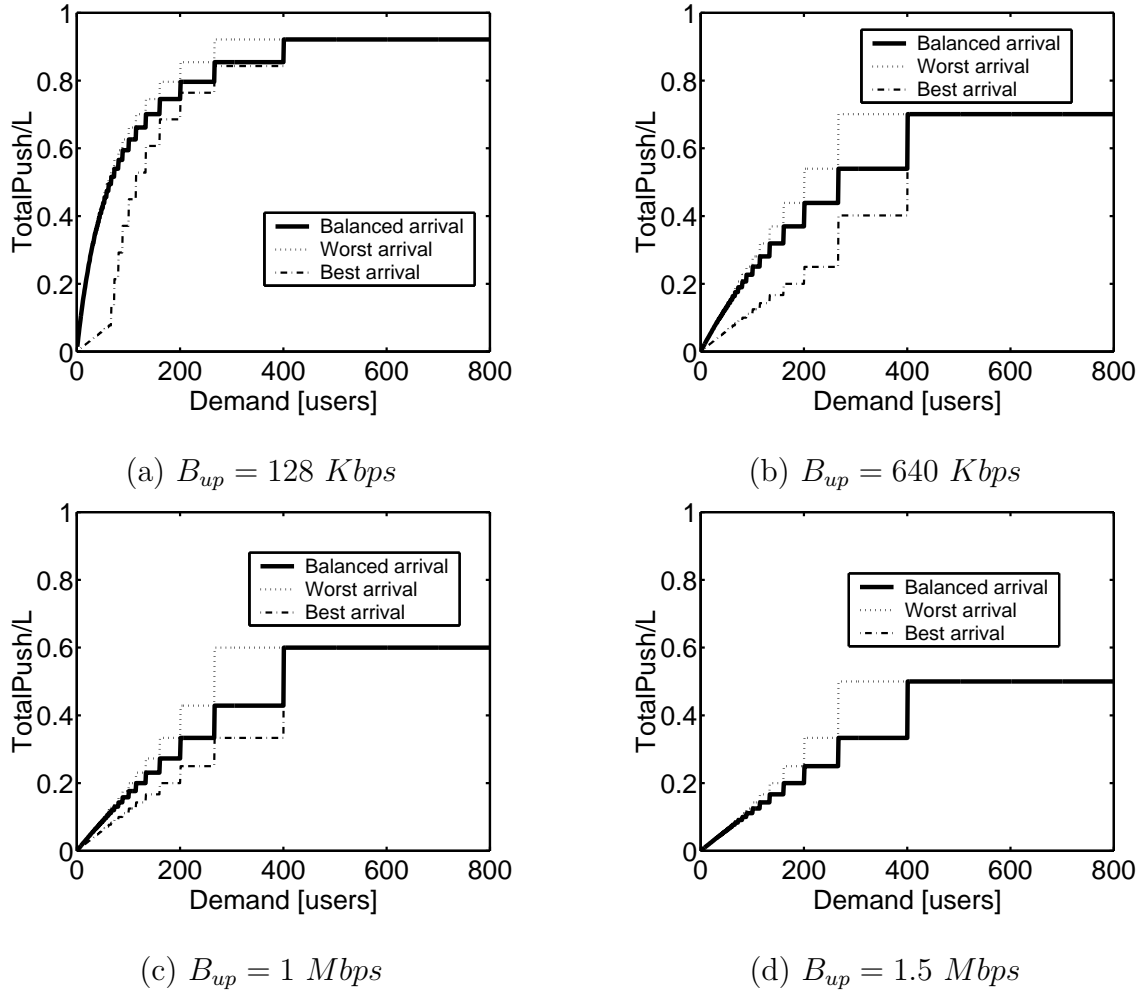


Figure 4.8: Fraction of L to push per gateway as a function of the demand - Deterministic Push - Balanced, Best, Worst arrival pattern - $L = 1 GB$, $R_v = 1.5 Mbps$, $N = 800$ - $D_{up} = 1$, $N_w = 1$

The best and worst arrival pattern represent two extreme cases. In our next analysis we will focus more on a balanced arrival pattern, since we saw it can be considered an average scenario. We can't predict users' behavior, a future work is to study some real data to extrapolate a realistic arrival pattern. Anyway here we understand the influence of an arrival pattern on the Deterministic Push.

4.2 The Source Coding Push - $D_{up} = 1$ - $N_w = 1$

The Source Coding push relies on a distribution of coded chunks. The strength of source coding is that it simplifies the issue of piece selection. The consequence is that performance is no more affected by the arrival pattern. The drawback is the overhead of the code and the impossibility to exploit the linear priority among coded chunks.

We analyze here the storage occupation for a Source Coding Push, $D_{up} = 1$ and $N_w = 1$. Figure 4.9 shows the ratio between the total volume to serve and the length of the movie L as a function of the demand.

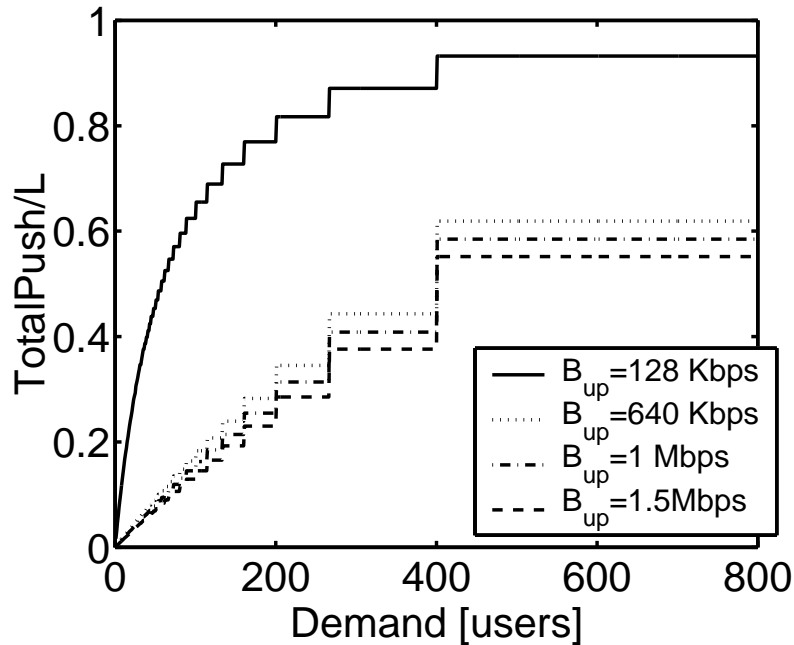


Figure 4.9: Fraction of L to push per gateway as a function of the demand - Source Coding Push - $L = 1$ GB, $R_v = 1.5$ Mbps, $N = 800$, $\varepsilon = 0.04$ - $D_{up} = 1$, $N_w = 1$,

The curves never completely overlap also for a small demand. The reason is that since we need to download all the coded blocks before being able to reconstruct the original content, the size of V_p can never be zero. So we have always a benefit in storage occupation in considering a larger value of B_{up} . There is no more the concept of a replication-factor X_r , since we don't consider real chunks but coded chunks. This is also the reason why there is no more piece selection.

In Figure 4.10 we plot the behavior of $\frac{V_p}{L}$ and $\frac{V_s}{L}$. As usual increasing the value of B_{up} we increase the capability of the peer phase, so the value of V_s . It confirms or previous results, that $V_p > 0$ even when $B_{up} = R_v$.

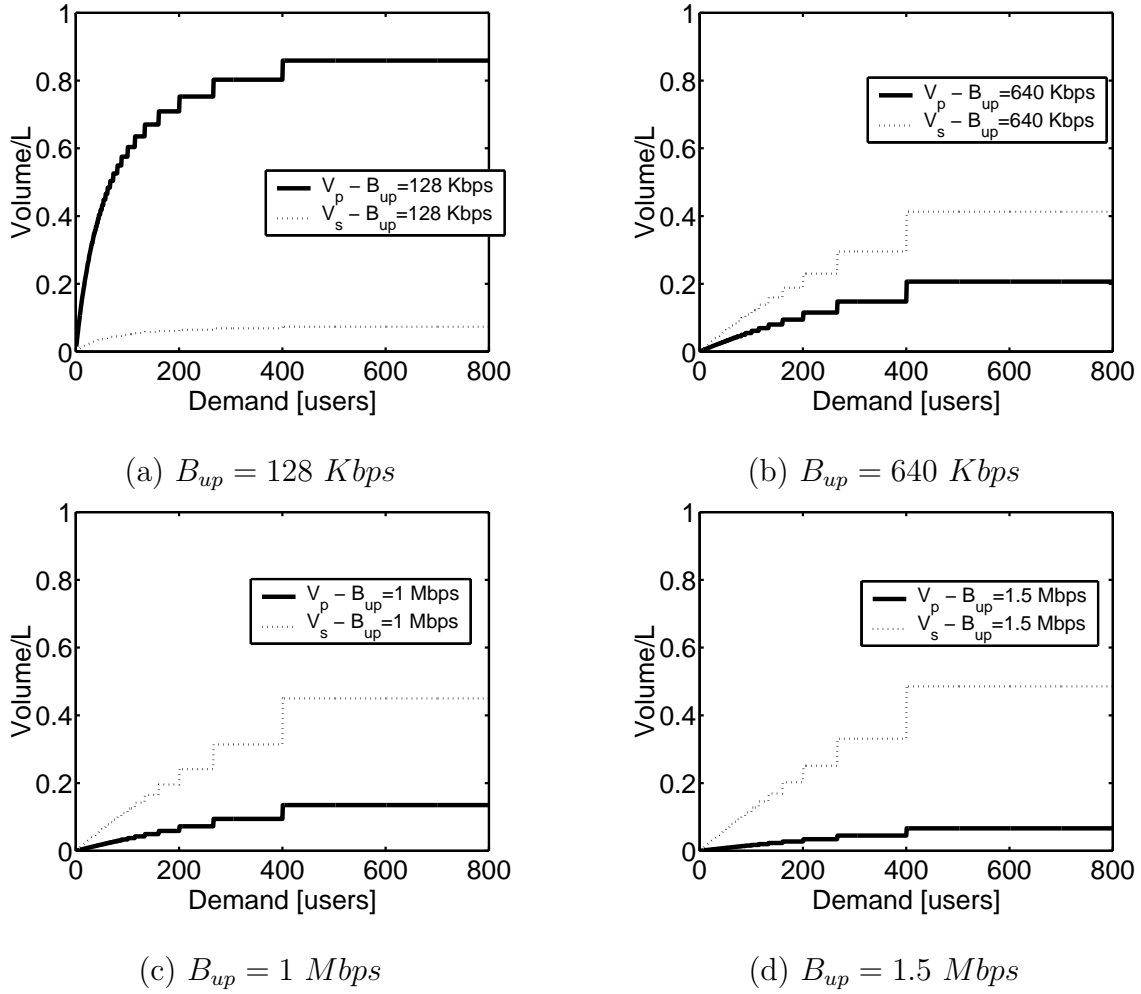


Figure 4.10: $\frac{V_p}{L}$ and $\frac{V_s}{L}$ as a function of the demand - Source Coding Push - $L = 1 \text{ GB}$, $R_v = 1.5 \text{ Mbps}$, $N = 800$, $\varepsilon = 0.04$ - $D_{up} = 1$, $N_w = 1$

A Source Coding Push as expected achieves a larger storage occupation than a Deterministic Push. It is interesting to notice that this kind of push frees completely from the concept of arrival pattern. Anyway in the next section we directly compare the results proposed above.

We can conclude that a distribution of coded-blocks can be interesting if the system is affected by a bad peers' behavior, i.e a non structured network.

4.3 Comparison: Deterministic Push - Source Coding Push

We compare here the storage occupation for a Source Coding Push and a Deterministic Push with balanced and worst arrival pattern. We consider $D_{up} = 1$ and $N_w = 1$. Figure 4.11 shows the ratio between the total volume to serve and the length of the movie L as a function of the demand.

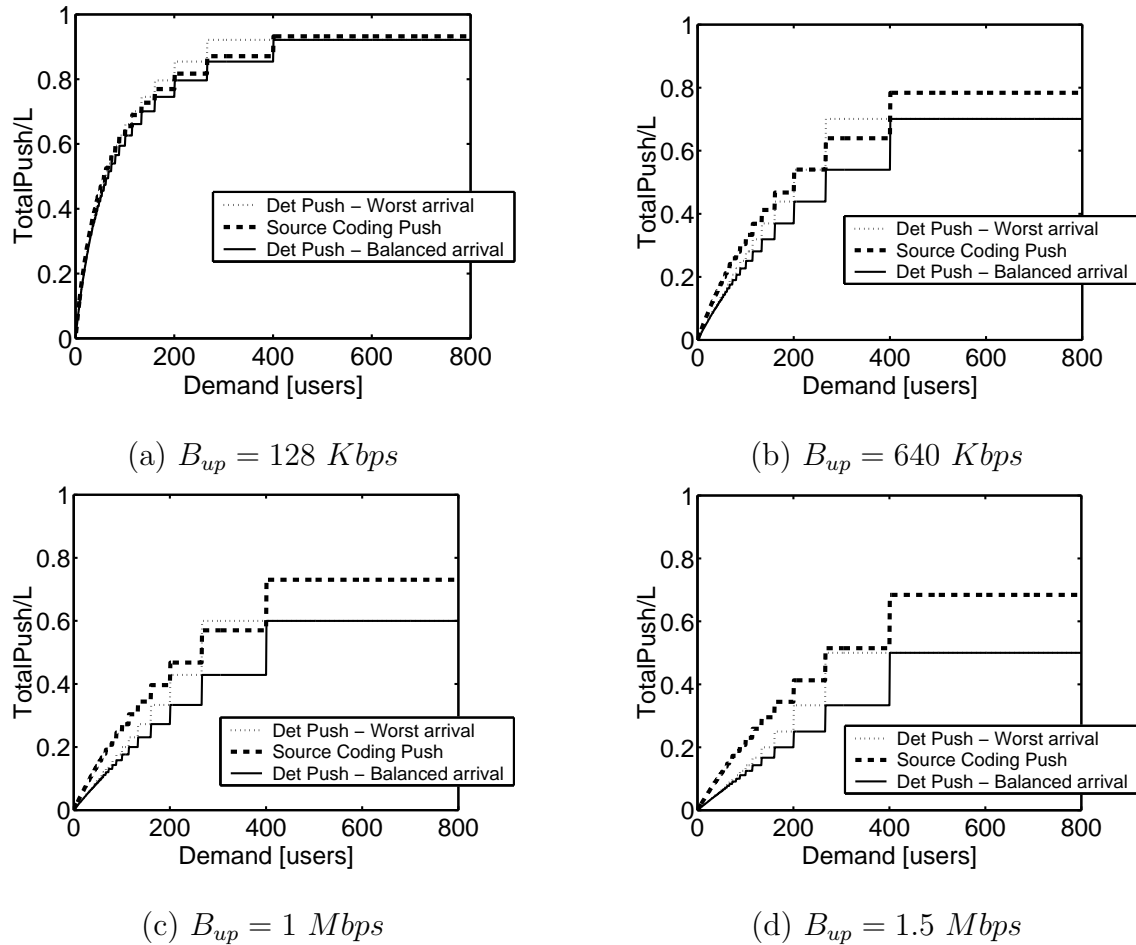


Figure 4.11: Fraction of L to push per gateway as a function of the demand - Deterministic Push and Source Coding Push - Balanced, Worst arrival pattern - $L = 1 GB$, $R_v = 1.5 Mbps$, $N = 800$, $\varepsilon = 0.04$ - $D_{up} = 1$, $N_w = 1$

Figure 4.11 shows that the Deterministic Push with balanced arrival pattern always minimizes the storage occupation. It is interesting to see that if the system is affected by the worst arrival pattern, only for some values of n_a we have an advantage in considering source coding. Since the addition of coding also increases

the complexity, we don't see any interests in a source coding strategy with respect to the parameters considered here.

According to what we see here, we conclude that the performance of the Push-to-Peer architecture can rely on a Deterministic Push strategy. In section 4.1.4 we conclude that the balanced arrival pattern represents an average and realistic scenario. We can say that the Push-to-Peer architecture can be dimensioned with a Deterministic Push strategy and on the assumption of a balanced arrival pattern.

4.4 Trick-mode models for the push phase

$$- D_{up} = 1 - N_w > 1$$

The basic model for the Push-to-Peer system (3.2) relies on $N_w = 1$. The limitation of this model is that the only way to allow the user to jump back and forward in the movie (trick-mode) is to first download and store the entire content. This model is defined as a zero latency scheme when the trick-mode is not supported.

In section 3.4.1 we presented the logical extension of the basic model for the trick-mode. The new scheme replicates the basic scheme in each window, simply considering $L \Rightarrow \frac{L}{N_w}$. It allows to achieve a completely zero latency solution. Which means each peer can jump in N_w points in time of the movie and watch the video without any delay. The total volume of data to push per peer stays the same, so we can refer to the results of 4.1.

Starting from the model of Figure 3.4 we understood which improvements we can have in storage occupation by considering $N_w > 1$ (see formula 3.35 and 3.37).

In Figure 4.12 we plot $T_d(N_w)$ considering $1 \leq N_w \leq \frac{V_m}{A}$. We consider an MPEG-4 good quality video, $N = 800$, $n_a = 200$, $B_{up} = 300 \text{ kbps}$, $V_{ch} = 256 \text{ KB}$ and a Deterministic Push with balanced arrival pattern. We get $V_p = 444.44 \text{ MB}$, $V_s = 111.11 \text{ MB}$, $V_m = 444.45 \text{ MB}$ and the corresponding V_p^w , V_s^w , V_m^w according to the value of N_w .

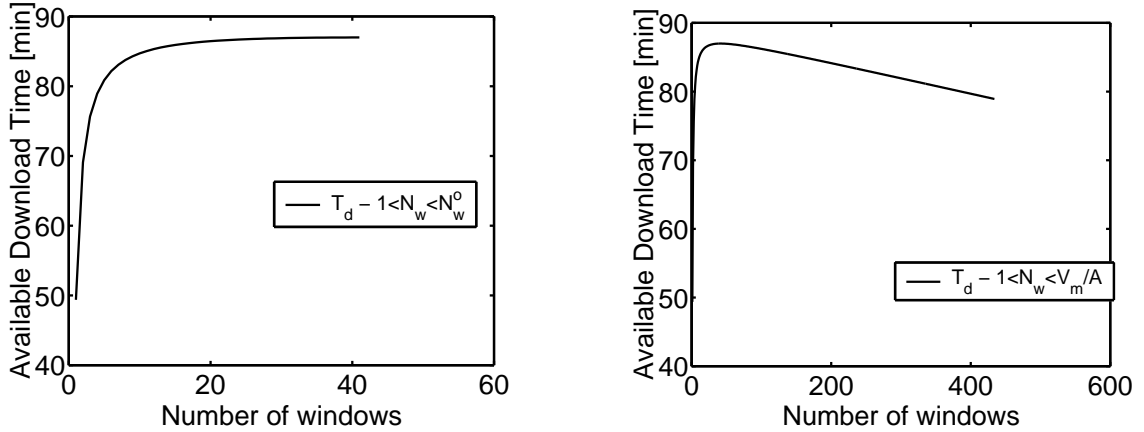


Figure 4.12: $T_d(N_w)$ - MPEG-4 good quality video - Deterministic Push - Balanced arrival pattern - ($N = 800$, $n_a = 200$, $B_{up} = 300 \text{ Kbps}$, $V_{ch} = 256 \text{ KB}$)

Figure 4.12 shows that considering $N_w = 1$ leads to have the smallest value for T_d . The maximum value for T_d is achieved with $N_w = 41$. This value confirms the result of equation 3.37 with the above parameters: $N_w^o = \sqrt{\frac{V_m}{V_{ch}}} = 41$.

In the mathematical analysis we considered an approximation on the term T_{ch} of formula 3.37. Simplifying the value of T_{ch} , equation 3.37 has no more a maximum. Anyway Figure 4.12 shows that for the content distribution we consider, choosing $N_w > 10$ allows to better exploit the linear priority among chunks respect to a single window scheme.

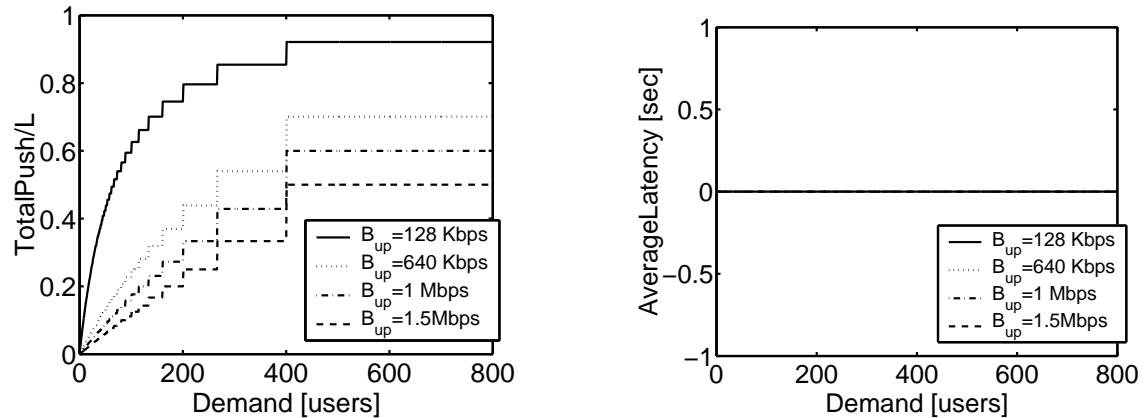
In this section we analyze the performance of three schemes for the trick-mode in term of volume of data to push and latency. The content distribution involved is of a single high definition movie. According to the previous results for the push strategy we consider a Deterministic Push with balanced arrival pattern.

The three models we analyze are :

- *Zero Latency solution per window*: this scheme is the extension of the basic push scheme. The total volume of data to push per peer is maximum but the latency is equal to zero in all the N_w windows.
 - *Zero Initial Latency solution*: this scheme minimizes the total volume of data to push per peer by keeping an initial latency equal to zero. It introduces a latency T_l when the trick-mode is used.
 - *Zero Latency solution per N_j windows*: this scheme defines a tunable trade-off between storage occupation and trick-mode capabilities. It guarantees a latency equal to zero in N_j windows.
-

4.4.1 Zero Latency solution per window

We start by analyzing a Zero Latency solution per window. The scheme in each window behaves exactly as the Deterministic Push with balanced arrival pattern with $N_w = 1$ and size of the content equal to $\frac{L}{N_w}$. Figure 4.13(a) shows the ratio between the total volume to serve and the length of the movie L as a function of the demand. The total volume to serve is computed as an average on the N_w windows.



(a) Fraction of L to push per gateway as a function of the demand

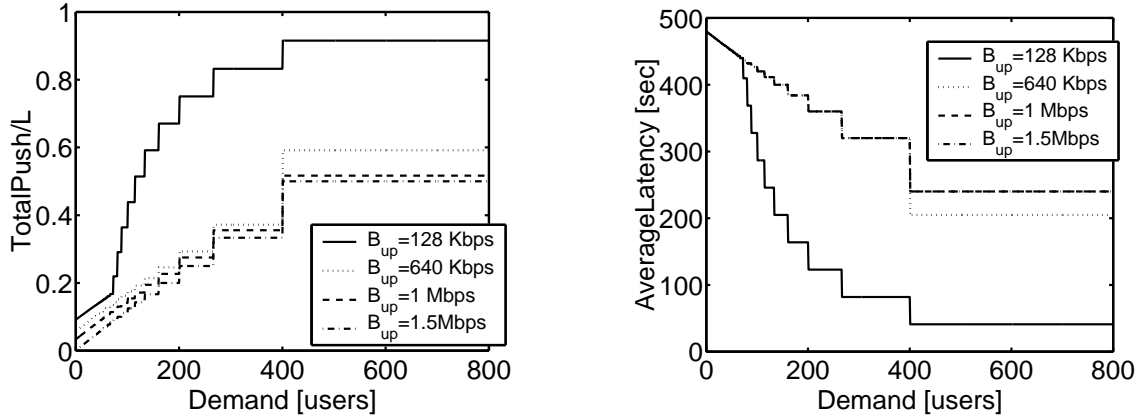
(b) Average initial latency on N_w windows

Figure 4.13: Deterministic Push - Balanced arrival pattern - Zero Latency solution per window - $L = 1$ GB, $R_v = 1.5$ Mbps, $N = 800$ - $D_{up} = 1$, $N_w > 1$

Figure 4.13(b) show the average latency in the system as a function of the demand. Each window is dimensioned as the scheme of Figure 3.1, which means the latency is equal to zero for all the windows. The scheme is a zero latency solution per window and the average latency is zero for every value of n_a .

4.4.2 Zero Initial Latency solution

The scheme we consider now foresees the same push per $(N_w - 1)$ windows and a specific push for the first window. Figure 4.14(a) shows the ratio between the total volume to serve and the length of the movie L as a function of the demand. The total volume to serve is computed as an average on the N_w windows.



(a) Fraction of L to push per gateway

(b) Average initial latency on N_w windows

Figure 4.14: Deterministic Push - Balanced arrival pattern - Zero Initial Latency solution - $L = 1 \text{ GB}$, $R_v = 1.5 \text{ Mbps}$, $N = 800$ - $D_{up} = 1 N_w$

Figure 4.14(a) shows a save in the volume of data to push compared to Figure 4.13(a). The drawback is the addition of an initial latency, as Figure 4.14(b) shows.

Figure 4.14(b) shows the average latency in the system. The scheme is designed to have a zero initial latency and then a continuous view. If the user watches the movie in trick-mode he perceives a latency equal to T_m^w . In fact each window except the first is affected by an initial latency equal to T_m^w .

In Figure 4.14(b) we notice that the value of the average latency decreases with the increase of the demand n_a . The reason is that this latency is a consequence of the scheme introduced and not a system parameter. As we said each window is affected by an initial latency equal to $T_m^w = \frac{V_m^w}{R_v}$. The value of V_m^w decreases with the increase of the demand. As a consequence the latency decreases. We can say that this latency is inversely proportional to the capability of the peer phase. For the same reason the average latency increases also with the increase of B_{up} .

Figure 4.14(a) shows that for small values of n_a the curves have all the same trend. The reason is that the condition of formula 3.42 is never verified also for the smallest value of B_{up} . As a consequence V_p^w is always zero. As we said before if $V_p^w = 0$ V_s^w doesn't depend on the value of B_{up} . Anyway the curves start from a different value. The reason is that when we compute the parameters for the first window, we always need a prefix according to formula 3.46. So here we see the effects of a large or small value for B_{up} .

Increasing the value of the demand n_a , we see the effect of V_p^w first for $B_{up} = 128$ Kbps then for $B_{up} = 640$ Kbps. For $B_{up} = 1.5$ Mbps and $B_{up} = 1$ Mbps the curves overlap for each value of n_a : we expect V_p^w to be always zero also for each n_a .

We confirm what we are saying in Figure 4.15. We plot $\frac{V_p^w}{L}$ and $\frac{V_s^w}{N_w}$ for several values of B_{up} .

Figure 4.15(a) shows that up to $n_a = \frac{N}{\frac{B_{up}}{L} - 1} \Rightarrow n_a \simeq 74$, $V_p^w = 0$ and so V_s^w follows the same shape of the other curves. Then for $n_a > 74$ we need to add a prefix. We can see that V_s^w assumes a constant value. The reason is in 3.43: V_s^w does not depend on A . The effect of A is only reflected in the size of the prefix.

Finally in Figure 4.15(c) and 4.15(d) we confirm what we say above. For $B_{up} = 1.5$ Mbps and $B_{up} = 1$ Mbps the curves overlap for each value of n_a since V_p^w is always zero, so we see for each n_a only the effect of V_s^w . This result is interesting, since it allows to say that if we don't want to modify the replication factor X_r the scheme saturates at a certain value of B_{up} . Which means we don't apport any improvements in the system increasing the value of B_{up} after a certain threshold.

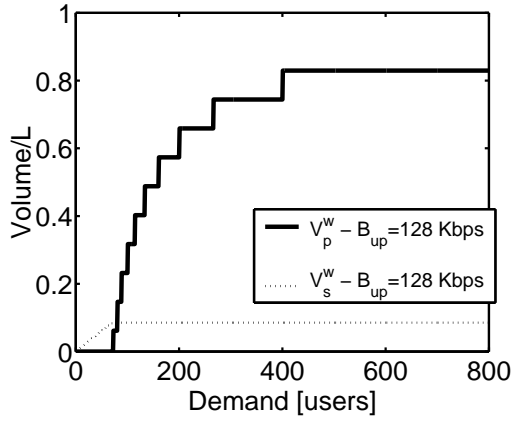
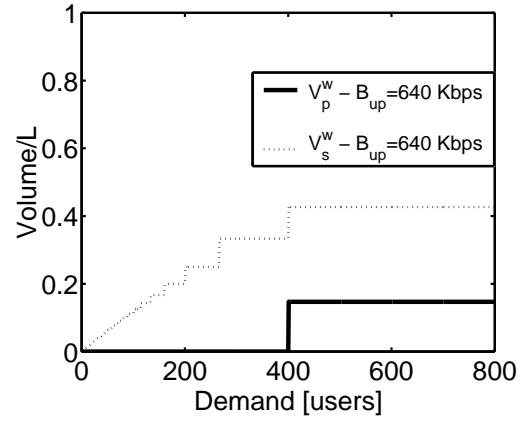
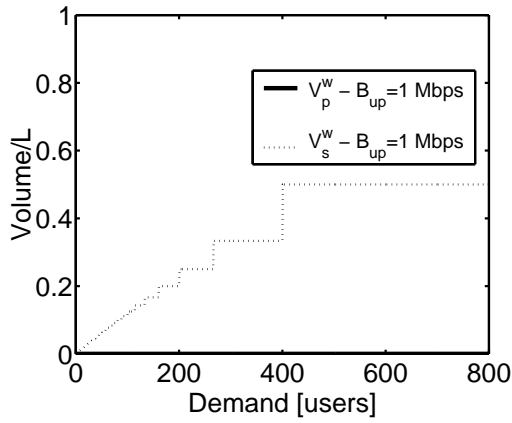
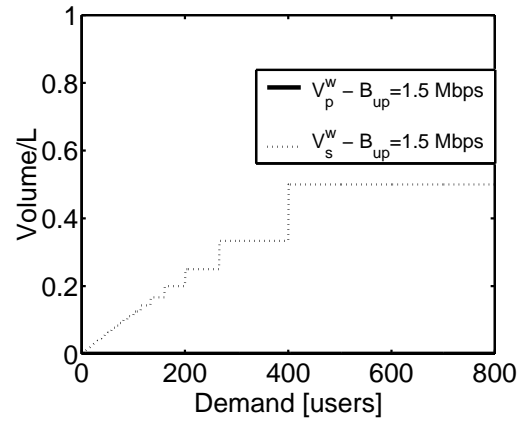
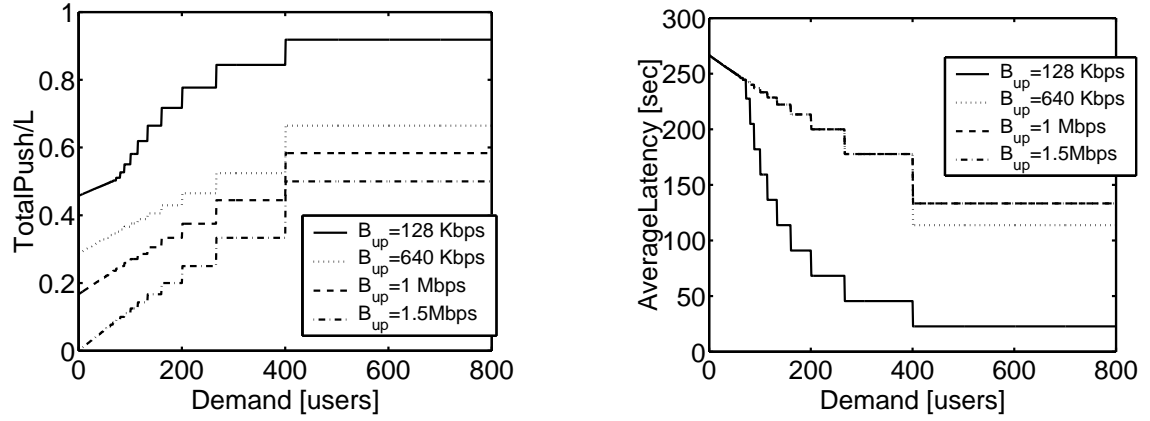
(a) $B_{up} = 128 \text{ Kbps}$ (b) $B_{up} = 640 \text{ Kbps}$ (c) $B_{up} = 1 \text{ Mbps}$ (d) $B_{up} = 1.5 \text{ Mbps}$

Figure 4.15: $\frac{V_p^w}{N_w}$ and $\frac{V_s^w}{N_w}$ as a function of the demand - Deterministic push - Balanced arrival pattern - Zero initial latency solution - $L = 1 \text{ GB}$, $R_v = 1.5 \text{ Mbps}$, $N = 800$ - $D_{up} = 1$, $N_w > 1$

4.4.3 Zero latency solution per N_j windows

The scheme we consider now foresees the same push per $(N_w - N_j)$ windows and a specific push for N_j windows. Figure 4.16(a) shows the ratio between the total volume to serve and the length of the movie L as a function of the demand. We consider to have $N_j = \frac{N_w}{2}$. The total volume to serve is computed as an average on the N_w windows. It's a generalization of the previous scheme, where we add as input parameter N_j .

Figure 4.16(b) shows the average latency in the system as a function of the demand. The scheme is designed to have a zero latency in N_j windows. An user who watches the movie in trick-mode perceives a latency equal to zero in N_j windows. In the remaining $(N_w - N_j)$ windows the latency is equal to T_m^w .



(a) Fraction of L to push per gateway as a function of the demand

(b) Average initial latency on N_w windows as a function of the demand

Figure 4.16: Deterministic Push - Balanced arrival pattern - Zero Latency solution per N_j windows - $L = 1$ GB, $R_v = 1.5$ Mbps, $N = 800$ - $D_{up} = 1$, $N_w > 1$, $N_j = \frac{N_w}{2}$

Figure 4.16(a) shows that for small values of n_a the curves have all the same trend but start from a different value. The reason is the same as before in Figure 4.14(a). Anyway the gap between curves is larger then before. In fact we need to replicate the prefix $V_p^{w'}$ (see formula 3.46) in N_j windows.

4.4.4 Comparison: Trick-mode models for the push

We compare here the storage occupation for the three trick-mode models described in sections 3.4.1, 3.4.2 and 3.4.3. We introduce a simplified notation:

- Scheme *A*: Zero Latency solution per window.
- Scheme *B*: Zero Initial Latency solution.
- Scheme *C*: Zero Latency solution per N_j windows.

Figure 4.17 shows the ratio between the total volume to push per peer and the length of the movie L as a function of the demand. We consider several values of B_{up} .

When the demand is very small Figure 4.17 shows that for each value of B_{up} Scheme *A* performs better than the other two schemes. The reason is that Scheme *B* and Scheme *C* are too much affected by the volume of $V_p^{w'}$ in the zero-latency windows. Since n_a is very small, for $(N_w - N_j)$ windows we don't need any prefix, $V_p^w = 0$. As a consequence in the N_j windows we need to push a very large prefix $V_p^{w'}$. When we compute the average on the N_w windows, the weight of $V_p^{w'}$ affects too much the value of the total volume of data to push.

Scheme *B* can be defined as Scheme *C* with $N_j = 1$. Figure 4.17(a) shows that when $n_a = 1$, the content-server has to push 5 times more in Scheme *C* than in Scheme *B*. The reason is connected with what we say above. Since n_a is very small the total volume of data to push is nearly only affected by the value of $V_p^{w'}$. In Scheme *C* the prefix $V_p^{w'}$ is replicated in $N_j = 5$ windows, while in Scheme *B* in only $N_j = 1$ window. Which means it weights 5 times more than in Scheme *B*.

Scheme *C* is interesting because it defines a tunable tradeoff between storage occupation and trick-mode capabilities. In fact we can tune the value of N_j , representing the points in the movie where the user can jump and watch the movie without perceiving any latency.

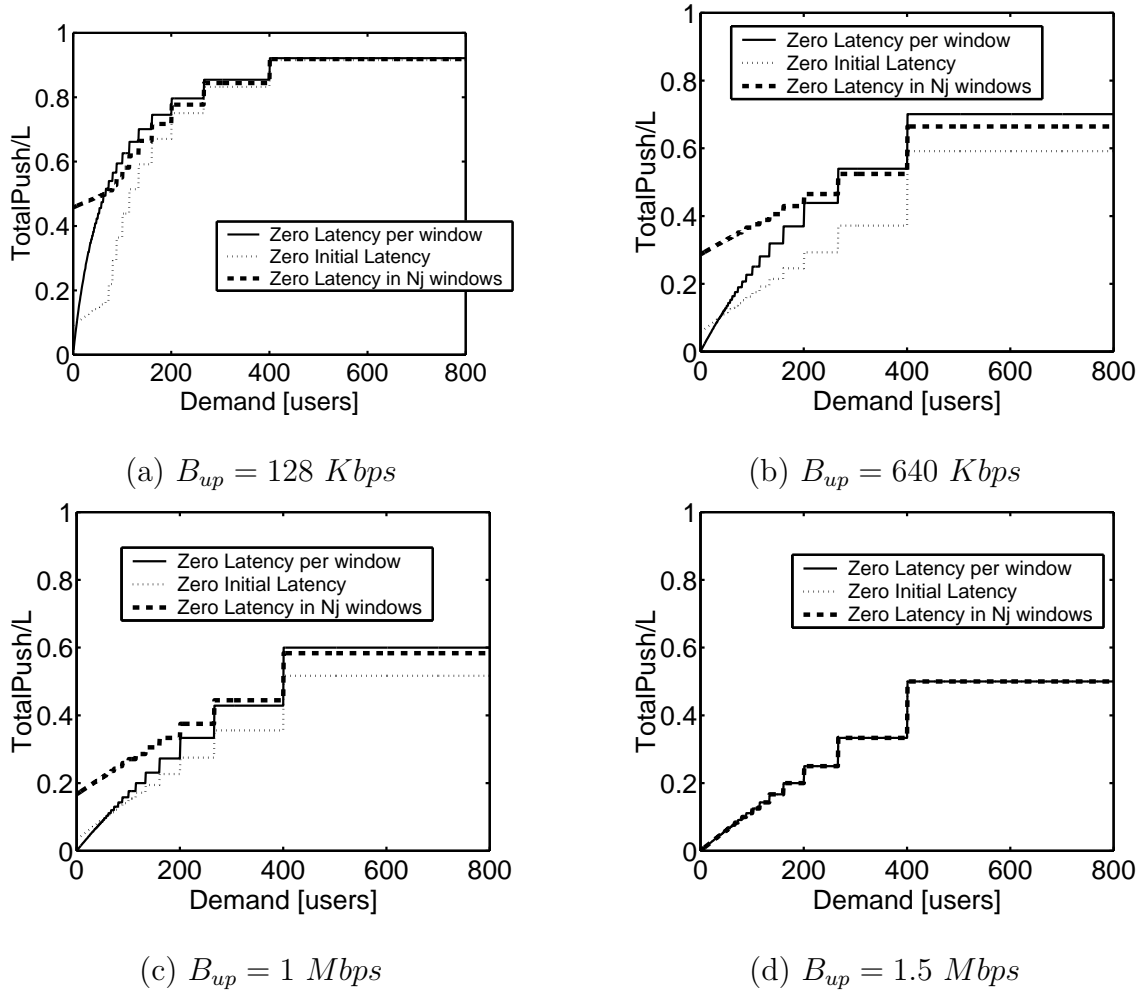


Figure 4.17: Fraction of L to push per gateway as a function of the demand - Comparison: Trick-mode models for the push - Deterministic Push - Balanced arrival pattern - $L = 1 \text{ GB}$, $R_v = 1.5 \text{ Mbps}$, $N = 800$ - $D_{up} = 1$, $N_w = 10$, $N_j = 5$

Figure 4.18 shows the fraction of L to push per gateway as a function of the demand for schemes A , B , C . We consider $N_j = [1; 3; 5; 8]$ and $B_{up} = 128 \text{ Kbps}$.

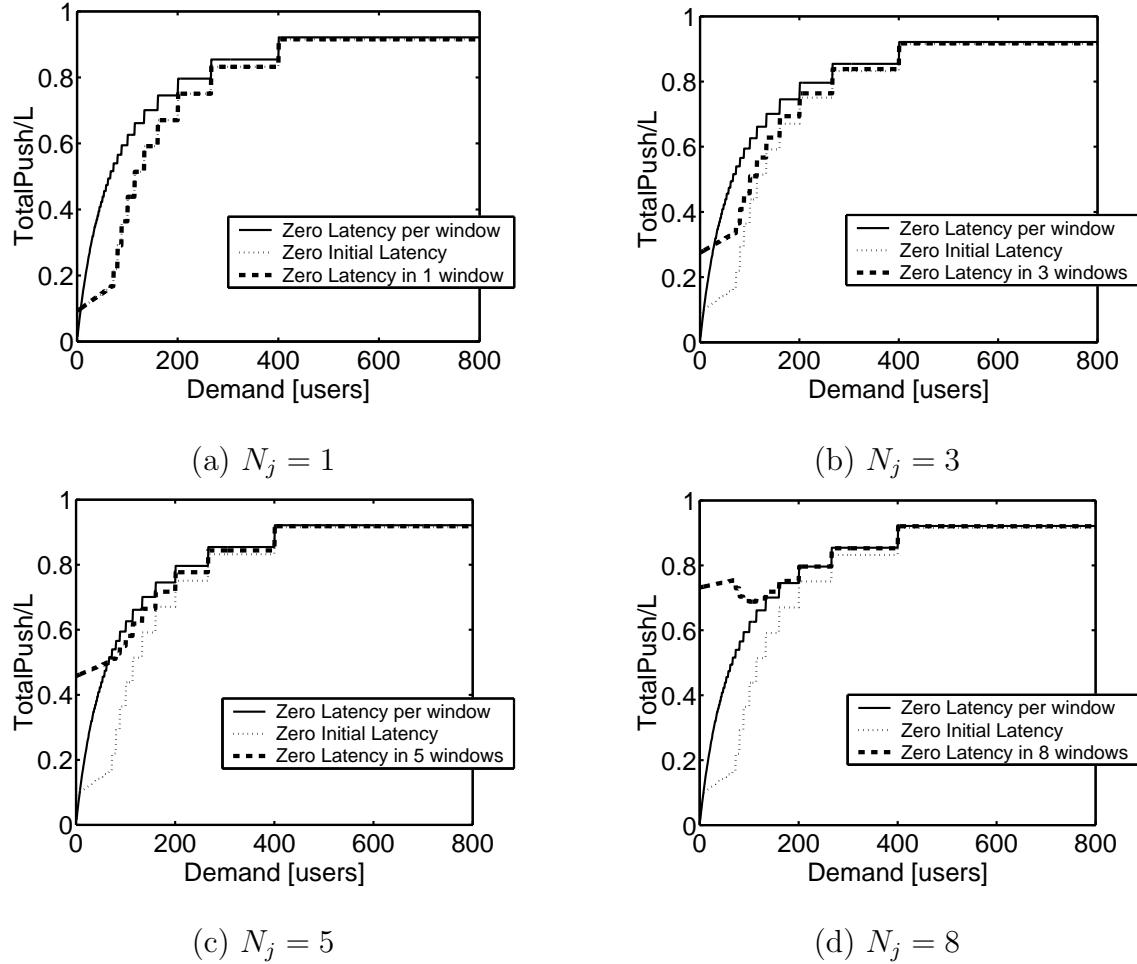


Figure 4.18: Fraction of L to push per gateway as a function of the demand - Comparison: Trick-mode models for the push - Deterministic Push - Balanced arrival pattern - $L = 1 \text{ GB}$, $R_v = 1.5 \text{ Mbps}$, $N = 800$ - $D_{up} = 1$, $N_w = 10$, $B_{up} = 128 \text{ Kbps}$

Figure 4.18(a) shows that Scheme B can be defined as Scheme C with $N_j = 1$, in fact the two curves completely overlap. From a general view of Figure 4.18 we confirm that Scheme B minimizes the storage occupation, while Scheme A maximizes it. Tuning the value of N_j the curve of Scheme C moves between the curve of Scheme B and of Scheme A .

Figure 4.19 shows the average latency as a function of the demand for schemes A, B, C. We consider $N_j = [1; 3; 5; 8]$ and $B_{up} = 128 Kbps$.

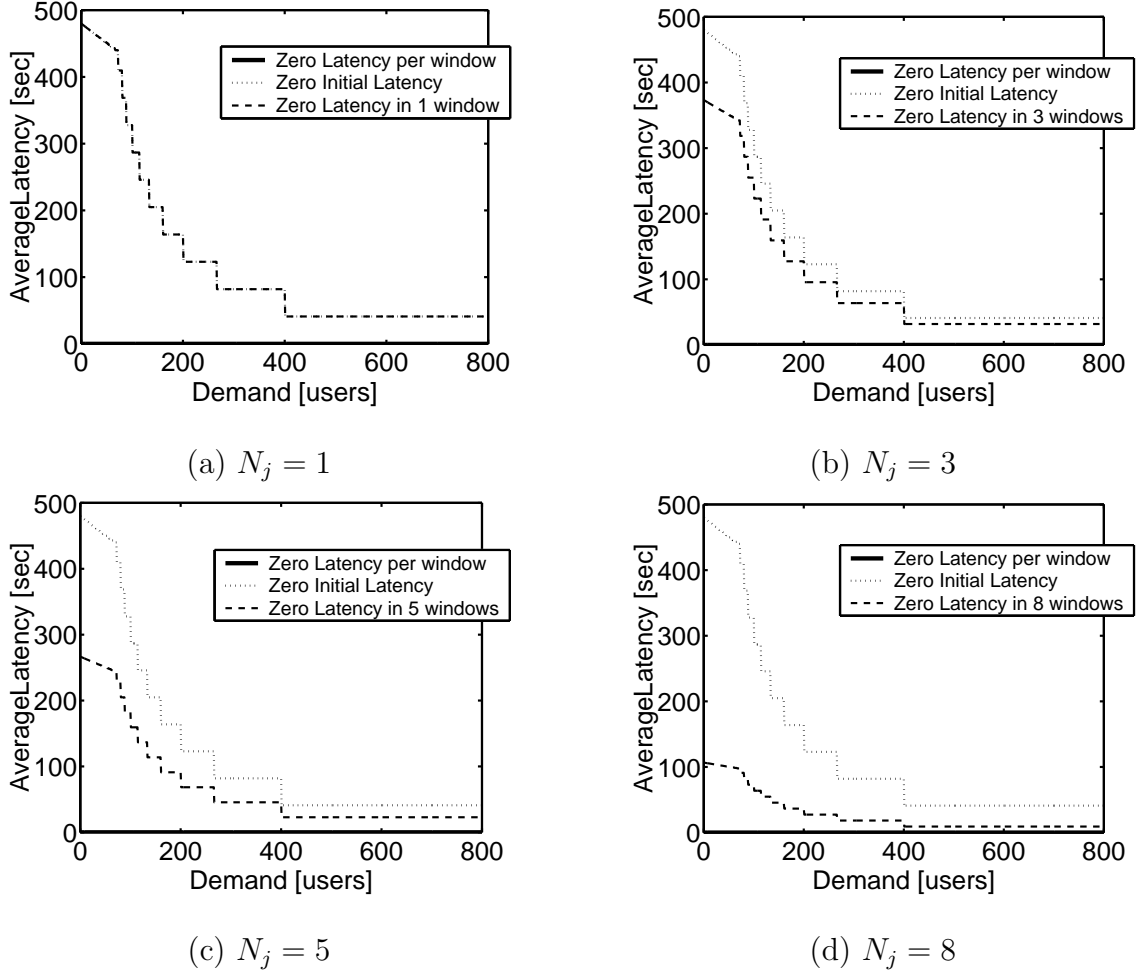


Figure 4.19: Average Latency as a function of the demand - Comparison: Trick-mode models for the push - Deterministic push - Balanced arrival pattern - $L = 1 GB$, $R_v = 1.5 Mbps$, $N = 800$ - $D_{up} = 1$, $N_w = 10$, $B_{up} = 128 Kbps$

Figure 4.19 shows the same behavior depicted above for the latency. We can say that Scheme C allows to introduce a clear tradeoff between storage occupation and average latency. If we observe Figure 4.19 and Figure 4.18 we notice how increasing the value of N_j we increase the storage occupation and decrease the average latency.

Now we move back analyzing Figure 4.18. Figure 4.18(d) shows a strange behavior. In the left side of the graph, increasing the value of the demand we decrease the total volume of data to push. The reason is that when the demand is very low, V_p^w is always zero and V_s^w assumes a value independent from the demand. As a consequence $V_p^{w'}$ assumes a large value for N_j windows. When V_p^w starts to be larger

then zero the value of $V_p^{w'}$ decreases for N_j windows. The speed of this decrease can be many time larger then the increase of V_p^w , which explain this strange behavior.

Figure 4.20 shows the maximum value of N_j for which Scheme C has a storage occupation smaller then Scheme A according to formula 3.50. In general we can say that Scheme C has no sense when we consider a value of N_j larger then what defined in formula 3.50. In fact we can easily rely on the basic push and have less storage occupation and latency.

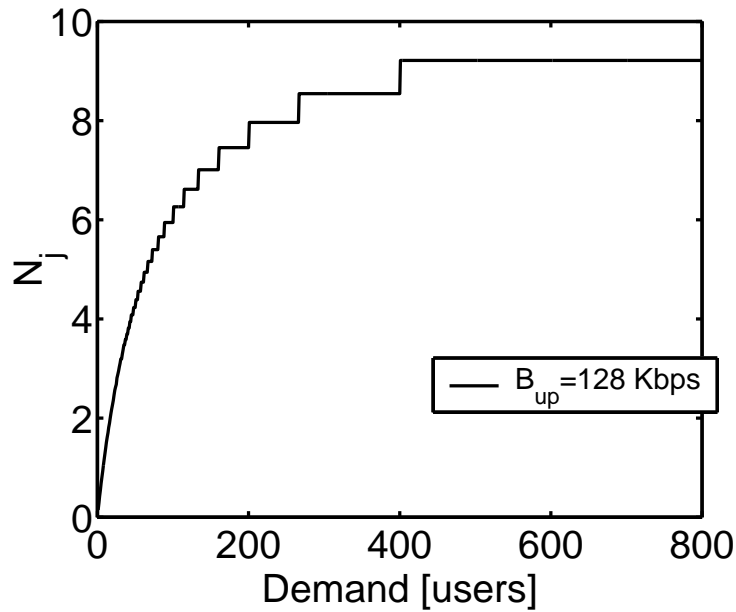


Figure 4.20: N_j evolution as a function of the demand - Deterministic Push - Balanced arrival pattern - $L = 1 GB$, $R_v = 1.5 Mbps$, $N = 800$ - $D_{up} = 1$, $N_w = 10$, $B_{up} = 128 Kbps$

In this section we compared the performance of several trick-mode models in term of storage occupation and latency. We presented a tradeoff between storage requirement and latency for the trick-mode. We came up with a general model where we can have a tradeoff between storage and latency. We learnt that this general model has sense only for certain values of the demand and of the parameter N_j . Moreover we understood that if we want a completely zero latency scheme we have to rely on Scheme A (see Figure 4.19). For this reason we believe that this scheme can represent the general model for the Push-to-Peer architecture with trick-mode. Anyway we showed that it is possible to reduce the storage requirement allowing a loose of performance.

4.5 Centralized vs Push-to-Peer - $D_{up} = 1$

In this section we compare the Push-to-Peer architecture with a centralized solution. In a centralized architecture the trick-mode is supported by assigning to each requesting peer a rate equal to R_v . A peer can jump to any point in time of the movie with a near to zero latency. In order to compare the two architectures in a fair way, we focus on the scheme presented in section 3.4.1. We choose a Deterministic Push strategy with balanced arrival pattern according to the results we got.

In a classical VoD system a cluster of servers serves requests. If there are n_a requests the total amount of data to push is linear with the number of requests, $n_a * L$. For the Deterministic Push with balanced arrival pattern we have to consider formula 3.32 and 3.33.

In Figure 4.21 we compare the amount of data the content-server has to push in a centralized and Push-to-Peer scenario considering several values of B_{up} . We notice that when the value of B_{up} is pretty small (Figure 4.21(a)), a centralized solution leverages on a smaller push. Considering $B_{up} \geq 1$ Mbps (Figures 4.21(c) and 4.21(d)) allows to see some benefits on the Push-to-Peer system. The reason is that to really exploit the peer phase we need to consider some high value of B_{up} if compared to the rate of the video R_v .

From Figure 4.21 it is also interesting to see a nice property of a p2p architecture. Even for the smallest value of B_{up} ($B_{up} = 128$ kbps), when the demand is very high the Push-to-Peer architecture performs better than a centralized solution. The reason is that we can always rely on a peer phase, also if limited, so we don't need to push the entire content to all the peers.

In order to compare the two architectures in a fair way, the simple amount of data to push is not a sufficient metric. We have to take into account the distribution of requests over time, and the cost it generates at a server. If we consider a system that want to sustain a peak demand equal to n_a , in a centralized solution the server has to pay for a link at speed equal to $n_a * R_v$.

The available time to push $n_a * L$ bytes in a centralized solution is $T_{sd}^C = T$. This value can't be tuned, otherwise we introduce a latency for the users, or we serve a smaller demand. The available time to serve the same amount of data in a Push-to-Peer architecture is a system parameter and can be several times larger than T , i.e. $T_{sd}^{P2P} \gg T$. Since most networks are unused during night time, the push phase could be organized during the night. So we can think about setting $T_{sd}^{P2P} = 8h$.

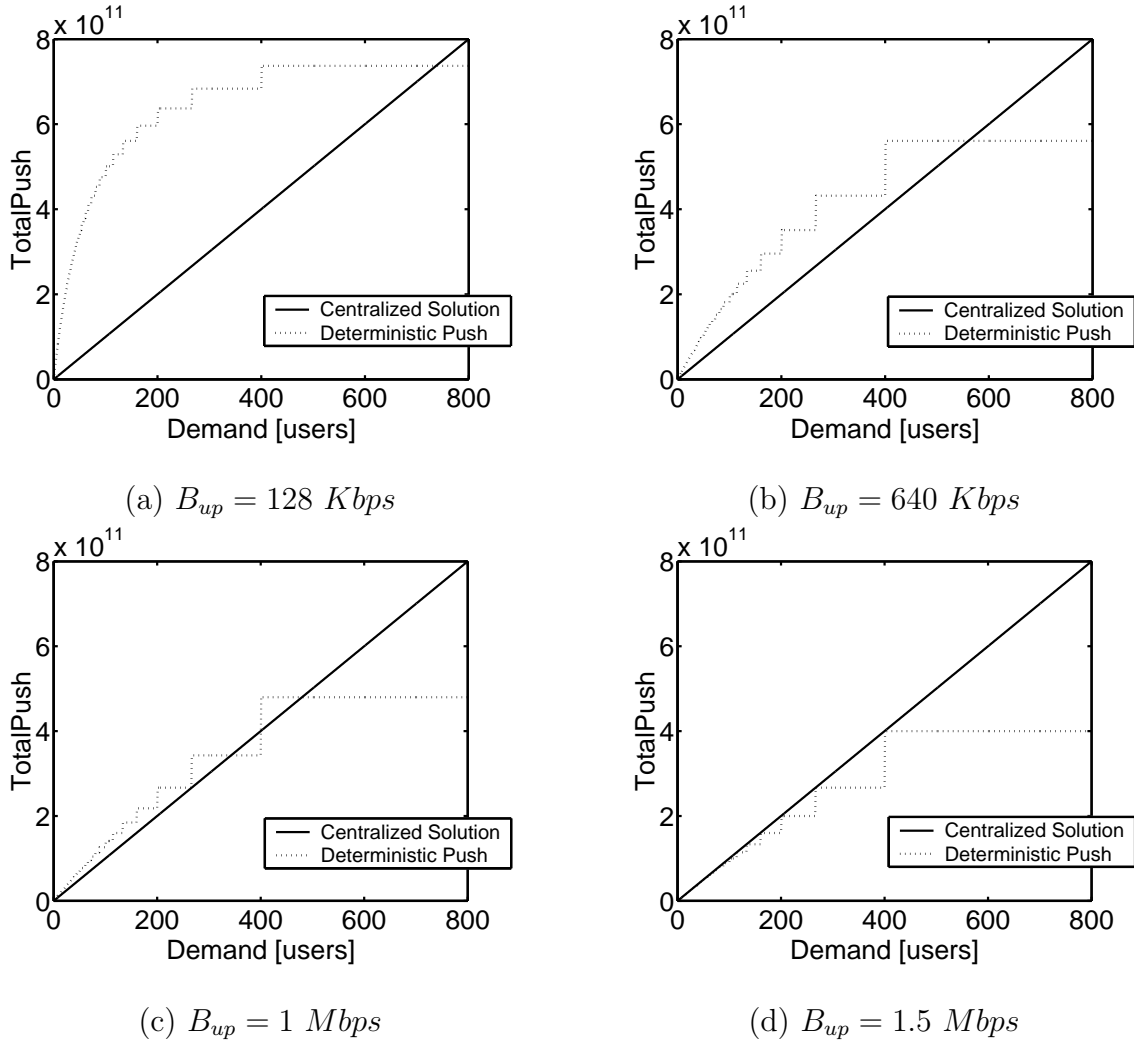


Figure 4.21: Total volume of data to push in the network - Centralized vs Push-to-Peer - $L = 1 GB$, $R_v = 1.5 Mbps$, $N = 800$ - $D_{up} = 1$, $N_w = 1$

We introduce R_S^{p2p} as the server-rate within a Push-to-Peer architecture and R_S^c as the server-rate within a Centralized architecture. If we consider the same amount of data to push in both architecture, we can define a relationship between R_S^{p2p} and R_S^c :

$$R_S^{P2P} = \frac{T_{sd}^C}{T_{sd}^{P2P}} * R_S^c \quad (4.1)$$

In Figure 4.22 we plot $R_S(n_a)$ for both architectures.

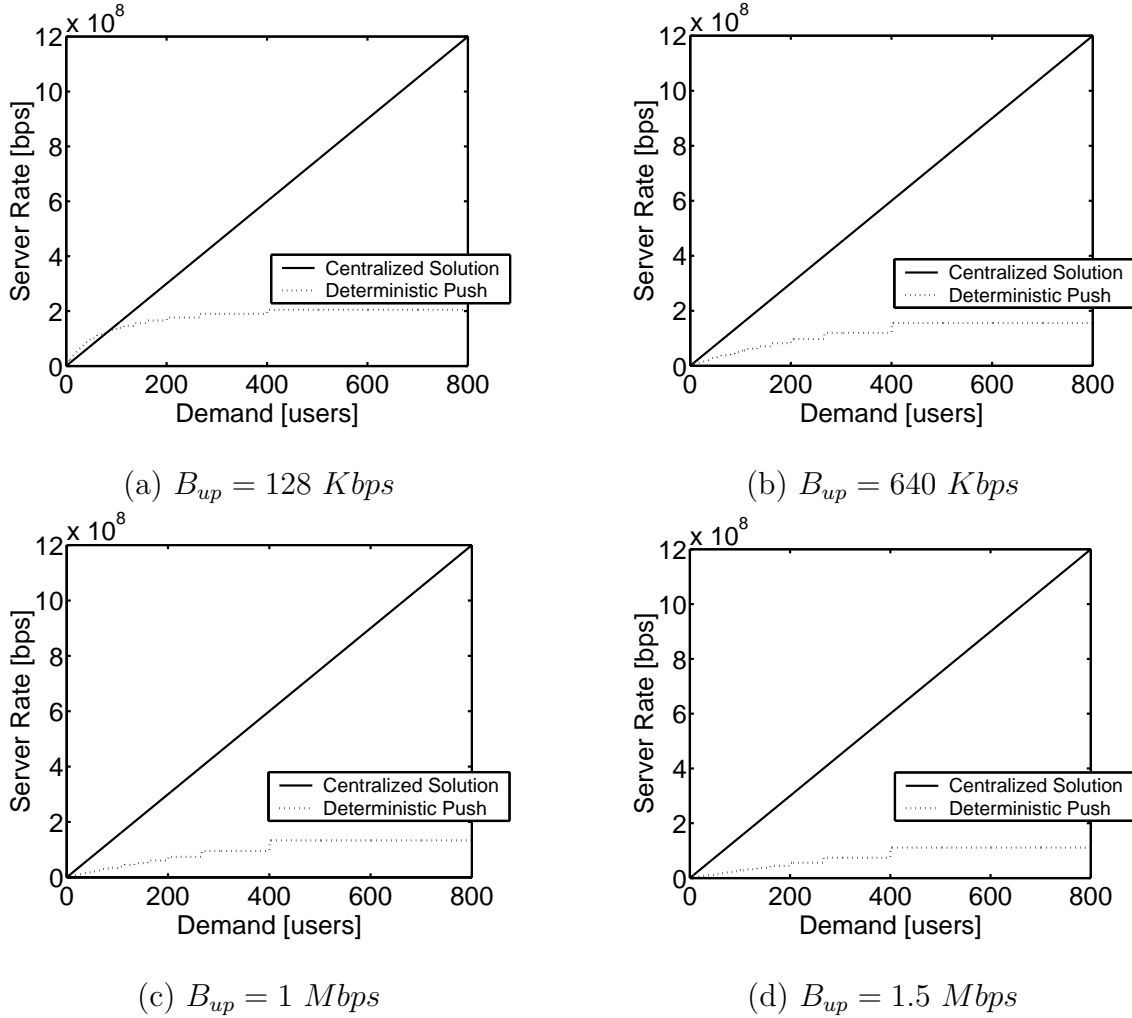


Figure 4.22: R_S as a function of the demand - Centralized vs Push-to-Peer - $L = 1 \text{ GB}$, $R_v = 1.5 \text{ Mbps}$, $N = 800$ - $D_{up} = 1$, $N_w = 1$

We consider the amount of data to push starting from the values of Figure 4.21. This analysis is more complete compared to the previous one, since take into account the actual cost of the architecture. In fact the content provider needs to pay a link according to the target $R_S(n_a)$. Moreover the value of $R_S(n_a)$ has an impact also on the cost of the server (cluster of servers).

From Figure 4.22 we see the great improvements we can have via a Push-to-Peer architecture. Also for $B_{up} = 128 \text{ Kbps}$, the minimum for nowadays ADSL connections, there is only a small interval of n_a where the centralized solution performs better. Then if we increase the demand value we see a big gap between the two solutions.

A more complete cost model to compare the two architectures should take into account the cost of the storage equipments and the cost of the bandwidth at different hours of the day. However we believe that this first analysis already gives interesting insight with respect to the cost of the two architectures.

Chapter 5

Conclusion and Future Work

Given the increasing popularity of VoD service, servers starts being overwhelmed by the volume of requests. In this context we suggested to introduce a p2p architecture. We presented the Push-to-Peer system, an architecture for the distribution of VoD service which leverages on a p2p architecture. We showed the feasibility of such an architecture and we defined the design-space. Moreover we showed the reduction on both load and cost compared to a centralized solution.

More generally we showed the benefits of a combined server/Peer-to-Peer solution. Peer-to-peer solutions can be used as an extension to centralized server-based solutions. I.e. combine some servers with p2p to get the best of both worlds at moderate cost.

A future work foresees the extension of the analytical model to the distribution of multiple movies. The interesting research issue that arises is how to match users' interest with the adopted push technique.

In a single movie distribution all users are interested in the same content. The goal of the push phase is to achieve to serve a certain demand. Extending the system to multiple movies means to add as parameter the interests of users. The goal of the push phase becomes to serve a certain demand minimizing the waste of data to push. A user should store a percentage of a movie proportional to the probability he has to watch it.

We are currently working on the definition of a test-bed for the Push-to-Peer system. Starting from the analytical results we got, we are going to implement a scheme based on a Deterministic Push. The goal is to evaluate its performance in presence of a realistic arrival pattern.

Bibliography

- [1] Arnaud Legout, Guillaume Urvoy-Keller, Pietro Michiardi. "Understanding BitTorrent: An Experimental Perspective". INRIA, Sophia Antipolis, July 2005.
- [2] Andrew Parker. "P2P in 2005". <http://www.cachelogic.com/>.
- [3] J. Murali. "Open-content distribution via P2P". <http://www.hinduonnet.com>.
- [4] <http://www.bittorrent.com/index.html>.
- [5] <http://www.akamai.com/>.
- [6] <http://www.coralcdn.org/>.
- [7] Christos Gkantsidis, Pablo Rodriguez. "Network Coding for Large Scale Content Distribution". IEEE Infocom, 2005.
- [8] Xinyan Zhang, Jiangchuan Liu, Bo Liz, Tak-Shing Peter Yum. "CoolStreaming/DONet: A Data-Driven Overlay Network for Efficient Live Media Streaming". IEEE INFOCOM, 2005.
- [9] "PPstream" <http://www.ppstream.com/>.
- [10] "Coolstreaming" <http://www.coolstreaming.us/>.
- [11] J. Liu, J. Xu. A survey of streaming media caching. IEEE communication magazine, 2004.
- [12] Pablo Rodriguez, Ernst W. Biersack. "Dynamic Parallel Access to Replicated Content in the Internet". Infocom, 2000.
- [13] Michael J. Freedman, Eric Freudenthal, David Mazieres. "Democratizing content publication with Coral". New York University. <http://www.scs.cs.nyu.edu/coral/>.

- [14] Hongliang Yu, Dongdong Zheng, Ben Y. Zhao, Weimin Zheng. "Understanding User Behavior in Large-Scale Video-on-Demand Systems". EuroSys, 2006.
 - [15] <http://www.napster.com/>.
 - [16] P. Maymounkov, D. Mazieres. "Rateless codes and big downloads". 2nd International Workshop on Peer-to-Peer Systems, 2003.
 - [17] M. Luby. "Lt codes". 43rd IEEE Symposium on Foundations in Computer Science, Nov. 2002.
-