Caching and prefetching for efficient video services in mobile networks

Ali Gouta

▶ **To cite this version:**

Ali Gouta. Caching and prefetching for efficient video services in mobile networks. Distributed, Parallel, and Cluster Computing [cs.DC]. University of Rennes 1, 2015. English. NNT : . tel-01256966

HAL Id: tel-01256966
https://inria.hal.science/tel-01256966

Submitted on 15 Jan 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

ANNÉE 2015

THÈSE / UNIVERSITÉ DE RENNES 1
sous le sceau de l'Université Européenne de Bretagne

pour le grade de

DOCTEUR DE L'UNIVERSITÉ DE RENNES 1

Mention : Informatique

Ecole doctorale Matisse

présentée par

Ali Gouta

Préparée à l'unité de recherche UMR 6074 IRISA
Equipes d'accueil : Orange Labs IDA/CDS - Irisa/ASAP
Convention Cifre N 1326/2011

Caching and prefetching for efficient video services in mobile networks

Thèse soutenue à Rennes
le 15 janvier 2015

devant le jury composé de :

Felber Pascal
Professeur - Université de Neuchâtel / *rapporteur*
Noel Crespi
Professeur - Télécom Sud Paris / *rapporteur*
Guillaume Pierre
Professeur - Irisa / *examinateur*
David Hausheer
Professeur - université de Darmstadt / *examinateur*
Anne Marie Kermarrec
Directrice de recherche - Irisa / *directeur de thèse*
Yannick Lelouédec
Ingénieur - Orange Labs / *co-directeur de thèse*

Wings are a constraint that makes

it possible to fly.

— Robert Bringhurst

To my parents. . .

Acknowledgments

I am most grateful to my advisors, Yannick Le louedec and Pr. Anne-Marie Kermarrec. Their guidance and insights over the years have been invaluable to me. I feel especially fortunate for the patience that they have shown with me when I firstly stepped into the field of data analysis, caching and prefetching in mobile networks. I am indebted to them for teaching me both research and writing skills. Without their endless efforts, knowledge and patience, it would have been extremely challenging to finish all my dissertation research and Ph.D study in 3 years. It has been a great honor and pleasure for me to do research under their supervision. I would like to thank Pr. Guillaume Pierre, Pr. David Hausheer, Pr. Noel Crespi and Pr. Pascal Felber for serving as my Ph.D committee members and reviewing my dissertation. I also owe thanks to Zied Aouini and Saifallh Ksibi, who helped me to address technical and scientific problems. Before coming to the University of Rennes 1, I received my engineering degree from Sup'Com in Tunis. I thank Dr. Yvon Ghourant for introducing me to the field of computer networking and for helping me to complete my thesis on computer networks.

I must thank my family in Tunisia, who supported me a lot. Without their endless love and encouragement I would have never completed this dissertation.

Abstract

Cellular networks have witnessed phenomenal traffic growth recently fueled by new high speed broadband cellular access technologies. This growth is in large part driven by the emergence of the HTTP Adaptive Streaming (HAS) as a new video delivery method. In HAS, several qualities of the same videos are made available in the network so that clients can choose the quality that best fits their bandwidth capacity. This strongly impacts the viewing pattern of the clients, their switching behavior between video qualities, and thus beyond on content delivery systems.

Our first contribution consists in providing an analysis of a real HAS dataset collected in France and provided by the largest French mobile operator. Firstly, we analyze and model the viewing patterns of VoD and live streaming HAS sessions and we propose a new cache replacement strategy, named WA-LRU. WA-LRU leverages the time locality of video segments within the HAS content. We show that WA-LRU improves the cache hit-ratio mostly at the loading phase while it reduces significantly the processing overhead at the cache.

In our second contribution, we analyze and model the adaptation logic between the video qualities based on empirical observations. We show that high switching behaviors lead to sub optimal caching performance, since several versions of the same content compete to be cached. In this context we investigate the benefits of a Cache Friendly HAS system (CF-DASH) which aims at improving the caching efficiency in mobile networks and to sustain the quality of experience of mobile clients. We evaluate CF-dash based on trace-driven simulations and test-bed experiments. Our validation results are promising. Simulations on real HAS traffic show that we achieve a significant gain in hit-ratio that ranges from 15% up to 50%.

In the second part of this thesis, we investigate the mobile video prefetching opportunities. Online media services are reshaping the way video content is watched. People with similar interests tend to request same content. This provides enormous potential to predict which

content users are interested in. Besides, mobile devices are commonly used to watch videos which popularity is largely driven by their social success. We design a system, named "Central Predictor System (CPsys)", which aims at predicting and prefetching relevant content for each mobile client. To fine tune our prefetching system, we rely on a large dataset collected from a large mobile carrier in Europe. The rationale of our prefetching strategy is first to form a graph and build implicit or explicit ties between similar users. On top of this graph, we propose the Most Popular and Most Recent (MPMR) policy to predict relevant videos for each user. We show that CPSys can achieve high performance as regards prediction correctness and network utilization efficiency. We further show that CPSys outperforms other prefetching schemes from the state of the art. At the end, we provide a proof-of-concept implementation of our prefetching system.

Key words: caching, HAS, DASH, HLS, prefetching, collaborative filtering, mobile networks, passive measurements, dataset.

Résumé

Les réseaux cellulaires ont connu une croissance phénoménale du trafic alimentée par les nouvelles technologies d'accès cellulaire à large bande. Cette croissance est tirée en grande partie par le trafic HTTP adaptatif streaming (HAS) comme une nouvelle technique de diffusion de contenus audiovisuel. Le principe du HAS est de rendre disponible plusieurs qualités de la même vidéo en ligne et que les clients choisissent la meilleure qualité qui correspond à leur bande passante. Chaque niveau d'encodage est segmenté en des petits vidéos qu'on appelle *segments* ou *chunks* et dont la durée varie entre 2 à 10 secondes. L'émergence du HAS a introduit des nouvelles contraintes sur les systèmes de livraison des contenus vidéo en particulier sur les systèmes de cache. Dans cette thèse, nous nous intéressons à l'étude de cet impact et à proposer des algorithmes et des solutions qui optimisent les fonctionnalités de ces systèmes. D'autre part, la consommation des contenus est fortement impactée par les nouvelles technologies du Web2.0 tel que l'émergence des réseaux sociaux. Dans cette thèse, nous exploitons les réseaux sociaux afin de proposer un service de préchargement des contenus VoD sur terminaux mobiles. Notre solution permet l'amélioration de la QoE des utilisateurs et permet de bien gérer les ressources réseaux mobile.

Nous listons nos contributions comme suit :

Notre première contribution consiste à mener une analyse détaillée des données sur un trafic HAS réel collecté en France et fournie par le plus grand opérateur de téléphonie mobile du pays. Tout d'abord, nous analysons et modélisons le comportement des clients qui demandent des contenus catch-up et live. Nous constatons que le nombre de requêtes par segment suit deux types de distribution : La loi log-normal pour modéliser les 40 premiers chunks par session de streaming, ensuite on observe une queue qui peut être modélisé par la loi de Pareto. Cette observation suggère que les clients ne consomment pas la totalité du contenu catch-up. On montre par simulation que si le cache implémente des logiques de caching qui

ne tiennent pas en compte les caractéristiques des flux HAS, sa performance diminuerait considérablement.

Dans ce contexte, nous proposons un nouvel algorithme de remplacement des contenus que nous appelons Workload Aware-LRU (WA-LRU). WA-LRU permet d'améliorer la performance des systèmes de cache en augmentant le Hit-Ratio en particulier pour les premiers segments et en diminuant le temps requis pour la mise à jour de la liste des objets cachés. En fonction de la capacité du cache et de la charge du trafic dans le réseau, WA-LRU estime un seuil sur le rang du segment à cacher. Si le rang du chunk demandé dépasse ce seuil, le chunk ne sera pas caché sinon il sera caché. Comme WA-LRU dépend de la charge du trafic dans le réseau, cela suppose que le seuil choisit par WA-LRU est dynamique sur la journée. WA-LRU est plus agressif pendant les heures chargées (i.e. il cache moins de chunks, ceux qui sont les plus demandés) que pendant les heures creuses où le réseau est moins chargé.

Dans notre deuxième contribution, nous étudions plus en détail les facteurs qui poussent les clients HAS à changer de qualité lors d'une session vidéo. Nous modélisons également ce changement de qualité en se basant sur des données empiriques provenant de notre trace de trafic. Au niveau du cache, nous montrons que le changement fréquent de qualité crée une compétition entre les différents profiles d'encodages. Cela réduit les performances du système de cache. Dans ce contexte, nous proposons Cache Friendly-DASH (CF-DASH), une implémentation d'un player HAS compatible avec le standard DASH, qui assure une meilleure stabilité du player. Nous montrons à travers des simulations et des expérimentations que CF-DASH améliore expérience client et permet aussi d'atteindre un gain significatif du hit-ratio qui peut varier entre 15% à 50%.

Dans la deuxième partie de cette thèse, nous proposons un système de préchargement de contenus vidéos sur terminaux mobile. La consommation des contenus vidéo en ligne est fortement impactée par les nouvelles technologies du Web2.0 et les réseaux sociaux. Les personnes qui partagent des intérêts similaires ont tendance à demander le même contenu. Cela permet de prédire le comportement des clients et identifier les contenus qui peuvent les intéresser. Par ailleurs, les smartphones et tablettes sont de plus en plus adaptés pour visionner des vidéos et assurer une meilleure qualité d'expérience. Dans cette thèse, nous concevons un système qu'on appelle CPSys (Central Predictor System) permettant d'identifier les vidéos les plus pertinentes pour chaque utilisateur. Pour bien paramétrer notre système

de préchargement, nous analysons des traces de trafic de type User Generated Videos (UGC). En particulier, nous analysons la popularité des contenus YouTube et Facebook, ainsi que l'évolution de la popularité des contenus en fonction du temps. Nous observons que 10% des requêtes se font sur une fenêtre de temps d'une heure après avoir mis les vidéos en ligne et 40% des requêtes se font sur une fenêtre de temps de un jour. On présente aussi des analyses sur le comportement des clients. On observe que la consommation des contenus vidéo varie significativement entre les clients mobiles. On distingue 2 types de clients :

- les grands consommateurs : Ils forment une minorité mais consomment plusieurs vidéos sur une journée.
- les petits consommateurs : Ils forment la majorité des clients mais consomment quelques vidéos par jour voir sur une période plus longue.

On s'appuyant sur ces observations, notre système de préchargement adapte le mode de préchargement selon le profil utilisateur qui est déduit à partir de l'historique de la consommation de chaque client.
Dans un premier temps, CPSys crée un graphe regroupant les utilisateurs qui sont similaires. Le graphe peut être soit explicite (type Facebook) ou implicite qui est construit à la base des techniques de colllaborative filtering dérivés des systèmes de recommandations. Une fois le graphe est créé, nous proposons la politique Most Popular Most Recent (MPMR) qui permet d'inférer quel contenu doit-on précharger pour chaque utilisateur. MPMR trie les vidéos candidats selon la popularité locale du contenu définit comme le nombre de vues effectués par les voisins les plus similaires, ensuite MPMR donne la priorité aux contenus les plus frais. Nous montrons que CPSys peut atteindre des performances élevées par rapport à d'autres techniques présentées dans l'état de l'art. CPSys améliore la qualité de la prédiction et réduit d'une manière significative le trafic réseau.
Finalement, nous développons une preuve de concept de notre système de préchargement.

Contents

Acknowledgments **i**

Abstract **iii**

List of figures **xiii**

List of tables **xv**

Publications **xvii**

1 Introduction **1**

1.1 The early days of content delivery over the internet 1

1.1.1 CDN 1.0 1

1.1.2 CDN 2.0 1

1.1.3 Streaming over HTTP 2

1.1.4 Video streaming in mobile networks 3

1.2 Why should we care and what can we do? 3

1.2.1 Growing trend of mobile video traffic VS ISP investments 3

1.2.2 User-engagement and QoE 4

1.3 Contributions 5

1.4 Thesis Organization 6

2 Traffic measurements, caching and prefetching: A review of the literature **9**

2.1 Data collection: Measurements and analysis 9

2.1.1 Data collection 10

2.1.2 Video streaming analysis 12

2.1.3 Positioning 14
2.2 Adaptive steaming over HTTP 15
2.2.1 HAS methods 16
2.2.2 Stability of HAS players 18
2.2.3 Positioning 19
2.3 Caching 20
2.3.1 Replacement strategies in context of HAS 20
2.3.2 Positioning 22
2.4 Prefetching 23
2.4.1 Prefetching systems in context of video streaming 24
2.4.2 Positioning 25

3 HTTP adaptive streaming in mobile networks : characteristics and caching opportunities 27

3.1 Introduction 27
3.2 The Dataset overview 28
3.2.1 Data collection 28
3.2.2 Content types 28
3.2.3 Data processing 29
3.2.4 Fields description 31
3.3 Clients' behavior analysis 32
3.3.1 Distribution of requested chunks per session 32
3.3.2 Analayis on user engagement 33
3.4 Caching HAS content 36
3.4.1 Presentation of WA-LRU 36
3.4.2 WA-LRU in action 38
3.4.3 Pseudo-code of WA-LRU 39
3.4.4 Evaluation 40
3.5 Conclusion 42

4 Improving caching efficiency and quality of experience with CF-Dash 43

4.1 Introduction 43

4.2 Analysis on the adaptation logic in HAS 44
4.2.1 Profiles in catch-up and live sessions 44
4.2.2 Video bitrate adaptation 45
4.2.3 Impact of HAS on caching performance 46
4.2.4 Markov characterization of the switching between profiles 48
4.3 Motivation to CF-DASH 51
4.3.1 Empirical study summary 51
4.3.2 QoE evaluation 52
4.4 Cache Friendly-Dash 53
4.4.1 Cache Friendly-Dash in a nutshell 53
4.4.2 PoC implementation 54
4.5 Evaluation 56
4.5.1 Simulation evaluation 56
4.5.2 Experiments evaluation 58
4.6 Conclusion 60

5 CPSys: A system for mobile video prefetching 61
5.1 Introduction 61
5.2 Background and related works 62
5.3 Traffic analysis 64
5.3.1 Dataset 64
5.3.2 Distribution of the number of views per video 65
5.3.3 Distribution of the number of views per user 65
5.3.4 Relationship between request frequency and request inter-arrival rate . 66
5.3.5 Video lifetime distribution 68
5.3.6 Load variation across the day 69
5.4 System Design 70
5.4.1 What to prefetch? 71
5.4.2 When to prefetch? 75
5.4.3 How many videos to prefetch? 76
5.5 Trace-driven simulation experiments 78

5.5.1 Simulation setup 79
5.5.2 Performance analysis 80
5.6 Prototype implementation 83
5.7 Conclusion 85

6 Conclusion & Perspective **87**
6.1 Achievements 87
6.1.1 Traffic analysis 87
6.1.2 Caching HAS videos 89
6.1.3 Prefetching videos on mobile devices 89
6.2 Future work 90
6.2.1 Future works on data analysis 90
6.2.2 Future works on caching 90
6.2.3 Future works on prefetching 91

Bibliography **104**

List of Figures

2.1 HLS configuration 16

2.2 DASH standard 18

3.1 Hosting servers for both catch-up content videos and live channels 29

3.2 Profiles presentation used for catch-up contents 30

3.3 Distribution of number of chunks per session for live and catch-up HAS sessions 33

3.4 Download throughput and Profiles interdependencies 34

3.5 Impact of the Delays in chunks delivery on the playing-time 35

3.6 CDFs of number of chunks per session when $D < 1$ and $D > 1$ 36

3.7 Workload pattern measured by the logging system from 08/11/2012 to 15/11/2012 37

3.8 Scenario 1: comparison between θ and $chunk^{A}_{index}$ 39

3.9 Scenario 2: comparison between θ and $|chunk^{A}_{index} - chunk^{B}_{index}|$ 39

3.10 Average hit-ratio 40

3.11 *Metrics evaluation: Average hit-ratio, update-ratio and average hit-ratio per chunk position* 41

4.1 Proportion of sessions requesting $profile_i$ 44

4.2 Distribution of requested profiles with respect to the $chunk_{index}$ for both catch-up and live video streaming 45

4.3 Switching during HAS sessions 47

4.4 single profile VS multi-profile: Implication on caching efficiency 48

4.5 Percentage of contents in which clients request different profiles when requesting the same $chunk_{index}$ 50

4.6 Average and Gain in hit ratio 57

4.7 *Metrics evaluation: Average hit-ratio,* G_H 57
4.8 Profiles distribution 59
4.9 Evaluation: Hit ratio and stability 60

5.1 Video popularity distribution on 3 different measurement periods: 1 day, 1 week and 1 month, starting from 03/28/2014 65
5.2 Number of views per user on 3 different measurement periods: 1 day, 1 week and 1 month, starting from 03/28/2014 66
5.3 YT+FB data 66
5.4 a: Relationship between the daily request frequency and the daily request inter-arrival rate on 01/13/2014; b and c : on the week starting from 01/13/2014 . . . 67
5.5 Lifetime distribution of videos made available on January 8, 2014 68
5.6 (a) Number of active sessions across the day; (b) Zoom in during peak hours . . 70
5.7 CPSys design 71
5.8 QNotif: Data structure which holds the prefetching candidates 74
5.9 Transition-state control mechanism running on the prefetcher agent 76
5.10 CPR with different content seleciton policies 81
5.11 Overhead with different content selection policies 81
5.12 FNR with different content selection policies 82
5.13 CPR 83
5.14 Overhead 83
5.15 Snapshots from CPClient 84

List of Tables

2.1 Datasets description 14

3.1 Breakdown of the number of proposed profiles per HAS content 30
3.2 Profiles settings 31
3.3 Empirical models 32

4.1 Sojourn time distribution 49
4.2 MoS of the perceived quality of experience 53

5.1 Properties of the two used datasets 64
5.2 Traffic trace used for the simulation 80

Publications

Gouta, A., Hong, D., Kermarrec, A. M., & Lelouedec, Y. (2013, August). HTTP adaptive streaming in mobile networks: characteristics and caching opportunities. In *Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2013 IEEE 21st International Symposium on* (pp. 90-100). IEEE.

Gouta, A., Hong, C., Hong, D., Kermarrec, A. M., & Lelouedec, Y. (2013, June). Large scale analysis of HTTP adaptive streaming in mobile networks. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2013 IEEE 14th International Symposium and Workshops on a* (pp. 1-10). IEEE.

Amann, N., Gouta, A., Hong, D., Kermarrec, A. M., & Le Louedec, Y. (2013). Large scale analysis of HTTP Adaptive Streaming over the Mobile Networks. *15èmes Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications (AlgoTel)*, 1-4.

Aouini, Z., Diallo, M. T., Gouta, A., Kermarrec, A. M., & Lelouedec, Y. (2014, March). Improving caching efficiency and quality of experience with CF-Dash. In *Proceedings of Network and Operating System Support on Digital Audio and Video Workshop* (p. 61). ACM.

1 Introduction

Internet forms a universe of information accessible via networked devices. Today, it becomes clear that distributing, delivering and servicing content have become the first challenge of today's internet. In this introduction, we survey the history of Content Delivery Networks (CDNs) and multimedia streaming over the internet and illuminate the key challenges raised by the new content delivery constraints.

1.1 The early days of content delivery over the internet

1.1.1 CDN 1.0

Back in the nineties, the first generation of CDN - named CDN 1.0 here - emerged to overcome the web's bottlenecks and slowdowns. CDN 1.0 was designed to serve mainly the frequently requested and static web content. That was achieved by moving and staging the content at an edge server close to the end users.

1.1.2 CDN 2.0

Over the past ten years, as storage capacity and bandwidth increased, videos started to be embedded in web sites and content providers gradually shifted their business model towards media content delivery. This allowed the emergence of Over The Top (OTTs) video providers since the 2000s, such as the introduction of a new DVD rental business by Hulu in 2007, and in the emergence of User Generated Content (UGC) services such as YouTube. So far, the BBC called the 2012 London Olympics the first truly digital games and streamed over 2.8 petabytes of data in one single day. CDNs had to evolve during this period to support the huge traffic growth caused by media delivery. A new generation of CDNs, we name CDN2.0, has thus

ermeged and has become a cornerstone of the current Internet.

In this evolutionary step, the most important features that were introduced in CDN2.0 were the focus on real time media delivery and caching functionality optimization. A key step to meet these requirements was to revisit the streaming protocols. Initially, the RTSP/RTP stack was considered the most prevalent protocol for streaming video, but it turned out to have issues such as to be blocked at routers or firewall settings. In this context, HTTP streaming had the potential to make a significant inroad to replace RTSP/RTP to deliver media content. It was gradually being adopted by CDNs for its low deployment costs, simplicity and flexibility.

1.1.3 Streaming over HTTP

In [39], authors report that today 98% of multimedia streaming traffic is delivered over HTTP. Streaming over HTTP has known multiple step evolution. The first wave of HTTP-based video streaming applications used the simple progressive download method, in which a TCP connection simply transfers the entire movie file as quickly as possible. The shortcomings of that approach are many, however. One major issue is that all clients receive the same encoding of the video, despite the large variations in the underlying available bandwidth both across different clients and over the time for the same client. This has recently led to the development of a new wave of HTTP-based streaming applications that we refer to as *HTTP Adaptive Streaming (HAS)*. Motivated by the appealing opportunities of such a content delivering scheme, several major players have developed their own implementations: Microsoft smooth streaming, Adobe OSPF, HTTP Live streaming (apple), GPAC, Move networks. In adaptive streaming, the server maintains multiple profiles of the same video, encoded at different bitrates and quality levels. Further, each profile is then segmented into smaller parts, called chunks or segments, which usually correspond to 2 to 10 second of video content. When a user requests a video content, the hosting server sends back to the client a formalized description of the media presentation, named manifest file or Media Presentation Description (MPD), describing all available profiles of the requested content. The MPD enables the client-player to choose the quality that matches best with the network state and device capacities.

Today, HTTP live streaming is the most popular streaming protocol used for this type of streaming transfer. Since 2010, MPEG started working on a new HAS standard. Later, in 2012, the first MPEG-Dash (Dash1.0) standard was published. It turns out that Dash1.0 did not

highlight the role of content delivery actors -typically CDNs and ISPs- within the Internet content delivery landscape. In July 2013, MPEG started working on the second edition of DASH (Dash2.0). This second edition addresses various issues related to HAS, among them was the impact on the Internet delivery infrastructures such as CDNs and caches.

1.1.4 Video streaming in mobile networks

The proliferation of mobile smartphones has significantly changed the landscape of mobile usage. Smartphones are now designed to provide end users with high quality video experience, and they are commonly used by users to watch TV-programs either in live or in replay. Cisco forecasts that mobile video will grow at a compound annual growth rate of 90% between 2011 and 2016. All these factors suggest that mobile streaming services will soon dominate the mobile communication landscape. Hence, ensuring a fast and a reliable content delivery accounting for users' network constraints is critical in this context. In this dissertation, we set out to discover the major challenges that mobile and CDN operators could incur while delivering video content in mobile ecosystem. Then, we provide guidelines and novel strategies to improve caching systems and to sustain the QoE.

1.2 Why should we care and what can we do?

In this section, we underline and discuss 2 major challenges CDN market players or service providers faces: The first is how to reduce or postpone investments in network and CDN infrastructures and the second is how to provide a high quality experience for every viewer, anytime, anywhere, on any device.

1.2.1 Growing trend of mobile video traffic VS ISP investments

Why is it important?

Recent studies point out the explosive growth in mobile data demand driven by wireless devices and video services [51]. This growth is in large part driven by mobile video applications and subsequently drives telcos to invest in their networks to keep pace with the increasing need for resources [1]. According to recent estimates [1], mobile carriers in US invested 75 billions of dollars in network infrastructure in 2012. This suggests that a small saving in traffic

[1] http://www.fiercewireless.com/tech/press-releases/us-carriers-will-outlay-105-billion-network-investments-2013-they-chase-4g

can therefore correspond to save millions of dollars.

What can we do?

Caching is still to be a prominent solution and a fundamental building block of the internet to offload the network and to achieve savings. However with the emergence of HAS, caching strategies should be reviewed. Yet few is known about the characteristics of this class of traffic and its network cost implications that may consequently arise. One prominent solution would be to investigate these characteristics and to design new cache replacement strategies tailored to the HAS properties.

1.2.2 User-engagement and QoE

Why is it important?

According to the 2013 Conviva Viewer Experience Report [2], poor quality streaming video solutions has resulted in an estimated revenue loss of 2.16 Billion of dollars for the media industry in 2012. Today encoding bitrates ranges from low bitrates typically to fit small screen sizes such as handled devices to Ultra High Definition typically 4K and 8K. For instance, Netflix moved beyond HD and started streaming the 4K video format. The increase in the number of encoding profiles within the network leads to new constraints on users' experience and caching efficiency. This is particularly challenging for content delivery actors and much more constraining for mobile carriers, since resources are expensive and bandwidth can significantly fluctuate over short periods. This makes the video delivery optimization more subtle, yet more challenging.

What can we do?

The following key actions briefly outline promising directions for improving the user QoE.

Characterizing HAS traffic To identify the challenges raised by HAS, a key step is by developing a deep understanding of how HAS videos are being consumed by clients. For instance, the knowledge of the video encoding distribution and video session abandonment distribution (number of chunks requested during a video session) could help to better understand and evaluate how much caching optimization opportunity one can achieve. This also allows identifying the appropriate optimization techniques and how to implement them.

Caching and HAS players stability We infer from the mobile data trace several key factors affecting the QoE and caching performance. Stability of the client player (switching between video qualities) has a straight implication on caching performance. In this thesis, we address the stability issue and propose a novel mechanism to reduce the switching rate between profiles and sustain the QoE.

Prefetching videos to reduce the response time perceived by users In context of content distribution, prefetching has potential benefits as it reduces the start-up delays and enables a smooth playback. However, prefetching is usually accompanied with costs. In this thesis, we deeply investigate the costs-benefits trade-off problem raised by mobile video prefetching.

1.3 Contributions

Adaptive streaming over HTTP has changed the way videos are delivered over the Internet. The caching paradigm needs to be revisited to meet the challenges raised by HAS. We therefore start to characterize the adaptive streaming traffic by collecting a real mobile traffic traces. Based on this trace, we uncover several findings that could be leveraged by content delivery networks to improve their caching strategies.
This thesis consists of two parts. In the first part, we investigate properties of HAS, then we propose 2 approaches to improve the caching performance and the user experience.
The first approach is driven by the video consumption pattern that we infer from the traffic traces:

- **HTTP Adaptive Streaming in Mobile Networks: Characteristics and Caching Opportunities:** We show that the distribution of the number of requested chunks per video session is a piece-wise combination of Log-normal and Pareto distributions. The Pareto distribution infers that few of users request the very last chunks of HAS videos. We show how this leads to a sub-optimal caching performance. Besides, we revisit the user-engagement based on several key metrics. Then, we propose a new caching strategy; we name it WA-LRU. WA-LRU infers from the state of the network which chunk should be cached and the ones that should be left away.

In a second step, we turn our interest towards the video quality adaptation logic:

- **Improving caching efficiency and quality of experience with CF-Dash:** In HAS, a wide range of video bitrates of the same video content are made available over the internet so that clients' players pick the video bitrate that best fit their bandwidth. Considering a very large number of clients switching between video bitrates, we show that this adversely affects the performance of CDNs and transparent caches since several versions of the same content compete to be cached. Relying on a large dataset, we analyze the frequency of switching between video qualities during HAS sessions, then we assess and quantify the performance of the cache through simulations. Then we carry out a subjective quality perception tests over 26 individual user. We observe that the user-engagement is guaranteed starting from a specific encoding profile. We leverage both findings and investigate the benefits of a Cache Friendly HAS player (CF-DASH), which aims at improving the caching efficiency and sustaining the quality of experience of mobile clients. We evaluate CF-Dash through simulations and prototype experiments.

The second part of this dissertation is dedicated to the design, evaluation and implementation of a mobile video prefetching system.

- **CPSys: A Centralized Predictor System to Prefetch Mobile Videos:** CPSys leverages users' preferences and machine learning techniques to predict relevant videos for each user. The design principles of CPSys is guided by findings and empirical analysis we have carried out on users' behavior. We then evaluated CPSys based on real traffic with millions of views. At the end, we provide a proof-of-concept implementation and first steps towards a large scale deployment of CPSys.

1.4 Thesis Organization

The dissertation is structured as follows:

In Chapter 2, we provide an overview of prior works related to the research areas we highlight in this thesis. First, we give a brief outline of the the most used data collection techniques and discuss some related works on traffic analysis and characterization. Then we overview several proposed caching algorithms and prefetching schemes and expose their strengths and weaknesses. For each research field we cite representative works to illustrate the range of

issues that have been addressed and position our contributions in that respect.

In Chapter 3, we introduce and evaluate a novel cache replacement strategy, we name it Workload Aware-LRU(WA-LRU) and show that it outperforms traditional LRU algorithm.

In Chapter 4, we deepen our analysis and investigate the rate-adaptation paradigm in HAS. We identify the trade-offs raised by the HAS quality-adaptation logic. More precisely, we show that the high frequency of rate adaptation adversely affects the cache performance which subsequently leads to a sub-optimal streaming performance. We then propose CF-Dash, a cache friendly Dash mechanism to improve caching efficiency and sustain the QoE.

In Chapter 5, we propose CPSys, a Central Predictor System to prefetch videos on clients' mobile devices. In this design we answer 3 fundamental questions: *What to prefetch*? *When to prefetch*? *How much to prefetch*?

Finally in Chapter 6, we conclude this dissertation by summarizing our main contributions and discuss remaining open questions.

2 Traffic measurements, caching and prefetching: A review of the literature

There is a vast literature on dataset collection and analysis, traditional web or media caching. However, previous caching strategies are not necessarily efficient with respect to today's media streaming technologies. The widespread adoption of HAS has raised new challenges to content distribution networks. Therefore we believe that revisiting caching strategies is ever more relevant now than before.

Additionally, prefetching has been advocated by researches to reduce latency and increase the user experience. However, prefetching has been broadly investigated in context of peer-assisted video streaming. Yet, very little work has been done in context of mobile video prefetching.

The purpose of this chapter is to give a state-of-the art overview of the research topics we highlight in this thesis. In section 2.1, we overview related works on datasets collection methodologies and analysis. In section 2.2, we survey different HAS implementations and overview the DASH standard. In section 2.3, we overview existing caching strategies and in section 2.4 we overview related works on prefetching systems, their design and performance.

2.1 Data collection: Measurements and analysis

Since the last decade, data collection and mining have become a common practice for the research community and networking actors to monitor the network and to provide a deep understanding on traffic characteristics. ISPs supply their networks with NMIs (Network measurement infrastructure) to continuously monitor the network performance and follow-up the dynamics of traffic flows. On the other hand, research communities always try to provide a comprehension of peculiar traffic patterns. So far, this helps modeling and characterizing

the traffic properties. In this section, we shed light on methodologies and techniques used to collect data, then we provide insights into related works on video traffic analysis.

2.1.1 Data collection

Data collection is an important aspect as it helps us to collect, study, and provide a comprehension about a specific topic or research area. Inaccurate data collection leads to a bias regarding lessons inferred from the analysis.

In this thesis, we collected and analyzed several datasets. We used different techniques to collect the datasets. Therefore, in this section, we provide a survey of the existing data collection methods. Overall, we classify them into 2 types: *On-line* and *Off-line* measurements. The former consists in: active and passive measurements [26], while the latter consists of crawling and API extraction.

On-line data collection

On-line method requires probe nodes being deployed in the network. These nodes serve as to monitor and report feedback about the state of the network.

Active measurements Active measurements require injecting test packets in the network for testing performance. A wide range of tools are proposed and have been largely deployed in the network to actively monitor the network. In the literature many have been developed: Iperf [3],OWAMP [109], Pathchar [38], Pathload [52], etc. These tools are designed to routinely monitor key performance metrics of the network such as: packet loss, jitter, Round Trip Time (RTT) and connection bandwidth.

Passive measurements Active measurements techniques have shown their limitations across several studies [97, 48] especially when probing the entire cellular network. In comparison to the active measurements, the passive measurements do not require any packet injection. The passive probe system remains transparent to the network and is usually deployed at strategic points in the network. The system captures flows established between the end-user terminal and the server. Such systems are usually supplied with packet sniffing and filtering modules to determine the signature of the flow which subsequently allows classifying it.

ISPs are likely to instrument their network with passive monitoring tools since it is more

practical to monitor the entire network[48]. In [39, 48, 110], authors deployed the monitoring probe at the Gn interface between the Gateway GPRS Support Nodes (GGSN) and Serving GPRS Support Nodes (SGSN). For the 2 former studies, authors characterized the traffic flowing through AT & T mobile network. In [84], authors adopted the passive measurement approach to monitor the major mobile carrier in Austria. They monitored links at 3 different levels: 1-Gn interface between the SGSN-GGSN. 2-Gi interface between GGSNs and internet. 3- between peering links and edge routers linking a national ISP. So far, this study is the unique work we are aware of, in which authors provide such a complete picture of the monitoring system. In [98], authors used tstat [71] -an open source passive sniffer supplied with classification capabilities- to collect YouTube flows records. Records are collected from 5 locations spread across 2 European ISPs and 2 university campus networks. In [57], authors proposed Nornet Edge (NNE) a dedicated infrastructure for passive measurements and experiments in Mobile Broadband networks.

Off-line data-collection

API extraction Crawling and Rest-based APIs are two off-line methods largely used to collect data. Content providers -mainly Online Social Networks- such as Twitter, Facebook and YouTube are publicly providing a sheer amount of structured data related to their services. This enabled researchers to carry out measurements and data analysis to provide a deep understanding on patterns of clients' subscribing to their services.
In this context, YouTube has attracted most interest and was extensively studied by the research community. In most of the studies [40, 40, 24] authors used the API provided by YouTube to study the properties of YouTube videos.

Crawling extraction Alternatively to API method, crawling was extensively used when the required data can not be accessed via an API or revealed data is insufficient for subsequent analysis. Conceptually, crawling is a technique derived from web data extraction. It consists of sampling a graph where the source node is called a seeder, then the crawler visits the list of connected nodes to this source. This operation is iteratively repeated until the crawler collects a representative view of the entire graph.
In [29], authors crawled all YouTube videos in the Science category for six consecutive days to understand the characteristics of globally popular videos. In [72], authors crawled 4 OSNs,

namely Orkut, YouTube, flicker and LiveJournal. They combined API data extraction with crawling technique to collect datasets and study the properties of these OSNs.

2.1.2 Video streaming analysis

Now, we outline the existing VoD datasets that represent similar or complementary characteristics to the ones we have collected and analyzed in this thesis. We give insights into the data collection process, particularities and major findings in each of them.

VoD datasets

Over the last decade, video streaming over the internet has been widely investigated. There are many measurements study of VoD streaming services [17, 18, 28, 29, 89]. In [28, 29], authors analyzed properties of YouTube videos, they deepened their analysis on viewing, popularity and age distribution of YouTube videos. They inferred that the popularity distribution could be modeled by a piece wise distribution: Power and exponential decay lows and that user's preference seems relatively insensitive to video's age.

In [89], authors inferred that YouTube, Dailymotion, and Metacafe videos exhibit a Zipf-like with truncated tail popularity distribution and concluded that caching popular videos can significantly reduce the server load.

In [37], authors studied how video quality affects the user engagement when streaming on-demand and live content videos. They observed that 1% increase in buffering ratio can reduce the user engagement by 3 minutes, they also correlated the playing time with several quality metrics. In this work authors did not report analysis on quality adaptation logic and its impact on the user-engagement. Additionally, authors did not investigate the root cause affecting the user-engagement which we believe to be the network conditions.

In [50] authors collected data from the popular MSN video site and studied popularity of MSN videos. Authors have also explored the growth in demand and video bit rate increase over a period of 9 months. They observed that the distribution of the bitrate for MSN videos shifted from 200 kbps to 300 kbps and that the aggregated number of requests increased by 57.4% .

In [80], authors observed that for short videos of 3 minutes or less, users abort their video session at any moment, while for videos longer than 3 minutes, users either stop downloading early or download the video entirely. When the reception quality deteriorates, fewer videos are fully downloaded, and the decision to interrupt the download is taken earlier. They also

correlated the viewing behavior with the quality reception.

In [65], authors compared users' behavior when they access the VoD catalog from WiFi and 3G connections. They collected a large mobile dataset from a Chinese TV provider namely PPTV[4] and inferred that users' behavior exhibits a strong daily and weekly patterns. While analyzing the behavior of clients individually, they observed a concentration of interest and peculiar access patterns which helps classifying users and predict their behavior. They also studied video popularity aspect and inferred that it fits the Pareto distribution.

To this end, VoD systems have been largely studied. However, HTTP adaptive streaming is not yet deeply investigated. A few measurements and analysis have been done to characterize this class of video traffic. In the next paragraph, we enumerate the very few studies that have investigated the HAS traffic.

Adaptive streaming datasets

In [64, 63], authors collected traces from CNLive[5] a leading mobile TV service provider in china. CNlive provides a platform to distribute content for TV and radio stations to broadcast programs to smart phones and other devices. In both works, authors highlighted several key observations about clients' behavior. First, they reported that the playback length of videos can be separated into two distinct groups. The first group represents users' video browsing, where the average playback length is 4.1 second. The second group represents users' video viewing, with the average playback length being 185.17 second. Second, they modeled the video playback time of active viewers. The joining-phase is modeled by Weibull distribution, while the viewing phase is a piece-wise combination of log-normal and Pareto distributions. Besides, they reported that the sojourn time of 3G users are generally shorter than that of WiFi users. For example, 58.40% of the audio sessions are from 3G connections, but they only consume 48.34% of the audio sojourn time. They further made a comparison between IPTV systems and accessing TV from mobile devices.

In [39], authors provided insights into HLS traffic characteristics such as the average bitrate, video duration, and object size distribution. They additionally studied other HTTP-based streaming methods such as progressive download. In [69] authors studied characteristics of Netflix, Hulu and YouTube on Android and IOS based devices. They observed that video players implementations vary significantly across mobile platforms and network types and

this comes at the expense of CDNs and caching performance.

2.1.3 Positioning

We compile in Table 2.1 a taxonomy of datasets that have been studied in the state of the art. The table reports the service provider (column 1), the number of unique users in the datasets referred by U (column 2), the number of views referred by V (column 3), number of content referred by C (column 4), the duration of data collection(comlumn 5), the method used to collect the data, and the last column refers to the topic and purpose for which the data was collected. The last 3 rows report the datasets we collected and analyzed all along this thesis.

Service provider	U 10^3	C 10^3	V 10^6	Time (days)	Method	Topic
YouTube	-	252	539	6	crawling	Dataset characterization[29]
	16	303	0.6	14	passive	Traffic characterization[112]
Dailymotion	15	-	2	120	passive	Traffic analysis[27]
	-	1194	1795	14	crawling	Dataset characterization[73]
Yahoo!	-	99	770	1	crawling	Workload analysis[73]
Veoh	-	269	588	1	crawling	Workload analysis[73]
Metacafe	-	239	3076	1	crawling	Workload analysis[73]
PowerInfo	42	8	20	210	request log	VoD system analysis[108]
Hulu	-	2	0.01	3	passive	prefetching[56]
Netflix	480	18	100	2725	ratings log	Recommendation[20]
HAS	247	-	2	63	passive	HAS characterization and caching
YouTube UGC	3,179	10,676	65	111	passive and API	Prefetching
Facebook UGC	400	2,856	15	111	passive	Prefetching

Table 2.1: Datasets description

Most studies related to mobile workload characterization relied on datasets collected at the G_n interface of the mobile carrier. However, our passive probing system is installed at all G_i interfaces of the mobile carrier, i.e. above the G_n interface. This way we have succeeded in

collecting a large dataset, in which we captured all requests of all subscribers in France over several time periods.

Characterizing the HAS traffic from an operator standpoint Li and al. [63] were first to provide insights on HAS traffic characteristics. However, their study was limited to count the number of chunks requested during live streaming sessions. In this thesis, we provide a thorough characterization of HAS traffic. First, we characterize and model live and on-demand HAS traffic at per video-chunk level. Second, it is crucial for content providers to identify the factors that may affect the user engagement. Therefore, we quantify the user engagement at per-view and per-chunk level for both live and VoD streams. Third, we unveil one major HAS property which is the bitrate adaptation mechanism. We quantify and model the switching between bitrates based on empirical observations. Then, we study the implication of such transitions on network caching systems.

Providing guidelines to design a mobile video prefetching system We collected a YouTube (YT) and a Facebook (FB) dataset for a period of 94 days. The FB dataset consists of logs of users requesting *akamaized* Facebook UGC videos. These videos are requested from Facebook pages and hosted by Akamai CDN [6] and served on-demand. The YT dataset consists of logs of users requesting videos from YouTube. Most of the studies related to YouTube are coarse-grained. Yet authors carried out analysis of aggregated data to study YouTube videos characteristics. Instead, our analysis is fine-grained and design oriented. The purpose of our YT and FB analysis is to gain broad insights into the design of a mobile video prefetching system.

2.2 Adaptive steaming over HTTP

In HAS, the bitrate selection decision is made at the client-player. Incorrect or ineffective selection leads to a sub-optimal streaming and network components' performance [106, 83]. In this section, we overview the existing adaptive streaming implementations, then we shed light on issues related to the client player stability.

2.2.1 HAS methods

Over the past few years, many commercial and opensource HAS players have been released. Each implementation has a different approach regarding the decision of bitrate adaptation and chunks' format. Following, we give insights on characteristics of the most popular HAS implementations.

HTTP Live Streaming

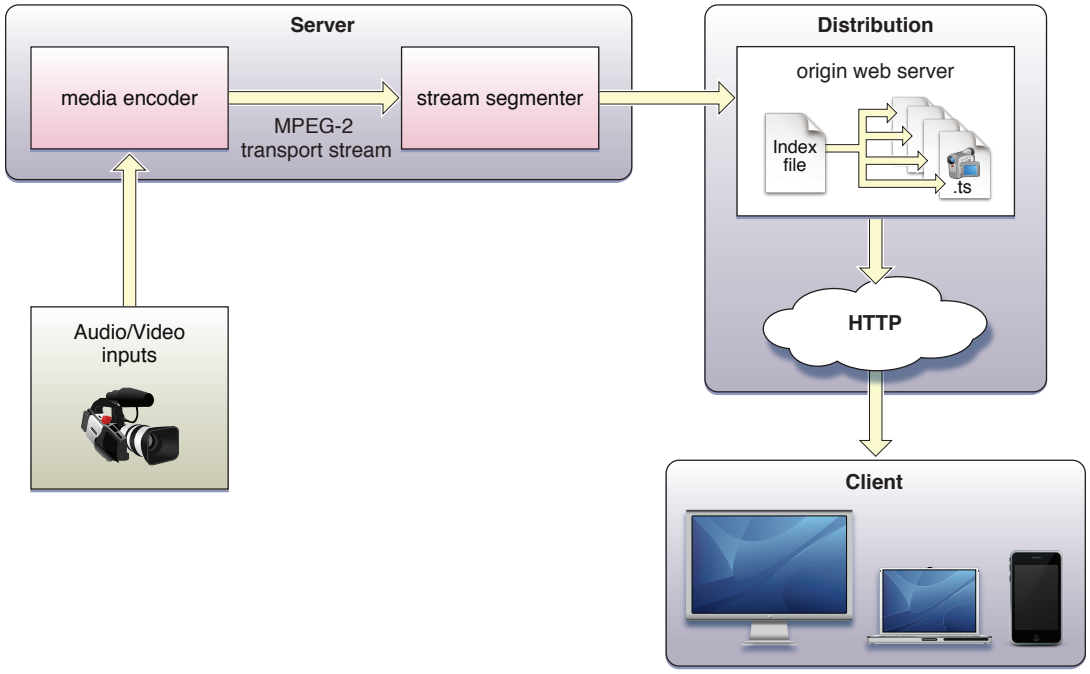


Figure 2.1: HLS configuration

HTTP Live Streaming (HLS) is the most popular video streaming technology released by Apple[7] and largely supported and deployed by Akamai[8]. Figure 2.1 depicts the key preparation steps of a HAS content. First, a source MPEG video file is converted into multiple MPEG-2 transport streams. Each stream is segmented into small videos, namely chunks, with duration of 10 seconds of video. Each stream is encoded at different bitrate, making the entire video available at several bitrates. For each stream a manifest or *playlist* is created which contains metadata about the video files in the stream, including the URL for each file. A master manifest is also created, with metadata about the existing streams. HLS supports live video streaming as well as video on demand. For a live video broadcast, the stream segment files are created

continuously and the manifests are regularly refreshed.

Microsoft Smooth Streaming

Microsoft smooth streaming (MSS) was first tested in 2008 during the Beijing Summer Olympic Games. Since then, MSS has gained popularity and has attracted the industry attention. There are several key differences between HLS and MSS.

The major difference is that MSS videos are no longer segmented into large number of chunks, but are instead implicitly segmented into fragments and then stored in a contiguous MP4 file. In MSS, users' requests encompass pointers to the desired quality and fragment of the video stream that the server holds. While in HLS, each chunk has a different URL which does not essentially include information related to the quality or number. The second important difference is that the client and the server holds two different manifest files .ismc and .ism, respectively. The former is served to the clients to learn about the available qualities, resolutions, codecs, and list of available chunks while the latter is held by the server. The .ism file is used to map the user's request to the corresponding .ismv or .isma file on disk. These 2 last files contain the video stream (.ismv) and the audio stream (.isma) that will be served to the client.

DASH protocol

A lot of standardization efforts are currently being conducted by researchers, MPEG and 3GPP to promote MPEG-DASH as to become the next major driver for multimedia streaming technology. Conceptually DASH has many common key points with the HLS approach.

Figure 2.2 summarizes the DASH standard. The server side holds the MPD and the chunks. The MPD has to follow a particular XML structure, while chunks should follow the ISO base media format[94]. The server side is fully standardized, but the client is not. The green blocks are left to the interested actors to define their own strategies of bitrate selection.

Christophe Muller has largely contributed in the emergence of Dash. In [77], Muller and al. developed a VLC media player plugin which enables VLC to play DASH videos. In [58], Le Feuvre and al. has implemented GPAC[9], an open source multimedia framework supporting DASH which comes with its own player.

In [76], Muller and al. collected a dataset in which they recorded the measured download throughput during separate freeway car drives linking 2 cities in Austria. They used this dataset as a landmark to emulate the fluctuations of the bandwidth and to benchmark their DASH

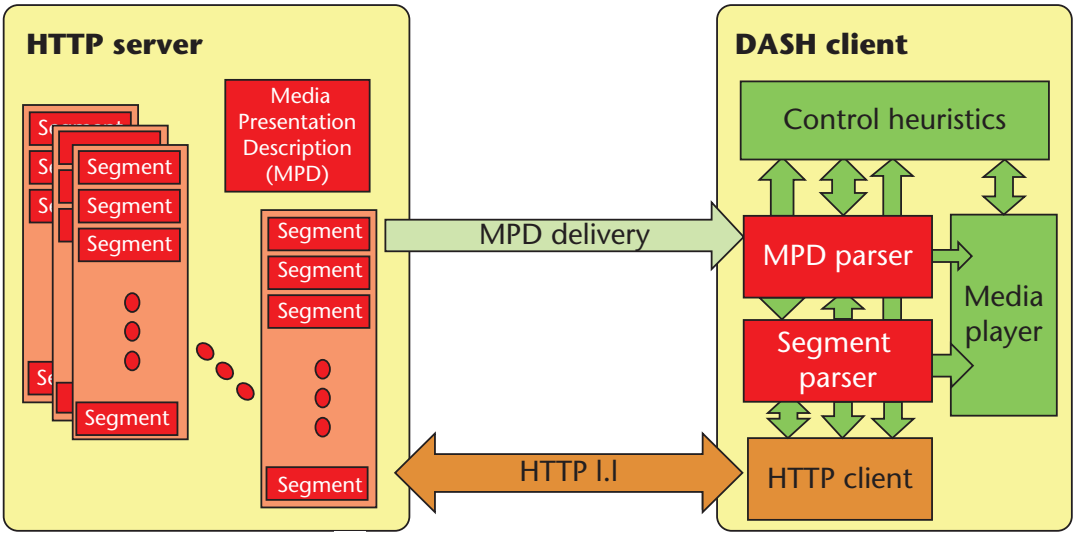


Figure 2.2: DASH standard

implementation with other commercial implementations, namely HLS, MSS and HDS. They demonstrated that their DASH implementation is more reactive to bandwidth fluctuations and achieves roughly a similar average bitrate compared to MSS implementation.

In [60], Muler and al. provide a public dataset[10] which consists of a set of videos encoded with respect to DASH standard. The videos are available in 20 different qualities. The dataset was created with a DASH encoder provided by the university of Klagenfurt. The dataset might be used by researchers to benchmark different DASH implementations.

2.2.2 Stability of HAS players

In the context of HAS, instability of the player means that the player switches very often between qualities, which consequently annoys the user experience[35].

The instability of the client players has been addressed in several studies [53, 14, 16, 15, 67]. All these works reported that the main root cause of the client-player instability occurs when multiple players compete for a shared bandwidth behind a bottleneck link. A client player supporting HAS starts a session with a *buffering-state*. In this state, the players try to build the playback buffer as quickly as possible. Then the buffer turns into the *steady-state* during which it maintains a constant playback buffer size. This steady-state consists of 2 sub-states: The player is either ON, and downloads a new chunk, or OFF where the buffer is full enough and does not require downloading a new chunk. In [16], authors reported that most of the

players including commercial players estimate the bandwidth based on the download speed of the last downloaded chunk to decide what would be the next bitrate of the next chunk. This suggests that during the OFF state, the player does not estimate the available bandwidth. In the case where multiple players are competing for a shared bandwdith behind a bottleneck link typically a broadband, one of the players might be ON while all the others are OFF, therefore, the player estimates a high available bandwidth which pushes her/him selecting the highest encoding profile with respect to the observed available bandwidth. Depending on the overlap between the ON-OFF periods; 3 key performance are affected, namely unfairness, instability and under-utilization of resources.

Saamer and al. [15] proposed to tackle the issue of instability at the server-side without any cooperation between the client and server. They considered that client players receive all chunks from the same video server, which is not true in the presence of caches or CDNs. Their proposal was to supply the server with a shaping module tied to each client player. Their proposal was to avoid the OFF period during the steady-state. This is by streaming the chunk at its playback rate. In Festive[53], authors used a different approach to cope with the instability issue. Their approach is client-based and does not require any interaction with the server. They estimated the current bandwidth using the harmonic mean of the download speed of the twenty last downloaded chunks. They demonstrated that this leads to a much more stable player.

Instability is an important issue that should be carefully investigated especially in the context of mobile video streaming where bandwidth is subject to frequent fluctuation. In the next section we position our contribution in this context.

2.2.3 Positioning

Most studies related to clients' instability are limited to the scenario where a small number of players are competing for a shared bandwidth. Yet, the mobile ecosystem has many different characteristics, for example the bottleneck is likely to be shifted towards the nodeB which is usually shared by a large number of users. Besides, resources are allocated in a different way with regard to fixed networks. Therefore, we believe that the instability issue -in mobile context- should be resolved differently. To this end, In this thesis, we provide the first empirical study and guidelines on clients switching between profiles at large scale. Then, we model this

behavior based on empirical observations. In this study, we answer the following questions: *how frequently users switch between profiles?, how far this switching adversely affect caching performance?*.

Answering these questions led us to design and implement a cache friendly-adaptive streaming player (CF-DASH), which aims at giving networking actors such as ISPs and CDN providers the ability to assist the player to select the video quality that both clients' players and content delivery actors find it convenient to serve. CF-DASH aims at enhancing both caching efficiency and QoE perceived by users. This contribution is in line with the emerging standard SAND, the second edition of DASH. Recently, MPEG has addressed various issues related to the actual HAS version [11], among them was the impact on the existing internet infrastructure such as servers, proxies, caches and CDNs. Therefore we consider our contribution as a first step towards implementing SAND principles.

2.3 Caching

Closer is better. Caches are widely deployed in today's internet infrastructure to reduce load on web servers and the backbone network. Caching replacement policies have been widely investigated in context of web content[101, 23, 54, 105, 81]. Caching techniques for streaming media have different characteristics. Streams have important real-time and time synchronization requirements. Therefore, caching polices need to be amended with respect to these requirements. The full scope of this research is too broad. However, this section describes the basic concepts of today's solutions and ongoing research in caching adaptive streaming videos. Several selected techniques are described in detail because of their practical importance and positioning regarding our contribution.

2.3.1 Replacement strategies in context of HAS

Storing the entire content would quickly exhaust the storage capacity of a cache. The segmentation nature of HAS videos has opened new perspectives to revisit the content replacement policies and amend them with respect to the content properties.

Once the cache is full, deciding to store a recently requested chunk requires a decision to remove and replace an object that is already stored. In the following, we overview a set of replacement algorithms that have been introduced and evaluated in context of adaptive

streaming:

FIFO [55]

Replace the oldest chunk based on when it was stored. Eventually, the cache will be filled with the recently requested chunk.

LRU [82]

Replace the chunk that has not been requested for the longest time. The most recently requested chunk will be placed at the top of the list of the cached chunks. LRU is the most popular and used algorithm. However, LRU does not consider the temporal locality of segments in the HAS content. In the next chapter, we show that this adversely affects the cache performance.

LFU [61]

Replace the chunk with the fewest request-rate. Eventually, the chunk with the highest frequency will remain longer in the cache. LFU is not widely adopted because of its log complexity and the necessity for a periodic check and eviction of the stale chunks.

MIN [100]

The MIN algorithm is considered to be near-optimal replacement policy. Chunks with no more requests are subject to replacement. Yet this requires a knowledge about the future which is hard to predict. This makes MIN unrealistic for real implementation.

CC [49]

The Chunk-based Caching (CC) algorithm takes into account the time structure of the video content. CC associates a score for each chunk, then chunks with lowest scores are subject to eviction. The rationale behind is as the following: Assuming that a content C has M chunks. If a user is requesting chunk i, where $i \in [0..M]$, then all chunks with indexes ranging from $i+1$ to M increment their scores. This gives priority to the chunks that will be requested shortly given the latest clients' requests.

RT [103]

The Reuse Time-based (RT) algorithm is a step forward towards enhancing CC. RT has many common points with CC strategy except that RT takes in addition the distance between chunk indexes of clients requesting the same content. This case is particularly suitable for long movies since it might be that one user is requesting the first chunk of the stream while the user ahead is at the end of the same stream. Therefore, it is worth weighting the scores of chunks in-between rather than simply incrementing all the scores.

WAVE [33]

Wave was designed for collaborative in-network caching such as content oriented networks. The number of chunks to be cached is adjusted based on the popularity of the content. As the request count increases, WAVE exponentially increases the number of chunks to be cached per file. WAVE aims at decreasing the overhead of cache management and improving caching efficiency.

2.3.2 Positioning

CC and RT algorithms are first algorithms to leverage the temporal locality of chunks in HAS to improve caching efficiency. However, these algorithms assumed that users watch the content entirely which is not always true[46, 47]. Subsequently, this introduces a bias into the real performance CC and RT could achieve. These replacement algorithms might be adopted to pay-per-view video services as the end user is motivated to watch the video she paid entirely. Furthermore, these algorithms are tunable, i.e. selecting the appropriate parameters depends primarily on traffic properties. In this thesis, we propose a workload aware LRU (WA-LRU) a cache replacement strategy to decide which chunk to replace and how to update the list of cached chunks. To do so, WA-LRU relies on 3 information: First, number of users requesting the same content simultaneously. Second, the workload on the cache. Third, the cache capacity.

WA-LRU adopts a dynamic behavior across the day. It evicts aggressively the video segments when the cache is under a heavy load and behaves less aggressive during the off-peak times. WA-LRU gives more priority to the first chunks to be cached since they are much more requested than the latest chunks of the same content. We demonstrate that WA-LRU improves

the hit-ratio and reduces the processing overhead required to update the list of cached objects.

2.4 Prefetching

Fundamentally prefetching is different from caching. The main principle of caching is to leverage the historic information about an object to decide if it is worth caching it or not, while prefetching consists of predicting then pre-staging the content that is likely to be consumed by clients in the future. Prefetching might be applied to various architectures and implemented in different ways. In this thesis, we choose to push the content items on the users' devices. Prefetching is a promising approach to reduce delays, optimize the QoE, ensure a smooth playback and to reduce energy consumption. For instance authors in [43] suggest prefetching videos when the WiFi connection is turned on, while authors in [74] suggest prefetching mobile-ads to achieve energy savings.

Recently, video prefetching for mobile users has gained considerable interest by the research community. In this section, we overview several prefetching systems and prediction models derived from recommendation techniques and applied to the context of content delivery optimization.

recommendation Video Recommendation is a hot topic. Currently there is 3 prevalent and widely adopted recommendation approaches, namely Collaborative Filtering (CF), Content-based filtering (CBF), and Hybrid Filtering (HF). The CBF approach suggests filtering video items based on the unseen and viewed videos by the user [44]. The CF approach compares a user's preference on items with others to find out people sharing simlar interests [19, 79]. The HF approach combines both CF and CBF approaches.

Other works has deepened this scope of research. For instance, in [111] authors propose attributing a score for each video candidate. The score is composed of two parts: The interest degree of this video by user's friends. This interest is based on textual (i.e. category of the video) and visual similarity with the other viewed videos. The second part is the relationship strength between the user and his friends, i.e. affinity. In[70], Mei and al. designed a video recommendation system they name it VideoReach. VideoReach recommends a list of videos related to the user's current viewing based on three modalities: textual, visual and aural relevance.

Recent studies suggest supplying the recommender system with additional input derived from online social networks properties. In [34, 90, 86, 87, 42], authors leveraged information derived from social cascades such as geographic information and shared YouTube videos to improve caching and to predict the information dissemination around the world. In [99], authors developed TailGate that exploits information available in Twitter so as to deliver long-tailed content while decreasing bandwidth costs and improving the QoE.
Other prediction algorithms have been proposed in the context of social networks. These algorithms estimated the relevancy of content items for a given user. EdgeRank [12] is used by Facebook and relies on 3 criteria to estimate how relevant a content item is for a given user user:

- *Tie-strengh (affinity):* The relationship strength between a user and the content creator.
- *Timedecay:* Timedecay refers to how long the content was popular and requested by users.
- *Weight:* Weight is related to the type of the displayed element (photo, status, video , etc.).

EdgeRank gives a score to each content item based on the 3 above criteria, then sorts and display the content items with respect to these scores.

2.4.1 Prefetching systems in context of video streaming

Several prefetching systems have been designed and evaluated in context of video streaming. In [31] authors designed NetTube a peer-to-peer streaming system tailored to the short YouTube video clips. They observed that YouTube videos have strong correlations with each other. NetTube creates an overlay, through which peers re-distribute videos that they have cached. NetTube implements a cluster-aware prefetching system to reduce delays during transitions between video playbacks. Authors consider that related videos in YouTube form a social network and that videos are highly clustered. Therefore, they propose prefetching video prefixes of the related video of the currently viewed video. Huang and al. [50] have designed a similar peer-assisted VoD system to reduce MSNs' servers stress. In [66], authors leveraged social closeness derived from OSNs to build the P2P overlay. For each source node, the overlay endorses nodes within 2-hops away from the source node. This is because they observe that more than 90% of the viewers of the video are within 2 hops away. Their prefetching scheme

consists on prefetching only one chunk of each video from the source to the rest of social neighbor peers. In [102], Zhi and al. observed that OSNs help predicting users' behavior. They designed a user preference guided prefetching strategy tailored to P2P systems. In their design, authors rely on social closeness and video popularity to decide if worth prefetching a video or not.

In [96], Samamon and al. proposed deploying a prefetching module at the client and at a proxy cache level to prefetch YouTube videos. Their prefetching strategy consists of two main algorithms. The first algorithm is based on user's search results. The search query sends back a list of videos displayed in a search result page, then authors propose prefetching the top-N videos from this list. The second algorithm consists on prefetching the top-N related videos. That is because the next video is likely to be confined by the related list of the current viewed video.

In [56], authors proposed prefetching the weekly most popular videos from Hulu and prestage them at a campus network. Similarly and in mobile context, authors proposed in [41] to periodically prefetch bundles of popular videos on mobile devices.

2.4.2 Positioning

While most of the prefetching systems have been designed and evaluated in context of Peer-assisted video streaming, few of works [43, 41]addressed prefetching in context of mobile video streaming. Yet, in these works authors did not address several questions which we see as being fundamental and with a prevalent importance in context of mobile video streaming.

Their study was coarse-grained and did not address the question of what to prefetch. Prefetching irrelevant videos may strongly decrease the user QoE and systematically increase the traffic in the network. We show that our strategy is worth to reduce the traffic overhead while enhancing the prediction accuracy. To do so, we deeply investigate the 3 following key design questions: *what to prefetch? when to prefetch?* and *how mush to prefetch?*

To evaluate our system performance, we use our own and unique dataset which is based on real world measurements with millions of views. Prior work depend on small-scale experiments with few users. For example, in [102], authors considered only the 500 most active user to assess the performance of their prefetching strategy.

For testing performance, in [21], authors considered users' ratings to simulate the traffic

workload. Potentially, this leads to biased results since only few users express their opinion on videos they watch.

In a nutshell, we design a pefetching system we call it CPSys. The system principle is driven by the video consumption and users' behavior patterns. Then we evaluate our system and show that it achieves considerable performance. At the end, we provide a prototype implementation of CPSys.

3 HTTP adaptive streaming in mobile networks : characteristics and caching opportunities

3.1 Introduction

Cellular networks have witnessed the emergence of HTTP Adaptive Streaming (HAS) as a new video delivery method. In HAS, the source content is encoded at multiple bitrates so that clients can pick the best quality that fits their bandwidth capacity. In the previous chapter, we showed that this has particular implications on caching strategies with respect to the viewing patterns and the switching behavior between video qualities. In this chapter and in Section 3.2 we present analysis of a real HAS dataset collected in France and provided by the country's largest mobile phone operator. We outline our methods for collecting and processing the data, then we give insights into the traffic properties and the variety of different categories and video services requested by clients. In Section 3.3, we analyze and model the abandonment rate and infer guidelines regarding caching. We also gain insights into the clients' attitude and study the factors that decreases the user engagement. In Section 3.4, we leverage all these observations and findings to design and evaluate a novel cache replacement strategy, namely WA-LRU. WA-LRU requires the knowledge about the traffic load and chunks requested by users to decide whether it is worth caching a chunk or not. We conclude this chapter in Section 3.5.

3.2 The Dataset overview

3.2.1 Data collection

We have connected a server-log system at a Gi interface of one of the Gateway GPRS Support Nodes (GGSNs) deployed by the European largest mobile career in France. We captured 1.763.516 adaptive streaming sessions. We logged exactly 92.595.115 HTTP GET requests. Each HTTP request corresponds to a chunk request. In our study, the considered GGSN serves 230 NodeB/BTS. This area is covered by 3G/HSPA/HSPA+ and 2G (EDGE) radio access networks and served 246.913 unique active client during the measurement period. The data collection was conducted over a period of 9 weeks from November 7^{th} 2012 until January 9^{th} 2013, involving mainly Apple HTTP Live Streaming (HLS) and Microsoft smooth streaming sessions (MSS). However, we limit our analysis to HLS sessions since they form more than 99% of HAS sessions in our dataset.

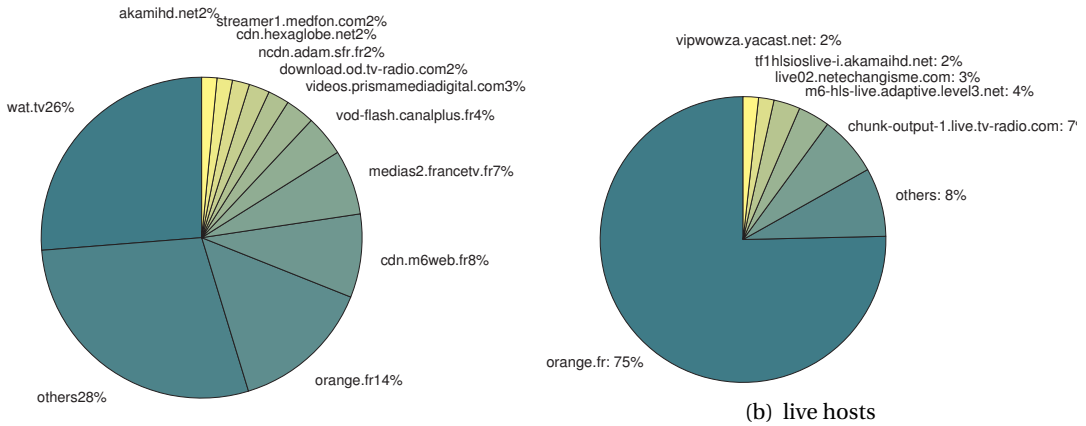
3.2.2 Content types

Currently, HTTP adaptive streaming is mainly used by Over-The-Top TV industries. Telecom companies, on their side, offer their subscribers the TV-service as a free value-added service. Our dataset mainly contains:

- *Live TV sessions*, where clients watch live TV sessions;
- *Catch-up TV sessions*, where TV content providers offer clients a catalog of videos previously broadcasted in live.

In our dataset, we observe that around 76% of HAS sessions correspond to Live streaming sessions, while the rest corresponds to catch-up sessions. In the following we give insights on the requested domain names:

Domain names for Catch-up videos Figure 3.1(a) shows that the domain *wat.tv* is the most frequently requested domain name (25% of client requests) by mobile clients when requesting catch-up videos. *wat.tv* is a TV service provider that hosts catch-up videos of most of the local French-TV broadcasters. Other TV channels delegates to world wide CDNs to deliver their content on their behalf. For example: *"vod-flash.canalplus.fr"* cnames Akamai servers.



(a) catch-up hosts

(b) live hosts

Figure 3.1: Hosting servers for both catch-up content videos and live channels

Other TV companies also play the role of content service providers (CSPs) by using their own streaming infrastructure such as the domain *cdn.m6web.fr*.

Domain names for Live sessions Figure 3.1(b) shows that 75% of live HAS sessions are streamed from servers with domain name *orange.fr*: This means that subscribers usually access live TV channels through the Orange application. Orange provides streaming servers to enable local TV channels to broadcast their videos to mobile clients. The domain *chunk-output-1.live.tv-radio.com* is the second most requested domain (7% of client requests) which cnames Akamai servers.

3.2.3 Data processing

We identify a session through the first HTTP-GET request of the first chunk for which the URL address ends with either a *.ts* for a HLS stream, or with a *.ism* for a MSS stream. Each HAS session corresponds to the set of chunks sent to a client over the TCP connection. Our logging system captures all packet headers of HAS streams which contain useful information, such as the packet size and sequence number. All information is aggregated per TCP connection and exported into a database, from where they are analyzed. Each persistent-TCP connection corresponds to one downloaded video segment. This means that the number of downloaded chunks during one session is equal to the number of persistent TCP connections established between the server side and the end-user over time. The encoding scheme of the different profiles may differ among content videos since each of the TV service providers may define

his own range of encoding profiles.

Figure 3.2(a) shows the distribution of the encoding profiles ($b_{i\in[1..N]}$) where N represents the number of profiles defined for each catch-up content within the data-set. We observe that most of the defined encoding profiles are below 1000 kbps.

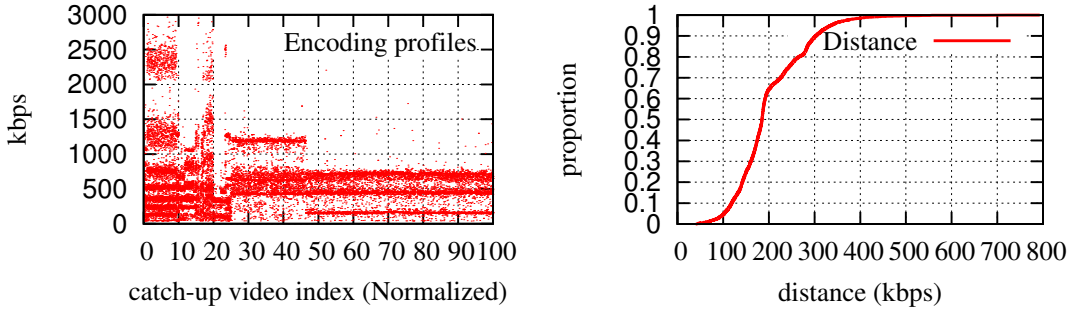
Nbr of profiles	1	2	3	4	5	6	7	8
percentage	17%	22%	32%	13%	7%	4%	4%	1%

Table 3.1: Breakdown of the number of proposed profiles per HAS content

In Table 3.1, we show the breakdown of number of the available profiles for catch-up TV videos requested during the measurement period. In Figure 3.2(b), we estimate the average distance in kbps between the encoding profiles defined for each catch-up video:

$$distance = \frac{\sum_{i=2}^{N-1}(b_{i+1} - b_i)}{N} \quad = \frac{b_N - b_2}{N}$$

We observe that in 95% of catch-up videos, the average difference between the encoding profiles is higher than 100 kbps. For this reason and since we are interested in aggregated statistics to draw conclusions about HAS characteristics, we define a scale of 8 different profiles (see Table 3.2) that will be used to map each requested chunk in each HAS session to the appropriate profile *(P)*.



(a) encoding profiles

(b) distance between profiles

Figure 3.2: Profiles presentation used for catch-up contents

We assume that the encoding rate is equal to the volume of data contained in the packet payloads per chunk and divided by the chunk's duration. The measurement of the encoding rate of each chunk is initiated the time we capture the HTTP-200 response from the hosting

server. When the download finishes, we map the encoding rate to the corresponding profile.

Profile i	Encoding bitrate (kbps)
Profile 0 (P_0)	< 50
Profile 1 (P_1)	[50-150)
Profile 2 (P_2)	[150-280)
Profile 3 (P_3)	[280-420)
Profile 4 (P_4)	[420-600)
Profile 5 (P_5)	[600-1000)
Profile 6 (P_6)	[1000-2000)
Profile 7 (P_7)	≥ 2000

Table 3.2: Profiles settings

3.2.4 Fields description

The dataset consists of hundreds of thousands of rows. A new row is added each time a user switches to a different encoding rate. For each HAS session and during each transition, we record the timestamp and the newly visited profile (P_i).

In the following we describe the fields we report in each row:

$Session_{id}$

- Session Start (in timestamp).
- Requested URL.
- Cumulative number of requested chunks in $session_{id}$.
- Current profile $P_{j\epsilon[0..7]}$.
- Next profile $P_{k\epsilon[0..7],k\neq j}$ (when a bitrate-switching happens).
- Cumulative number of requested chunks within profile $P_{j\epsilon[0..7]}$.
- Cumulative Bytes downloaded in profile $P_{j\epsilon[0..7]}$.
- Cumulative time to deliver all requested chunks in profile $P_{j\epsilon[0..7]}$.

3.3 Clients' behavior analysis

3.3.1 Distribution of requested chunks per session

In HAS, the video content is segmented into multiple chunks and subsequently requested independently. This opens the opportunity to study the behavior of clients at a fine-grained level. To start, we show in Figure 3.3 the distribution of the number of requested chunks per catch-up TV and Live HAS sessions.

Using the maximum likelihood (MLE) estimation, we show that the Log-normal (μ, σ_1) distribution is best to model the requested number of chunks for both live and catch-up TV sessions for the first 40 chunks per session. Log-normal distribution is ubiquitous to describe several mobile communication patterns such as the call holding times [93], the file data transfer in the mobile networks [62]. We observe that this still to be the case for the first 40 chunks per session, which applies for more than 90% of HAS sessions. Then, we observe a sharper distribution when the number of chunks per session exceeds 40 chunks. The generalized pareto distribution (GPD (k, σ_2, θ)) is best to model that tail. In general, the GPD is well known to model *long-tailed* profiles such as the non-popular video distribution [28]. In our case, the tail suggests that in very few sessions users watch a content for a long time.

We show in Table 3.3, the parameters of both log-normal and GPD distributions used to model the requested number of chunks per session.

	Log-normal ($x \leq 40$)		Generalized pareto ($x > 40$)		
params	μ	σ_1	k	σ_2	θ
catch-up	1.75145	1.01198	0.034218	118.55	40
live	1.87679	1.00993	0.412856	78.5267	40

Table 3.3: Empirical models

In the case where the chunks are delivered by caches or edge servers deployed by CDNs, such a tailed profile could degrade the caching efficiency especially for a limited storage capacity and the use of a highly reactive replacement strategy. Requesting these latest chunks would result in a competition between chunks that belong to the head of the distribution which are frequently requested and chunks that belong to the tail even though few of clients will reach that latest chunks of the videos. This becomes more critical when the cache adopts the LRU (Least recently used) algorithm, where these chunks will be top-ranked the time they

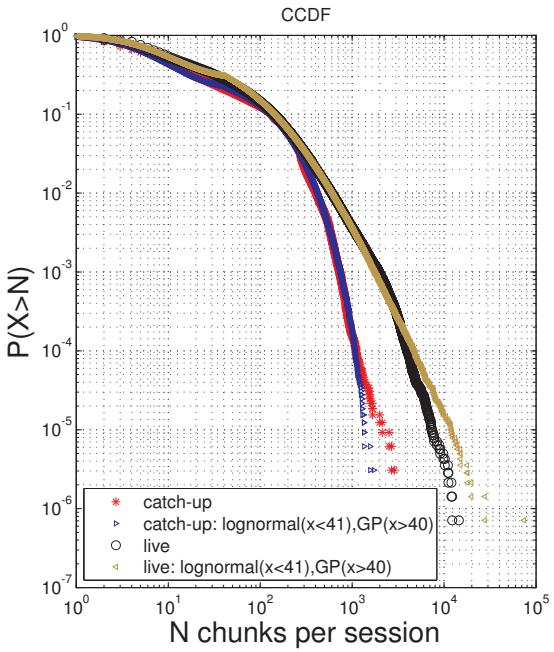


Figure 3.3: Distribution of number of chunks per session for live and catch-up HAS sessions

will be requested, thus pushing down chunks that will probably get requested in the near future. In the last section of this chapter, we evaluate through simulations the benefit of not systematically caching the latest chunks of HAS videos when deploying a proxy-cache between the clients and servers holding the videos.

3.3.2 Analayis on user engagement

In HAS, client-players try to adapt at best the playback quality to match the available bandwidth. Authors in [16] studied throughout experiments the behavior of the playback buffer under bandwidth restrictions using open source tools. However, in the real world and in the context of mobile streaming, bandwidth fluctuation is potentially the root cause that pushes client-players to decide what would be the next bitrate to select. Figure 3.4 shows that the profile selection is potentially correlated to the Download Throughput (DT) experienced by the clients. In Figure 3.4, we bin DT every 500 kbps and we assess the profiles picked by clients when they experience a DT that ranges from 500 kbps to higher than 2000 kbps. When DT is equal to 500 kbps, we observe that 82% of requests are to download profiles equal or below P_3. However the distribution of profiles equal or higher than P_4, is superior to 60% when clients experience a DT higher than 2 Mbps.

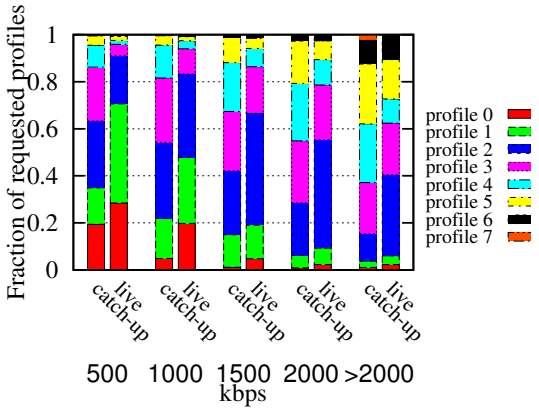
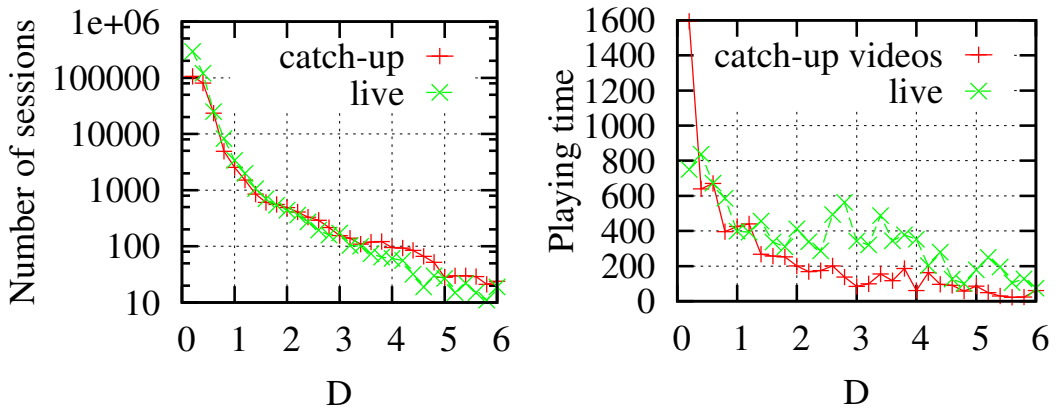


Figure 3.4: Download throughput and Profiles interdependencies

So far, we study the *user-engagement* [37] by correlating the average playing time across all sessions as a function of the experienced average delay while delivering video chunks. The playing time in our case corresponds to the length of data that was downloaded during a session (*i.e. nb of requested chunks per session* $* Chunk_{duration}$). To do so, we define the delay metric (D) that we associate for each session. D is equal to the average delivery time of one chunk ($\tilde{T}_{chunk}$) per session divided by the average chunk duration per session:

$$D = \left[\frac{\tilde{T}_{chunk}}{Average chunk duration} \right]_{session}$$

If ($D > 1$), then the playback was potentially interrupted at least one time during the stream. If the time spent in the network to deliver one chunk exceeds the chunk duration, this would certainly turn the client-player into the buffering mode. In Figure 3.5(b), we bin D in units of 0.2 and evaluate the playing time as a function of D. We stopped at $D = 6$, since as shown in Figure 3.5(a), the number of sessions in which D is superior to 6 is handful and thus not enough representative. For catch-up TV sessions, Figure 3.5(b) shows that the average playing time is exponentially distributed as a function of D. When $D = 1$, we observe that the user-engagement decreases with a ratio of $\frac{3}{4}$, and still decreases as far as D becomes higher. However, surprisingly, for live sessions, we observe that the average playing time is still higher for $D > 1$ and does not show a clear pattern as the case for catch-up sessions. One possible reason is that when $D > 1$ clients might let their live session active while doing something else and waiting until the stream comes back.



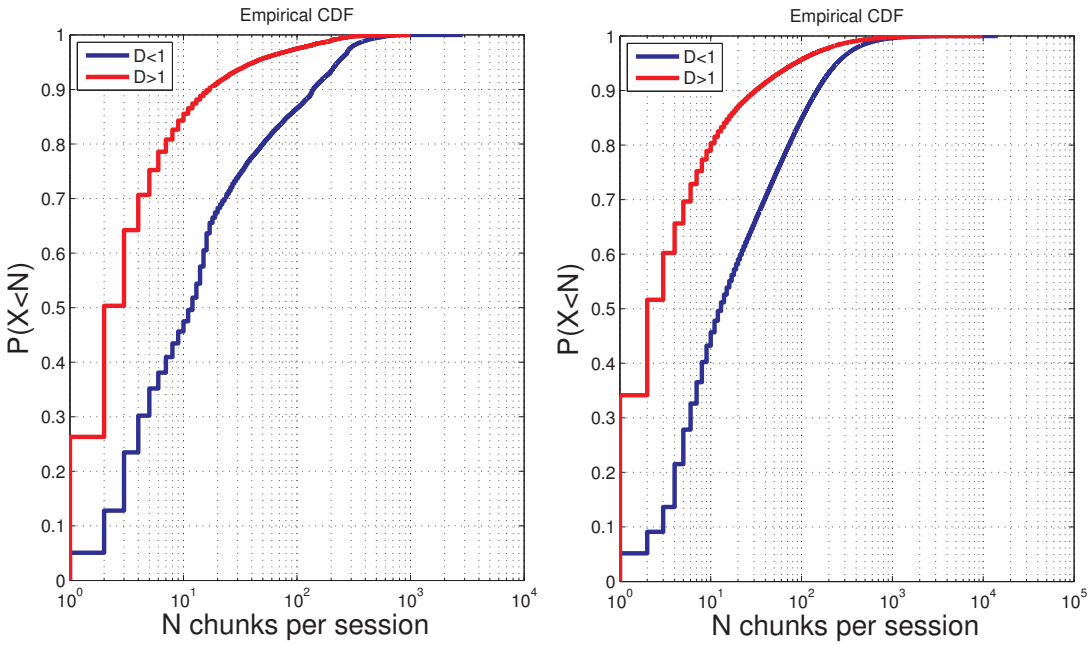
(a) Number of sessions = f(D)

(b) user-engagement = f(D)

Figure 3.5: Impact of the Delays in chunks delivery on the playing-time

We show in Figure 3.6(a) that when $D < 1$, clients are more likely to maintain their catch-up sessions active. We find that in 13% of catch-up sessions clients request at least 100 chunks. However, clients spend less time viewing a video when $D > 1$, mainly due to the video interruptions. We find that only in 2% of HAS sessions, clients request more than 100 chunks. We also observe that 50% of HAS sessions are aborted after requesting only 3 chunks per session when $D > 1$. This suggests that while experiencing long delays in the *joining-phase* (*i.e. D>1)*, we observe that one out of two sessions is aborted from the beginning of the stream. While delivering video chunks, it is worth to give priority to the first chunks of video content to be served to guarantee a smooth *joining-phase* to the end-users. Our algorithm WA-LRU -which we present in the last section- takes up this challenge since it gives priority to the first chunks to be cached. For live sessions (Figure 3.6(b)), we observe that in 15% of live sessions, clients request more than 100 chunks per session when $D < 1$, while only 3% of live sessions reach 100 chunks per session when $D > 1$.

In summary, we observe that around 60% of HAS sessions are aborted when users experience delays at the *joining-phase*. This clear correlation between the user-engagement and the joining-phase suggests that prestaging the first chunks of a video content at a cache -close to the clients- will clearly improve the user-engagement. This finding is inline with the observation we did in 3.3.1, where the piece-wise of log-normal and Pareto distribution suggests that clients do not systematically watch the content until the end. These 2 findings



(a) catch-up (b) live

Figure 3.6: CDFs of number of chunks per session when $D < 1$ and $D > 1$

drive the design principle of WA-LRU which we describe in the next section.

3.4 Caching HAS content

In this section, we introduce and evaluate WA-LRU using a real mobile traffic trace.

3.4.1 Presentation of WA-LRU

WA-LRU takes advantage of the temporal locality of the chunks within a HAS video to increase the performance of the cache.

LRU blindly caches the video chunks without considering any property of the content (*i.e.* popularity, temporal locality of video chunk, etc.). A cache-miss will systematically triggers a pull request to download and cache a new chunk from the parent-server or origin server, thus evicting the least recently requested chunk. It turns out that the first and latest chunks of a HAS content have equal opportunities regarding caching, although we previously reported that the access to chunks is not uniformly distributed. Yet we observe a heavy tail leading to a disproportional access to video chunks. To tackle this issue, WA-LRU does not systematically cache the latest chunks of a video content. The decision of caching a chunk depends on its

position within the stream. WA-LRU needs to learn about the traffic load on the cache and the number of simultaneous users watching the same video content to decide if it is worth caching a chunk or not. WA-LRU adopts a dynamic behavior across the day since the caching strategy is function of the traffic load.

In Eq 3.1 we define the traffic load on the cache over a period of T seconds as a multiplication of the *encode-rate* (in kbps) of each requested chunk between (t) and (t+T) seconds and the *chunk-duration* (in seconds).

$$[W]_T = \int_t^{t+T} \textit{encode-rate}(t) * \textit{chunk-duration}(t)\,\mathrm{d}t \tag{3.1}$$

In Figure 3.7, we picked one representative week data from our dataset and we plot the aggregated traffic load on the Gi interface at a granularity of one hour (T=3600s). We observe that the traffic load follows a strong diurnal pattern which suggests that the caching mechanism *should* consider the variation of the workload along the day. It should be aggressive at peak hours, and less conservative at off-peak times.

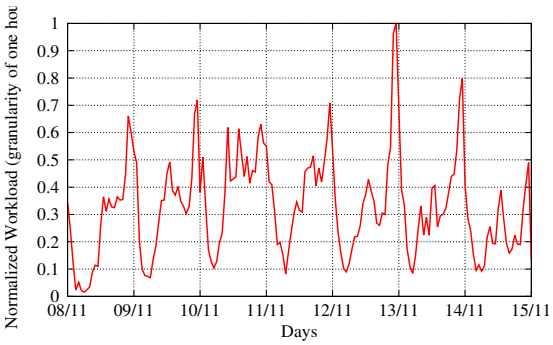


Figure 3.7: Workload pattern measured by the logging system from 08/11/2012 to 15/11/2012

Under a high workload, a cache -running LRU- will be highly reactive and will intensively evict chunks. We define the parameter R (Reactiveness of the cache, c.f. Eq 3.2) to measure the bytes' ratio $(B^E)_T$ to be evicted periodically from the cache each T seconds. We have: $(B^E)_T = 1 - (B^S)_T$, where $(B^S)_T$ represents the ratio of the bytes served from the cache.

C represents the cache capacity.

$$R_T = \frac{[B^E]_T}{C} = \frac{[W]_T - [B^S]_T}{C} \tag{3.2}$$

R_T is sensitive to the diurnal pattern since it depends on the traffic load. Under a high load, R_T will increase and thus reach its maximum at peak hours. We set T to 10 seconds to match it to the chunk length. Subsequently, if we assume that $R_{T=10s}$ is static or varies slightly at each round of $T = 10s$ (next we show that this assumption could be retained). Additionally, if a particular chunk is requested at time t_0 and does not receive any additional request for the next $t_0 + 10 * \lfloor \frac{1}{R_T} \rfloor$, this chunk will be subject for eviction. For the rest of this chapter, we note $\Delta_T = \lfloor \frac{1}{R_T} \rfloor$. Using LRU, if one chunk is not requested at least twice in a time frame of $10 * \lfloor \frac{1}{R_T} \rfloor$, then this chunk will be evicted.

$\Delta_{T=10s}$ varies across the time since it is a function of the traffic load. At a high load, $\Delta_{T=10s}$ becomes small and vice versa. We then consider the following weighted moving average method to periodically compute the value of the threshold θ_T at each 10 seconds. θ_T is used as a landmark to decide what to cache and what to leave. Following, we show how we compute the value of θ_T:

$$\begin{cases} \theta_0 = \frac{1}{R_{T=10s}} \\ \theta_T = (1-\beta) * \theta_{T-10} + \beta * \frac{1}{R_{T=10s}} \end{cases}$$

If θ_T is lower than the $index$ of the chunk, then this chunk is not cached; else it will be cached. β is used to avoid the outliers that may happen such as a drastic variation of $R_{T=10s}$. However, in our dataset we find that the coefficient of variance of the traffic load ($Cov = \frac{\sigma}{\mu}$) over the day is equal to 0.7 which remains less than 1. This suggests that the variation of R is smooth across the time. Based on this information and for the sake of simplicity we set β equal to 1.

3.4.2 WA-LRU in action

Now, we describe the mechanism of WA-LRU based on 2 use cases. Let $chunk^A_{index}$ represents the position of client A in the video content, where *index* ranges from 1 to the last chunk of the

video content (m). The two following scenarios (Figures 3.8 and 3.9) outline the principle of WA-LRU:

Use case 1:

In Figure 3.8, client A requests a new chunk where the $index$ is superior to θ. If this chunk is not already cached, then it will be ignored by the cache.

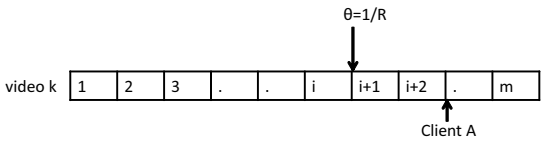


Figure 3.8: Scenario 1: comparison between θ and $chunk^A_{index}$

Use case 2:

In Figure 3.9, client A and B are simultaneously requesting the same video content. If $chunk^A_{index} > chunk^B_{index} + \Delta_{T=10s}$, then it is useless to cache $chunk^A_{index}$ since after $10 * \Delta_{T=10s}$ seconds, it should be evicted and client B will be redirected to the origin server to download it. If $chunk^A_{index} \leqslant chunk^B_{index} + \Delta_{T=10s}$, then $chunk^A_{index}$ should be cached since we suppose that in less than $10 * \Delta_{T=10s}$ seconds, client B will eventually reach $chunk^A_{index}$ of the video stream. However, in this scenario $chunk^B_{index}$ should be ignored by the cache since the *index* is higher than θ.

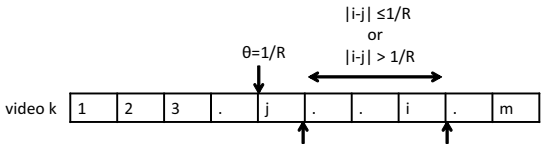


Figure 3.9: Scenario 2: comparison between θ and $|chunk^A_{index} - chunk^B_{index}|$

3.4.3 Pseudo-code of WA-LRU

Let $S_{T=10s} = [s_0, s_1, .., s_N]_{T=10s}$ be the set of active sessions updated each $T = 10s$. s_i could be represented by a 3-dimension vector: $s_i = [Video_{id}, Chunk_{id}, P_k(profile)]_i$. In Algorithm 1, we show the pseudo-code of WA-LRU:

- Line 1: updates S_T periodically to capture the active sessions.

- Lines [3-4]: The cache is not full yet and no policy is applied. We cache all chunks.
- Lines [5-14]: The cache is full and chunks ($s_i^{chunk_{id}}$) are cached and evicted with respect to the WA-LRU policy.
- Line 17: After T seconds, the value of R is updated with respect to the number of bytes evicted from the cache.

```
Require: S_T;
1: update(S_T)
2: if R=0 then
3:     cache
4: else
5:     for i = N → 1 do
6:         for j = N → 1 do
7:             if s_j^{Video_id} == s_i^{Video_id} then
8:                 if s_j^{Chunk_id} + 1/R ⩽ s_i^{Chunk_id} then
9:                     cache
10:                else
11:                    forward
12:                end if
13:            end if
14:        end for
15:    end for
16: end if
17: update(R)
```

Algorithm 1: WA-LRU

3.4.4 Evaluation

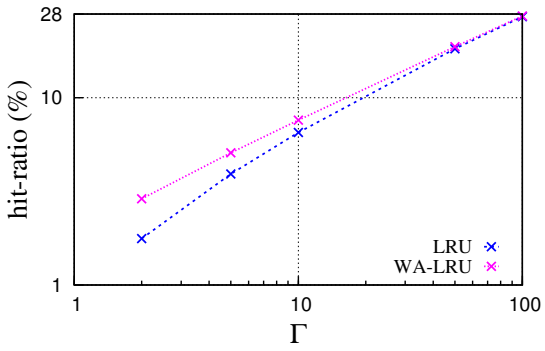
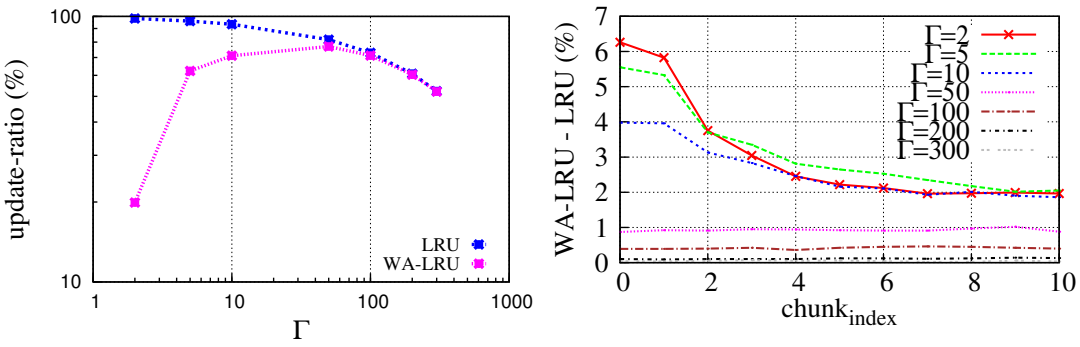


Figure 3.10: Average hit-ratio

We evaluate our algorithm based on the 3 following metrics:

- *Average hit-ratio.*



(a) Average update-ratio

(b) Average Hit-ratio per $chunk_{index}$

Figure 3.11: *Metrics evaluation: Average hit-ratio, update-ratio and average hit-ratio per chunk position*

- *Update-ratio*: Ratio of requests leading to update the list of cached chunks. This ratio has an impact on the processing overhead to update the list of cached chunks.
- *Per-chunk hit ratio*: Average cache hit-ratio per each $chunk_{index}$.

An efficient replacement strategy should reduce the update-ratio and increase the hit-ratio. We evaluate the performance of the cache based on trace-driven simulations and with different values of the storage capacity C such as: $C = \Gamma * S$, where S represents a video size of 50 MB and Γ represents the number of videos that the cache can hold. For the sake of simplicity, we assume that:

- All chunks are 10 seconds length.
- we only consider the Catch-up TV sessions.
- We use the following encoding profiles: $P_{i\epsilon[0..7]} = [40, 64, 240, 360, 440, 640, 1840, 2540]_{kbps}$.
- Clients do not make any jump forward/backward during the catch-up session [1].

We observe in Figure 3.10 that WA-LRU outperforms LRU when $\Gamma < 50$. This means that when the cache avoids updating blindly the list of cached chunks -especially chunks driving the tail distribution- Earlier chunks are more likely to hold longer within the cache. On the other hand, we observe in Figure 3.11(a) that WA-LRU reduces significantly the update-ratio when $\Gamma < 100$.

[1] Previous studies on video streaming in mobile context show that up to 80% video sessions are without any trick mode (no pause and no jump forward/backward) [50], [107]

This relieves the cache from unnecessary cache-updates and thus reduces significantly the amount of fetching and processing time within the cache.

In Figure 3.11(b), we assess the gain in hit-ratio at per-chunk level. We show that for the low cache sizes, we may achieve at least a gain of 3% for the 3 first requested chunks of videos. This is important for the $joining - phase$ since clients will be served from the cache rather than the origin server, and since this is the most critical phase that impacts the *user-engagement* as we demonstrated in the previous section. In addition, we observe that WA-LRU enhances the hit ratio for the advanced chunks position even though they are less frequently requested.

One limitation of our approach is that we can not predict when clients abort their sessions. For example, in scenario 2 represented in Figure 3.9, if clients A and B are watching the same video content, such as:

$chunk^{A}_{index} \leqslant chunk^{B}_{index} + \Delta_{T=10s}$, then WA-LRU will decide to cache $chunk^{A}_{index}$ because it considers that client B will request it soon. However, client B may abort his session before reaching $chunk^{A}_{index}$, hence, our prediction regarding caching the $chunk^{A}_{index}$ will be inaccurate. Yet, this is very uncommon and can only happen in one particular case: The cache has a very limited storage capacity with regard to the traffic load. In this case, WA-LRU gives more priority to the first chunks to be cached and thus improves significantly the hit-ratio for these chunks, but it behaves less efficient regarding the latest chunks.

3.5 Conclusion

The findings on users' behavior could be leveraged to provide guidelines to design adequate content delivery systems and mechanisms for mobile networks, including for content caching logic, as well to model clients' behavior in that context. In this chapter, we collected the first *large* and *timely* dataset of its kind that allowed us to characterize the users' behavior and engagement at the level of chunk. We ended this chapter by presenting and evaluating WA-LRU a cache replacement strategy that leverages the time-structure of video chunks of a video content. We show that WA-LRU improves the average hit-ratio, in particular the first chunks of HAS videos, while it significantly decreases the processing overhead.

4 Improving caching efficiency and quality of experience with CF-Dash

4.1 Introduction

In HAS, the client player dynamically adjusts the video bitrate as a function of the network condition and CPU usage. While this may reduce the playback interruptions at the client side, it still adversely affect the user experience with the rise of the number of switching between qualities during the video session [78]. In this chapter we deeply investigate the users' bitrate adaptation logic based on empirical traffic observations. In Section 4.2, we address the following questions: How frequently do mobile clients switch between the encoding bitrates during the sessions? and to which extent this affects the performance of caches and CDNs. To answer these questions, we made a thorough characterization and modeling of the switching pattern we observe in the real mobile traffic. We also show through simulations that the high rate of transitions leads to a sub-optimal caching performance.

This motivated us to design a Cache-Friendly Dash player which we present and evaluate in section 4.3. Following, we outline the key steps towards the design of CF-DASH:

- First, we identify the encoding profiles that are commonly requested by clients. We carried out subjective quality perception tests over 26 individual user to quantify the satisfaction of clients when requesting HAS videos. This study aims at identifying which profile should sustain longer within the cache so that we may increase the hit-ratio, we call this profile: *profile-limit.*

- We promote the *profile-limit* within the cache: By default clients with high bandwidth do not systematically move beyond the profile limit since we assume that the user-

engagement is guaranteed at that profile.

- We carry out testbed experiments and trace-driven simulations to evaluate CF-Dash. Simulations show that we may achieve a gain in hit-ratio that ranges from 15% up to 50%.

- Clients still have the option to move beyond the profile-limit. They should manually select profiles higher than the profile limit.

4.2 Analysis on the adaptation logic in HAS

In this section, we rely on the dataset we described in the previous chapter to unveil key characteristics of the rate-adaptation in HAS. More precisely, we study the distribution and frequency of switching between profiles during live streaming and catch-up sessions. Then, we quantify the implication of such switching on cache performance.

4.2.1 Profiles in catch-up and live sessions

A key advantage and characteristic of HTTP Adaptive Streaming is the possibility for the users to dynamically change the encoding bitrate of the video content as a function of the state of the network. Figure 4.1 shows the fraction of sessions in which $profile_i$ (P_i) has been requested. We observe that P_3 was requested in 63% of catch-up sessions, followed by P_2 (60%), followed by P_4 and P_5 (52% of catch-up sessions). While in live sessions, we observe that P_2 figures out in more than 80% of sessions.

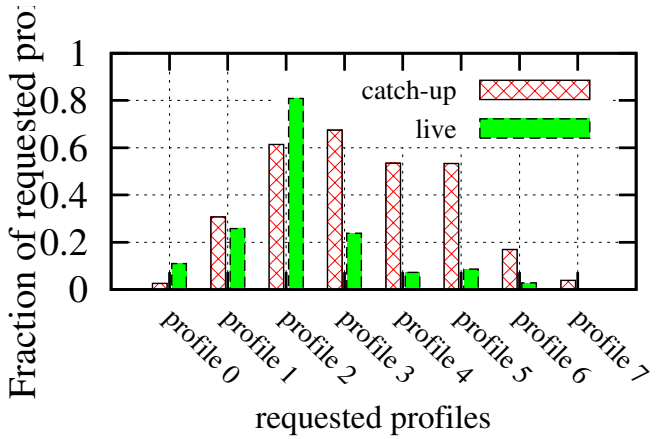
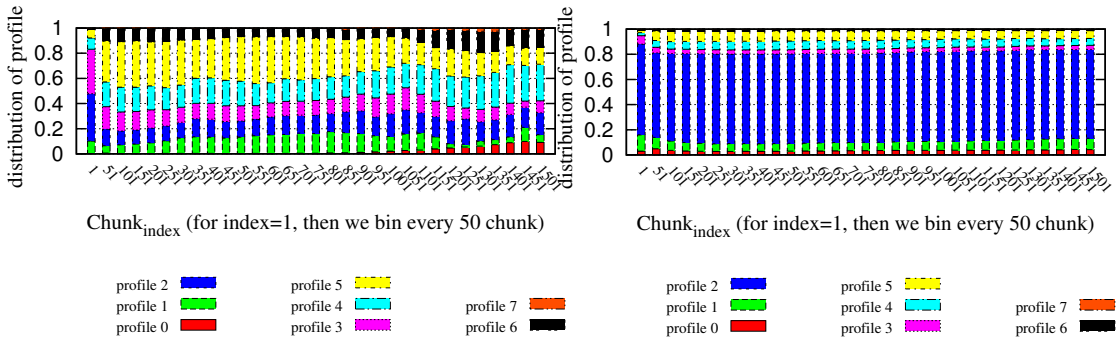


Figure 4.1: Proportion of sessions requesting $profile_i$



(a) catch-up (b) live

Figure 4.2: Distribution of requested profiles with respect to the $chunk_{index}$ for both catch-up and live video streaming

In Figure 4.2(a) and 4.2(b), we bin $chunk_{index}$ every 50 chunk over all HAS sessions, where the $index$ ranges from 1 to the last requested chunk in the session and points to the number of the $chunk$ within the stream. We show in these figures the proportion of the requested profiles in each bin. Interestingly, in Figure 4.2(a), we observe that the breakdown of P_3 is respectively 30% for the first chunk, then it consistently falls below 20% for $chunk_{index>50}$. However, in Figure 4.1, we observe that P_3 was requested in at least 63%. This is because clients tend to start with the lowest encoding profiles. For example, we observe that in 82% of catch up sessions, clients start with a profile lower than P_4. Then, the player upgrades the quality as soon as it captures a high bandwidth. In HLS, the first profile is set by default. Content providers usually define low encoding rates for the first 3 chunks. The rationale is to build the playback buffer as quickly as possible in order to guarantee a smooth playback at the joining-phase. Once the buffer is full, clients decide to switch to higher profiles as far as they experience a high available bandwidth. This is well illustrated in Figure 4.2(a), since profiles higher than P_3 accounts for more than 60% of requests when $chunk_{index\geq 51}$.

4.2.2 Video bitrate adaptation

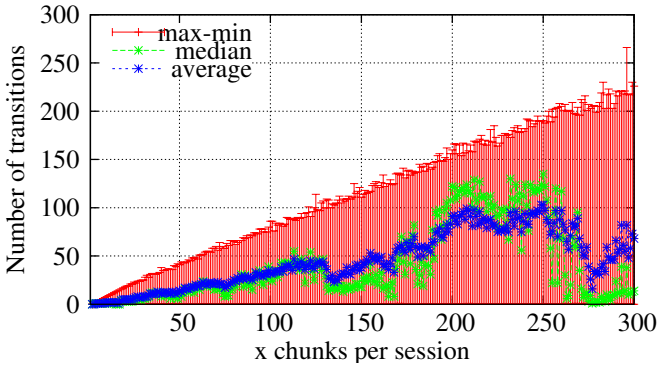
We show in Figure 4.3(a), the maximum, minimum, average and median number of transitions when requesting N chunks per session. We stopped at N equal to 300 chunks per session since a minority of sessions reaches that stage of catch-up sessions (around 25 catch-up sessions, see Figure 4.3(b)), which is not enough representative. Figure 4.3(a) shows that regardless of the number of requested chunks per session, the minimum number of transitions during

a session is always 0. This suggests that in these sessions, clients never switched between bitrates either because the content provider might have defined a single video bitrate for the considered catch-up content, or clients may experience a high available bandwidth that hindered them from making any transition during the session. In the same figure, we also observe that clients start making transitions after requesting at least 2 chunks. This is a property of HLS where clients start buffering low profiles before moving to higher ones in order to guarantee a smooth loading time. We also observe that on average, the number of transitions during a HAS session is in between $[1/6, 1/2]$ of the total requested chunks per session which is considered important and may adversely affect the end-to-end quality of service. We believe that switching between bitrates may sustain the video stream by allowing clients suffering from resources scarcity to switch to low profiles. However, assuming that we have a cache-server as a middle-box between clients and content-hosting servers, we show in the following that switching from one quality to another may affect the performance of the cache.

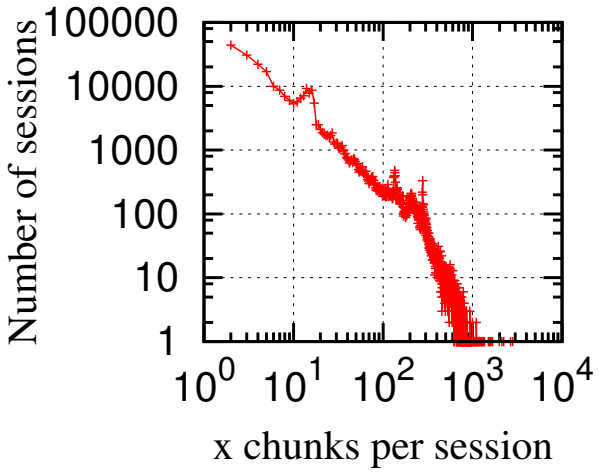
4.2.3 Impact of HAS on caching performance

To quantify the implication of the switching between profiles on caching, we conduct a 15 day trace-driven simulation from the dataset and compare the case where the source is encoded at one single profile (*e.g.* progressive download) to the case where the source is encoded at multiple profiles (*e.g.* HAS). We assume that the cache is deployed at the Gi interface, *i.e.* on the same link where we have fixed our probing system.
For the sake of simplicity, we assume that:

- All chunks are 10 seconds length.
- we only consider the Catch-up TV sessions.
- When the source is provided at multiple profiles, we consider the following profiles: $P_{i\epsilon[0..7]} = [40, 64, 240, 360, 440, 640, 1840, 2540]_{kbps}$.
- When the source is provided at a single-profile, we consider that all chunks are encoded at $640kbps$ (*i.e.* P_5).
- We use LRU as a cache replacement strategy.



(a) Number of transitions during HAS sessions



(b) Number of sessions reaching $chunk_i$

Figure 4.3: Switching during HAS sessions

- Clients do not make any jump forward/backward during the catch-up session [1].

We evaluate the performance of the cache for different values of the capacity C such as: $C = \Gamma * S$, where S represents a video size of 50 MB and Γ represents the number of videos that the cache may hold. Γ ranges from 2 to 300.

In Figure 4.4, we compare the average hit-ratio for both single and multi-profile cases. When $\Gamma \geq 100$, we observe that the difference in hit-ratio can reach 15%, even though the encoding rate we have defined for the single-profile is high. This simulation clearly shows that the instability of client players potentially reduces the performance of the cache. Raising the

[1] Previous studies on video streaming in mobile context show that up to 80% video sessions are without any trick mode (no pause and no jump forward/backward) [50], [107]

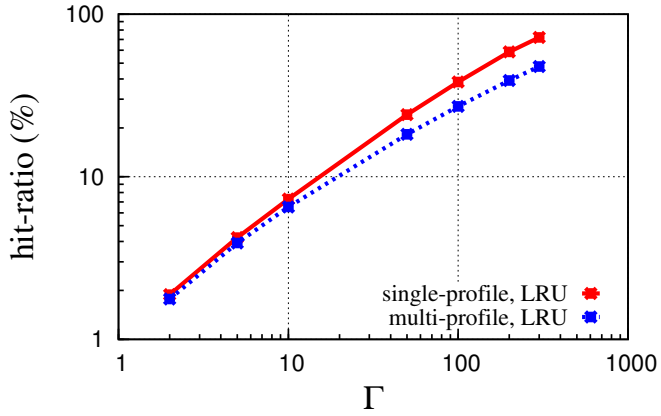


Figure 4.4: single profile VS multi-profile: Implication on caching efficiency

number of qualities of the same $chunk_{index}$ from the same video content in the network will potentially decrease the probability to find this chunk within the cache. Consequently, clients will be redirected to the origin server which subsequently adds further delays to download the video segment. So far this pushes client-players to trigger more and more the adaptation mechanism. To tackle this issue, one possible solution would be to decrease the number of transitions (*i.e.* maximizing the sojourn time per profile) during the video session, while sustaining the user quality of experience. In this chapter, we propose CF-Dash which aims at enhancing the cache performance and sustain the QoE.

4.2.4 Markov characterization of the switching between profiles

Markov chains are best to describe and model the transitions between profiles [104] during a HAS session. Deciding for the next profile does not require any prior knowledge about the past requested profiles. Additionally, since the switching between bitrates is time-dependent, i.e. clients remain for a while in a particular profile before visiting a new profile, then the continuous-time Markov chain (CMTC) is best to characterize the switching between profiles. To this end, we use our dataset to characterize the CTMC. More precisely, we investigate:

- The sojourn time per profile $P_{i\in[0..7]}$
- The initial distribution $\pi(0) = [\pi_i(0)]$
- The infinitesimal generator
- State of the CTMC ($\pi(t)$) at instant $t > 0$

We start to formulate the switching process between profiles, and then identify step-by-step, all parameters needed to feed that model based on empirical observations. Formally, let t_0 be the time during which a client decides to switch from profile P_i to $P_{j\neq i}$ in time slot dt, after spending an amount of time T_i in P_i. We note $p_{i,j}$ the probability to move from P_i to profile P_j.Then we have:

$$\begin{aligned} P_{\{i,j\}}(dt) &= P[(X(t_0+dt)=j|X(t_0)=i] \\ &= P[Ti \leqslant dt] * p_{i,j} \end{aligned} \tag{4.1}$$

Sojourn time per profile: $\mu_{i\epsilon[0..7]}$ $P[Ti \leqslant dt]$ represents the cumulative distribution for staying T_i slot at P_i. We use MLE estimation to evaluate the law that best fits the sojourn time in P_i. We find that the exponential distribution fits the empirical data well. We show in Table 4.1, the parameters per each profile.

$Profile_i$	exponential μ_i (seconds)
P_0	583.898
P_1	865.413
P_2	934.798
P_3	995.341
P_4	1107.89
P_5	1088.39
P_6	1128.9
P_7	1236.49

Table 4.1: Sojourn time distribution

Then using Taylor series, Eq (4.1) becomes:

$$\begin{aligned} P[Ti \leqslant dt] &= (1-e^{-\mu_i * dt}) * p_{i,j} \\ &= \mu_i * dt * p_{i,j} + o(dt) \end{aligned}$$

This suggests that the *transition rate* ($\mu_{i,j}$) from P_i to P_j is also exponentially distributed, such as: $\mu_{i,j} = \mu_i * p_{i,j}$

The initial distribution $\pi(0) = [\pi_i(0)]$ In HLS, the client begins by retrieving the master MPD file which depicts the list of encoding profiles proposed by the content provider. The media player begins the playback with the first bitrate listed in the master MPD; it is expected that the first bitrate to be selected is the one suggested by the content provider[68]. Hence, all clients should start with a preset profile. This is consistent with what we observe in Figure 4.5. In this figure we show the percentage of videos in which clients request the same $chunk_{index}$ from the same catch-up videos, but with different encoding bitrates. We observe that for $chunk_{index=1}$, all clients requesting the same catch-up video start always with the preset profile in the manifest file. Then client-players change the video-bitrate as soon as they experience a high bandwidth.

The initial distribution $\pi(0) = [pi(0)_i]_{i\in[0..7]}$ is derived from the analysis we carried out in section 4.2.1:

$$\pi(0) = \begin{pmatrix} 0.0029 & 0.0981 & 0.3704 & 0.3473 & 0.0888 & 0.0843 & 0.0082 & 0 \end{pmatrix}$$

$\pi(0)$ shows that in around 81% of sessions, clients start always requesting profiles lower than or equal to P_3, since clients start always requesting low profiles to reduce delays at the *joining-phase*, then the distribution in mostly concentrated between profiles 3,4 and 5 for $chunk_{index>50}$.

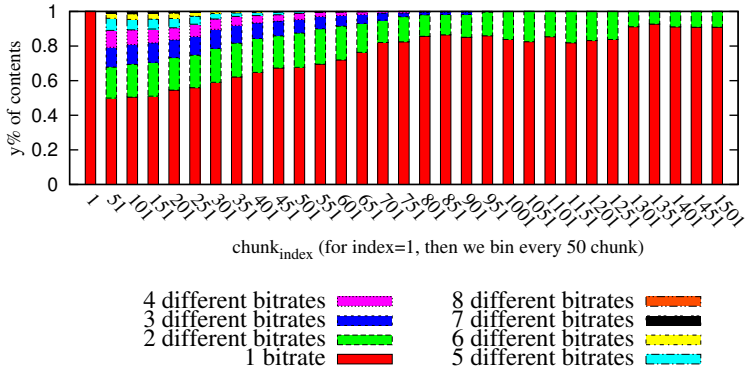


Figure 4.5: Percentage of contents in which clients request different profiles when requesting the same $chunk_{index}$

The infinitesimal generator: Transition from $i \rightarrow j$ Let $Q = [q_{i,j}]$ be the infinitesimal generator of the CTMC we are characterizing, such as:

$$q_{\{i,j\}} = \begin{cases} \mu_{i,j} = \mu_i * p_{i,j}; \forall\, i \neq j \\ -\sum_{j=0;j\neq i}^{j=7} \mu_{i,j} = -\sum_{j=0;j\neq i}^{j=7} \mu_i * p_{i,j};\ i = j \end{cases}$$

The transition probabilities ($p_{i,j}$) are the coefficients of the transition probability matrix P from one state i (rows) to the next state j (columns). we derive these coefficients from our dataset:

$$P = \begin{pmatrix} 0 & 0.0997 & 0.2650 & 0.2685 & 0.1784 & 0.1358 & 0.0460 & 0.0066 \\ 0.0053 & 0 & 0.1533 & 0.3482 & 0.2683 & 0.1753 & 0.0464 & 0.0029 \\ 0.0036 & 0.0405 & 0 & 0.4237 & 0.28 & 0.2034 & 0.0456 & 0.0028 \\ 0.0021 & 0.0410 & 0.2496 & 0 & 0.4204 & 0.2486 & 0.0339 & 0.0041 \\ 0.0007 & 0.0192 & 0.1271 & 0.3867 & 0 & 0.4080 & 0.0500 & 0.0079 \\ 0.0009 & 0.0099 & 0.0817 & 0.2005 & 0.4790 & 0 & 0.2016 & 0.0260 \\ 0.001 & 0.0055 & 0.0303 & 0.0739 & 0.1617 & 0.6060 & 0 & 0.1213 \\ 0.0004 & 0.0018 & 0.0078 & 0.0160 & 0.0441 & 0.2266 & 0.7031 & 0 \end{pmatrix}$$

We infer the state of the CMTC as follows: Let $\pi(t)$ be the distribution of $P_{i\in[0..7]}$ at $t > 0$. Since the sojourn time in one profile is exponentially distributed, then we have:

$$\pi(t) = \pi(0) * e^{Q*t}$$

$\pi(t)$ achieves the stationary regime. In figure 4.2(a), we observe that the distribution of $P_{i\in[0..7]}$ tends to stabilize. We observe that the distribution of profiles is likely to be uniform for $chunk_{50<index<900}$. However, we observe a bias for $chunk_{index>900}$. That is because the set of sessions in this range is not enough representative to conduct inference.

4.3 Motivation to CF-DASH

Our motivation stems from the results of the empirical study that we carried out in the previous section. Next, we report the main findings. Then, we introduce CF-DASH.

4.3.1 Empirical study summary

Our motivation for the present study is derived from the 3 following observations:

- We showed that clients' players often switch between the encoding bitrates. On average the number of transitions during a HAS session is in between [1/6, 1/2] of the total requested chunks per session. In case where a caching system (e.g. CDN or transparent cache) is deployed at the Gi interface of the mobile carrier, our simulations show that this high switching behavior and video bitrate selection heterogeneity reduces the cache hit ratio by 15%.

- The analysis of the encoding bitrates in the collected data led us to classify them into a set of representative ranges. Profile 5 was clearly the most frequently requested profile over all HAS sessions, while the lowest profiles are mostly requested at the beginning of the sessions.

- We used markov chain to characterize the switching between profiles. We estimated the probability to move from $P_{i\in[0..7]}$ to profile $P_{j\in[0..7];j\neq i}$ (c.f. transition matrix P). We observed that the probability to move from $P_{i=5}$ to $P_{j>5}$ is 22%, then when clients are in $P_{i=6}$, they are more likely to switch to $P_{j=5}$ with a probability equal to 60%. When being in P_7, clients move to P_6 and P_5 with a probability of 92%. All this suggests that when being in P_5, we observe that a significant proportion of transitions are limited within the highest profiles. When being in P_4, around 45% of transitions are made to the upper profiles, and that 48% are made from $P_{i=5}$ to $P_{j=4}$. This intermittently switching between the highest profiles does not necessarily yield improvement of qualities of experience. But instead, it adversely affects some network components such as caches.

These three observations show that there is a potential opportunity to improve the caching efficiency in the mobile network by favoring one specific profile in the HAS adaptation logic without compromising the user engagement and the ability of HAS clients to react adequately in case of network congestion and adverse conditions on the server and client devices.

4.3.2 QoE evaluation

In this section we show the results of a subjective quality perception tests we conducted on 26 individual users. Tests were conducted with five ranges of quality from bad 1 to excellent [4,5]. The purpose of the experiment was to gain insights into the user quality of experience to draw reliable conclusions regarding the user engagement. Type of videos we used fits well the type

of HAS videos we observe in the dataset: News, Animation and Film. Subjects were invited to watch video content twice: on smart phone, then on Tablet. The resolution description and results of the average Mean Opinion Score (MOS) of each profile are reported in Table 4.2.

Profile	Video resolution	User perception (MOS)
Profile 1 (P_1)	176 x 144	bad (1)
Profile 2 (P_2)	280 x 160	bad (1.2)
Profile 3 (P_3)	320 x 180	medium (2.2)
Profile 4 (P_4)	400 x 224	good (3.3)
Profile 5 (P_5)	480 x 270	good (3.8)
Profile 6 (P_6)	640 x 360	Excellent (4)
Profile 7 (P_7)	1024 x 576	Excellent (4.5)

Table 4.2: MoS of the perceived quality of experience

In [88], authors suggest that a minimum guaranteed MOS equal to 3 is required to ensure an acceptable service quality for any connected user. This requirement is guaranteed with P_4 and P_5. Based on these considerations, we observe that by defining a wide range of encoding profiles -such as content providers do actually- brings clients' players to acute instability which pushes them to switch very often. However, the tests that we have conducted underline that the user engagement is guaranteed at a specific profile. In CF-Dash, we leverage these key findings to improve on-network caching.

4.4 Cache Friendly-Dash

In this part, we introduce CF-Dash adaptation logic and detail our implementation and test-bed setup.

4.4.1 Cache Friendly-Dash in a nutshell

In CF-Dash, we intentionally promote one specific profile within the cache. This is achieved by pushing clients' players to request one specific and commonly requested profile, we call it *profile-limit*. By doing so, the *profile-limit* will be populated within the cache and clients will be more likely to be served from the cache. In CF-Dash, even though clients experience a high bandwidth, by default clients' players never ask for profiles above the *profile-limit*. The *profile-limit* is chosen in such a way we guarantee a good user-engagement. Hence switching to the highest profiles would be unnecessary and would not affect the user-engagement. For

clients who want to move beyond the *profile-limit*, they have to select manually profiles higher than the *profile-limit*. The rationale behind this idea is to prevent clients to turn systematically to the highest profiles when they experience a high bandwidth.

4.4.2 PoC implementation

In our PoC, we chose GPAC [59] as an Open Source multimedia framework installed on end-terminals, since it incorporates the major DASH standard components. We chose Squid as a proxy cache and Apache as HTTP server. We integrated our proposed changes within the core of GPAC, and carried out test-bed experiments to evaluate our approach and validate our implementation.

In CF-Dash, clients' players and cache-servers share information so that the clients' player learn about the *profile-limit*. In MPEG-SAND, 3 options have been considered to make clients' players and content delivery servers communicate: either by modifying the MPD at some network level (*i.e.* CDN), or by adding new fields in the HTTP header, or by adding a new interface through which clients' players and network components exchange messages. In our PoC, when a client starts a new video session. First, her HTTP-GET request gets forwarded to the proxy-cache server. Then, the proxy-cache is configured so that it does not cache the initialization segment [2] that points to the *profile-limit* but caches all the rest. Squid implements the *X-CACHE* header. Therefore, if the init segment is not cached, the X-CACHE header will contain a miss. The HTTP-Response being sent to the client, the latter parses the header fields and maps the miss into the *profile-limit* candidate. Herein we consider 2 cases:

Leader This is the case when a new video is being added to the catalog of VoDs, and a first user (we call it leader) requests that video. This first client will not be able to identify the *profile-limit*, since none of the init segments is cached. This pushes the leader to download all init segments from the origin server and admits that the latest downloaded init segment (highest profile) corresponds to the profile limit.

Followers Followers refer to all users who will successfully identify the *profile-limit* based on the aforementioned mechanism.

[2] During the connection setup, the clients' player downloads all the init segments defined within the Media Presentation Description. Each init segment is associated to one encoding profile. The init segment has the metadata needed by the player to decode the associated profile.

Pseudo code

Algorithm 2 depicts the pseudo-code of CF-Dash rate-adaptation logic:

Require: Client, AdaptationSet, Representation, Limit_Profile,
1: **if** (disable_switching = TRUE) **then**
2: return
3: **end if**
4: Go_Up=FALSE
5: DL=compute_download_rate(Client)
6: **if** ($Representation.bandwidth \leqslant DL$) **then**
7: Go_Up=TRUE
8: **end if**
9: **if** ($DL \leqslant AdaptationSet.Min\_Representation\_Bitrate$) **then**
10: DL=AdaptationSet.Min_Representation_Bitrate
11: **end if**
12: N=AdaptationSet.total_representations
13: **for** $k = 0 \rightarrow N$ **do**
14: SR=Get_Representation(AdaptationSet,k)
15: **if** ($DL \geq Selected\_Representation.bandwidth$) **then**
16: **if** (!New_Representation) **then**
17: New_Representation=SR
18: **else if** (Go_Up) **then**
19: **if** ($SR.bandwidth \geqslant New\_Representation.bandwidth) and (k \leq AdaptationSet.Limit\_Profile$) **then**
20: New_Representation=SR
21: **end if**
22: **else**
23: **if** ($SR.bandwidth \geqslant New\_Representation.bandwidth) and (k \leq AdaptationSet.Limit\_Profile$) **then**
24: New_Representation=SR
25: **end if**
26: **end if**
27: **end if**
28: **end for**
29: **if** ($disable\_switching = FALSE) and (New\_Representation) and (New\_Representation \neq Representation$) **then**
30: Representation = New_Representation
31: **end if**

Algorithm 2: CF-Dash: rate adaptation logic

- In Require: *AdaptationSet* and *Representation* are dash standard words naming.*Representation* refers to the encoding profile.

- Line 1: *disable switching* becomes true when clients select manually profiles higher than the profile limit.

- Lines [6,8]: DL corresponds to the download speed of the last downloaded chunk. In these lines, we compare the last representation (last requested profile) with the new DL.

- Lines [13,28]: This loop parses all existing profiles and identifies the profile that fits the user bandwidth. Conditions 19 and 23, forces the clients' player to not surpass the profile limit whatever the value of DL.

- Line 30: We save the actual representation, so that we can compare it with the next computed DL, when requesting the next chunk.

4.5 Evaluation

In this part, we demonstrate through simulations and test-bed experiments that CF-Dash adaptation logic improves the cache efficiency on large scale.

4.5.1 Simulation evaluation

We performed a trace-driven simulation to evaluate the impact of CF-Dash on caching efficiency. The trace considered in the simulation corresponds to HAS sessions collected over a period of 2 weeks from the dataset that we introduced in the previous chapter. Each record in the trace corresponds to one HAS session; it includes the timestamp of the connection setup, the reference of the requested video, the number of requested chunks, and for each video bitrate switching the timestamp and the newly requested Profile. The caching system is deployed at the Gi interface on the mobile network. For the sake of simplicity, we assume that: All chunks are 10 second long. The encoded profiles are selected with respect to the following encoding profiles:
$P_{i\epsilon[0..7]} = [40, 100, 210, 250, 510, 900, 1500, 3500]_{kbps}$. LRU is the content replacement algorithm of the caching system. We evaluate the performance of the cache for different values of the capacity C such as: $C = \Gamma * S$, where S represents a video size of *10 MB*, typically a short video-clip of 5 minute long, and Γ represents the number of videos that the cache holds. In Figure 4.6, we evaluate CF-Dash with the baseline (i.e. native trace) based on 3 key metrics:

- Average hit-ratio
  ($H = \frac{requests\ served\ from\ the\ cache}{all\ requests}$): average ratio of requests successfully served from the cache.

- Gain on hit-ratio
  ($G_H = \frac{H_{CF-Dash} - H_{baseline}}{H_{basline}}$), to assess the gain we may achieve on hit-ratio with CF-Dash in comparison to the baseline.

- G_H per $chunk_{index}$.

We observe on Figure 4.6 a significant improvement in the hit-ratio when clients adopt CF-Dash adaptation logic. When the *profile-limit* is set to P_4, we reach more than 40% of G_H. When the *profile-limit* is set to P_5, we still reach a gain of 15%.

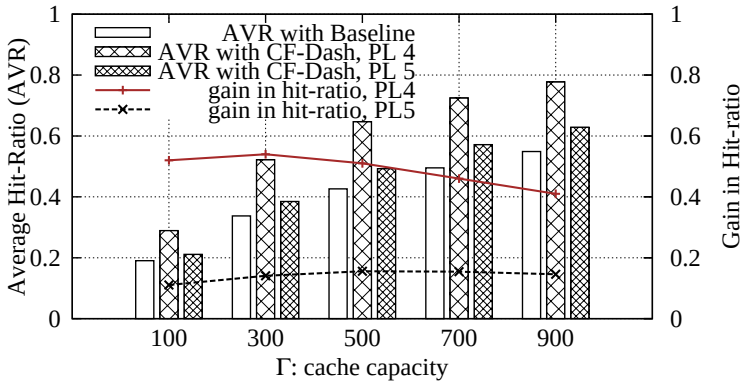
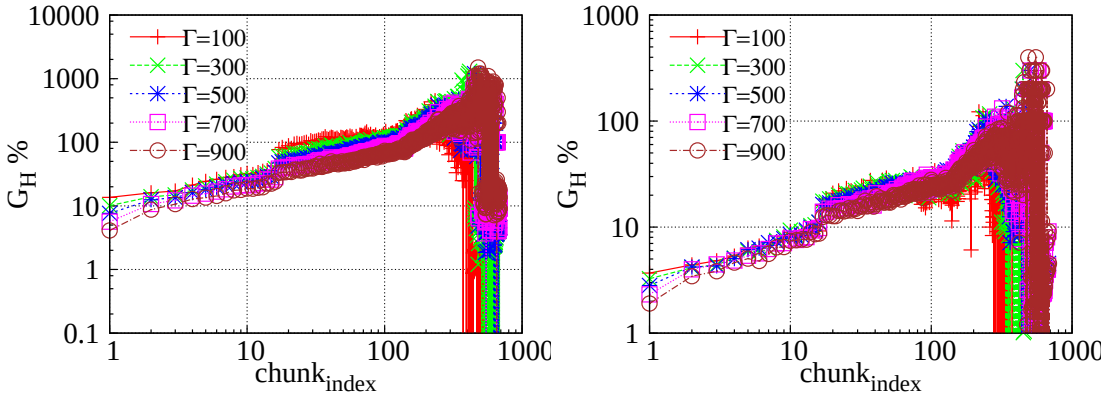


Figure 4.6: Average and Gain in hit ratio



(a) Average Hit-Ratio per chunk index when setting profile limit to 4

(b) Average Hit-Ratio per chunk index when setting profile limit to 5

Figure 4.7: *Metrics evaluation: Average hit-ratio, G_H*

In [47], we observed that around 65% of HAS sessions are being aborted from the start when clients experience long delays during the *joining-phase*. Figures 4.7(a) and 4.7(b) show that reducing the aggregated number of switching gives more chance to the earlier $chunk_{index}$ to hold in the cache. This is because the earlier chunks compete less with the latest chunks for whom CF-Dash takes action. This is important for the *joining-phase* since clients will be served from the cache rather than from the origin server, and since this is the most critical phase that impacts the *user-engagement*.

4.5.2 Experiments evaluation

Test bed and scenario

The experimental test-bed and setup is similar to that described in [75]. We install Gpac on both end-terminals. The configuration of each component of the test-bed is as follows:

- The Apache HTTP server hosts a catalog of 20 videos encoded with respect to the profiles used in the previous simulations. Each video is hosted jointly with its MPD. Videos are segmented into 12 chunks, 10 second long each one. We skip P_0, since it is rarely requested (c.f. figure 4.2(a)).

- The cache proxy is placed between the clients and the origin server. Evaluation tests are carried out by varying the cache size capacity C. We keep on the same parameters used in the simulations: $C = \Gamma * S$, where S represents a video size of *12 MB*, which is the average size of one video from our catalog, and Γ represents the number of videos that the cache may hold. We set the *profile limit* to P_5. We use Dummynet on both clients to emulate the bandwidth variation. We use real world bandwidth variation traces [85] of clients being covered by 3G/HSPA network in Norway. We periodically reproduce the network conditions when moving by train from Oslo to Vestby, and the way back (c.f. [13]).

Finally, we simulate 200 clients requesting videos from the catalog. Content popularity is distributed according to Zipf law with parameter equal to 1.

Results and interpretation

We evaluate the following:

- Profiles distribution: We analyze the proportion of each profile per requested chunk. Our goal is to promote the *profile-limit* within the cache.

- Gain in hit-ratio (G_H)(c.f. Section 4.5.1).

- Stability of clients' players: We show how CF-Dash reduces the aggregated number of switching during HAS sessions.

Profiles distribution Most HAS implementations (commercial and open source) agree that the first requested chunk from a video content should be preset in the MPD by the content provider. Usually, the content provider defines the start-up profile as the lowest encoding profile, so that to guarantee a smooth playback at the joining-phase. In figure 4.8, we observe that this holds the same in Gpac for $chunk_1$. Then, clients decide to move to the profile that fits best their available bandwidth. We observe that the *profile-limit* is largely requested by client-players. Profiles higher than the *profile-limit* (i.e. P_6 and P_7 in our case), are still being requested. This is because leaders do not recognize the *profile-limit* as do the followers, since all init segments are downloaded from the origin server.

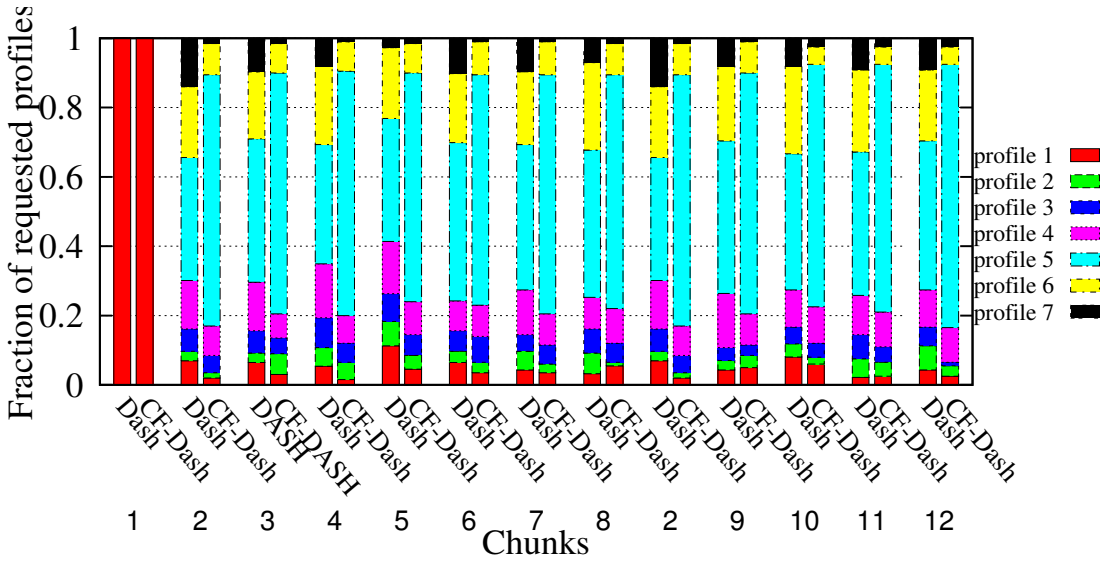
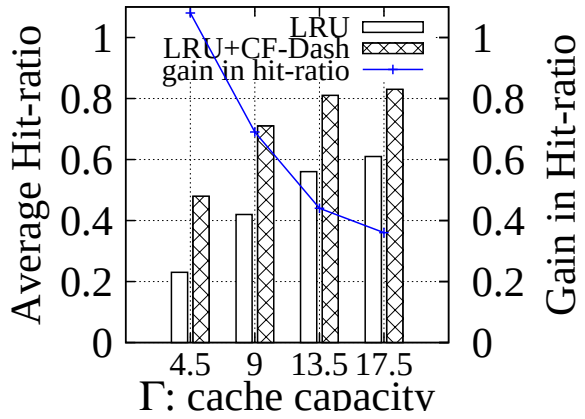


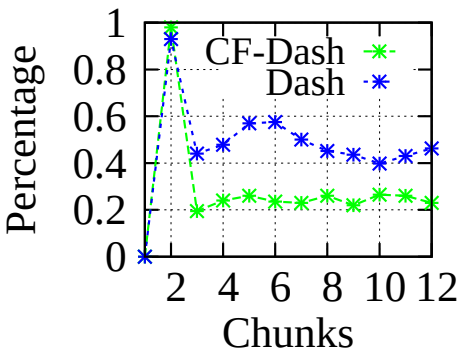
Figure 4.8: Profiles distribution

Gain in hit-ratio Figure 4.9(a) confirms that the G_H significantly increases. When $\Gamma = 4.5$, we observe that CF-Dash adaptation logic allows doubling the performance of the cache with regard to the dash adaptation logic. Then, although the G_H decreases as the cache size increases, we still maintain a promising gain in hit-ratio that reaches around 38% when $\Gamma = 17.5$. Our evaluations demonstrate that CF-Dash improves significantly the hit-ratio.

Stability of client-players Figure 4.9(b) shows that most HAS sessions upgrade their video quality when moving from the first to the second chunk of the stream. This is because the emulated bandwidth variation is most of the time higher than the preset profile. With CF-Dash,



(a) Average Hit ratio



(b) Percentage of switchings per $chunk_{index}$

Figure 4.9: Evaluation: Hit ratio and stability

we decrease the aggregated number of switching over all HAS sessions by 20%. Hence, this promotes the *profile-limit* within the cache and increases the opportunity to the followers to be served from the cache.

4.6 Conclusion

In this chapter, we investigated the effect of the rate-adaptation in HAS on cache performance. Having observed that the QoE is guaranteed from a specific encoding bitrate, we designed CF-Dash - a cache friendly DASH player- with the aim to have further control on the rate-adaptation and to promote one specific encoding profile within the cache. In our future work, we will further investigate the ideal profile to be cached based on the content characteristics and to define incentive strategies to encourage clients requesting the same encoding profiles.

5 CPSys: A system for mobile video prefetching

5.1 Introduction

Today, mobile devices are commonly used to watch videos everywhere. Within a few years mobile devices are likely to become the users' preferred choice for accessing the Internet [1] while according to [92], multimedia content represents already a significant portion of the mobile traffic today. This growing trend is to a large extent driven by social networks. Online social networks (OSNs) are reshaping the way videos are being consumed. First, by boosting popularity of video content within groups of users with similar interest [45]. Second, by providing viewing recommendation for each user. EdgeRank [25] is used by Facebook to sort items on the news feed of the individual users based on affinity, weight and time decay scores. These key factors drive users to conduct a specific behavior when browsing their news feed and allow for a prediction of content a user is interested in. This social or interest based interaction can be leveraged by networking actors, in particular over-the-top (OTT) content providers and content delivery networks (CDNs) to predict future behavior of users and decide if it is worth pushing videos to the interested users at a particular time. Hence, if properly designed, prefetching videos can alleviate the network during peak traffic periods, i.e. flash crowds. Besides, it can improve the user experience since it avoids buffering delays or stalling of the streaming video as the content can be played from local storage.

In this chapter, we design and implement CPSys, a Central Predictor system to prefetch videos on users' mobile devices. Our prefetching scheme aims to answer the 3 following questions:

[1] http://www.morganstanley.com/about/press/articles/4659e2f5-ea51-11de-aec2-33992aa82cc2.html

- *Which content should be prefetched?* To determine which content the user is interested in is hard in general. The distribution of lifetime video views combined with the preferences of the users are key factors to build an accurate prediction model. Our model relies on an enhanced recommendation techniques tailored to the prefetching requirements. We capture users with similar interests and give priority to the most viewed and most recent videos that have been requested by the user's neighbors to be prefetched.

- *When to trigger prefetching?* We define two control mechanisms: a network-oriented and a state-transition control mechanism. The former allows an efficient use of network resources while the latter aims at controlling the prefetcher agent running on the user device. Combined, these control mechanisms avoid the agent to prefetch videos aggressively or randomly.

- *How many videos are to be prefetched?* This is a design choice. We differentiate between 2 kinds of users: Heavy and light users. We correlate the number of videos to prefetch with the user's past activity and conclude on the number of videos to be prefetched.

Our system design can be leveraged by all networking actors, in particular by OTTs, CDN providers, and Telcos. We implemented several features to make it open and flexible for future extension.
The rest of this chapter is organized as follows. Section 5.2 describes the background and related works. Section 5.3 represents observations on the behavior of clients through analyzing the used traffic trace. In Section 5.4, we introduce our system and configure it with respect to the observations drawn from the previous section. In Section 5.5, we extensively evaluate our system with different assumptions. Section 5.6, we provide a proof-of-concept implementation of our system. Finally, we conclude this chapter in Section 5.7.

5.2 Background and related works

Recent studies have brought forth the benefits of prefetching videos on mobile devices. NetTube [30] and SocialTube [66] were designed to leverage P2P overlays to download YouTube videos. In these systems, authors proposed a prefetching scheme for prefetching prefixes of videos to improve the user experience at the joining phase. In SocialTube, the authors assumed optimistic hypothesis to evaluate their prefetching strategy. They limited their study

to 2000 videos shared among 5000 nodes. While this holds reasonable to assess the maximum performance the system may achieve, we believe that this introduces a bias on the performance results. In contrast, the present chapter exploits real traffic traces collected over a large mobile carrier in Europe; thus all the specific characteristics of mobile video traffic are captured and taken into account in the assessment works reported in this chapter.

Recommendation systems have been widely investigated by the research community. In [22], authors proposed a decentralized system for disseminating news items in a large-dynamic setting with no central authority. In [95], authors represent a survey of the state-of-the art of existing recommendation techniques. In this chapter, we use a k-nearest neighbor-based collaborative filtering (CF) technique to identify people with similar interests. Once we we identify the neighbors, our new prefetching strategy selects videos to be prefetched for each individual user: We present the MPMR policy which gives priority to the *M*ost *P*opular and *M*ost *R*ecent videos viewed among similar users to be prefetched.

In [43], authors show that prefetching has potential benefits regarding energy savings. Prefetching when the device is connected to Wifi can reduce the energy consumption by 10% with respect to a 3G connection. Yet, the authors did not investigate the fundamental and prior question that should be addressed: which content should be prefetched to individual users? Even if connected to Wifi, an aggressive prefetching strategy, i.e. prefetching all videos, would drain the battery very fast which, as well, leads to a bad user experience. In this respect, we focus on the primarily question: *what to prefetch ?* Once we identify the video candidates, we address the second question: *when to prefetch?*

Prashanth and al. [74], proposed to prefetch advertisements (ads), to achieve energy savings. Ads form a very small set of videos being watched by the user over a day. In this chapter we do not limit our study to ads. Instead we prefetch all videos that may interest an individual user including the ads. Alessandro et al. [41] proposed to periodically prefetch bundles of popular content videos on mobile devices. In the prefetching context, we believe that the term *popularity* has no absolute meaning. A content might be locally popular inside a group of users sharing similar interests, but not globally popular and vice versa.

As briefly discussed above, none of the presented related works address several fundamental questions that we see as being of fundamental importance to prefetching. To this end, we carry out analysis and draw lessons using real mobile traffic traces. Based on these observations, we

design CPSys a network-friendly prefetching system. Then, we assess the proposed mechanism using real traffic traces. At the end, we provide a PoC implementation of CPSys.

5.3 Traffic analysis

In this section, we introduce our dataset and provide analysis and findings on users' behavior. Then, we use these findings to infer the design principles of our prefetching system.

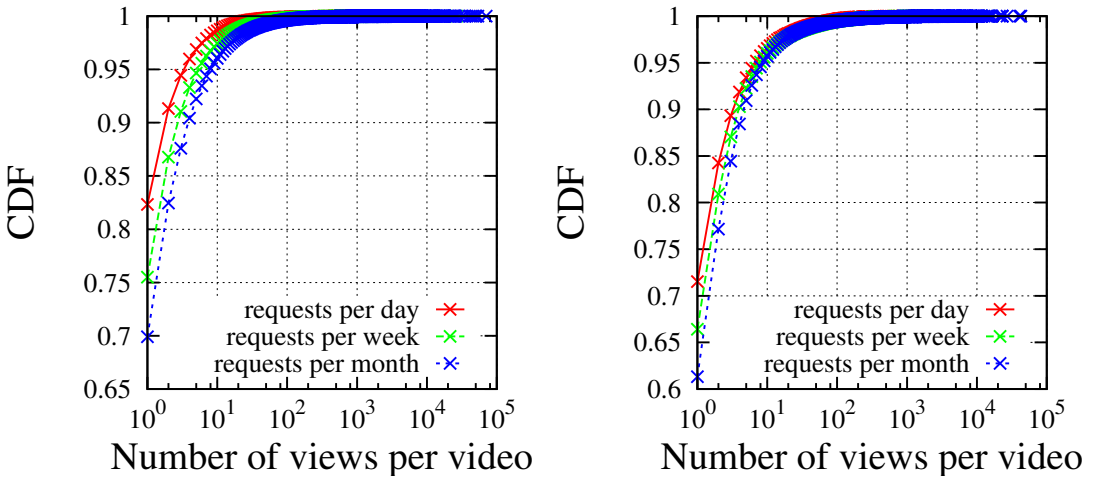
5.3.1 Dataset

We rely on a large dataset gathered at all G_i interfaces of all GGSNs deployed by the aforementioned mobile carrier in France. The dataset consists of logs of video streaming sessions generated by all connected devices of the carrier's subscribers. The logs were collected from 8 January 2014 to 28 April 2014. Due to maintenance reasons, the monitoring infrastructure was disabled for 27 days, which makes the real period of data collection lasting about 94 days. These disruptions do not introduce any bias in the data analysis and simulations reported in this chapter since they are not achieved over the whole duration of the dataset. We limit our study to 2 subsets of video traffic: requests for YouTube (YT) videos and requests for Facebook (FB) videos. Table 5.1 gives an overview of both these subsets of traffic. Parsing the HTTP header in FB and YT traffic flows enables to extract the unique video identifier (noted *reference ID*) requested by the users. For illustrative purpose, typical URI from YT and FB are given in Table 5.1, the field in bold pointing to this *reference ID* of the video. To preserve confidentiality and privacy, our dataset is anonymized during an early stage in the collection process.

Dataset name	Number of unique users	Number of unique videos	Number of requestes	Service provider	Typical URI
YT (Youtube dataset)	3,179,296	10,676,156	64,722,755	Google CDN	r8—sn-4g57kue6.youtube.com/videoplayback?id=↩ **d1875abcee9b6d33**&itag=36&source=youtube&....
FB (Facebook dataset)	399,645	2,856,321	14,305,404	Akamai	video.ak.fbcdn.net/hvideo-ak-prn2/v/↩ **1608327_750958311600439_982850231**_n.mp4?...

Table 5.1: Properties of the two used datasets

In the following, we provide empirical traffic observations and Findings (noted **F**), upon which we establish the design principles of our prefetching system. More precisely, we investigate 5 fundamental traffic properties: *video popularity distribution*, *distribution of the number of views per user*, *relationship between request frequency and request inter-arrival rate*, *video lifetime distribution*, and *load variation across the day*.



(a) YT data (b) FB data

Figure 5.1: Video popularity distribution on 3 different measurement periods: 1 day, 1 week and 1 month, starting from 03/28/2014

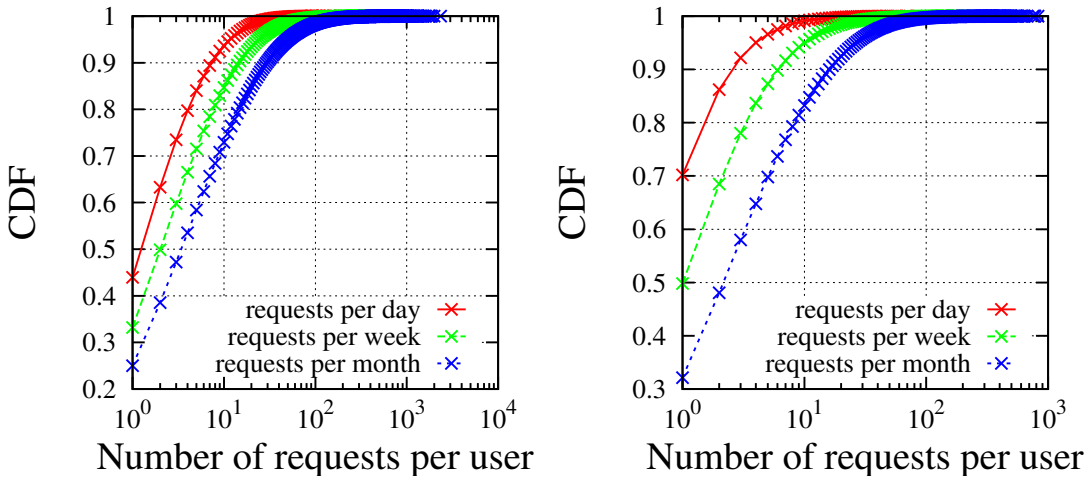
5.3.2 Distribution of the number of views per video

Figure 5.1 shows the popularity distribution of YT and FB videos over 3 different measurement periods: one day, one week, and one month starting from Monday, March, 3, 2014. The skewness of popularity distribution of YT and FB videos is very similar and follows a Zipf-like distribution. Around 75% of YT videos and 65% of FB videos are requested only once.

F1: Most of the videos are requested only a few times, and a majority only once. Therefore, their future consumption is hard to predict. Popularity is thus an important factor to maximize the prediction accuracy and to best manage network resource utilization: a safe prefetching strategy would be to favor the prefetching of popular videos.

5.3.3 Distribution of the number of views per user

Figure 5.2 shows the CDF of the number of requests per user over the 3 periods of data collection. Figure 5.2 shows that users exhibit various viewing patterns. Few users request far more frequently video contents – we call them heavy users – while the majority is less active – we call them light users. Over a month, we observe that 72% and 83% of active users requested at most 10 YT and FB videos, respectively. Predicting the behavior of light users is hard in general and turns into a typical cold start situation where it is hard to learn the preferences of the user



(a) YT data (b) FB data

Figure 5.2: Number of views per user on 3 different measurement periods: 1 day, 1 week and 1 month, starting from 03/28/2014

from a small set of viewed videos. In prefetching context, this has to be carefully considered since these users access videos on their devices less frequently.

F2: It is important to adapt the prefetching strategy with regard to the user activity. Aggressively prefetching videos to light users does not make sense: Our prefetching system differentiates between heavy and light users based on their past activity.

5.3.4 Relationship between request frequency and request inter-arrival rate

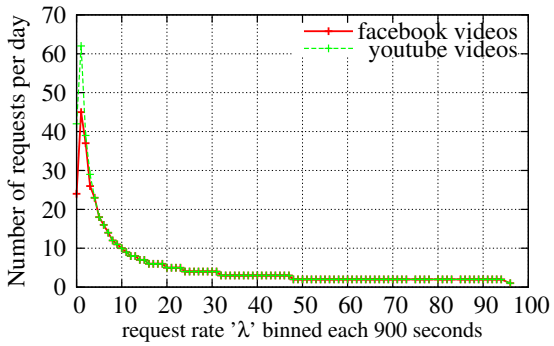
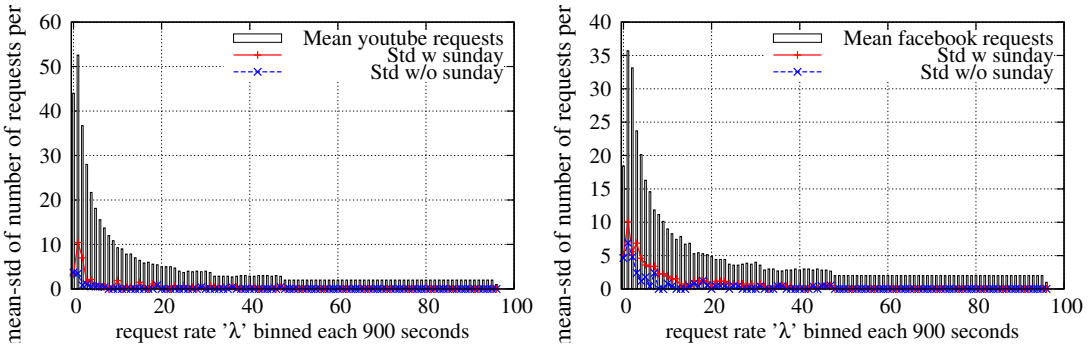


Figure 5.3: YT+FB data

Figure 5.3 quantifies the relationship between the request rate and the request inter-arrival



(a) YT data (b) FB data

Figure 5.4: a: Relationship between the daily request frequency and the daily request inter-arrival rate on 01/13/2014; b and c : on the week starting from 01/13/2014

rate for YT videos, respectively FB videos, on one given day, here Monday, January 13, 2014.

Given a row data - a vector of $(x_\lambda, y_N)_u$ values representing one single user, where x_λ represents the inter-arrival rate of requests over one day and y_N represents the number of viewed videos per day - on the x-axis, we start creating bins of 900 second long. This subsequently generates 96 bins to cover the entire day. Then each user is associated to one bin. The bin 0 in the x-axis refers to users having their inter-arrival request rate ranging from 0 to 900 seconds. Bin 1 corresponds to the range [900 seconds, 2*900 seconds[, etc. On the y-axis, for each of these 96 groups, we show the number of requests per day (y_N) of the 99-percentile most active user within each bin. Figure 5.3 shows that the activity of users could be modeled and quantified with an exponential decay. Users do not request more than 3 videos per day when their request inter-arrival rate is higher than $20 * 900$ seconds.

Figures 5.4(a) and 5.4(b) generalize this observation for the rest of the days of a week (until 20 January). On these figures, users are binned on the x-axis as per the daily average inter-arrival rate of the requests they generated in the week of Monday, January, 13, 2014. On the y-axis, for each of the 96 bins, we show the mean over the seven days of number of requests of the 99-percentile most active user within each bin. The standard deviation is also given twice: once over the seven days, and once over six days from Monday to Saturday (excluding Sunday).

Figures 5.4(a) and 5.4(b) show that the request inter-arrival rate remains slightly similar across the days of the week. The standard deviation gets quickly close to zero for the least active users and it also remains relatively low for the heaviest users. The standard deviation is slightly

higher when it includes Sunday. This illustrates that users have more heterogeneous consumption behaviors on Sunday than on other days. On Sunday, we record a lower activity on mobile devices. This suggests that the majority of users consume less FB and YT videos on their mobile devices while a minority is much more active on Sundays. The patterns are quite similar for YT (Figure 5.4(a)) and FB (Figure 5.4(b)); only the mean request inter-arrival rate of the heaviest users is higher for YT.

F3: The request inter-arrival rate might be used to identify the heaviest users from the least active ones in order to enforce them a specific prefetching strategy. Moreover Figures 5.4(a) and 5.4(b) provide additional insights to fine tune the prefetching system: The 99-percentile most active users request no more than 35 FB videos and 52 YT videos per day on average, which gives an insight on the daily number of videos to prefetch.

5.3.5 Video lifetime distribution

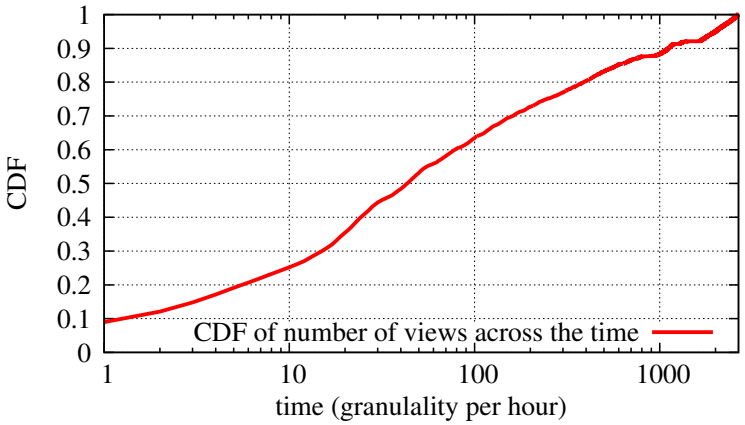


Figure 5.5: Lifetime distribution of videos made available on January 8, 2014

In Figure 5.5, we show the lifetime distribution of YouTube videos across time. We limited our study to videos that have been uploaded to YouTube on January 8, 2014, which is the starting day of the dataset. In this study, the following process was applied:

- First, the unique *reference IDs* of the YT videos in the dataset are extracted from the URIs (as explained in Section 5.3.1).

- Second, the *reference-id* of each of these videos is decoded into a base64 encoding scheme to get the unique identifier of the video attributed by YouTube to index the content.
- Third, the YouTube API [2] is used to retrieve the day the videos were made available online and their category.
- Fourth, we observe that 4554 YouTube videos were uploaded on 8 January 2014 and are present in the dataset.
- Fifth, the aggregated cumulative distribution of the number of views of these videos is computed at a granularity of 1 hour and for the whole period of the dataset (94 days). An offset is associated to each of these video to shift the first request of each video to the origin. Then we use the same offset to shift the rest of requests of the same content.

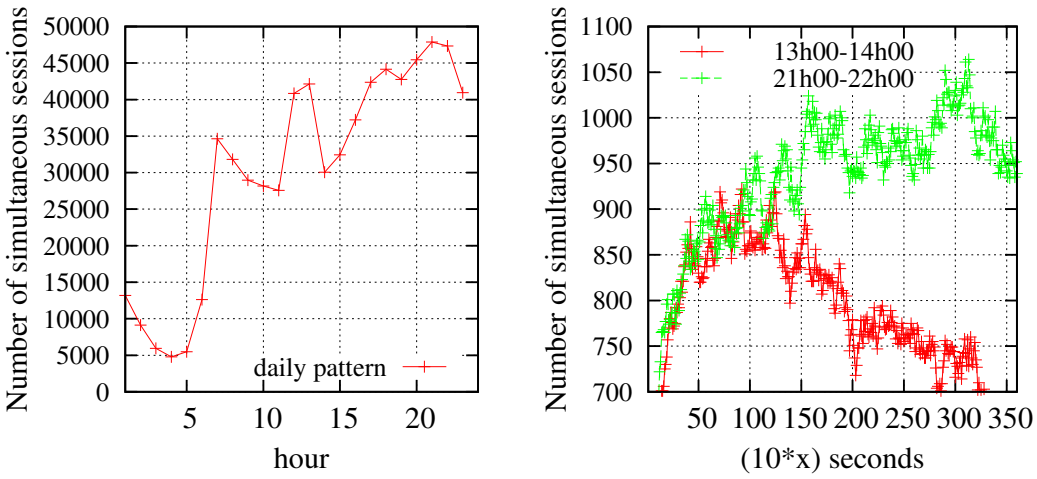
We plot in Figure 5.5 the cumulative distribution of the number of views of these videos. The figure clearly shows that most of the views happen in a short time frame after the videos are made available: 10% the first hour and 40% the first day.

F4: According to Cha et al. [28] a large part of content items is immutable which means that users tend to lose interest in an item immediately after they consumed it. Figure 5.5 confirms that any prefetching strategy shall be proactive and quickly anticipate the interest of each user towards each video.

5.3.6 Load variation across the day

We plot in Figure 5.6(a) the aggregated number of simultaneous YT and FB sessions in the dataset across one representative day. As expected, it follows a classic daily pattern with peaks at 1pm and in the evening. Figure 5.6(b) concentrates during the most loaded hours: 1-2pm and 9-10pm. At a granularity of seconds we observe that traffic is not uniformly distributed. A clever prefetching strategy would be to leverage the local minimum in these most loaded hours to push content on mobile devices. Digging further, one might investigate the best way to allocate the mobile spectrum. One possible solution would be to efficiently reuse white

[2] https://developers.google.com/youtube/hl=fr



(a) Load variation during the day

(b) Load variation during peak hours

Figure 5.6: (a) Number of active sessions across the day; (b) Zoom in during peak hours

spaces [32] and push content during these periods. Yet, we investigate this in future work.

F5: Load is not uniformly distributed, including at peak hours. Therefore, prefetching should be scheduled either at off-peak hours or at much smaller time scales at the least loaded instants during the peak hours, and in coordination with the mobile carrier's resource allocation control.

5.4 System Design

CPSys (Central Predictor System) is designed to leverage the user's interest or social ties to determine a personalized list of content items to be prefetched for each user. Figure 5.7 depicts the CPSys architecture. The system consists of two main components:

- *Prefetcher agent.* Installed on the user device, this component runs as a background service to ensure two main functions. First it provides the centralized predictor with reports on the user' activities. Second it triggers and controls the prefetching of the videos from a list it receives from the centralized predictor.

- *Central predictor (CP).* This component holds and updates profiles of all users running the prefetcher agents by using the reports provided by the latter ones. It also creates and updates social ties between users. Finally it exploits all these sets of information to

predict the candidate videos to prefetch for each user running the prefetcher agent.

The 3 following questions are used to drive the design of our system: *What to prefetch? When to prefetch? How many videos to prefetch?*

5.4.1 What to prefetch?

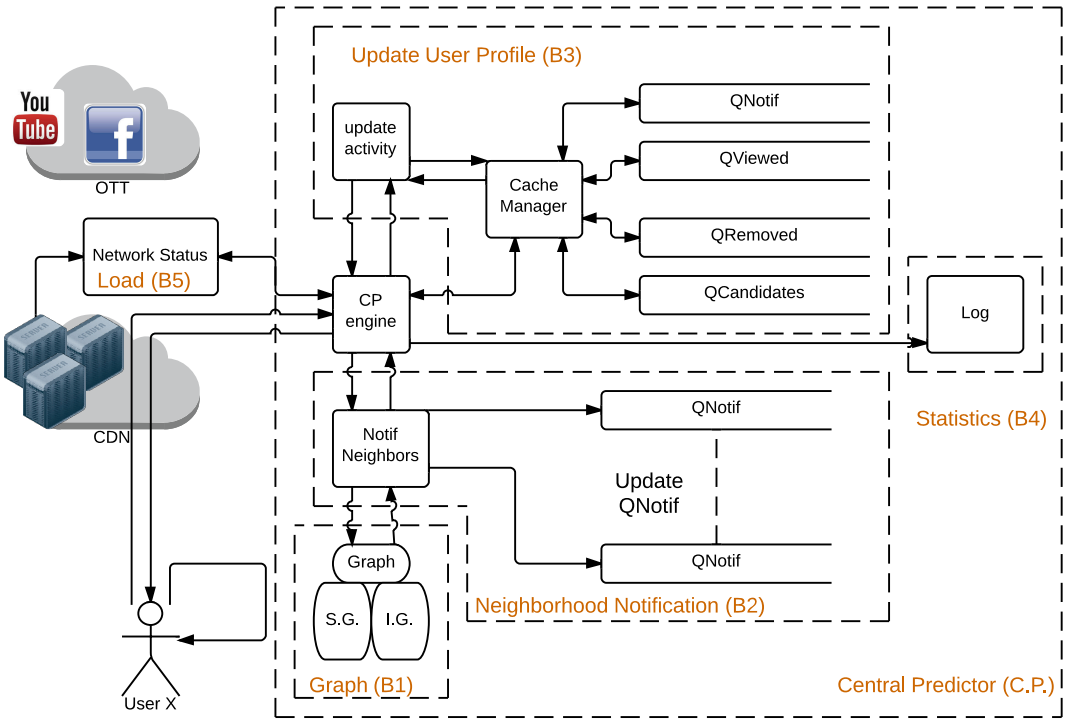


Figure 5.7: CPSys design

Our prefetching strategy is derived from the recommendation techniques, yet tailored to the mobile networks requirements and constraints. First, we identify users with common interests. The rationale is to affiliate for each user a group of users - we call them neighbors - who tend to request similar content items, hence we build a directed graph. Second, on top of this graph and for a given user X, the centralized predictor tracks the videos that has been requested by neighbors of user X and defines a personalized list of prefetching candidates. In the following, we detail how the CP creates the graph and updates the users' profiles:

Building the graph

The graph (B1 on Figure 5.7) is one of the most important components of the CP. It builds and maintains ties between users based on social or interest affinity. It implements the following two interfaces:

Social graph interface (SG) The prefetcher agent, installed on the user device, reports the IDs of all social neighbors running the prefetcher agent to the central predictor. Hence, the social graph is inferred from the OSN such as we do for Facebook. The social neighbors should as well install an instance of the prefetcher agent so that CPSys could create the edge between two users socially connected.

Interest graph interface (IG) The central predictor updates in daily routine the list of neighbors affiliated to each user. It computes similarities between users based on all past and recorded user preferences, hence re-affecting the implicit ties with respect to the new similarity scores. The more we learn about the user preferences the more accurate the prediction model can become. There exist many similarity measures between pair of sets such as cosine, euclidean, Pearson, Jaccard, etc. We use the Jaccard index [91] to compute the Affinity (A) score between all users. The reason of this choice is twofold. First, we suppose that a user is interested in a video only if she watches it. This leads to a binary preference either the video content is releavent or not for a user. Second, in CPSys we avoid creating edges between the heaviest and the lightest users. This reduces the time overhead required to update the profile of light users, since an update is triggered each time a neighbor watches a video. Formally, the affinity score is computed as the following:

$$A(u,y)_d = \frac{|L_u(v)_{t..d-t-1} \cap L_y(v)_{t..d-t-1}|}{|L_u(v)_{t..d-t-1} \cup L_y(v)_{t..d-t-1}|} \tag{5.1}$$

$L_u(v)_{t..t-d}$ is the set of videos viewed by user u over the time window of d days. We associate for each user the K-Nearest Neighbors (KNN), i.e. the ones with the highest affinity scores. The decision for an appropriate K value is a design choice.

Content selection process

Having established the graph, when User X requests a content item v at instant t, the centralized predictor captures this request in real time and proceeds as follows:

- Neighborhood Notification (B2 on Figure 5.7). The CP manages (creates and updates) a queue named *QNotif* which contains the identifiers, i.e. URLs, of the videos watched by the user's neighbors. This queue is updated each time one of these neighbors watches a video. More precisely, each element in the queue, named index, is a data-structure composed of the unique identifier of a video and several attributes of the video, including its local popularity, the source(s) of the request(s) for that video (the neighbor(s) who requested the same content), its freshness (the date of the latest request for that content), and the pointers to the next and previous indexes in QNotif. QNotif may implement different policies to rank the indexes in the queue. We opted for the Most Popular Most Recent (MPMR) policy for the queue QNotif in CPsys, as detailed below.

- Update User Profile (B3 on Figure 5.7): The CP updates the profile of user X. First, it updates the request rate associated to this user and increments the number of requested videos viewed per day. Then the CP inserts the index of the viewed video into the Qviewed queue. In CPsys we do not prefetch the same content multiple times. The rationale behind this design choice is twofold:

  First, according to Cha et al. [28] a large part of content items is immutable which means that users tend to lose interest in an item immediately after they consumed it. This is especially the case for, e.g. catch-up [47] or user-generated content, where users tend to watch the content only one time.

  Second, we consider the local cache of a user being large enough to hold content items for a considerable period of time before removing it. As a result, requests for already prefetched items can be easily served locally, even for multiple requests.

Figure 5.8 presents the queue QNotif implemented with the MPMR policy. QNotif is divided into a set of classes ($C_{j=0..\Gamma}$):

- Each class C_j includes at least the indexes of the videos that have been watched by a part or all of the user's neighbors on day $d = (N - j)$, where N points to the actual day

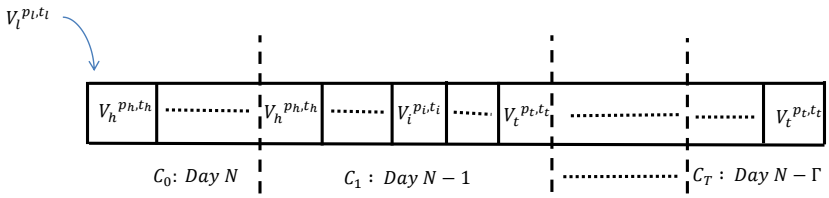


Figure 5.8: QNotif: Data structure which holds the prefetching candidates

(d equals to N in that case).

- Γ is a parameter, expressed in terms of number of days. Γ is used to prevent the queue from growing indefinitely. The setting of this parameter is a design choice. The higher the value of Γ, the more costly the look_up and update operations will be. Given that the number of views in the considered dataset drops significantly 3 days after the upload (c.f. Finding F4 in Section 5.3.5), we set Γ to 3.

In MPMR and within each class C_j, the rank of the index is attributed with respect to the popularity (p) as a first criteria, then with respect to the freshness of the content. The index *heading* the class is referred by (${v_h}^{p_h,t_h}$) which should point to the most viewed content by neighbors, then as far as we iterate through the list of indexes, the popularity should either remain the same or drops until we reach the *tail* of the class (${v_t}^{p_t,t_t}$).

Based on the notations used in Figure 5.8, we illustrate how QNotif is maintained with respect to the MPMR policy, we suppose that one of the neighbors of user X requests a content item V_l. As a result, CP updates the list of indexes $Q_n(v)$ of user X. Algorithm 3 and the following paragraphs detail where exactly in the list to insert the index v_l. v_l is a pointer to content item V_l.

- *Line 1,2*: This is the case where index v_l does not exist in $Q_n(v)$, hence the index is created and inserted into class C_0. A newly created index starts always with $p = 1$, since only one of the neighbors watched the content. When we insert v_l, first, we determine its target location which is index $v_{i_{target}}$ where : ($p_{i_{target}-1} > 1$) and ($p_{i_{target}} = 1$), then we shift each index after this target location $v_{i_{target}}$ by one position to make room for the new insertion.

Require: $Q_n(v), v_l$;
1: **if** $v_l \nexists Q_n(v)$ **then**
2: insert(v_l, N, $p_{i_{target}} > 1$, $p_{i_{target}} == 1$)
3: **else**
4: $d_l \leftarrow get\_day(Q_n(v_l))$
5: $p_l \leftarrow get\_popularity(Q_n(v_l))$
6: $p_l \leftarrow p_l + 1$
7: **if** $d_l == N$ **then**
8: move(v_l, N, $p_{i_{target}} + 1 > p_l$, $p_l \geq p_{i_{target}}$)
9: **else**
10: $d_l \leftarrow d_l + 1$
11: move(v_l, d_l, $p_{i_{target}} > p_l$, $p_l \geq p_{i_{target}}$)
12: **end if**
13: **end if**

Algorithm 3: Update of user X's QNotif upon the request for video V_l with index v_l by one of User X's neighbors

Now, if the index v_l already exists within QNotif, we take both the C and p parameter to decide where to move it.

- *Line 7*: This line corresponds to the case where C equals to N. This index remains within class N and only the number of views (p_l) is incremented. Then, v_l is moved ahead to the position before index $v_{i_{target}}$ with property ($p_{i_{target}-1} > p_l$) and ($p_{i_{target}} <= p_l$).

- *Line 8*: This is the case where $C < N$. In this case, v_l jumps to class $C+1$, increments the (p_l) score, and is moved ahead before index i_{target} with property ($p_{i_{target}-1} > p_l$) and ($p_{i_{target}} <= p_l$).

In the next section, we show that MPMR outperforms other policies by improving the prediction accuracy and decreasing the overload. In particular we compare MPMR to LRU and FIFO policies.

5.4.2 When to prefetch?

To efficiently manage the prefetching process, the system includes a double control mechanism, consisting of a network-oriented and a state-transition control mechanism. The prefetching -triggered and performed by the user device- is hindered until both control mechanisms meet.

Network control mechanism

The prefetching shall be hindered when the network is overloaded. And, ideally, it should be achieved in coordination with the mobile carrier's resource allocation control (See Finding F5

in Section 5.3.6).

Hence, the Network status component (B5 on Figure 5.7) is used to monitor and report the traffic load to the centralized predictor. If the load exceeds a certain threshold, prefetching is not allowed to be performed.

State-transition control mechanism

The prefetcher agent is also controlled by a state-transition control mechanism. Figure 5.9 shows the transition states that the prefetcher agent should follow-up.

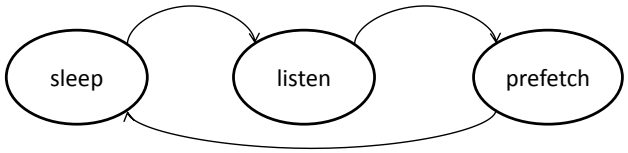


Figure 5.9: Transition-state control mechanism running on the prefetcher agent

Figure 5.9 depicts the 3 states the prefetcher agent goes through. By default, the prefetcher agent is in the "listen" state and tracks all user requests. If the network control mechanism allows it (i.e. if the network conditions are met) and if the prefetcher agent detects a session established between the user device and one of the content applications for which the prefetching system is deployed, the agent switches to the "prefetch" state and the video prefetching process starts effectively. When this process ends, the agent turns to the "sleep" state for a certain period of time Δ, predefined at the central predictor. Then it turns back to the "listen" state.

This double-check control mechanism aims at maintaining a solid control on the prefetcher agent, so as to prevent too aggressive video prefetching plans especially for light users. This is in line with the Findings F2 and F3 from Sections 5.3.3 and 5.3.4.

5.4.3 How many videos to prefetch?

Finding F3 in Section 5.3.4. shows that a small set of users behaves as heavy users. However, the users may change their behavior across the time. The number of prefetched videos should also follow the evolution of each user's behavior accordingly. In CPsys, the Update User Profile block (B3 on Figure 5.7) models the user behavior as a function of time, based on her past activity, and it determines the final list of videos to prefetch with these three parameters:

$[\tilde{S}_{max}]_d$, $N_{prefetch}$ and p_{th}.

- $[\tilde{S}_{max}]_d$ is a prediction of the number of content items the user would request at day d. It is updated on a daily basis and equals to the average number of requested videos per day over the 10 past days.

$$[\tilde{S}_{max}]_d = \frac{\sum_{i=d-11}^{d-1} (number\ of\ requests\ per\ day)_i}{10} \tag{5.2}$$

- $N_{prefetch}$ is the maximum theoretical number of videos from the queue QNotif that the prefetcher agent is allowed to prefetch when the prefetching process is executed.

$$N_{prefetch} = \tilde{S}_{max} * r + 1 \tag{5.3}$$

- The threshold popularity score, p_{th}, is used to achieve a final filter among the $N_{prefetch}$ best ranked candidate videos in QNotif. The rationale is to avoid prefetching the very unpopular content videos. Hence only the subset of the videos satisfying the property ($p_i >= p_{th}$) are moved to the QCandidates queue in the Update User profile block and communicated as a list to the prefetcher agent on the user device to be prefetched sequentially. In the next section, we investigate the impact of fine-tuning this parameter p_{th} on the performances of CPsys.

The queue QRemoved on Figure 5.7 is just used for debugging purposes. If the user requests a content which had already been prefetched previously but removed before the user watches it, due to storage capacity limitation for example, QRemoved is updated with the index of the removed video. Later, this information is used to feed the Statistics component (B4 on Figure 5.7).

5.5 Trace-driven simulation experiments

We developed Prefsim [3], written in Java, a simulator implementing the CPSys architecture as presented in Figure 5.7 and described in the previous section. Prefsim runs either in social or interest mode. It is a trace-driven simulator. In both modes, Prefsim requires a traffic trace as an input file which contains the timestamp, the userID, the videoID, and the duration for individual user sessions. Additionally, if it runs in social mode, the simulator requires the social graph as input file.

The task performed by the prefetching system might be considered as an instance of a recommendation problem: the system should be able to predict the user's interest and to timely prefetch the content of interest. The approaches applied in studies in the area of recommendations usually rely on the collection and exploitation of preferences expressed by users - mostly ratings - to build recommendation algorithms and engines, as well as to assess their performances - mostly recall and precision - [36]. In recommendation systems, it is commonly known that a very small set of users express their opinions on items through rating or liking, while the majority consume items without expressing their opinion. Instead, our dataset captures real users' views and not their expressions, hence we are able to further characterize the users' interests.

The finding F1 in Section 5.3.2 suggests that the heavy long tail of the video popularity distribution leads to a high sparsity levels of the user-video matrix, which includes all users and videos from the trace and describes how the two are linked by observed sessions. Even if the prediction runs perfectly, it is expected that the recall -later we call it Hit-Ratio (HR)- of approaches based on this matrix will be too low to show a considerable benefit in the context of prefetching.

We evaluate CPsys based on 3 metrics which are network oriented. We evaluate the following:

- Correct Prediction Ratio (CPR): Ratio of requests served from the user's local cache out of the number of prefetched videos.

- Overhead: Ratio of videos being prefetched and not requested by the user divided by the number of requests that were not served from the user's local cache. Formally, it is

[3] http://www.ict-ecousin.eu/public-deliverables-dissemination/public-deliverables/ecousin-deliverable-d3.2-v1.0-public.pdf/view

equal to:

$$Overhead = \frac{(1 - CPR) * N_{\text{prefetched videos}}}{(1 - HR) * N_{\text{requested videos}}} \tag{5.4}$$

- False Negative Ratio (FNR): Ratio of requests that the prediction policy failed to detect, although clients have already been notified about these contents: The QNotif holds an index pointing to the requested content, but the content was not prefetched since it was not considered to be relevant for the user. The reason for this failure might be that the content was not considered popular enough ($p_v < p_{th}$), and/or the freshness of the content was not good enough to position it among the N_{prefetch} best ranked content items in QNotif, i.e. among the ones selected as prefetching candidates. Thus the sum of FNR and CPR gives insights into the optimal CPR we may achieve.

The goal of the Prefsim implementation is to get insights into the prediction accuracy that one can achieve.

5.5.1 Simulation setup

In Prefsim, the simulation setup is defined in an XML-based configuration file (input.xml). We evaluated our system using the FB dataset that was introduced in Section 5.1. In order to ensure fast simulation processing, we only considered the users who requested at least 100 videos over the whole data collection period (in average 1 video per day). Table 5.2 summarizes the traffic trace used for the simulation. The sparsity level ($1 - \frac{\text{number of requests}}{\text{number of user} * \text{number of videos}}$) is extremely high and, thus, sticks to the real world traffic properties. To the best of our knowledge, no studies have used such a sparse user-video matrix in context of video prefetching; hence this makes the prefetching task tougher, yet more realistic and makes this work unique in this respect.

The interest graph is updated in a daily routine. Each user has at most 20 similar neighbors. The cache size used for users is limited to 50 videos and LRU is used as a cache replacement policy. We maintain a history size of the 10 past requests to model the user activity. r is equal to 0.334. For a given user, we do not prefetch videos unless she requests at least 20 videos. This is to get a first insight on users' preferences and to better assign neighbors for each user

in the system. The sleep state period (Δ) is set to 1 hour. Regarding the configuration of the queues QNotif and QCandidates, Γ is set to 3 days, and only the videos that belong either to the classes C_0 or C_1 of QNotif are candidates for prefetching.

Number of FB sessions	Number of unique users	Number of unique videos	Sparsity level
4,152,885	23,548	962,406	0.9998

Table 5.2: Traffic trace used for the simulation

5.5.2 Performance analysis

In the first experiment, we evaluate the average CPR and the overhead of several policies adopted by QNotif: FIFO, LRU, and MPMPR-p_{th}, i.e. the MPMR policy with different values for the threshold popularity score p_{th} ($p_{th} \in [1..5]$). In the same experiment, we also compare the case where $N_{prefetch}$ is either dynamic and equal to S_{max} as described in Section 5.4.3 or static and fixed to 6. Figure 5.10 shows that regardless of the values of p_{th}, MPMR outperforms the LRU and FIFO policies. We observe a significant increase in the CPR when $p_{th} > 1$, reaching up to 18%. The CPR improves when the value of p_{th} increases, as this makes the prediction policy more conservative: the prefetching process is triggered only when the content was viewed at least p_{th} times by the neighbors.

Figure 5.11 shows that the overhead decreases significantly when adopting MPMR, especially when p_{th} is high. We observe that if prefetching is not well tuned, then the traffic increases significantly and may even double, which obviously cannot be accepted by Telcos. However, a fine tuning of parameters significantly decreases the Overhead. We show that it drops below 5% when ($p_{th} > 2$). Besides, adapting the number of prefetched videos with respect to the past user activity ($N_{prefetch} = S_{max}$) also reduces the overhead, hence leads to a better network experience.

The lesson we learn from this result is in line with Finding F1: it is safer to avoid prefetching very unpopular content and wise to control the overhead caused by content prefetching on the network load.

Figure 5.12 shows that the FNR increases as far as the prediction policy becomes more conservative, which in return increases the CPR and decreases the overhead. Relying on users' neighbors, FNR reaches 12% and CPR reaches 18% when p_{th} is equal to 5, this suggests that

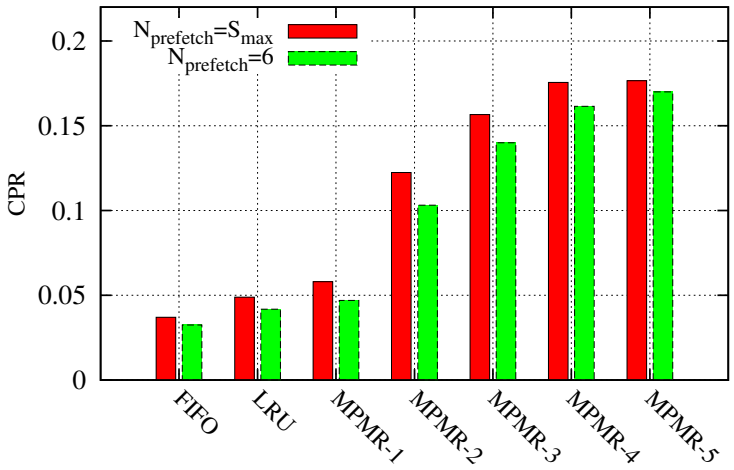


Figure 5.10: CPR with different content seleciton policies

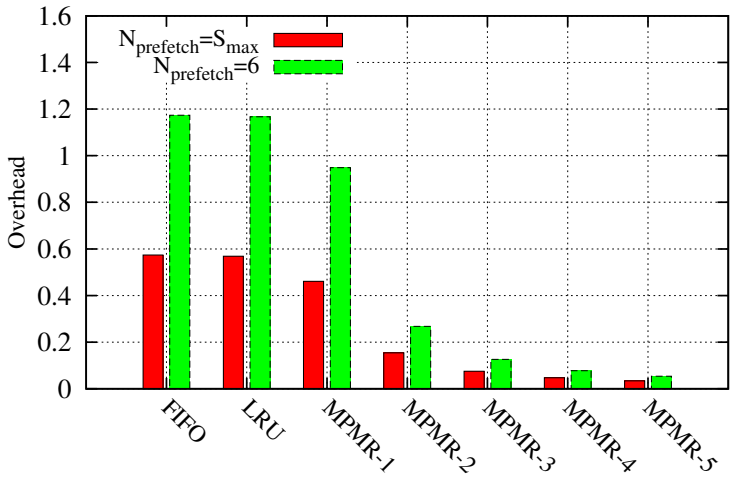


Figure 5.11: Overhead with different content selection policies

while setting p_{th} to 5, an optimal content selection policy would rise the CPR to 30%. However, decreasing the FNR will systematically increase the overhead. This trade-off should be carefully handled in operational networks.

The lesson we learn from the past simulation results is that regardless of the policy used to select the prefetching candidates, prediction is still hard in general. While the MPMR suggests pushing the most popular and fresh content items that have been seen by the most similar neighbors, we observe that the large majority of viewed videos by individual users are considered as personalized content items. Prefetching these personalized content items

is risky and leads to an acute trade-off between the overhead and prediction accuracy. In CPsys, the more we learn about users' preferences, the more accurate the prediction model is. Unfortunately, the dataset we used does not cover all users' preferences since it was collected from mobile networks. We could have successfully identified and pushed a content the user is interested in. However, this user was connected to a fixed network through a WiFi connection the time he requested that content. In this case, we do not capture this request in our mobile traffic traces. This suggests that the performance assessment we carry out in this section represents the lower bound of the real performance we may achieve.

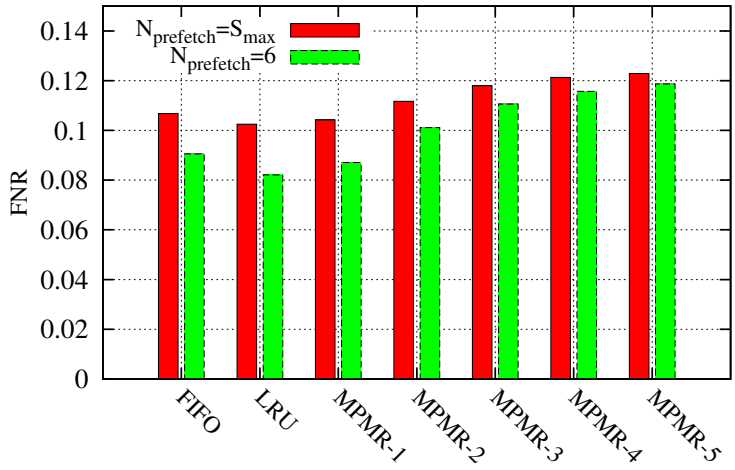


Figure 5.12: FNR with different content selection policies

In the second experiment, we limit the simulation to 4 days and compare our prefetching system with the authors' proposition in [41] which consists of pushing bundles of popular content on mobile devices. Mapping this to Prefsim, if one user watches a video, then every user should update her list of prefetching candidates with the most recently watched videos. Figures 5.13 and 5.14 show that limiting the neighborhood to the 20 most similar users and setting p_{th} to 3 improves the CPR up to 3 times, while it decreases the overhead by 5 times. This means that pushing only popular content to everyone is not necessarily the best option with regard to the prediction efficiency. Yet, it is better to personalize the list of prefetching candidates according to the preferences of the user's most similar neighbors.

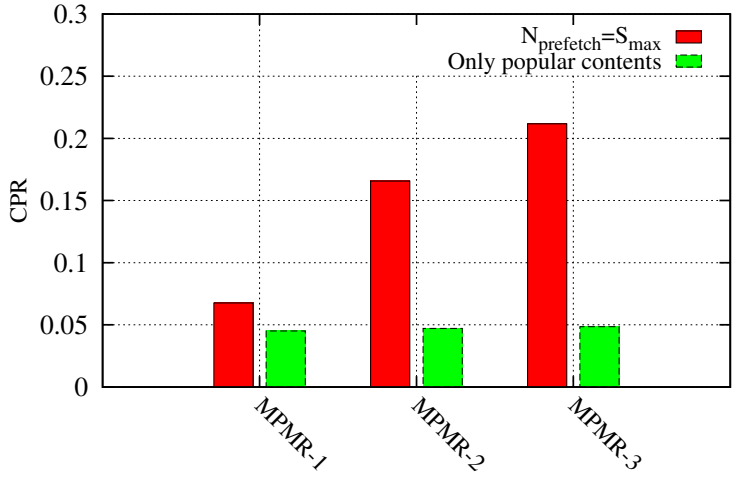


Figure 5.13: CPR

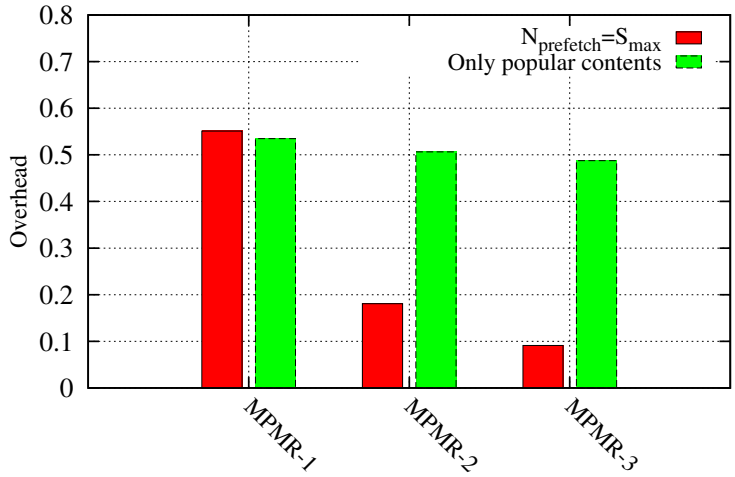


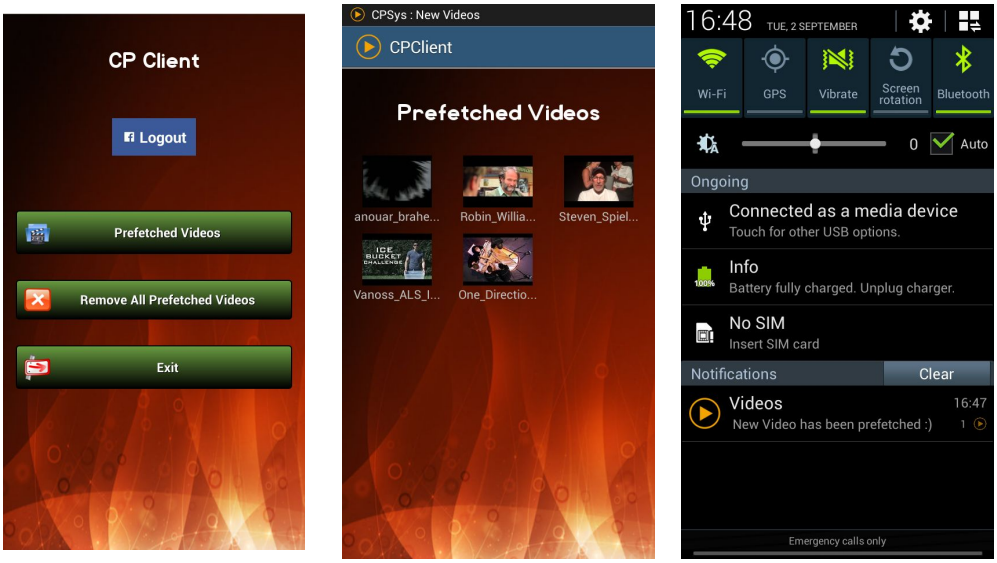
Figure 5.14: Overhead

5.6 Prototype implementation

We have implemented a prototype of CPsys using a client-server model. The central predictor runs as a third-party server holding profiles of all users running the prefetcher agent.

At the server side, we used an Apache Tomcat server [4] as well as the Jersey framework [5] to implement the restful web services. We used the Mahout framework [6] to build and update the interest graph and compute the Jaccard affinity scores. As a content selection policy, we

[4] http://tomcat.apache.org/
[5] https://jersey.java.net/
[6] https://mahout.apache.org/



(a) CPClient interface (b) Prefetched videos (c) Notification

Figure 5.15: Snapshots from CPClient

implemented MPMR.

At the client side, users should install the CPclient which is an Android-based application. The CPclient endorses the prefetching agent and a frontend interface (c.f. Figure 5.15(a)). The CPClient is linked to the user's Facebook account, i.e. the user is asked to log in to his Facebook account to initiate running the CPClient. The reason for this is that, in CPSys, we use the Facebook_ID as a unique user identifier. The same ID is used at the server to update all data structures, including databases, QNotifs, QViewed, etc.

We use an intent-filter mechanism [7] to follow users' activities and report the list of videos watched to the CPserver. When prefetching is executed, CPClient pulls the video IDs from QNotif. Subsequently, the agent asynchronously prefetches these videos. When prefetching is complete, thumbnails of the videos are displayed (cf. Figure 5.15(b)). In this example, QNotif holds the IDs of YouTube videos which are considered for prefetching.

At last, in our prototype implementation, we enriched the prefetching-app with a notification mechanism (c.f. Figure 5.15(c)). When the prefetching process is over, a user – running CPclient receives a notification to motivate him or her to watch the recently prefetched videos.

[7] http://developer.android.com/guide/components/intents-filters.html

We believe that this incentive strategy improves both CPR and HR, since clients are more likely to consult and watch the prefetched videos.
A video demonstration of this work is available at
http://tinyurl.com/pq2v28s.

5.7 Conclusion

In this chapter, we designed, evaluated and implemented CPSys, a prefetching system we have designed based on traffic patterns and clients' behavior that we observed in a real operational mobile network. We addressed a series of key design issues. Subsequently, CPSys relies on recommendation techniques to build the implicit or social graph, then we use the MPMR policy to select the video prefetching candidates. At the end, we evaluated CPSys through trace-driven simulations. We show that the highest lower-bound performance of CPSys regarding CPR ranges from 18% to 22% while we show that the traffic overhead decreases significantly. We observed that prefetching performance is strictly related to content characteristics. When content items become further personalized, prediction becomes harder and potentially reduces the prediction accuracy.
There are many possible venues towards enhancing CPsys. One primary direction would be to focus on the personalized content items, hence supplying CPsys with additional information to further personalize the list of prefetching candidates. In parallel, we plan to enhance our system implementation and make it faster and more scalable. The goal is to study how CPSys performs at large scale and study how notifications can improve the system performance.

6 Conclusion & Perspective

This thesis has explored the design of streaming media content distribution techniques to reduce the delivery costs and increase the user quality of experience. In the context of caching, costs include the server load and the total volume of data crossing the transit links, while in context of prefetching, costs consist of the traffic overhead resulting from the false negative prefetched videos. First, we analyzed and provided an in-depth understanding of user behavior and video traffic properties. More precisely, we have deeply investigated and modeled the user video access patterns. Then, we analyzed the user-engagement and quantified the video abandonment rate as a function of the bandwidth. We have further studied and modeled the switching between qualities that the client may incur while watching a HAS content.

The findings on users' behavior have served as guidelines for the design of new caching strategies for conventional CDN and transparent caches. Additionally, we leveraged these observations to motivate our work on prefetching.

Section 6.1 outlines our contributions in this thesis. In section 6.2, we provide possible directions for future work.

6.1 Achievements

We now provide a brief summary about our contributions.

6.1.1 Traffic analysis

We have mainly investigated the properties of 2 classes of traffic. The first class consists of HAS traffic, and consists mainly on catch-up and live streaming sessions. The second class consists of UGC traffic, mainly YouTube and Facebook user generated videos. In the following

we outline the major findings we derived from the data analysis.

HAS traffic characterization

- The number of requested chunks in live and on-demand HAS sessions could be modeled by a piece-wise distribution. The head of the distribution follows the log-normal distribution while the tail follows the generalized Pareto distribution.

- Delays perceived at the joining-phase has a straight impact on the user engagement. If clients experience video interruptions or long loading delays during the 3 first requested chunks, the user-engagement decreases by a factor of 7.

- Most of the requested profiles are encoded between 600 and 1000 kbps. The lowest encoding profiles are mostly requested at the beginning of the session. Then users rapidly switch to the highest profiles.

- We used Markov chain to model the switching between profiles. We estimated the probability to move from $P_{i\in[0..7]}$ to profile $P_{j\in[0..7];j\neq i}$. We observed that the probability to move from $P_{i=5}$ to $P_{j>5}$ is 22%, then when clients are in $P_{i=6}$, they are more likely to switch to $P_{j=5}$ with a probability equal to 60%. When being in P_7, clients move to P_6 and P_5 with a probability of 92%. This suggests that when being in P_5, a very limited number of transitions happen among the highest profiles. When being in P_4, around 45% of transitions are made to the upper profiles, and that 48% are made from $P_{i=5}$ to $P_{j=4}$. This intermittently switching between the highest profiles does not necessarily yield improvement of quality of experience. But instead, it adversely affects some major network building blocks such as caches. On average the number of transitions during a HAS session falls within [1/6, 1/2] of the total requested chunks per session. In case where a cache is deployed at the G_i interface of the mobile career, our simulations show that this high switching behavior and video bitrate selection heterogeneity results in a significant decrease of the cache hit-ratio.

UGC traffic: YouTube and Facebook traffic characterization

- The YT and FB video popularity distribution follows the Zipf distribution. We observe that around 75% of YT videos and 65% of FB videos are requested only once.

- Users do not request more than 3 videos per day when their request inter-arrival rate is higher than 1800 seconds and few users request far more frequently videos, while the majority is less active. Over a period of one month measurement, we observed that around 72% and 83% of active users requested at most 10 YT and FB videos respectively.

- The lifetime distribution of the user generated videos could span over a long period. However, most of the views happen over a short time frame. we observe that 10% of the views happen only one hour after the content becomes available online, while near the half of the views are done the first day.

6.1.2 Caching HAS videos

In this thesis, we proposed WA-LRU as a new caching strategy that leverages the time locality of the segments within the HAS content. The principle of WA-LRU is driven by 2 main findings about the client browsing behavior. First, we observed a heavy long tail distribution which suggests that the probability to abandon the viewing becomes high as far as users remains watching the video. Second, long start up delays increases significantly the abandonment rate. Therefore, in WA-LRU we give priority to the first chunks to be cached. The requested chunk is compared with the threshold index which is computed with respect to the cache capacity and traffic load on the cache. Simulations show that WA-LRU decreases significantly the processing time at the cache while increases the average cache hit-ratio.

We have also proposed CF-DASH a cache friendly Dash player to tackle the player instability issue. In CF-Dash, we reduce the number of switching between qualities while we sustain the user quality of experience. We evaluated CF-DASH through simulations and test-bed experiments. Both evaluation methods show that we achieve more than 15% of gain in hit-ratio.

6.1.3 Prefetching videos on mobile devices

The last part of this thesis was dedicated for the design, evaluation and implementation of CPSys a system for mobile video prefetching. In CPSys, we build implicit and explicit ties between users sharing similar interests. Then, we proposed MPMR policy to sort content items that have been viewed by the most similar neighbors with respect to the popularity and freshness scores. Prefetching is executed when two control mechanisms meet: a state control

mechanism implemented at the user device and a network control mechanism defined by the mobile carrier. At the end, the number of items to be prefetched depends on the engagement of the user. We classify clients into heavy and light users and prefetch items accordingly. We developed prefsim a java simulator implementing the CPSys architecture. Simulations show that CPsys achieves a satisfactory performance. Based on our simulation settings, the lower bound prediction accuracy could reach 22% while we decrease significantly the traffic overhead down to 5%. We also provided a proof of concept implementation of CPSys by implementing an android application that runs as a prefetcher agent and the central predictor as a web server.

6.2 Future work

We conclude this dissertation by highlighting some open problems and items for future work.

6.2.1 Future works on data analysis

- In the HAS dataset, we only analyzed live and catch-up traffic. It would be interesting to study other types of videos such as UGC videos and infer additional guidelines regarding caching as well prefetching mechanisms.

- We provided a coarse-grained analysis on the user-engagement since we collected the data from a vantage point in the mobile network. However, it would be interesting to get closer to the client and assess the user-engagement from a user perspective. For instance by instrumenting the video player with an agent that reports fine grained information related to each buffering or stalling event that may happen during the video viewing.

- Using our datasets, we were not able to derive inference on users profiles in Facebook or YouTube. Mining the different attributes that are accessible through the user profile and combining them with the user activity at the network would help to further characterize and better predict the user behavior.

6.2.2 Future works on caching

- In CF-DASH, it would be interesting to further investigate the ideal profile to be cached. In our work, we were limited to the MOS criteria to decide which profile should be cached and served to the clients. However, several other criteria could be involved such

as the category of the content, the playback buffer, etc.

- Caching performance depends crucially on the type of content to be cached. In this thesis we evaluated WA-LRU using a catch-up traffic trace to drive our simulations. We demonstrated that WA-LRU performs better than the widely adopted LRU. That is because we observe a heavy tailed distribution of the requested number of chunks per session. The heavier the tail the better WA-LRU performs, otherwise if the number of views per segment is uniformly distributed, then WA-LRU will perform the same like LRU.

6.2.3 Future works on prefetching

- We used the collaborative filtering techniques to infer the interest graph. One limitation of this method is the cold start problem. In our simulations, we assign the interest ties for a user after requesting at least 20 videos. One promising way to solve the cold-start problem is to further investigate the social ties and infer which content is more likely to be consumed by the user based on his social neighborhood. Then as far as CPSys learns about the user preferences, we may therefore consider the interest graph as an alternative to the social graph.

- In context of prefetching scalability and sparsity should be addressed when the number of clients and items in the system grows exponentially. Therefore, new methods have to be developed to reduce the computational costs and to tackle this potential issue. This should drive us to investigate new research directions such as cloud computing or distributed systems to make the system highly deployable on large scales.

Bibliography

[1] http://www.progressivepolicy.org/wp-content/uploads/2013/09/2013.09-Carew-Mandel_US-Investment-Heroes-of-2013.pdf.

[2] http://blogs.msdn.com/b/interoperability/archive/2014/01/03/mpeg-dash-tutorial-embedding-an-adaptive-streaming-video-within-your-html5-application.aspx.

[3] https://iperf.fr/.

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

[5] http://www.cnlive.it/.

[6] http://www.akamai.fr/.

[7] https://developer.apple.com/streaming/.

[8] http://www.akamai.fr/enfr/html/resources/http-live-streaming.html.

[9] http://gpac.wp.mines-telecom.fr/.

[10] http://www-itec.uni-klu.ac.at/dash/?page_id=207.

[11] http://cmm.khu.ac.kr/korean/download.php?id=808&sid=6a267a6d7b2f771ab06f776dc67a21ba.

[12] http://www.whatisedgerank.com/.

[13] http://home.ifi.uio.no/paalh/dataset/hsdpa-tcp-logs/.

[14] S. Akhshabi, L. Anantakrishnan, A. C. Begen, and C. Dovrolis. What happens when http adaptive streaming players compete for bandwidth? In *Proceedings of the 22nd international workshop on Network and Operating System Support for Digital Audio and Video*, pages 9–14. ACM, 2012.

[15] S. Akhshabi, L. Anantakrishnan, C. Dovrolis, and A. C. Begen. Server-based traffic shaping for stabilizing oscillating adaptive streaming players. In *Proceeding of the 23rd ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*, pages 19–24. ACM, 2013.

[16] S. Akhshabi, A. C. Begen, and C. Dovrolis. An experimental evaluation of rate-adaptation algorithms in adaptive streaming over http. In *Proceedings of the second annual ACM conference on Multimedia systems*, pages 157–168. ACM, 2011.

[17] M. Arlitt and T. Jin. A workload characterization study of the 1998 world cup web site. *Network, IEEE*, 14(3):30–37, 2000.

[18] M. F. Arlitt and C. L. Williamson. Internet web servers: Workload characterization and performance implications. *IEEE/ACM Transactions on Networking (ToN)*, 5(5):631–645, 1997.

[19] S. Baluja, R. Seth, D. Sivakumar, Y. Jing, J. Yagnik, S. Kumar, D. Ravichandran, and M. Aly. Video suggestion and discovery for youtube: taking random walks through the view graph. In *Proceedings of the 17th international conference on World Wide Web*, pages 895–904. ACM, 2008.

[20] R. Bell, Y. Koren, and C. Volinsky. Modeling relationships at multiple scales to improve accuracy of large recommender systems. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 95–104. ACM, 2007.

[21] W. Bellante, R. Vilardi, and D. Rossi. On netflix catalog dynamics and caching performance. In *Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), 2013 IEEE 18th International Workshop on*, pages 89–93. IEEE, 2013.

[22] A. Boutet, D. Frey, R. Guerraoui, A. Jégou, and A.-M. Kermarrec. Whatsup: A decentralized instant news recommender. In *Parallel & Distributed Processing (IPDPS), 2013 IEEE 27th International Symposium on*, pages 741–752. IEEE, 2013.

[23] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. Web caching and zipf-like distributions: Evidence and implications. In *INFOCOM'99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 1, pages 126–134. IEEE, 1999.

[24] A. Brodersen, S. Scellato, and M. Wattenhofer. Youtube around the world: geographic popularity of videos. In *Proceedings of the 21st international conference on World Wide Web*, pages 241–250. ACM, 2012.

[25] T. Bucher. Want to be on the top? algorithmic power and the threat of invisibility on facebook. *New Media & Society*, 14(7):1164–1180, 2012.

[26] P. Calyam, D. Krymskiy, M. Sridharan, and P. Schopis. Active and passive measurements on campus, regional and national network backbone paths. In *Computer Communications and Networks, 2005. ICCCN 2005. Proceedings. 14th International Conference on*, pages 537–542. IEEE, 2005.

[27] L. Carlinet, T. Huynh, B. Kauffmann, F. Mathieu, L. Noirie, and S. Tixeuil. Four months in daily motion: Dissecting user video requests. In *Wireless Communications and Mobile Computing Conference (IWCMC), 2012 8th International*, pages 613–618. IEEE, 2012.

[28] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, and S. Moon. I tube, you tube, everybody tubes: analyzing the world's largest user generated content video system. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 1–14. ACM, 2007.

[29] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, and S. Moon. Analyzing the video popularity characteristics of large-scale user generated content systems. *IEEE/ACM Transactions on Networking (TON)*, 17(5):1357–1370, 2009.

[30] X. Cheng and J. Liu. Nettube: Exploring social networks for peer-to-peer short video sharing. In *INFOCOM 2009, IEEE*, pages 1152–1160. IEEE, 2009.

[31] X. Cheng and J. Liu. Exploring interest correlation for peer-to-peer socialized video sharing. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP)*, 8(1):5, 2012.

[32] R. I. Chiang, G. B. Rowe, and K. W. Sowerby. A quantitative analysis of spectral occupancy measurements for cognitive radio. In *Vehicular Technology Conference, 2007. VTC2007-Spring. IEEE 65th*, pages 3016–3020. IEEE, 2007.

[33] K. Cho, M. Lee, K. Park, T. T. Kwon, Y. Choi, and S. Pack. Wave: Popularity-based and collaborative in-network caching for content-oriented networks. In *Computer Communications Workshops (INFOCOM WKSHPS), 2012 IEEE Conference on*, pages 316–321. IEEE, 2012.

[34] G. Christodoulou, C. Georgiou, and G. Pallis. The role of twitter in youtube videos diffusion. In *Web Information Systems Engineering-WISE 2012*, pages 426–439. Springer, 2012.

[35] N. Cranley, P. Perry, and L. Murphy. User perception of adapting video quality. *International Journal of Human-Computer Studies*, 64(8):637–647, 2006.

[36] J. Davis and M. Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240. ACM, 2006.

[37] F. Dobrian, V. Sekar, A. Awan, I. Stoica, D. Joseph, A. Ganjam, J. Zhan, and H. Zhang. Understanding the impact of video quality on user engagement. *ACM SIGCOMM Computer Communication Review*, 41(4):362–373, 2011.

[38] A. B. Downey. Using pathchar to estimate internet link characteristics. In *ACM SIGCOMM Computer Communication Review*, volume 29, pages 241–250. ACM, 1999.

[39] J. Erman, A. Gerber, K. Ramadrishnan, S. Sen, and O. Spatscheck. Over the top video: the gorilla in cellular networks. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, pages 127–136. ACM, 2011.

[40] F. Figueiredo, F. Benevenuto, and J. M. Almeida. The tube over time: characterizing popularity growth of youtube videos. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 745–754. ACM, 2011.

[41] A. Finamore, M. Mellia, Z. Gilani, K. Papagiannaki, V. Erramilli, and Y. Grunenberger. Is there a case for mobile phone content pre-staging? In *Proceedings of the ninth ACM conference on Emerging networking experiments and technologies*, pages 321–326. ACM, 2013.

[42] W. Galuba, K. Aberer, D. Chakraborty, Z. Despotovic, and W. Kellerer. Outtweeting the twitterers-predicting information cascades in microblogs. In *Proceedings of the 3rd conference on Online social networks*, pages 3–3. USENIX Association, 2010.

[43] N. Gautam, H. Petander, and J. Noel. A comparison of the cost and energy efficiency of prefetching and streaming of mobile video. In *Proceedings of the 5th Workshop on Mobile Video*, pages 7–12. ACM, 2013.

[44] M. Gibas, G. Canahuate, and H. Ferhatosmanoglu. Online index recommendations for high-dimensional databases using query workloads. *Knowledge and Data Engineering, IEEE Transactions on*, 20(2):246–260, 2008.

[45] P. Gill, M. Arlitt, Z. Li, and A. Mahanti. Youtube traffic characterization: a view from the edge. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 15–28. ACM, 2007.

[46] A. Gouta, C. Hong, D. Hong, A.-M. Kermarrec, and Y. Lelouedec. Large scale analysis of http adaptive streaming in mobile networks. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2013 IEEE 14th International Symposium and Workshops on a*, pages 1–10. IEEE, 2013.

[47] A. Gouta, D. Hong, A.-M. Kermarrec, and Y. Lelouedec. Http adaptive streaming in mobile networks: characteristics and caching opportunities. In *Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2013 IEEE 21st International Symposium on*, pages 90–100. IEEE, 2013.

[48] E. Halepovic, J. Pang, and O. Spatscheck. Can you get me now?: estimating the time-to-first-byte of http transactions with passive measurements. In *Proceedings of the 2012 ACM conference on Internet measurement conference*, pages 115–122. ACM, 2012.

[49] D. Hong, D. De Vleeschauwer, F. Baccelli, et al. A chunk-based caching algorithm for streaming video. In *NET-COOP 2010-4th Workshop on Network Control and Optimization*, 2010.

[50] C. Huang, J. Li, and K. W. Ross. Can internet video-on-demand be profitable? *ACM SIGCOMM Computer Communication Review*, 37(4):133–144, 2007.

[51] C. V. N. Index. Global mobile data traffic forecast update, 2012–2017 http://www. cisco. com/en. *US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-520862. html*, 2013.

[52] M. Jain and C. Dovrolis. Pathload: A measurement tool for end-to-end available bandwidth. In *In Proceedings of Passive and Active Measurements (PAM) Workshop*. Citeseer, 2002.

[53] J. Jiang, V. Sekar, and H. Zhang. Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive. In *Proceedings of the 8th international conference on Emerging networking experiments and technologies*, pages 97–108. ACM, 2012.

[54] S. Jin and A. Bestavros. Greedydual< sup>∗</sup> web caching algorithm: exploiting the two sources of temporal locality in web request streams. *Computer Communications*, 24(2):174–183, 2001.

[55] T. F. Joyce. First in first out activity queue for a cache store, Mar. 25 1980. US Patent 4,195,340.

[56] D. K. Krishnappa, S. Khemmarat, L. Gao, and M. Zink. On the feasibility of prefetching and caching for online tv services: a measurement study on hulu. In *Passive and Active Measurement*, pages 72–80. Springer, 2011.

[57] A. Kvalbein, D. Baltrūnas, K. Evensen, J. Xiang, A. Elmokashfi, and S. Ferlin-Oliveira. The nornet edge platform for mobile broadband measurements. *Computer Networks*, 61:88–101, 2014.

[58] J. Le Feuvre, C. Concolato, J.-C. Dufourd, R. Bouqueau, and J.-C. Moissinac. Experimenting with multimedia advances using gpac. In *Proceedings of the 19th ACM international conference on Multimedia*, pages 715–718. ACM, 2011.

[59] J. Le Feuvre, C. Concolato, and J.-C. Moissinac. Gpac: open source multimedia framework. In *Proceedings of the 15th international conference on Multimedia*, pages 1009–1012. ACM, 2007.

[60] S. Lederer, C. Müller, and C. Timmerer. Dynamic adaptive streaming over http dataset. In *Proceedings of the 3rd Multimedia Systems Conference*, pages 89–94. ACM, 2012.

[61] D. Lee, J. Choi, J.-H. Kim, S. H. Noh, S. L. Min, Y. Cho, and C. S. Kim. On the existence of a spectrum of policies that subsumes the least recently used (lru) and least frequently used (lfu) policies. In *ACM SIGMETRICS Performance Evaluation Review*, volume 27, pages 134–143. ACM, 1999.

[62] C. W. Leong, W. Zhuang, Y. Cheng, and L. Wang. Call admission control for integrated on/off voice and best-effort data services in mobile cellular communications. *Communications, IEEE Transactions on*, 52(5):778–790, 2004.

[63] Y. Li, Y. Zhang, and R. Yuan. Measurement and analysis of a large scale commercial mobile internet tv system. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, pages 209–224. ACM, 2011.

[64] Y. Li, Y. Zhang, and R. Yuan. Characterizing user access behaviors in mobile tv system. In *Communications (ICC), 2012 IEEE International Conference on*, pages 2093–2097. IEEE, 2012.

[65] Z. Li, J. Lin, M.-I. Akodjenou, G. Xie, M. A. Kaafar, Y. Jin, and G. Peng. Watching videos from everywhere: a study of the pptv mobile vod system. In *Proceedings of the 2012 ACM conference on Internet measurement conference*, pages 185–198. ACM, 2012.

[66] Z. Li, H. Shen, H. Wang, G. Liu, and J. Li. Socialtube: P2p-assisted video sharing in online social networks. In *INFOCOM, 2012 Proceedings IEEE*, pages 2886–2890. IEEE, 2012.

[67] Z. Li, X. Zhu, J. Gahm, R. Pan, H. Hu, A. C. Begen, and D. Oran. Probe and adapt: Rate adaptation for http video streaming at scale. *Selected Areas in Communications, IEEE Journal on*, 32(4):719–733, 2014.

[68] K. J. Ma and R. Bartos. Http live streaming bandwidth management using intelligent segment selection. In *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*, pages 1–5. IEEE, 2011.

[69] A. Mansy, M. Ammar, J. Chandrashekar, and A. Sheth. Characterizing client behavior of commercial mobile video streaming services. In *Proceedings of Workshop on Mobile Video Delivery*, page 8. ACM, 2014.

[70] T. Mei, B. Yang, X.-S. Hua, L. Yang, S.-Q. Yang, and S. Li. Videoreach: an online video recommendation system. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 767–768. ACM, 2007.

[71] M. Mellia, R. Lo Cigno, and F. Neri. Measuring ip and tcp behavior on edge nodes with tstat. *Computer Networks*, 47(1):1–21, 2005.

[72] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee. Measurement and analysis of online social networks. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 29–42. ACM, 2007.

[73] S. Mitra, M. Agrawal, A. Yadav, N. Carlsson, D. Eager, and A. Mahanti. Characterizing web-based video sharing workloads. *ACM Transactions on the Web (TWEB)*, 5(2):8, 2011.

[74] P. Mohan, S. Nath, and O. Riva. Prefetching mobile ads: Can advertising systems afford it? In *Proceedings of the 8th ACM European Conference on Computer Systems*, pages 267–280. ACM, 2013.

[75] C. Mueller, S. Lederer, and C. Timmerer. A proxy effect analyis and fair adatpation algorithm for multiple competing dynamic adaptive streaming over http clients. In *Visual Communications and Image Processing (VCIP), 2012 IEEE*, pages 1–6. IEEE, 2012.

[76] C. Müller, S. Lederer, and C. Timmerer. An evaluation of dynamic adaptive streaming over http in vehicular environments. In *Proceedings of the 4th Workshop on Mobile Video*, pages 37–42. ACM, 2012.

[77] C. Müller and C. Timmerer. A vlc media player plugin enabling dynamic adaptive streaming over http. In *Proceedings of the 19th ACM international conference on Multimedia*, pages 723–726. ACM, 2011.

[78] P. Ni, R. Eg, A. Eichhorn, C. Griwodz, and P. Halvorsen. Spatial flicker effect in video scaling. In *Quality of Multimedia Experience (QoMEX), 2011 Third International Workshop on*, pages 55–60. IEEE, 2011.

[79] J. Park, S.-J. Lee, S.-J. Lee, K. Kim, B.-S. Chung, and Y.-K. Lee. Online video recommendation through tag-cloud aggregation. *IEEE MultiMedia*, 18(1):0078, 2011.

[80] L. Plissonneau and E. Biersack. A longitudinal view of http video streaming performance. In *Proceedings of the 3rd Multimedia Systems Conference*, pages 203–214. ACM, 2012.

[81] S. Podlipnig and L. Böszörmenyi. A survey of web cache replacement strategies. *ACM Computing Surveys (CSUR)*, 35(4):374–398, 2003.

[82] T. R. Puzak. Analysis of cache replacement-algorithms. 1985.

[83] X. Qiu, H. Liu, D. Li, S. Zhang, D. Ghosal, and B. Mukherjee. Optimizing http-based adaptive video streaming for wireless access networks. In *Broadband Network and Multimedia Technology (IC-BNMT), 2010 3rd IEEE International Conference on*, pages 838–845. IEEE, 2010.

[84] F. Ricciato, E. Hasenleithner, and P. Romirer-Maierhofer. Traffic analysis at short timescales: an empirical case study from a 3g cellular network. *Network and Service Management, IEEE Transactions on*, 5(1):11–21, 2008.

[85] H. Riiser, P. Vigmostad, C. Griwodz, and P. Halvorsen. Commute path bandwidth traces from 3g networks: analysis and applications. In *Proceedings of the 4th ACM Multimedia Systems Conference*, pages 114–118. ACM, 2013.

[86] T. Rodrigues, F. Benevenuto, M. Cha, K. Gummadi, and V. Almeida. On word-of-mouth based discovery of the web. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, pages 381–396. ACM, 2011.

[87] N. Sastry, E. Yoneki, and J. Crowcroft. Buzztraq: predicting geographical access patterns of social cascades using social networks. In *Proceedings of the Second ACM EuroSys Workshop on Social Network Systems*, pages 39–45. ACM, 2009.

[88] A. Saul. Wireless resource allocation with perceived quality fairness. In *Signals, Systems and Computers, 2008 42nd Asilomar Conference on*, pages 1557–1561. IEEE, 2008.

[89] M. Saxena, U. Sharan, and S. Fahmy. Analyzing video services in web 2.0: a global perspective. In *Proceedings of the 18th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, pages 39–44. ACM, 2008.

[90] S. Scellato, C. Mascolo, M. Musolesi, and J. Crowcroft. Track globally, deliver locally: improving content delivery networks by tracking geographic social cascades. In *Proceedings of the 20th international conference on World wide web*, pages 457–466. ACM, 2011.

[91] S. Shafer and D. Rogers. Similarity and distance measures for cellular manufacturing. part i. a survey. *THE INTERNATIONAL JOURNAL OF PRODUCTION RESEARCH*, 31(5):1133–1142, 1993.

[92] M. Z. Shafiq, L. Ji, A. X. Liu, and J. Wang. Characterizing and modeling internet traffic dynamics of cellular devices. In *Proceedings of the ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems*, pages 305–316. ACM, 2011.

[93] D. S. Sharp, N. Cackov, N. Laskovic, Q. Shao, and L. Trajkovic. Analysis of public safety traffic on trunked land mobile radio systems. *Selected Areas in Communications, IEEE Journal on*, 22(7):1197–1205, 2004.

[94] T. Stockhammer. Dynamic adaptive streaming over http–: standards and design principles. In *Proceedings of the second annual ACM conference on Multimedia systems*, pages 133–144. ACM, 2011.

[95] X. Su and T. M. Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in artificial intelligence*, 2009:4, 2009.

[96] V. Suranuntakul and C. Srinilta. Pp caching: Proxy caching mechanism for youtube videos in campus network. In *Proceedings of the International MultiConference of Engineers and Computer Scientists*, volume 1, 2011.

[97] R. Teixeira, K. Marzullo, S. Savage, and G. M. Voelker. In search of path diversity in isp networks. In *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, pages 313–318. ACM, 2003.

[98] R. Torres, A. Finamore, J. R. Kim, M. Mellia, M. M. Munafo, and S. Rao. Dissecting video server selection strategies in the youtube cdn. In *Distributed Computing Systems (ICDCS), 2011 31st International Conference on*, pages 248–257. IEEE, 2011.

[99] S. Traverso, K. Huguenin, I. Trestian, V. Erramilli, N. Laoutaris, and K. Papagiannaki. Tailgate: handling long-tail content with a little help from friends. In *Proceedings of the 21st international conference on World Wide Web*, pages 151–160. ACM, 2012.

[100] B. Van Roy. A short proof of optimality for the min cache replacement algorithm. *Information processing letters*, 102(2):72–73, 2007.

[101] J. Wang. A survey of web caching schemes for the internet. *ACM SIGCOMM Computer Communication Review*, 29(5):36–46, 1999.

[102] Z. Wang, L. Sun, S. Yang, and W. Zhu. Prefetching strategy in peer-assisted social video streaming. In *Proceedings of the 19th ACM international conference on Multimedia*, pages 1233–1236. ACM, 2011.

[103] T. Wu, K. De Schepper, W. Van Leekwijck, and D. De Vleeschauwer. Reuse time based caching policy for video streaming. In *Consumer Communications and Networking Conference (CCNC), 2012 IEEE*, pages 89–93. IEEE, 2012.

[104] S. Xiang, L. Cai, and J. Pan. Adaptive scalable video streaming in wireless networks. In *Proceedings of the 3rd multimedia systems conference*, pages 167–172. ACM, 2012.

[105] Q. Yang, H. H. Zhang, and T. Li. Mining web logs for prediction models in www caching and prefetching. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 473–478. ACM, 2001.

[106] J. Yao, S. S. Kanhere, I. Hossain, and M. Hassan. Empirical evaluation of http adaptive streaming under vehicular mobility. In *NETWORKING 2011*, pages 92–105. Springer, 2011.

[107] H. Yin, X. Liu, F. Qiu, N. Xia, C. Lin, H. Zhang, V. Sekar, and G. Min. Inside the bird's nest: measurements of large-scale live vod from the 2008 olympics. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, pages 442–455. ACM, 2009.

[108] H. Yu, D. Zheng, B. Y. Zhao, and W. Zheng. Understanding user behavior in large-scale video-on-demand systems. In *ACM SIGOPS Operating Systems Review*, volume 40, pages 333–344. ACM, 2006.

[109] M. J. Zekauskas, A. Karp, B. Teitelbaum, S. Shalunov, and J. W. Boote. A one-way active measurement protocol (owamp). 2006.

[110] Y. Zhang and A. Årvidsson. Understanding the characteristics of cellular data traffic. *ACM SIGCOMM Computer Communication Review*, 42(4):461–466, 2012.

[111] X. Zhao, J. Yuan, R. Hong, M. Wang, Z. Li, and T.-S. Chua. *On video recommendation over social network*. Springer, 2012.

[112] M. Zink, K. Suh, Y. Gu, and J. Kurose. Characteristics of youtube network traffic at a campus network–measurements, models, and implications. *Computer Networks*, 53(4):501–514, 2009.