POLITECNICO DI TORINO

SCUOLA DI DOTTORATO

Dottorato in Ingegneria Elettronica e delle Communicazioni – XXV ciclo

Tesi di Dottorato

Social distributed content caching in federated residential networks

Jin JIANG
Matr. 167942

Tutore
prof. Claudio Casetti

Coordinatore del corso di dottorato
prof. Ivo Montrosset

Febbraio 2013

Summary

The amount of data that residential users generate, store and share with their friends via a multitude of devices has grown significantly in the past few years. As a consequence, there is a clear need for an intelligent content distribution system that can help the residential users exchange data between devices, assure safe backup and share it with other friends.

In order to address this need, gateway-centric network architecture is proposed, whereby each household is equipped with a gateway that stores, tags and manages the data collected by the residential users. The gateway can provide the Internet access, manage content storage and collect the users' social networking data with the user's permission. Also in keeping with the "federated home" vision, multiple neighbouring or remote home gateways can be connected in a collaborative fashion, and can exchange various kinds of information.

Moreover by leveraging typical social networks indicators, such as interests, hobbies and preferences, and by having all personal digital data appropriately tagged, content replicas can be proactively cached on those gateways whose users are most likely to be interested in accessing the cached content. So that the matching of remote users and content to cache would allow to catch two birds with a stone: safe, redundant online backup and social content sharing.

In our work, we devise an efficient content placement schema to determine where to cache the content from a user's gateway to remote gateways belonging to his/her social friends. Moreover, placing content replicas "outside" the home (i) consumes transmission bandwidth for uploading the content and (ii) incurs a storage cost

on the remote friends' home gateways. So for the case of content backup, we aim at a strategy that maximizes the friends' benefit by trying to match content type and friends' interests while considering the bandwidth constraint between the local gateway and the remote friend gateways. Meanwhile in the case of content sharing, we also take into account the interest of content owner's other friends and the bandwidth from the replica gateway to the corresponding interested friend gateways.

Firstly, we formulate this content placement problem as a Budgeted Maximum Coverage (BMC) problem which is NP-hard, and we use Gurobi solver, which runs a variant of the branch-and-cut algorithm, to numerically solve the optimization model and to obtain the optimal content placement solutions. We also compare it with two different content placement strategies for gateways with various quota sizes, under a realistic simulation scenario with synthetic social network and realistic network environment.

We then devise and evaluate some low-complexity, distributed heuristic algorithms which can be implemented on federated residential gateways to realize a social caching strategy and use simulations in the same synthetic social network scenario to show the final content placement among "friendly" gateways well approximates the optimal solution under different network settings. Finally we evaluate the impact of different content caching strategies on the content retrieval performance in terms of average access delay, hops and distance using NS3 simulator.

To make our simulation scenario more realistic, we need to set up a synthetic social network that shares the basic common properties of real social networks, namely realistic degree distribution and the distribution of friends and their position/distance on the network topology. We choose Facebook as a target, since it is one of most popular and largest online social networking sites nowadays. In particular, we use the findings in other related works to characterize the network properties and to establish a relationship between geographical distance and friendship probability that matched the one that can be measured in Facebook.

In the implementation work of federated residential gateways, a framework prototype to connect the home gateway with the online social network (OSN) service is proposed and implemented through API services exposed by OSN (i.e. Facebook Graph API). So each gateway can be allowed to collect its user's social networking data with the user's permission, and to obtain the interest information shared by their friends who also belong to the federation residential networks after checking with lookup service (which is a naming service set up by us to manage the information on the home users and their associated gateways). Firstly, the overview of gateway implementation architecuture will be provided; followed by the design of a framework for caching module and the detailed description on the caching module implementation. Finally, we use Common Open Research Emulator (CORE) to emulate a realistic network environment for home gateways to test and evaluate our content caching algorithms implementation.

Acknowledgements

I would like to thank my thesis advisor, Prof. Claudio Casetti for his kindness in helping me to find my research direction and giving the basic lessons on how to do research. The completion of this study would have been impossible without his valuable helps and advices.

And the special thanks to all the staffs in TNG group of POLITO for their helps during my more than three years here.

Finally, I am deeply indebted to all members of my family, who constantly support me during my study.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The Internet has today more than 1.6 billion users and over 400 million households connected through broadband access; this corresponds to a growth of over 350% during the last decade. An almost exponential increase in user generated data is expected, with media content being an increasing fraction of the total data. Similarly, the storage capacity will continue to grow and include distributed data storage, both across end-users devices and storage networks “in the cloud”. End users will be both producers and consumers of content, and social networks will spread and grow based on users’ needs and desires to connect, interact and exchange data. Today, there is a clear need for an intelligent content distribution system that can provide a unified content storage and access from within the home and via the Internet, and help the home users exchange data between devices, assure safe backup and share it with other friends.

In this first chapter, we introduce the motivation and scope of the research carried in the context of this thesis, followed by the main contributions and an outline of the subsequent chapters.

1.1 Problem

The means of communication has significantly changed during the last years, starting with the invention of the telegraph, followed by the fixed/mobile telephone and radically disrupted with the wide adoption of the Internet. While people wound originally communicate via voice or limited text, new technologies have added a lot of new dimensions, allowing not only to have verbal discussions, but rather exchange messages, images, videos and generally experiences.

The Internet nowadays has connected a total of more than 1.6 billion users and over 400 million households through broadband access [1]. This corresponds to a growth of over 350% during the last decade. Along with its growth and increased adoption, the Internet has continuously evolved to adapt to new services, applications and operational requirements. We do not expect this growth and evolution to slow down, but rather the opposite. We foresee the future Internet to have a continued growth of users and connected households, including more interconnected networks, wireless networks and mobile users. And it is clear that the future Internet is moving towards to a user- and content-centric network.

Meanwhile, thanks to the continuously evolving of facilities and broadband access, we expect an almost exponential increase in data, with media content constituting an increasing fraction of the total data. For example, as digital cameras became common place, average consumers were able to generate their own content, directly in digital format, and transfer it to their personal computers for further processing, such as archiving, printing or sharing. Also, mobile phones with built-in cameras have brought the content creation devices to the masses.

Similarly, the storage capacity will continue to grow and include distributed data storage, both across end-users devices and storage networks "in the cloud". End-users will be both producers and consumers of content and there will be further development and increase of social networks based on the users' need and desire to connect, interact and exchange data. Furthermore, users will expect to be able to access and consume their services and content in a ubiquitous manner even while being remote and potentially mobile. Around this ecosystem, new content distribution channels became available, with dedicated on-line services, facilitating content storage and sharing, among family, friends and the wider community in general.

The next disruption is expected to happen in the home domain, as residential broadband connections have become a reality for many. The single home computer for accessing the Internet has been replaced with residential networks containing many interconnected multimedia devices, at least in the well developed countries, which promise to make communication even richer. Trends show that between 2007 and 2014 the total amount of personal digital content in a typical digital home could increase from about 1 Terabytes (TB) to almost 12 TB [2], in which half are for the entertainment in the home. Fig. 1.1 presents the estimated cumulative growth of digital content in average American home from 2007 through 2014.

The explosion of content amount and the wealth of digital devices and appliances have brought about the dramatic changes in our habits in everyday's life. Some content is acquired from a service provider; some is collected independently from any service provider; some is autonomously generated by the user. This complication brings about some critical difficulties and complexities in the management and delivery of the content for the home users. These entail to rethink the residential network architecture to support the home users' need to store, find and access content regardless of its location.

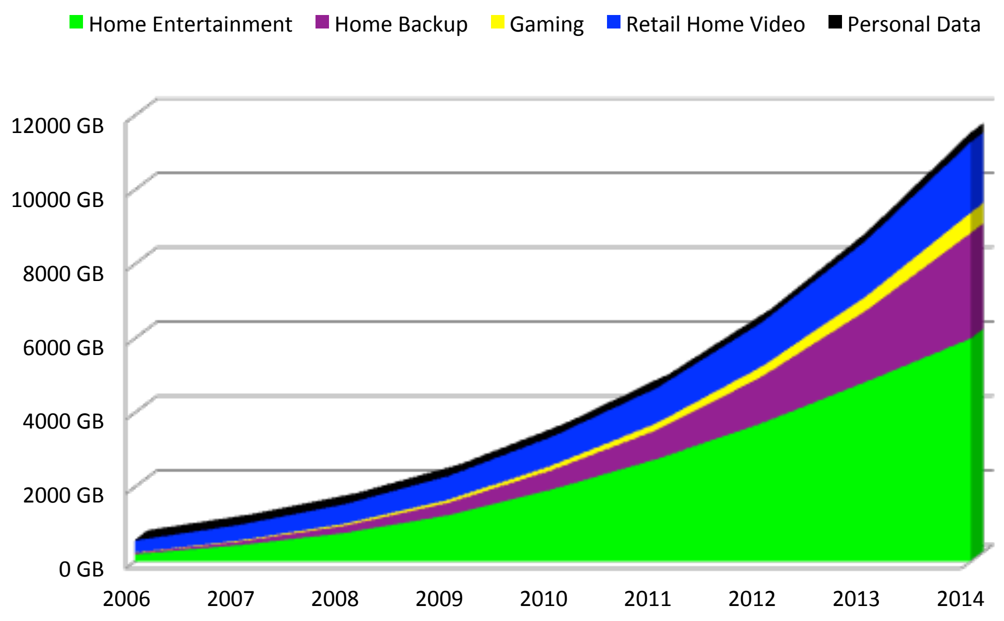


Figure 1.1: Estimated amount growth of digital content in average American home.

1.1.1 Residential network environment

The residential network is a vision of PCs, mobile devices (still cameras, video cameras, phones) seamlessly collaborating through wired or wireless networks to share content and enrich user experience. Today, home users are continuously acquiring, viewing and managing a rapidly increasing amount of digital content using an increasing number and variety of such devices. Due to this proliferation of content and devices, home users desire to enjoy content easily and conveniently. But today's reality is a complex environment of heterogeneous services, networks and devices; all operating independently. Current systems do not fulfil the home user expectations of simple management, demonstrated by some real use cases illustrated as following:

- Bob is watching his favourite TV show but he's late to dinner at some friends', so he needs to start recording the rest of the show, convert it into a suitable format and upload it to his mobile device: he will watch it the following morning while commuting by train.

- Alice is visiting Bob and wants him to listen to the new rock album she just downloaded. So Bob requires to remotely stream the songs from Alice's mobile device onto his home Hi-Fi set. After listening through it, Bob is so mesmerized by it that he asks Alice if she has previous works by that same rock band. She

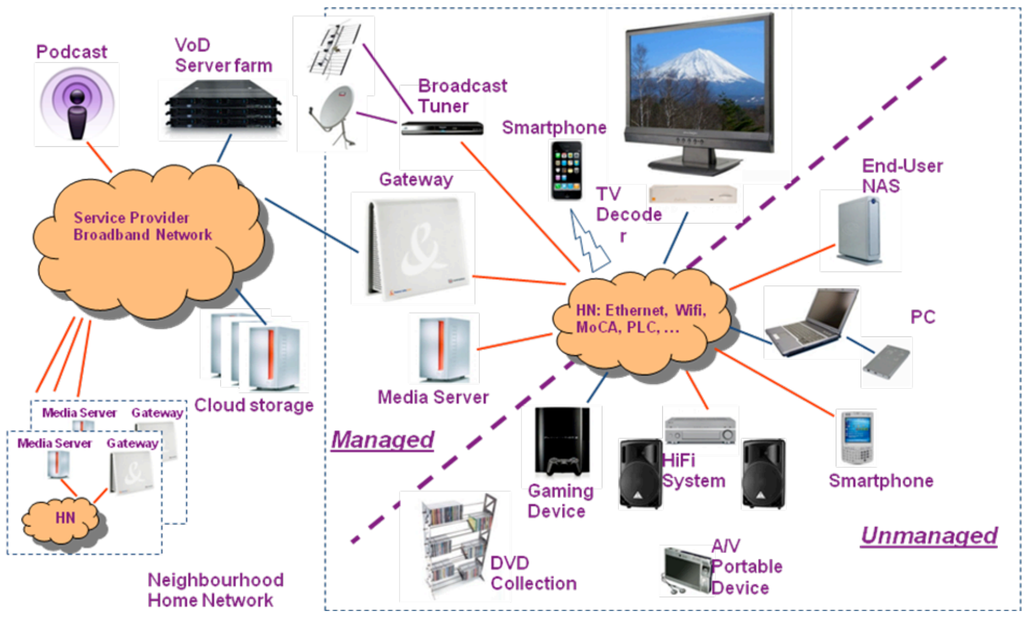


Figure 1.2: Residential network environment.

does, however they are not on her mobile devices: they are stored "in the cloud". So, she needs to fetch the content and streams it over the Internet to Bob's Hi-Fi set.

- Bob wants a "paperless" home, so he starts scanning all of his bills and receipts (unless they were in electronic form in the first place), and jots down his notes/lists/recipes on his laptop or mobile. He wants these notes to be indexed, filed under the proper category (using tags that he adds) and stored in his electronic household storage system for later browsing. In case of need, he will want to be able to access these notes on-demand from outside his home.

- Alice carries her laptop home from work. As she relaxes before dinner, she leaves the laptop on and an automated wireless backup onto a storage device is started. In the course of the backup, the updated content of her hard-disk is scanned for specified tags she may have appended to documents or media files: beside being bundled in the daily backup, these will be made available to any home appliance or digital document repository that matches the tags.

From the vivid scenarios above, we can derive a number of challenges for the future residential network architecture to better support the users' everyday life in interacting with the Internet and handling their content. First, residential networks connected at the edge of the Internet are becoming an integral part of the Internet. These residential networks are typically controlled by non-technical end-users, which give rise to new challenges in terms of simple network management for regular end-users. There needs to be support that alleviates the users from manually configuring, monitoring, and optimizing their networks, as well as support that aids them in case troubleshooting is needed. Second, new networking mechanisms are needed to better support the users having easy remote access from the Internet to their residential networks and their content/services traditionally residing in the residential network. Third, following this, it is clear that the future Internet must include an improved support for the users to easily handle their networked digital content. The wealth of available content, whatever its source, needs to be indexed and associated to a contextual application or appliance (e.g., news segments to a TV set or to a smart phone; music files to a hi-fi set or any mp3 player). Improved content management needs network architecture support to efficiently provide storage, search and access of digital content. Furthermore, the content should be easily accessible regardless from where the users are connected to the Internet. It must also support content privacy as well as easy content sharing, depending on content type and the owners' preferences. Fourth, the support for community-oriented networking must be improved in the future Internet. Innovative systems and network solutions must be developed to better support and exploit community networks to provide improved value-added services to the users. Finally, the integration of other sectors with the future Internet poses challenges for how to interconnect different networks and systems to ultimately provide a common service infrastructure. Moreover, the user interaction with these services must also be revisited for a successful integration.

1.2 Research objectives

As described previously, home users have become producers, importers and exporters of large amounts of digital content via a multitude of devices that are all managed independently. Perhaps one of the most remarkable complications is the reliance on digital storage for whatever information content we own or produce. So, there is a clear need for an intelligent content distribution system that can help the home users exchange data between devices, assure safe backup and share it with other users.

The user usually collects content in an anarchic way, and often the storage and archival of this content is done in an ad-hoc fashion with copies being made on external media such as USB hard disks or DVDs. Currently, a user may store some content on storage units he owns at home. If data are to survive a disk crash, the user must define his own management policies (using e.g., RAID storage, do regular backup on a dedicated disk, etc.). Of course, no savvy user would rely solely on storing precious, irreplaceable data in a single device and backup systems are now common in most households. More recently, the availability of “cloud” storage services, aimed at consumers and companies alike has introduced a new opportunity. The user can rely upon of a cloud storage provider for personalized data backup and sharing, such as Dropbox, Box.net, Apple’s MobileMe or Amazon’s Simple Storage Services (S3). In the latter case, a wide-band Internet connection can be exploited during idle periods to run background data transfer onto cloud storage. Some of these services charge users based on the storage volume and network bandwidth consumed, other are free and include premium options for a fee.

However, using just a single such provider of cloud storage has a number of shortcoming: (i) The terms of the service contract as of today are such that the service provider does not give any guarantees against loss and also is not liable in case of data loss and (ii) the service provider may simply go out of business at any time - as has happened often in the past with Internet-based hosting companies - in which case all data may be lost. In addition, storing on a remote server can be excessively time-consuming if done directly from the user devices, while a home storage device could trickle the content to a cloud storage using space available bandwidth.

Furthermore, accelerated content access, as well as data protection against loss in case of disaster (“single point” failure described above) can be addressed by distributing the content across multiple storage devices that are located in geographically distinct places including storage space that not only can be rented at data centers (cloud-based content storage) but also provided by other households (p2p-based content storage).

Another drawback of personal, cloud-based or p2p-based storage approaches is the fact that data of potential interest of other users sit unused in a storage device. Let us consider the following example. George has a set of pictures of the latest family vacations and he wants to show them to his friend John or other friends, while, at the same time backing them up. George remotely uploads the pictures to John’s NAS, where a storage quota is reserved for such purpose; also this replica on John’s NAS can be shared with George’s other friends. John is then notified that a copy of the pictures now exists in his NAS and that he is welcome to have a look, while keeping it in its NAS as a backup for George or a caching for George’s other friends. For fairness, a similar quota for John’s backups should be set aside at George’s. The example could be extended to a close group of friends, as defined within social networks, and the potential of such a scheme instantly become apparent. By leveraging typical social networks indicators, such as interests, hobbies and preferences, and by having all personal digital data appropriately tagged, the matching of remote users and content to cache would allow to catch two birds with a stone: safe, redundant social backup and social content sharing.

To improve efficiency of data exchange for home users to manage, backup and share their own generated content, caching techniques could be exploited for improving content delivery performance, in our vision including:

- **Gateway-centric architecture**

  The federated residential gateways can be used for caching the interested content for their home user's social friends. The residential gateways are ideal to act as stable caches: they lay at the edge of the network between the residential network and the Internet, and are highly available since they remain powered-on most of the time.

  And, keeping with the "federated homes" vision under the FIGARO [3] project (which will be introduced later), multiple neighbouring or remote home gateways can be connected in a collaborative fashion, and can exchange various information. So the federated residential gateways, which formed as the federation overlay network, can be tuned as a "caching layer" of content distribution for home users.

- **Social-aware**

  The intuition that is pursued in our work is the following: if a gateway is allowed to collect its users' social networking data, such information could be exploited to combine content sharing and content backup. Social data could include (but not be limited to) social contacts and social interests, friends' locations and whether they are in the federated home network or not. Federated gateways could then be designed so as to reserve a part of their storage quota to store content from other gateways belonging to friends from their users' social networks. The content could reflect common interests among such friends. For example, instead of (or in addition to) an anonymous "cloud" backup, a gateway could autonomously choose to upload a set of Mozart concertos to a music-lover friend's gateway. Beside creating a backup of the music files in a trusted location, these could also be enjoyed by the friend, who can access them on his/her own gateway.

  So content caching can exploit the user's social network in P2P-based storage solution. By combining content caching and information on the content owned by the user's social network, this can create a more efficient content caching mechanism. Some examples: if I buy some music, which a friend of mine also has, we can use each other's home gateways for caching; if I share my family photos with my parents, they will have a copy for purposes of looking at the photos but also as a backup for me. This can be taken further: the user doesn't actually have to copy data for making a backup, if that data is already available somewhere on another home gateway. One just needs a control mechanism that ensures one can only restore content from another location if he originally owned that content. This not only saves a lot of bandwidth, i.e. actual copies do not have to be made if the content is already available elsewhere, but also saves a lot of storage capacity and, indirectly, energy for powering all this bandwidth and storage capacity.

- **Optimization in content placement**

  Creating "redundant copies" that are cached on multiple residential gateways can assure reliability of irreplaceable content and accelerate content access for home users. We need a placement scheme that determines where to cache the redundant copies: storing the redundant copies "under the same roof" certainly increases the reliability, but may not protect against events such as theft or natural disasters, which can only be addressed by storing the data on geographically distributed households.

  Placing redundant copies "outside" the home (i) requires transmission bandwidth and (ii) incurs a cost for the additional storage space needed. We plan to develop strategies that use information about the social interactions between the members of the federated environment in order to optimize the system performance to maximize the benefit for all the home users.

- **Distributed heuristics**

  Also the distributed content caching and dissemination algorithm that can approximate to the optimal solution, along with a dispatching protocol must be devised to allow the gateway to efficiently deliver the content to the appropriate friend gateway. We will investigate the cost-reliability trade-offs provided by the different redundancy and replacement schemes and implement a prototype of caching module and test it under a realistic network environment.

  Several practical considerations, however, force us to draw a more complicated picture. Firstly, there are gateway selections issues. Choosing a friend's gateway to cache data only because interests match is not a sound policy from a networking point of view. The remote gateway could have poor connectivity or it could be overloaded. The gateway could be located in a far away country (even though friends in social networks are more likely to be in nearby areas [4]). The remote gateway should implement a solid quota management to avoid being swamped by friends' contents.

> Additionally, there are management details to address: the user must rely on the cached content to be readily available on the remote gateway (or, at least, it should be notified when the content is about to be deleted). If the content is deleted, a second-best choice should be identified, based on the same criteria that guided the former selection. Also a reliable updating policy should also be devised.

However, the outlook is not as simple as the description implies. Some issues outlined below might be beyond the focus of this thesis. For example, the home user should not be required to explicitly manage and schedule caching. Instead caching should occur automatically and transparently to the user. Lastly, copyright and ownership issues also should need a second thought.

1.3 Contributions

The basic assumption is that all the content is stored on the home gateway, i.e. whenever a satellite device comes in the communications range of the home gateway, all the digital content is automatically transmitted to the home gateway. A transfer in the other direction from home gateway to satellite device could also happen. For instance somebody may want to have copies (maybe in reduced resolution) of his pictures on his smartphone. You can attach a USB hard disk or NAS (Network Attached System) to expand the residential gateway's storage capacity. Another assumption is the residential gateway could gather the user's social data (e.t., on Facebook) after authorized by its home users using their credentials.

In this thesis, we choose instead to focus on content caching in the federated residential gateways and, specifically, to devise an efficient content placement scheme to determine where to cache the content from a user's gateway to remote gateways belonging to his/her social friends. As remarked above, placing content replicas "outside" the home (i) consumes transmission bandwidth for uploading the content and (ii) incurs a storage cost on the remote friends' home gateways. So we aim at a strategy *that maximizes the friends' benefit by trying to match content type and friends' interests while taking into account both the bandwidth constraints among home gateways as well as the storage space at the remote gateway.* Thus, we model this optimization problem as a Budgeted Maximum Coverage (BMC) problem, as preliminary introduced in [5], and numerically obtain the optimal content placement solutions using the Gurobi solver [6] under a synthetic social networking scenario, whose set up will be discussed in the following. Next, we design a collaborative social-aware placement strategy for federated home gateways and develop heuristic distributed caching algorithms taking the dynamic nature of users' social networking information into account to achieve the near-optimal results. Finally we evaluate the heuristics and discuss the conditions under which they can approximate the optimal solution.

The following is a summary list of the contributions of this thesis:

- We formulate this content placement problem as a Budgeted Maximum Coverage (BMC) problem which is NP-hard, and we use Gurobi solver [6], which runs a variant of the branch-and-cut algorithm, to numerically solve the optimization model and to obtain the optimal content placement solutions. We also compare it with two different content placement strategies for gateways with various quota sizes, under a realistic simulation scenario with synthetic social network and realistic network environment.

- We then devise and evaluate some low-complexity, distributed heuristic algorithms which can be implemented on federated residential gateways to realize a social caching strategy and use simulations in the same synthetic social network scenario to show the final content placement among "friendly" gateways well approximates the optimal solution under different network settings. And we evaluate the impact of different content caching strategies on the content retrieval performance in terms of average access delay, hops and distance using NS3 simulator.

- To make our simulation scenario more realistic, we need to set up a synthetic social network that shares the basic common properties of real social networks, namely realistic degree distribution and the distribution of friends and their position/distance on the network topology. We choose Facebook as a target, since it is one of most popular and largest online social networking sites nowadays. In particular, we use the findings in other related works to characterize the network properties and to establish a relationship between geographical distance and friendship probability that matched the one that can be measured in Facebook.

- In the implementation work of federated residential gateways, a framework prototype to connect the residential gateway with the online social network (OSN) service is proposed and implemented through API services exposed by OSN (i.e. Facebook Graph API). So each gateway can be allowed to collect its user's social networking data with the user's permission, and to obtain the interest information shared by their friends who also belong to the federation residential networks after checking with lookup service (which is a naming service set up by us to manage the information on the home users and their associated gateways). Meanwhile, we use Common Open Research Emulator (CORE) to emulate a realistic network environment for residential gateways to test and evaluate our content caching algorithms implementation.

1.4 Thesis organization

The rest of this thesis is organized as follows. In the next chapter we introduce our research background and present a list of related works. Chapter 3 describes the assumptions of our model and the modelling procedure of the content placement problem, which is formulated as an budgeted maximum coverage (BMC) optimization problem. Chapter 4 addresses the same problem from the point of view of distributed heuristic algorithms. Chapter 5 introduces the simulation scenario in which we present a procedure to construct a synthetic social network. Also the performance comparisons of the optimization approach and of the heuristics are shown in Section 5.2. In Chapter 6, the overview of caching module architecture and the details of implementation on the residential gateway is presented. Finally, in Chapter 7, we draw some concluding remarks of our study and outline directions for future work.

Chapter 2

Background and related works

In this chapter, we introduce the research and technology domains related with our work. Firstly, we will give an overview description to the framework of FIGARO project which is an European project under *FP7*. It proposes a research agenda that will spawn in parallel across multiple areas reflecting the multi-dimensionality of the problem we are called to address. Then, we review and discuss the other related works in multiple fields under our main works.

2.1 FIGARO project

The Future Internet Gateway-based Architecture of Residential netwOrks (FIGARO) project envisions that residential networks connected at the edge of the Internet are becoming an integral part of the Internet, and that content will be created and delivered to/from hundreds of millions households hosting these residential networks. To address the challenges above, FIGARO therefore proposes an evolvable future Internet architecture based on gateway-oriented federation of residential networks. The fundamental concepts of FIGARO and its overall architecture are *gateway-centric networking* and *federation of residential networks*. In this section, we recapitulate the definition of these two core concepts needed for the understanding of the overall FIGARO system and the description in the subsequent sections.

2.1.1 Gateway-centric networking

Traditionally, residential gateways interconnect the residential network with the Internet, and are responsible for aggregating a multitude of devices and services within the residential network, and are control points where many Internet-based services pass through [7,8]. The residential gateway clearly has a central role in our vision of proposed "gateway-centric" network architecture [3,9], whereby we assume, each home is equipped with a gateway and a large number of interconnected devices (satellite appliances) within the household.

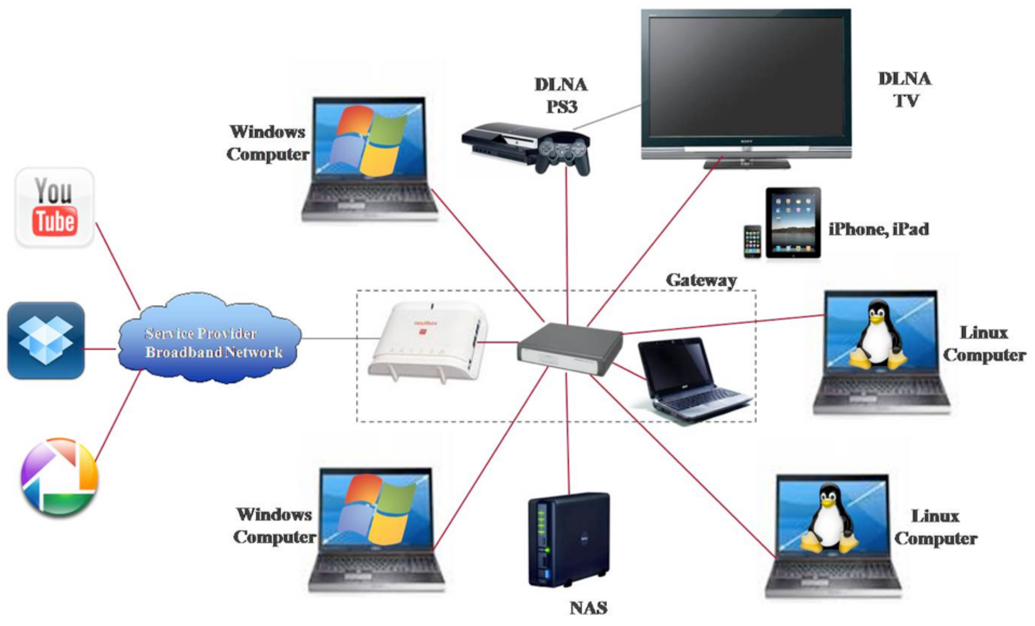


Figure 2.1: A gateway-centric residential network.

The home gateway will play an increasingly important roll in the management and storage of the content for the home users. The gateway allows any content to be downloaded from outside the household, stored on it and accessed by satellite devices in communications range of the home gateway. Upload to the gateway occurs either through the same device where the content was generated or temporarily stored, or through dedicated input devices allowing appliances with no wireless interface, or with no predefined uploading software, to access the gateway-centric home network (e.g., a digital camera equipped with a USB interface can connect with an external input devices that, in turn, uploads the pictures to the gateway for storage in the appropriate repository within the household).

To preserve the content from possible data loss, by configuring a backup plugin, the gateway will take care of backing-up the selected resources. The configuration of both the resources to be scheduled for backup and the parameters of the backup phase can be set by the user. Also, the gateway can participate in the operation as a cache among home network devices, and allow storage on other users' gateways by exploiting social information of home users.

Taking a simple example, Jane has to manage her large photo collection (10GB). She has her photos tagged in such a way that the gateway can discriminate which are personal, which are for family and relatives, which can be shared with friends, and which can be for public viewing. Jane's laptop uploads the photos on the gateway via the LAN. The gateway can both keep a local copy and cache the pictures onto friends' and families' gateways (according to the tagging) for additional storage capabilities. Once there, the pictures can also be viewed by the interested users.

Access to the user's home resources and content should be available (in a secure fashion) also when the user is not at home. To this end, multiple remote gateways (beyond the user's neighbourhood) can be configured in a collaborative manner to act as "virtual home gateways": each user on the move would be assigned to such "virtual" home gateway wherever they are connected, and they would provide computing and storage resources needed for the users on the move. Such access may occur on demand or on a scheduled basis (e.g., the user will need access to his medical records stored at home between 2pm and 4pm on a specific date when a hospital check-up is planned).

Furthermore, the home gateway can be used for coordination across different sectors. The future Internet will not be restricted to the IT, telecom and media sectors, however. It will also include services and user-centric content from other sectors, such as utility (energy, water, etc.), e-health care and security. Currently, those sectors develop their own communication and system solutions. Such intra-sector optimization often leads to badly standardized, poorly scalable and closed system. It's also being recognized among the world's governments (most notably in the US, Australia and the Netherlands [10]) that intra-sector optimization leads to only sub-optimal solutions for the society as a whole; It is simply too expensive and no longer acceptable that every single service requires its own separate infrastructure deployment along with installation of additional control devices. One example is the push from the energy industry to install elaborate energy meters, whereas many of the included communication functions could be handled by the existing Internet infrastructure with help of the residential gateway. Another example is the multiple movement sensors next to each other in the house, one for e-health and the others for security. We foresee a rapid growth of Internet-based applications and services in these other sectors. Moreover, user will expect to access these services and their content in the same manner as any other Internet-based service. The Future Internet must therefore evolve to support not only the increasing demands in the IT/telecom/media sector, but also meet the new requirements of these other sectors. For example, end-to-end quality of service and user friendliness for these emerging services and applications typically differs significantly from those currently governing the Internet. Furthermore, while the Internet Protocol (IP) clearly plays an important role in the integration of sectors, the services and applications in these other sectors may still use non-IP technologies. There is therefore a need for co-ordination across different sectors in order for them to interconnect and ultimately collaborate.

To conclude, the residential gateway is deployed where network domains are interconnected; it interconnects the residential network with the Internet and is responsible for aggregating a multitude of devices and services within the residential network. With the convergence of networking technologies, the residential gateway has become a critical infrastructure at home. The gateway is only the invariant and indispensable element of the residential network and a natural control point where many Internet-based services pass through. Most Internet-enabled end-user devices are connected to it while at home, and could in the future also access the gateway remotely over the Internet. Furthermore, a residential gateway today is nearly as powerful as a PC, and is capable of supporting the increasing requirements originating from the future Internet challenges outlined above. A gateway-centric approach enables efficient network management in terms of monitoring of network, application and services, as well as automatic troubleshooting and network optimizations. Furthermore, it also enables efficient management of digital content.

So, the residential gateway serves as the primary platform to implement and demonstrate the strength of our proposed future Internet architecture.

2.1.2 Federation of residential networks

FIGARO proposes an evolvable Future Internet architecture based on gateway-oriented federation of residential networks. We define "federation of networks" as follows. A federation of networks is composed by two or more independent networks that are interconnected and can operate, at least partly, in a coordinated fashion. The network in a federation can be independent and heterogeneous in terms of their ownership, characteristics, and technologies, as well as in their targeted services and applications. However, these networks share a set of common objectives, such as providing a certain set of services to one or more users, exchange of information, or sharing and optimization of network resources. Hence, in a federation of networks the sum is greater than its parts since the federation can offer functionality and services beyond what an individual network can. Naturally, network federation necessitates definition of key system requirements for successful operation. For example, common interfaces for querying about data, content, services and resources must exist. The federation should also have an agreement regarding common representation of information, such as network configuration and network monitoring data.

We extend the above concept and define "federation of residential networks" as follows. The federation is structured around the residential networks connected at the edge of the Internet. We consider the residential gateways additionally to undertake the role as federators, internally as well as externally, in the FIGARO architecture. We call a federation *external* when multiple gateways interconnect to form federated cooperative overlays across residential networks. In contrast, we call a residential network federation *internal* when a gateway interconnects (and federates) two or more networks within a single residence. We note that external network federation ultimately provides a scalable approach to Internet-wide ubiquitous access and sharing of data, content, services and resources. It will enable to offer added value in terms of e.g., collaborative services such as resource sharing, collaborative network optimizations, content distribution and management. Internal network federation, on the other hand, ultimately aims to converge the networks and services of regular IP-based networks and other sector-specific (possibly non-IP) networks used by e.g., home automation and e-health. More specifically, it targets to be an enabler of cross-sector services by providing a common interface to these networks and their services through the gateway.

These two types of federations: *External Federation* and *Internal Federation* within FIGARO will be described next.

External federation

Externally, gateways interconnect to form federated cooperative overlays across residential networks. These so-called external federations enable to offer added value in terms of collaborative services such as resource sharing, collaborative network optimizations, content distribution and management. For example, access network sharing for bandwidth bundling and multi-path video streaming. These services leverage neighbouring home networks to use their unused access network resources to increase the access bandwidth and improve the video streaming device robustness, respectively. Another example is optimizing WiFi networks on a neighbourhood scale. In this case, collaboration among neighbouring homing is leveraged to enable network optimizations such as load-balancing and interference management. A third example is collaborative content management, such as social-aware content caching and distributed backup, which exploit unused storage resources on gateways across the federation. In the first two examples, the federation is formed by gateways of home networks in the same neighbourhood, typically within WiFi radio range of each other. In the last example, the federation does not have an explicit geographical boundary but can potentially comprise gateways (and users) located anywhere. To this end, we consider that external federations can have different scope and potentially different functional or operational restrictions.

We give three examples of different types of external federations considered in FIGARO. A “neighbourhood federation” is a geographically localized federation consisting of members in some type of neighbourhood community. For example, this may include homes in the same apartment buildings or potentially span several co-located buildings. It is easy to imagine that these apartment owners are already part of another neighbourhood community, such as an apartment co-op or community league. Our neighbourhood federation use cases mentioned above rely

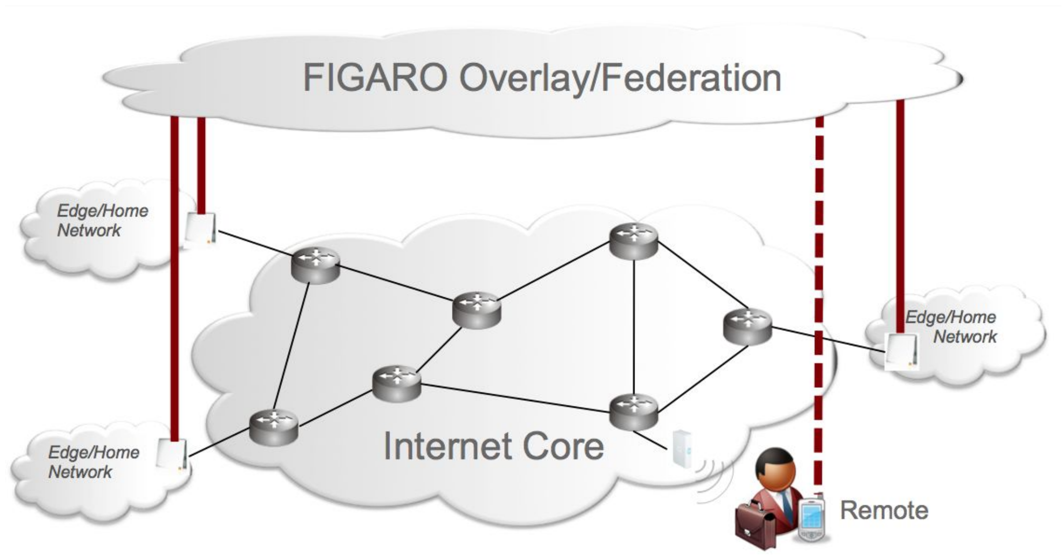


Figure 2.2: Simplified FIGARO residential network overlay/federation.

on inter-home wireless connectivity and thus require a certain density to operate and be efficient. However, one can imagine other services that do not have the wireless interconnectivity restriction, such as a collaborative neighbourhood micro-grid or local virtual power plant service. Another possible federation type is “service provider federation”. To some extent, such federation forms exist today and offer simple services. For example, subscribers to the operator FREE in France may subscribe to a service that allows them to access the Internet through the WiFi access points of other FREE subscribers. As per the use cases mentioned above, FIGARO shares the idea of such community WiFi concept, but extends it with more value added services. A service provider oriented federation clearly benefits from simplified operation, such as authentication, authorization and accounting, as a single service provider can handle all this. Such a federation type may be suitable for e.g., an access network bandwidth bundling service. While a service provider federation limits the federation to include only members (subscribers) of that particular service provider, it is technically feasible to implement provider-independent or cross-provider federations. A third interesting type of federation is the “social federation”. This is based on users’ social network and may be based on e.g., family, friends, or online social networks (OSNs) such as Facebook, Flickr or Google+. For example, the FIGARO social-aware caching device, though nominally capable of leveraging any OSN, is based on a user’s Facebook network. Conceptually, a “social federation” may be either service provider specific or service provider independent, each with its obvious pros and cons.

Internal federation

Current innovations in e-Health services and remote energy management are still hindered by a tendency to create vertically isolated (sometimes referred to as “stove-pipe”) ICT solutions (i.e., dedicated to a specific service), leading to non-scalable, expensive and closed systems. As a result, common technologies that are present in home networks are not only IP-based communication networks, but also other network technologies such as Zigbee and Bluetooth. They often provide their own, inseparable network and service delivery layer, and are usually only meant to interconnect function-specific devices with the home network. In the FIGARO architecture, the internal federation is designed to define proxy functionality between these non-IP networks and the common network and service layers within a single residential network. As such, the internal federation facilitates the integration of services from sectors such as e-Health, energy management and domestics, by interconnecting different networks and systems that do not necessarily use IP technology. The federation enables features such as communication, resource, and content sharing among the involved networks as well as a common interface to these networks through the gateway. Part of the internal federation is the common service delivery infrastructure. In providing this infrastructure we will leverage reliability, control and interface provided by the home gateway. Ultimately, the internal network federation facilitates cross-sector convergence with a unified access to different types of devices and services.

In summary, FIGARO has great potential to evolve the current Internet to meet current and future demands of applications, services and end-users. It will deliver solutions that contribute to the population well being through improved and simplified interaction with the Internet, their residential networks, and the digital content transported and services delivered over these networks. By developing technologies and solutions for integrating and supporting other sectors to converge with ICT, FIGARO will significantly reduce the economic costs that otherwise would arise when each sector continue to deploy their own communication and service infrastructure. Furthermore, through this convergence FIGARO facilitates solutions for improved interaction and control of energy management services. We expect that this ultimately becomes a green milestone for the society. As a general result from FIGARO, employment should benefit from the rapid growth of the ICT and multimedia markets as well from the potential of cross-sector innovation. This is expected to create new jobs, particularly in SMEs and start-up companies that develop and sell new networking solutions and content services for the Internet and home users. Finally, the project is expected to result in technologies that will strengthen Europe’s position and give competitive advantage to European industry in Future Internet technologies, residential gateway business and the home automation industry.

FIGARO’s scope can be categorized into the following four areas. (i) *Network* *organization and optimizations,* exploiting gateway-based overlay network federations. (ii) *Content management,* enabling unified access and distributed caching and backup. (iii) *Converged services,* extending current intra-sector control and interface platforms. (iv) *Monitoring,* enabling network and application performance characterization and troubleshooting.

Our works in this thesis are mainly focusing on the second category *Content management*, to address the need for content sharing, content confidentiality and protection against accidental loss through a unified, federated content management system that hides most of the complexity and low-level aspects from the end user, and to study and achieve more effective content management and delivery mechanism for sharing and backup of home user's content in federated residential networks using caching techniques, which will be described in details in the following sections. Rather, it leverages the availability of storage space both in the home environment and outside it. The home gateway plays a key role in our architecture and will act as a hub that performs content indexing and either holds a copy of (the most sensitive) content available on the different devices, or records the location of the content for prompt retrieval. We will now briefly recall the critical issues and other works related to our tasks.

2.2 Related works

Distributed caching techniques

In order to facilitate the access to distributed content (especially in appliances with low storage capabilities), our framework will be integrated with caching algorithms. Caching is a well known method to improve efficiency of data exchange in a client/server or p2p application. Deployed on an end host, caching allows the application to reuse elements that have been recently used. As most users of computer networks expect the instantaneous service provided by an infrastructure, delay for any information request may cause users' impatience and ask for more interactivity. Caching helps mitigating this problem by providing as quickly as possible part of the answer from a nearby host whenever possible.

Caching in cooperative fashion and cache placement in wireless networks have been explored by several works. That's due to the opportunistic nature of connectivity in a mobile network makes the design of an efficient caching scheme a very difficult issue. In particular, the work in [11] proposes a cooperative caching scheme that requires the nodes to periodically broadcast their identity as well as their cache contents. In [12], Yin and Cao present distributed caching strategies for ad hoc networks, according to which nodes may cache highly popular contents that pass by, or record the data path and use it to redirect future requests. The work in [13] presents both a centralized and a distributed solution to the cache placement problem of minimizing data access cost when network nodes have limited storage capacity. While some of these works address topics that are relevant to our project (cooperative caching, nodes with limited storage capacity) they tend to follow simple LRU (Last Recently Used), LFU (Least Frequently Used) policies.

"Locality of reference" which is a classical method to assess the performance of caching algorithms does not include geographical locality. We will work around these limitations by devising location-aware caching mechanisms, allowing content to be stored on devices that are physically close to the most likely content consumers within the federated home.

Among these methods or architectures, content distribution networks (CDNs [14–17]) which serve to deliver web objects have seen tremendous growth since its emergence. To minimize the retrieving delay experienced by a user with a request for a web object, caching strategies are often applied - contents are replicated at edges of the network which is closer to the user such that the network distance between the user and the object is reduced.

This paper [18] explains the caching techniques exploited for content delivery in both CDN and p2p overlay from the viewpoint of network provider, especially when discussing the localization of caches, which is in line with our simple cache-per-city topology used in the simulation scenario.

This work [19] uses the aspect of *social friend are geographically closer to each other* [4] to improve the caching in CDN. It describes how geographic information extracted from social network can be exploited to improve caching of multimedia files in a content delivery network. They take advantage of the fact that social graph can propagate in a geographically limited area to discern whether an item is spreading locally or globally. This informs cache replacement policies, which utlize this geographical information to ensure that content is kept close to the users who may be interested in it.

And our work is also similar with the p2p or friend-to-friend storage system, in which [20] is to critically assess the current state of the centralized social web, identify several novel research problems like privacy, data ownership and data portability etc. and outline the possible solution: friend-to-friend computing(F2F). F2F is a completely decentralized architecture in which two computers can communicate only if their owners know one each. But we are based on the home gateway, which is with much higher availability than other devices.

Distributed content replication

The only way to assure reliability of irreplaceable content is to distribute such content by creating "redundant copies" that are stored on multiple devices; so a redundancy scheme is needed to determine how to create redundant copies: the two techniques at hand are (i) replication [21, 22], i.e. making one ore more identical copies and (ii) coding [23, 24], where the original data are transformed (encoded) in such a way that only a portion of the encoded data is sufficient to reconstruct the original data.

In our vision, considering the potential interest of other friends, to make the content sit in the storage of residential gateway easily be shared, we follow the former method - replication. So, our work falls into the same category as several recent research efforts tacked the problem of multiple replicas across different resources. Solutions such as PRACTI [25] and Cimbiosys [26] additionally provide partial replication capabilities to better utilize the available storage capabilities. Podbase [27] provides a framework for automatically ensuring that multiple copies are stored across devices. But none of these works leverages or exploits the potential of social networking.

Parallel to the problem of cache placement is the one of replication, i.e., of determining whether it makes sense (and to what extent it does) for multiple copies of a content to be created and actively disseminated across the network (where they may in turn be cached). Simple, widely used techniques for replication are gossiping and epidemic dissemination [28,29], where the information is forwarded to a randomly selected subset of neighbours. Another viable approach to replication is represented by quorum-based [30] and cluster-base protocols [31]. Both methods, although different, are based on the maintenance of quorum systems or clusters, which in mobile networks are likely to cause an exceedingly high overhead. In our work, we will focus on replication mechanism that exploits inference techniques to tweak the rate of replication on the basis of the actual content demand or interest, in order to lower the network overhead.

Also, for content backup there have been a number of research projects that investigate peer-to-peer based backup [32–34]. There is also one paper [35] that describes an experiment using S3 for backup. The innovation of this proposal is to consider multiple storage providers that can be either located at the edges (peers) or in the cloud and to insist on being both storage and bandwidth efficient at the same time. What is missing, however, is a comprehensive approach that integrates cloud backup, federation backup and local backup in a seamless, user-transparent fashion, as the one pursued in our vision.

Social-aware content placement

Related to our problem there are also the works on content placement exploiting information from social networking. The work in [36] proposes ContentPlace, which is a social-oriented framework for data dissemination taking into consideration user interest with respect to content. ContentPlace assumes that nodes can be aware of the social communities they belong to. Using a general utility-based optimization framework, ContentPlace defines distributed algorithms for nodes to select which content to locally replicate, out of what is available on encountered nodes. These algorithms take into consideration the estimated distribution of content in the network, and the interests of the users with respect to content. A similar approach is taken in [37] where it is shown that mobility and cooperative content replication strategies can help bridge social groups. Another relevant work on an efficient social-aware content placement in opportunistic networks is [38] in which the authors model the content placement as the facility location problem.

[39] investigates how mobile systems could exploit people's social interactions to improve these systems' performance and query hit rate. They build a trace-driven simulator that are able to recreate the behaviour of mobile systems in a social environment. Finally, they find that mobile systems can benefit substantially from exploiting social information.

Tribler [40] introduces the notion of "friends" to peer-to-peer networks as special nodes among which content sharing gets a boost. This can be, in turn, applied to the federated home environment where nodes carrying "preferred content" within the residential network or a neighbour's residential network can be labeled as "friends", while nodes outside the federated environment are treated as "normal" peers. Such vision can be further extended to include a user's social networking contacts, whether they are in the federated home or not. On the one hand, one of the main functionality of a social networking service is to keep users updated on what events occurred in their local social environment, through notifications. On the other hand, a social networking service must allow any users to locate on demand the closest content that matches a particular request.

Also, [41] presents a design of an overlay constructed on top of a social network and shows that it gives a sizeable improvement in lookups, average round-trip delay and scalability as opposed to other overlay topologies. Herein it uses a popular real-world social network namely "Orkut" by evaluating the clustering behaviour of its graph structure and the socializing pattern of its members.

[42] is to use the P2P network to deploy a directory service facilitating search for friends by exploiting existing trust relations of the social network links. This work is to demonstrate that only a subset of the whole social network is adequate to build an efficient and reliable service.

In [43] "social cache" acting as the bridge among friends is introduced to alleviate efficient information delivery in the distributed online social networks (dOSNs). Given the topology of social graph, some nodes are selected as the "social caches" such that 1) each node should either be a social cache or connect to at least one social cache; and 2) a pair of friends should be connected by at least one social cache if none of them is a social cache. So this social cache selection problem further is formulated as the Neighbor-Dominating Set (NDS) problem. Because a node only contacts the social caches it is associated with by pushing its content updates and fetching the updates of its friends. The use of social caches can significantly reduce the total social traffic in the network. However, the selection decision is made when the information of global social graph is given [44]. It also can bring "hot-spot" problem especially for the nodes with high friend degree, due to not considering the bandwidth allocation and user interest locality.

Social graph generation

In the simulation works, we need to derive a realistic social graph with the suitable scale of users for simulation. So we referred other works on the social graph generation.

[45] uses short-range technologies (e.g., Bluetooth) on users' mobile phones to sense and keep track of other phones in the proximity. Then proximity records are processed using a variety of algorithms that are based on social network theories of geographical proximity and of link prediction. The result is a personalized and automatically generated list of people the user may know.

[46] deals with a methodology to generate a social graph of users' actions and predict the future social activities of the users based upon the existing relationships. This graph is updated dynamically based on the changes in the selected social network site.

[47] presents a large-scale measurement study and analysis of the structure of multiple online social networks (e.g., Flickr, YouTube, LiveJournal and Orkut). They crawled the publicly accessible user links on each site, obtaining a large portion of each social network's graph. Their results confirm the power-law, small-world and scale-free properties of online social networks including some observations of that the in-degree of user nodes tends to match the out-degree; that the networks contain a densely connected core of high-degree nodes; and that this core links small groups of strongly clustered, low-degree nodes at the fringes of the network.

[48] studies in detail about user interactions in the social network. This paper proposes the use of *interaction graphs* derived from Facebook user traces to impact meaning to online social links by quantifying user interactions and shows they exhibit significantly lower levels of the "small-world" properties shown in their social graph counterparts. This means that these graphs have fewer "super nodes" with extremely high degree, and overall network diameter increases significantly as a result. The results reveal that studies of social applications should use real indicators of user interactions in lieu of social graphs.

[49] is to define both a measure of local community structure and an algorithm that infers the hierarchy of communities that enclose a given vertex by exploring the graph one vertex at a time. They can use this algorithm to extract meaningful local clustering information in the large recommender network of an online retailer.

[50] studies online auction networks rather than other online social networks

by modelling and characterizing the structure and evolution of the network of user interactions on eBay. This work observes that the graph exhibits both significant differences and similarities to commonly studied graphs and studies the feedback behaviour of users. Finally, they develops an intuitive model that captures key properties of the graph in a visual and memorable way.

Chapter 3

Optimization model

Our optimization problem aims at maximizing the utilities/benefits of users whose residential gateways provide the storage space for caching their friends' content. To this optimization problem a series of constraints are added, including the boundedness of the total resource capacity of every gateways.

3.1 Model assumptions

Consider there are N households in the federation network, each equipped with one residential gateway, hence N is the number of residential gateways in the network. A residential gateway GW_i ($i = 1, 2, \ldots, N$) acts as a repository managing and storing content for all home users in the corresponding household. Its storage capacity is split into a local data storage (i.e., for data primarily stored by its local users and synchronized with the local devices) and into a "friend quota", Q_i, which we define as the available storage capacity for caching the data uploaded by friends of its associated home users. We define the upload and download bandwidth of the gateway GW_i as $C_i^{(u)}$ and $C_i^{(d)}$, respectively. The bandwidth C_{ih} from gateway GW_i to gateway GW_h is assumed to be:

$$C_{ih} = \frac{min\{C_i^{(u)}, C_h^{(d)}\}}{\alpha(d_{ih})} \tag{3.1}$$

where $\alpha(d_{ih})$ is a factor depending on the distance d_{ih} between gateway GW_i and GW_h, and $1 \leqslant \alpha(d_{ih}) \leqslant 10$. In Section 5.1 we will provide a possible definition of $\alpha(d_{ih})$ used by realistic simulation scenario.

We then assume that there are also M home users in the federation network. Each user registers himself/herself on the corresponding residential gateway equipped inside of the household, where the users can access/store their content. For the purpose of identifying which users are registered to which gateway, we define an $N \times M$

matrix $\mathbf{P}$ whose generic element is given by:

$$P_{ij} = \begin{cases} 1 & \text{if } U_j \text{ registered on } GW_i \\ 0 & \text{otherwise.} \end{cases} \tag{3.2}$$

where i indicates the gateways, $i = 1,2,\ldots,N$, while j is the user index, $j = 1,2,\ldots,M$.

As explained earlier, we assume that a crucial gateway functionality is the capability to collect the social information of its users by extrapolating such data from the social networks they belong to. In particular, we are interested in collecting *user's friend lists* and *user's interests*.

To represent the first dataset, we model the friendship between user U_j and user U_f through a friendship function $F(j,f)$:

$$F(j,f) = \begin{cases} 1 & \text{if } U_j \text{ and } U_f \text{ are friends, } j \neq f; \\ 0 & \text{otherwise.} \end{cases} \tag{3.3}$$

The friend list E_j of user U_j can thus be denoted as follows:

$$E_j = \{U_f : F(j,f) = 1\} \tag{3.4}$$

Secondly, the user's interests are mapped from one or more social interest communities that the user belongs to. The degree of user involvement with each such community - which will come to represent the distribution of content type preferences of the user - is captured by the user's *interest vector*, defined as follows. Let I_{jl} denote the interest factor of user U_j in interest type l, with $0 \leqslant I_{jl} \leqslant 1$, $l = 1,2,\ldots,L$ (L is the size of the interest area, i.e., the total number of interest types considered in our system). Let the interest vector of user U_j be the collection of all interest factors of the user associated with all the interest types, denoted by

$$\overline{I}_j = (I_{j1}, I_{j2}, \ldots, I_{jl}, \ldots, I_{jL}) \tag{3.5}$$

where $\sum_{l=1}^{L} I_{jl} \triangleq 1 - r^j$. r^j is the probability of the user U_j to be interested in the interest type out of the interest area L. Without loss of generality and in order not to burden the presentation of this problem, we will just assume users to only have interest in the interest types considered in the system, thus, $\sum_{l=1}^{L} I_{jl} = 1$ or $r^j = 0$.

We assume that the content items in the network are finite, i.e., each user may own any number of items out of K possible items. A generic item k, $k = 1,2,\ldots,K$ has the size $D^{(k)}$ and belongs to interest type l. The association between an item and its interest type is assigned according to a uniformly random distribution. For the sake of notation simplicity, we also assume that every user has the same average number of items to share or backup.

3.2 Mapping onto a BMC problem

Many applications arising in circuit layout, job scheduling, facility location, and other areas, may be modelled using the maximum coverage problem (see [51] and the references therein for examples of applications). The Budgeted Maximum Coverage (BMC) problem introduces a more flexible model for the applications mentioned above [52]. Consider, for example, the problem of locating k identical facilities so that the market share is maximized, introduced in [53]. Namely, there exist clients with associated profits, situated in known locations. A client will use a facility if the facility is within the specified distance from the client. The goal is to locate k facilities so that the total profit of the clients served by the facilities, is maximized. In another version of this problem, considered in [54,55], and known as optimal location of discretionary service facilities, facilities gain profits associated with travel paths of customers; a facility covers a path if it is located in one of the nodes on the path, or at some vertex "close" to the path. Clearly, the problems described above can be modelled by the unit cost maximum coverage problem. However, in practice, the cost of constructing a facility may depend on certain factors associated with the location of the facility. For example, each candidate site for constructing a facility is associated with some cost, and one is assigned a limited budget for constructing the facilities. This generalization of the problem of locating facilities to maximize market share is also discussed in [53]. The budgeted maximum coverage problem is a model that allows us to handle this type of applications.

Our objective is to find a selection of friends from the user's friend list where to cache the user's items onto these friends' associated residential gateways; such selection should maximize the benefit of the hosting users whose residential gateway provide the storage space for caching their friends' content, i.e., by closely matching his/her interests; and it should also maximize the data transfer effectiveness, i.e., by maximizing the bandwidth between the respective gateways. So we cast this optimization problem as a BMC problem [52, 56].

In BMC problems, a collection of sets $S = \{S_1, S_2, \ldots, S_m\}$ with associated costs $\{c_i\}_{i=1}^m$ is defined over a domain of elements $X = \{x_1, x_2, \ldots, x_n\}$ with associated weights $\{w_j\}_{j=1}^n$. The goal is to find a collection of sets $S' \subseteq S$, such that the total cost of elements in S' does not exceed a given budget L, and the total weight of elements covered by S' is maximized. The BMC problem is NP-hard, and [52] presents a $(1 - 1/e)$-approximation algorithm for it.

We assume that gateway GW_i has already collected the user U_j's friend list E_j and the friends' registration information (which friend of U_j is registered on which gateway). In our case, the problem is that the user U_j has a content item k with the size of $D^{(k)}$ to share or backup, and the content item k belongs to the interest type l, so *bins* and *elements* of the BMC problem can be mapped in the following way.

- The *bin* set B_j for user U_j is defined as follows: $B_j = \{b_{j1}, b_{j2}, \ldots, b_{jh}, \ldots, b_{jN}\}$, where bin b_{jh} denotes the set of friends of user U_j who are registered on gateway GW_h, $h = 1, 2, \ldots, N$:

$$b_{jh} = \{U_f \in E_j : P_{hf} = 1\}. \tag{3.6}$$

  We recall that $P_{hf} = 1$ means that user U_f is registered on gateway GW_h, and that $U_f \in E_j$ means that user U_f is in the friend list of user U_j, so $b_{jh} \subseteq E_j$. The cost $c^{(k)}_{(b_{jh})}$ of selecting the bin b_{jh} is defined as the cost of caching the content item k of size $D^{(k)}$ onto the gateway GW_h, which can be defined as:

$$c^{(k)}_{(b_{jh})} = D^{(k)} \tag{3.7}$$

- The *element* set in our problem obviously is the user U_j's friend list E_j.

  For each element/user $U_f \in E_j$ (user U_f is a friend of user U_j), we can define the weight as the benefit $w^{(k)}_{(U_f)}$ that element/user U_f can obtain when item k is cached onto the gateway GW_h where U_f is registered. The definition of benefit is depending on the use case of home user. For example, in the vision of "social backup" mentioned previously, the storage space on the friend gateway is only reserved for caching the backup replica of friend owned content, which is disabled to be shared with others. In this case, just considering the interest locality of social friends and network bandwidth contributed for the fast content backup recovery, the definition of benefit will depend both on the interest that friend U_f will have in the cached content, and on how easily accessible that content will be for the content owner U_j (i.e., on the bandwidth between the uploader gateway and the hosting gateway). We can thus define $w^{(k)}_{(U_f)}$ as:

$$w^{(k)}_{(U_f)} = I_{fl} \cdot C_{hi} \tag{3.8}$$

  where I_{fl} denotes the friend interest of U_f in items of type l which content k belongs to and C_{hi} is the uploading bandwidth from the hosting friend gateway GW_h which provides the storage space for caching back to the content owner's gateway GW_i, on which users U_j and U_f are registered, respectively ($P_{ij} = P_{hf} = 1$). In short, the benefit is calculated in the view point of content owner contributed by the friend gateway using its reliable storage space and wide uplink bandwidth.

  Another case of content caching for collaborative sharing is taken into account not only for content backup, but also towards content sharing among the social friends. In other words the cached file on the remote friend gateway is not only used as a backup replica for the content owner, but also provided as another sharing source for the owner's friends. So that the definition of user benefit should consider the several aspects from more than one counterpart: one is from the content owner, same as the Eq.(3.8) for content backup use case; the other is from the other friends of the content owner.

$$w_{(U_f)}^{(k)} = I_{fl} \cdot C_{hi} + \sum_{\alpha} I_{\alpha l} \cdot C_{h\alpha} \tag{3.9}$$

The first half of the equation above is the same with Eq.(3.8) regarding to the friend's interest value and corresponding gateway's network bandwidth, and the second half takes the potential interest of the owner's other friends α and the available bandwidth from the contributing gateway to those friends' gateways into account.

One constraint is about the gateway friend quota, Q_i, which we recall is the available storage capacity for caching the data uploaded by friends. The other is about the selection constraint, if one user is covered by the cached item ($y_f^{(k)}$ equals one), the corresponding home gateways must be selected as replica for that item ($x_h^{(k)} = 1$).

Finally, our problem can be formulated as follows:

$$\begin{aligned} &\textit{maximize} && \textstyle\sum_{k=1}^{K} \sum_{U_f \in E_j} w_{(U_f)}^{(k)} \cdot y_f^{(k)} \\ &\textit{subject to} && \textstyle\sum_{k=1}^{K} c_{(b_{jh})}^{(k)} \cdot x_h^{(k)} \leqslant Q_h \\ &&& \textstyle\sum_{P_{hf}=1} x_h^{(k)} \geqslant y_f^{(k)} \\ &&& x_h^{(k)}, y_f^{(k)} \in \{0,1\} \end{aligned} \tag{3.10}$$

where $x_h^{(k)} = 1$ indicates that gateway GW_h is selected to cache a replica of content item k, while $y_f^{(k)} = 1$ means that user U_f is covered by the cached content item k. And K is the total number of the content items to share or backup by all the users in the system.

3.3 Problem solving

As we know, the linear programming problem previously defined in Eq.(3.10) is NP-hard. We solve it through the Gurobi solver [6], which is a powerful optimization solver for linear programming (LP), mixed integer linear programming (MILP) and quadratic programming (QP) etc., using advanced implementations of the latest optimization algorithms.

3.3.1 Problem size

The number of Boolean decision variables ($x_h^{(k)}$ and $y_f^{(k)}$) is $O(K\langle N\rangle)$, where $\langle N\rangle$ denotes the average number of the friends per user. And the number of constraints is $O(K\langle N\rangle + M)$. The solution time for an instance with approximately 1,000 gateways, 3,000 users and an average of 5 content items for each user to share or backup, is about 30 minutes using a 4-core 2.3 GHz system and a 4 GB RAM.

3.3.2 Optimal solutions

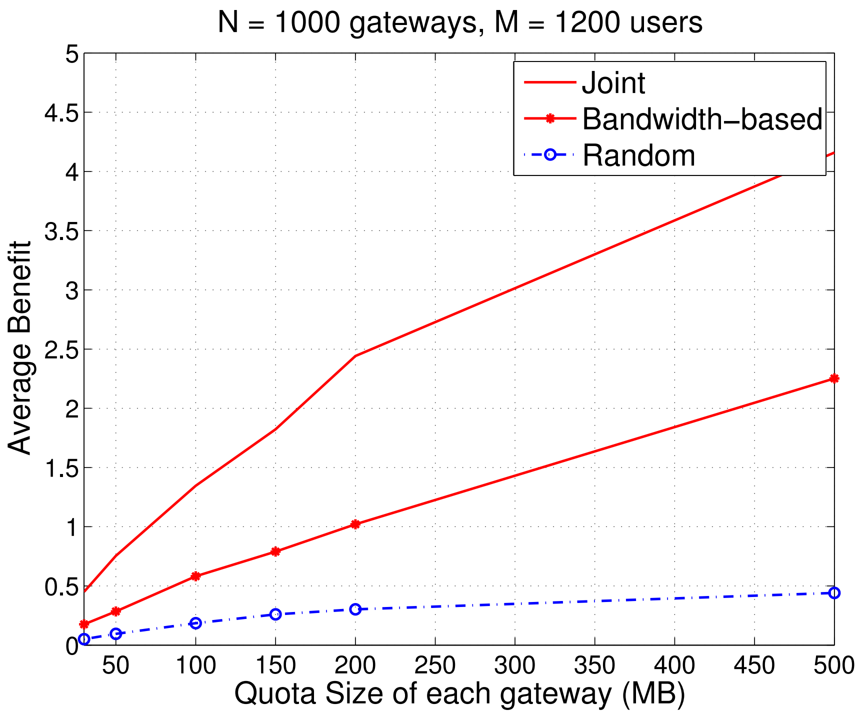


Figure 3.1: Average benefit per user as a function of gateway quotas. ($M = 1200$)

In this section we only consider three different optimization methods of content caching for the backup use case, and compare the optimal solution of these strategies in the same scenario. We use a realistic synthetic social network assembled following the procedure which will be outlined in the Section 5.1.1 later, and the parameters listed in Table 3.1 for the test scenario. As previously described, we assume that each user has an interest vector on the different content types, and the user's interest vector is shared among his/her friends in the whole network.

The first strategy is the *joint* optimization method, described in Section 3.2, in which the friends who have the larger interest in the corresponding content item

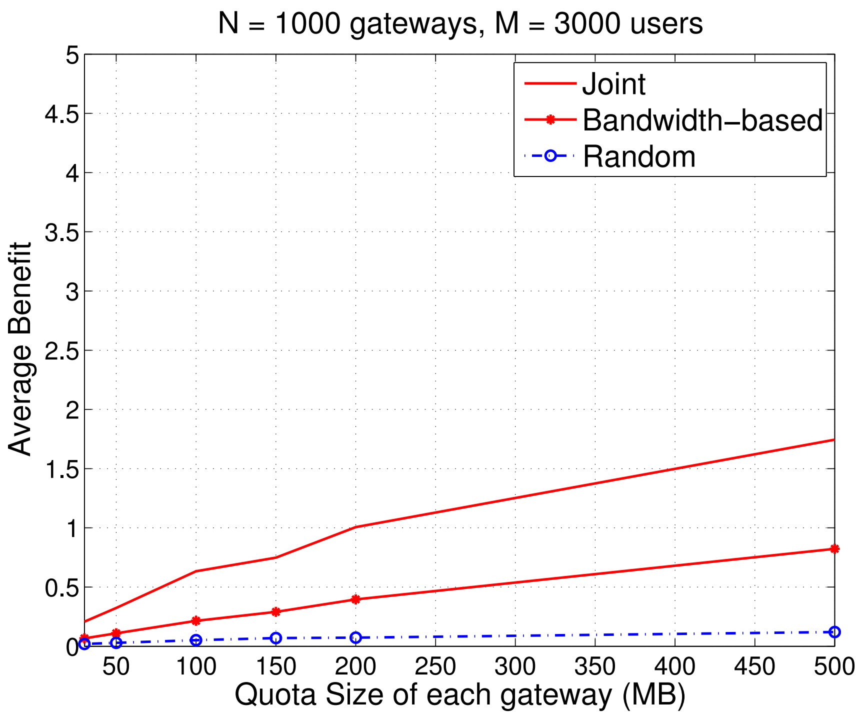


Figure 3.2: Average benefit per user as a function of gateway quotas. ($M = 3000$)

Table 3.1: Parameters used in the scenario

Parameter Name	Notation	Value
Number of Gateways	N	1,000
Number of Users	M	1,200 - 3,000
Number of content items	K	6,000 - 15,000
Items to backup per user		5
Number of interest types	L	10
Content item size	$D^{(k)}$	$10MB$
Uplink bandwidth	$C_i^{(u)}$	$4Mb/s$
Downlink bandwidth	$C_i^{(d)}$	$8Mb/s$

and the higher bandwidth connected will be selected (assuming their friend quota is not used up). The second strategy is the *bandwidth-based* method, in which friends reachable through gateways with the highest bandwidth will be selected, regardless of their interest in the cached items. The last one is the *random* method, in which the user just randomly chooses up to 10 friends to share the content item with, as long as the friends have enough quota to store the item, not considering any other factors.

We find the optimal allocation for the first two strategies through the Gurobi solver, which uses a variant of the branch-and-cut algorithm. Solving the budgeted maximum coverage problem in Eq.(3.10) yields the optimal *joint* content item placement, i.e., the set of candidate friend gateways to select for each content item, as well as the optimal benefit value that each user can obtain by being selected. The optimal *bandwidth-based* placement is obtained again from Eq.(3.10) by changing the benefit definition as:

$$w^{(k)}_{(U_f)} = C_{ih} \tag{3.11}$$

Using the obtained optimal total benefit of all the users, we compute the average per-user benefit obtained in the system, for various quota constraints, shown in Figures 3.1and 3.2, for $M = 1200$ and $M = 3000$ users, respectively (resulting in 1.2 and 3 users per gateway). Even though the average benefit is lower when 3000 users are considered (due to the larger number of users per gateway sharing the same quota), the advantage of finding an optimal allocation for content caching *jointly* depending on interest and bandwidth is clearly visible in both plots. Such advantage amounts to twice the benefit obtained with a *bandwidth-based* strategy only, and to ten times the benefit of a *random* placement selection.

Chapter 4

Distributed heuristics

The greatest hurdles in translating the optimization model into a working implementation are (i) that the model paints a *static* picture, where all users take instantaneous decisions and (ii) that decisions are taken by a centralized, knowledgeable entity.

In this chapter we propose a set of distributed heuristic algorithms that strive to achieve the same goal as the model outlined in the previous section. The algorithms take two different viewpoints: that of participating content owners who have data to share or backup and that of remote gateways who provide their own storage space for their social friends. In both cases, we follow the same arguments used in the optimization problem definition.

From the viewpoint of content owners, not only do they wish to backup or share the data as fast as possible, but, in the long run, they also wish that the remote gateway keeps the content for as long as possible. Therefore, content owners are naturally disposed to choose friends from whose gateways they can retrieve the cached content items more quickly. Also, they would like friends to be interested in the content they cache, because such friends are more inclined to store it for a long time. If one wants to cache their kid's pictures, what better place than the grandparent's gateway?

At the receiving end, the remote gateway can display two types of behaviour that are arguably worth investigating. One is a selfish behaviour: regardless of the caching requests received by friends of its users, the remote gateway will devote its "friend quota" only to maximize the interests of its associated users, i.e., based only on the first factor in the benefit $w^{(k)}_{(U_f)}$ of Eq.(3.8). (Please notice that the selfish behaviour is not considered for the sharing use case, due to its collaboration nature.) The other one is a cooperative behaviour: the remote gateway fills up its friend quota while trying to maximize the whole benefit of Eq.(3.8) or Eq.(3.9) depending on the use case for backup or sharing, hence accounting for both its users' interest and the bandwidth toward the content owner's gateway or other friends' gateways.

We will address either viewpoint through a specific distributed algorithm: a Greedy Placement Algorithm (GPA) run by content owner gateways in order to identify the most suitable places where to cache their content items, and the RePlacement Algorithm (RPA), run by each remote gateway upon receiving a caching request.

4.1 The Greedy Placement Algorithm

We assume that a user has available all items it wants to share or backup when the GPA procedure is started. Further, we assume time to be slotted in intervals of fixed length and that the starting time slot of GPA procedure on a gateway is random. On each gateway, the sequence in which users start GPA is also randomly determined. To achieve the fairness among all the users in the system, each user can run GPA *only once* per time slot.

When starting the GPA, a gateway GW_i will have already collected the following information *from each of its associated users* U_j:

- the friend list E_j;
- the remote friend gateway list RG_j which includes the remote gateways on which user U_j's friends are associated;
- list K_j of items to share or backup;
- for each item $k \in K_j$, the benefit $w_{(U_f)}^{(k)}$ of each friend $U_f \in E_j$ as defined in Section 3.2;
- for each remote friend gateway $GW_h \in RG_j$, a quantity referred to as *gateway aggregate benefit* $w_h^{(k)} = \sum_{U_f \in E_j} w_{(U_f)}^{(k)} \cdot P_{hf}$ (recall that $P_{hf} = 1$ indicates the friend U_f is associated to gateway GW_h);
- a query list Z_j where each element is a pair (k, GW_h) representing an item and the IDs of a remote friend gateways, sorted by their gateway aggregate benefit $w_h^{(k)}$.

The main idea behind GPA, detailed in Algorithm 1, is the following: every time the algorithm is scheduled, user U_j sends a caching request to the remote friend gateway whose ID is in the element that tops the query list Z_j. Such element is then removed from the list if the request is accepted; otherwise, it is pushed back to the bottom of Z_j. After sending caching requests on behalf of a user U_j for a total item size of S bytes, GPA stops and it is rescheduled randomly in the next time slot.

Algorithm 1 Greedy Placement $GPA(U_j, Z_j)$

```
Require: RETRY counters for all elements of $Z_j$
  $size \leftarrow 0$
  loop
    pop_front element $(k, GW_h)$ from $Z_j$
    if RETRY$(k, GW_h)$ > MAX_RETRY then
      continue
    end if
    if $size + D^{(k)} > S$ then
      insert_head element $(k, GW_h)$ into $Z_j$
      break loop
    else
      $size = size + D^{(k)}$
    end if
    send Caching_REQ to $GW_h$ for $k$
    if Caching request rejected then
      push_to_back element $(k, GW_h)$ into $Z_j$
      RETRY$(k, GW_h)$ = RETRY$(k, GW_h)$ +1
    end if
  end loop
  if $Z_j != \emptyset$ then
    schedule $GPA(U_j, Z_j)$ next time slot
  end if
  return  RETRY counters for all elements of $Z_j$
```

Since the query list Z_j of user U_j is sorted by the gateway aggregate benefit for the corresponding remote gateway, the pop_front operation corresponds to extracting from the list the item k and the ID of the best candidate gateway where it can be cached in the current time slot. A *Caching_REQ* message is then sent to such gateway.

Once a gateway receives the *Caching_REQ*, it will first check whether it has already cached this item. If not, and there is enough free space in its friend quota[1], it will set aside the corresponding size for this item in its storage space. A *Caching_REP* message is returned to the item owner notifying it whether the caching request was accepted. If the request is accepted, the content owner will start the upload. If the request is denied, or no reply is received, the corresponding list element is pushed at the bottom of the query list, for a later retry, up to a limit of MAX_RETRY times.

Upon reaching the S bytes caching request limit, and if the query list is not empty, the gateway schedules the next run of GPA, and the next batch of caching requests for user U_j, at the next time slot. In order to achieve fairness across the federated network, all users should use the same upper caching request limit S.

We finally remark that GPA can easily be modified so that the gateway attempts to cache a single item k only onto a limited set of friends' gateway. In this work, we have only considered the most general (and most challenging) case in which the gateway tries to cache all items on all the friends' gateways. Using the above notation, when GPA eventually stops, an always successful gateway will have dislocated $|RG_j| \cdot |K_j|$ items across the federated network, for each of its users.

4.2 The Replacement Algorithm

If remote gateways "passively" accepted all caching requests until their quota is filled up, their collection of cached items would not match the optimum allocation, being strongly dependent on which users start the GPA procedure first, hence which greedy requests they receive first. After all, the friends of the users associated to a gateway share the quota on this gateway by competing with each other. In order to alleviate such imbalance, we introduce the second algorithm, called RPA, RePlacement Algorithm, to be run by every gateway upon receiving a caching request and discovering that the quota is already filled up. We assume that a gateway GW_i holds a list B_i of items cached in its storage space, sorted by benefit, while we indicate by q_i the free space still available for caches out of the friend quota.

As explained above, when a gateway GW_i receives a *Caching_REQ* message for

[1]We will discuss the case of no free space in the next section.

Algorithm 2 The RePlacement Algorithm $RPA(GW_i, k)$

```
Require: $B_i, q_i, w_i^{(k)}, D^{(k)}$
  $replace \leftarrow$ false, $FreeSpace \leftarrow q_i$
  $DropWeight \leftarrow 0, DropSize \leftarrow 0$
  while $B_i \neq \emptyset$ do
    select $k' \in B_i$ with the lowest benefit
    $DropWeight \leftarrow DropWeight + w_i^{(k')}$
    $DropSize \leftarrow Dropsize + D^{(k')}$
    $FreeSpace \leftarrow FreeSpace + D^{(k')}$
    if $w_i^{(k)} < DropWeight$ then
      $replace \leftarrow$ false
      break
    else if $w_i^{(k)} == DropWeight$ then
      if $DropSize \leqslant D^{(k)}$ then
        $replace \leftarrow$ false
        break
      else
        $replace \leftarrow$ true
        break
      end if
    else
      if $D^{(k)} > FreeSpace$ then
        $B_i \leftarrow B_i \setminus k'$
        continue
      else
        $replace \leftarrow$ true
        break
      end if
    end if
    if replace and $D^{(k)} \cdot w_i^{(k)} \leqslant DropSize \cdot DropWeight$ then
      $replace \leftarrow$ false
    end if
  end while
  return $replace$
```

a new item k of size $D^{(k)}$, it will check whether it has the enough storage space for it; if the space is not sufficient, the gateway will compute the aggregate benefit $w_i^{(k)}$ of each cached item according to Eq.(3.8) or Eq.(3.9) depending on the use case for backup or sharing respectively and start the RePlacement Algorithm, described in Algorithm 2.

The gist of the RPA procedure is the following. In order to maximize the benefit of the users associated to the receiving gateway, the replacement strategy considers the removal of cached items with lower benefit than the incoming item. The B_i list is sorted by benefit, and RPA checks if there are enough items with lower benefit than $w_i^{(k)}$ that can be dropped to leave room for the incoming item. A second check verifies if the product of the total benefit and total size of the items selected for dropping is smaller than the benefit/size product on the incoming item. If so, the latter replaces the dropped items in the storage space of GW_i. Ideally, the second check is aimed at preserving the network efficiency, so that a large cached item is not easily dislodged by much smaller item with a marginally higher benefit.

If the remote gateway GW_i replaces content item k', a $Caching\_DEL$ message must be sent to inform the content owner that it needs to find a new gateway where to cache k'. The content owner will thus place a corresponding element in the Z_i queue of algorithm GPA (or start a new instance of GPA if Z_i has been emptied in the meanwhile).

4.3 Complexity analysis

The running time cost of GPA and RPA is minimal. For each user for which GPA is run, the length of the query list Z_j is $O(\hat{K}\hat{E})$, where $\hat{K}$ is the average number of items per user and $\hat{E}$ is the average number of remote friend gateways. So for the individual gateway the running time is $O(\hat{K}\hat{E}m/n)$ with m/n denoting the average number of users per gateway. As for the running time of RPA, the algorithm searches the whole the cached item list B_i to check what can be replaced, while the maximum number of iterations depends on the number of remote friends and their own items to share or backup. So the complexity of RPA for one gateway also is $O(\hat{K}\hat{E}m/n)$.

Chapter 5

Simulation and evaluation

In this chapter, we will now investigate the validity of our distributed caching approaches by realistic simulations.

Firstly, we numerically solve the model and derive the maximal benefit obtained for all the users according to Eq.(3.10). The results will be benchmarked against two other, simpler caching (re)placement strategies, to evaluate the gain that comes at the expense of extra complexity in the caching strategy. Secondly, we compare the content allocation resulting from the optimal solution with what is achieved through the distributed heuristics. In this case too, we will explore variants of GPA and RPA.

Our evaluation will necessarily target a synthetic scenario. Recreating all the conditions and variables of an actual online social network would be a daunting task. We thus extrapolate its essential features and create a scaled-down version for our simulation following the procedure we outlined in the following.

5.1 Simulation scenarios

Our first problem, however, is the definition of a suitable scenario that must exhibit realistic features of a social network, namely the distribution of friends and their position/distance on the network topology.

Then, we will give a brief description on the realistic gateway bandwidth and network topology used in the simulation scenario.

5.1.1 A synthetic social network

To make our simulation scenario more realistic, we need to set up a synthetic social network that shares the basic common properties of real social networks. We choose Facebook as a target, since it is one of most popular and largest online social networking sites nowadays. In particular, we use the findings in [57] and [4] to characterize the network properties and to establish a relationship between geographical distance and friendship probability that matched the one that can be measured in Facebook. We then proceed according to the following three phases [58].

Phase 1: location and bandwidth assignment

At first we assign the geographical location information for each home gateway in the scenario. We uniformly distribute the N gateways in an area of 1,000 × 1,000 square miles. Next, we compute the geographical distance between any two gateways and we evaluate the bandwidth between them through equation (3.1). We define the factor $\alpha(d_{ih})$ so that, intuitively, it is small (hence the bandwidth is large) for nearby gateways (up to 0.1 miles apart). Then, we let it grow linearly (hence the bandwidth decreases) up to 1,000 miles, after which we keep it constant. The choice of the values is clearly arbitrary, but it will serve our purpose of introducing a distance-dependent inter-gateway bandwidth. The $\alpha(d_{ih})$ factor is defined as:

$$\alpha(d_{ih}) = \begin{cases} 1 & 0 < d_{ih} \leqslant 0.1 \\ \frac{d_{ih}-0.1}{111.1} + 1 & 0.1 \leqslant d_{ih} < 1000 \\ 10 & d_{ih} \geqslant 1000 \end{cases} \tag{5.1}$$

where d_{ih} denotes the beeline distance between gateways GW_i and GW_h.

Phase 2: user assignment

The next step consists in distributing the M users onto the gateways, in a uniformly random fashion, so that the average number of users per gateway is the same. Each user will therefore have a geographical location according to the home gateway on which it has been registered. We then compute the geographical distances between each pair of users.

Phase 3: friendship grooming

The final step is the crucial one. The user graph we have constructed in the previous two steps has some degree of plausibility but it does not reflect yet any properties of actual social networks. In particular, if we established friendship between users picked at random on the graph, we would lose the typical locality that is exhibited by social networks, where most online friends tend to live in nearby areas, due to habits, employment or existing relationships. In order to create a plausible synthetic social graph based on the geographical location we were inspired by [59]. Such work presents a stochastic model for spatially embedded social networks based on the ideas of spatial interaction models. In it, each user is assigned an identical budget, and if two users establish a friendship link, the two users should both consume a cost amount from their respective budgets. When a user's budget reaches zero, the user will not be assigned any more friends. We call this cost a "friendship cost" and following [59] we define it depending on the geographical distance as follows:

$$c_{F(j,f)=1} = \gamma ln(d_{jf}) + const \tag{5.2}$$

where d_{jf} denotes the geographical distance between the user U_j and U_f, and $F(j,f) = 1$ means user U_j and U_f are friends, as defined earlier. In our study, we choose $\gamma = 1.05$ and $const = 1$ following the suggested values in [59].

Two details are still missing though: if every user has the same budget, the friendship graph will not resemble a social graph. And we still need to factor in the information on the distance distribution among friends in a social network.

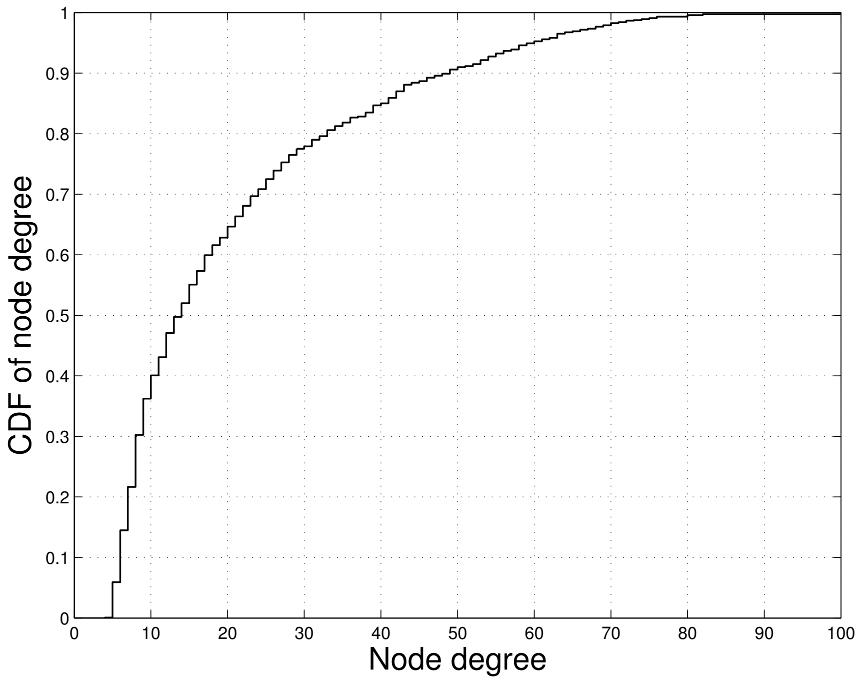


Figure 5.1: CDF of node degree distribution.

To address the first concern, we change the budget assignment so that users initially receive a randomly assigned budget which follows a power-law distribution with exponent −1.5. Thus, we can let the degree distribution in the social graph (i.e., the number of friends per user) follow a power-law distribution as in realistic social networks. Also we can adjust the average budget value to obtain the desired average node degree.

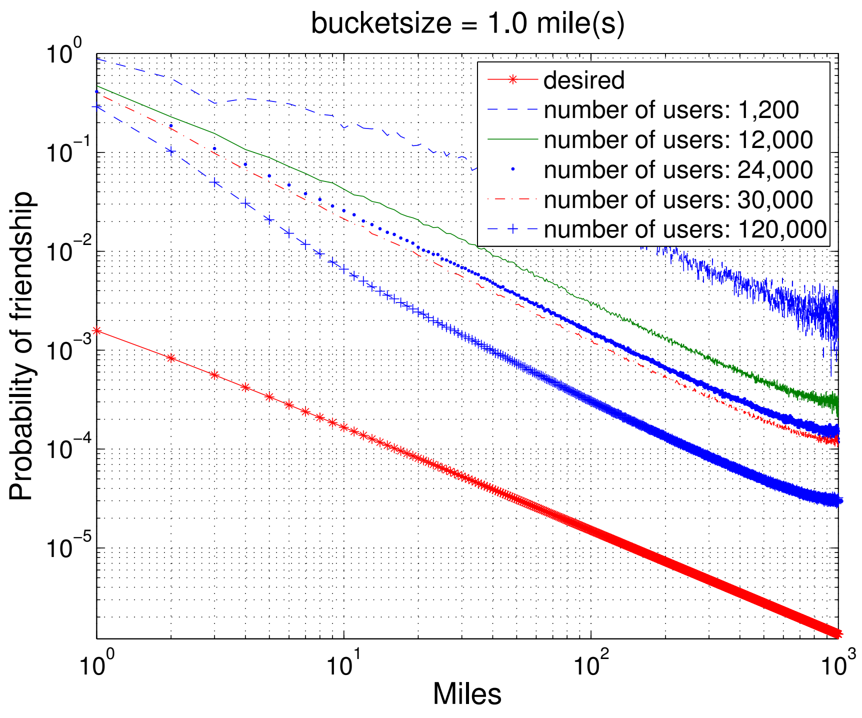


Figure 5.2: Probability of friendship as a function of distance.

Secondly, to tie in the distance distribution among friends we follow the findings in [4], where the probability of a friendship link between two users in Facebook is empirically determined given the two users' geographical distance, as follows:

$$p_{F(j,f)=1} = 0.0019 \times (d_{jf} + 0.196)^{-1.05} \tag{5.3}$$

where $p_{F(j,f)=1}$ denotes the probability that user U_j and U_f are friends.

The resulting social graph is then costructed through the following steps: (i) randomly pick a pair of users among those defined in Phase 2; (ii) use the friendship probability in (5.3) to determine whether to establish a friendship link among them; (iii) if the link is established, the budgets of U_j and U_f are decreased by the amount in (5.2).

The above procedure is repeated until all budgets of all users have been consumed[1].

From the generated social graph, a sample CDF of the number of friends can be shown in the Figure 5.1, we can see that 50% users have about 15 friends, 60% users have less than 20 friends, and just 10% users have larger than 50 friends. It's a power-law distribution.

[1]up to a minimum tolerance value, since it is unlikely that a budget becomes exactly zero

In order to validate whether this approach can get the realistic distribution of friendship over the physical distance, we compare the resulting friendship distribution as a function of user distance with the empirical results in [4]. The comparison is shown in Figure 5.2. The curve labeled "desired" is plotted following the results in [4], where about one million Facebook users were sampled. The other curves show the distributions derived from our procedure for varying number of users. Although we were yet unable to find an optimized implementation of our procedure that allows us to handle millions of users, the trend shown in Figure 5.2 is a clear indication that the distribution we obtain can be considered plausible.

5.1.2 Realistic gateway bandwidth

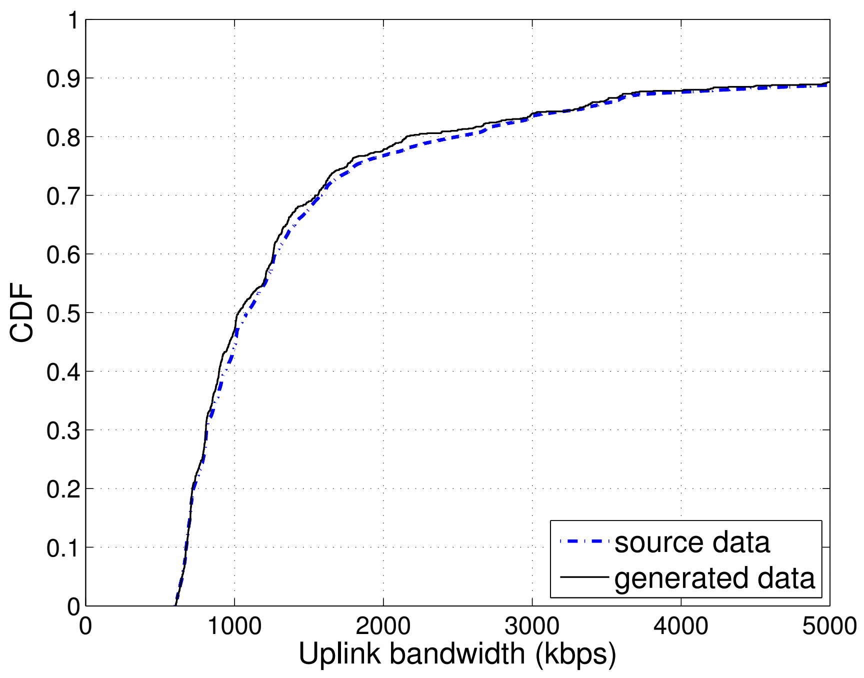


Figure 5.3: CDF of gateway uplink bandwidth.

To set up the realistic simulation scenario, we also need the data set of uplink/downlink bandwidth of each home gateway. Uplink capacities of gateways are obtained by sampling a real bandwidth distribution measured at more than 300,000 unique Internet hosts for a 48 hour period from roughly 3, 500 distinct ASes across 160 countries [60]. We randomly generate the uplink bandwidth of each gateway with the identical distribution of the realistic source data, shown in Fig. 3. These values have a highly skewed distribution, with a median of around 1030.4$kbps$ and a mean of about 5409.6$kbps$. To represent typical asymmetric residential Internet lines, we assign to each gateway a downlink speed equal to four times its uplink.

5.1.3 Simulated network topology

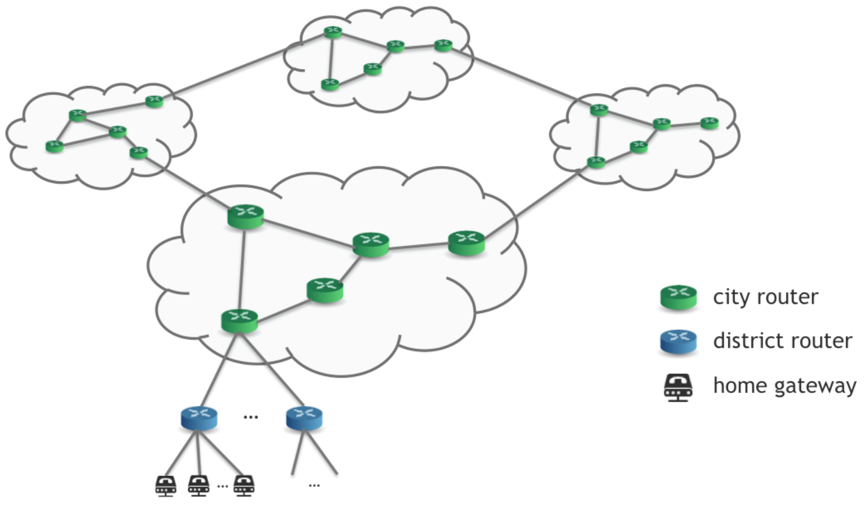


Figure 5.4: Simulation network topology.

Also we create a realistic tiered network topology through organizing network of routers structured in levels. In the first level there are 4 autonomous systems (ASes) totally. In each AS, locate 5 Tier 3 city routers (e.g., Turin, Milan, Rome), which have been suitably chosen latency. And inside each city, there are multiple (e.g., 10) districts, which is made up of set of district routers with which the residential gateway are connected through different types of connections provided by ISPs allowed. Obviously, herein the district is used to be acting as a local neighbourhood inside which there are around 10 households on average.

Furthermore, we can test the distribution of friendship probability among these tiered network structure. From Fig. 5.5 more than 40% friendship is assigned in the same AS, and more than 20% and 10% friendship are distributed in the same city and district respectively. This is another evidence that the friends are much more closer to each other in terms of geographical distance shown in our simulation scenario.

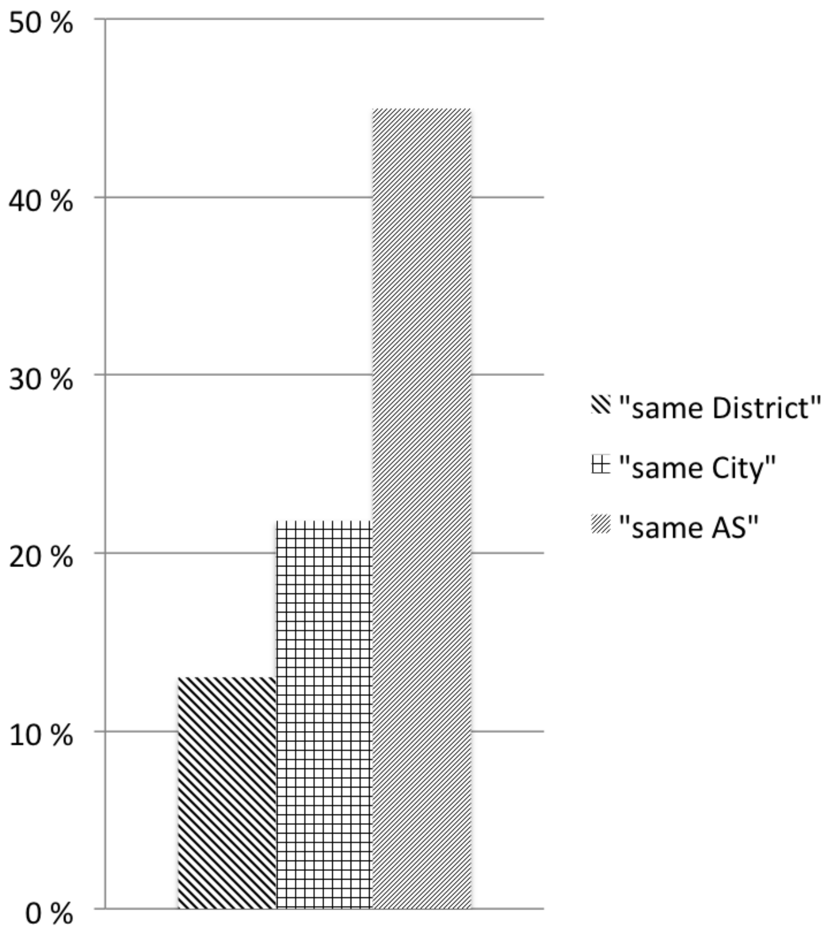


Figure 5.5: Percentage of friendship vs. distance.

5.2 Evaluating optimal model and distributed heuristics

We will now investigate the validity of our approach by following two main directions. Firstly, we numerically solve the model and derive the maximal benefit according to Eq.(3.10). The results will be benchmarked against other, simpler content placement strategy, to evaluate the gain that comes at the expense of extra complexity in the placement strategy. Secondly, we compare the content allocation resulting from the optimal solution with what is achieved through the distributed heuristics. In this case too, we will explore variants of GPA and RPA.

5.2.1 Optimization, heuristics and variants thereof

In order to extract meaningful comparisons between the optimization approach and the distributed heuristics, sharing the same scenario is not enough. On the one hand, we used Gurobi [6], which runs a variant of the branch-and-cut algorithm, to numerically solve the BMC problem in Eq.(3.10). The solution yielded an optimal joint content item placement, i.e., the set of candidate home gateways to be selected for each content item, as well as the optimal benefit value that each user can obtain by being selected. On the other hand, we simulated the heuristic approach in the network simulator ns-3 [61]. So that we could not only focus on the resulting allocation after requests, allocations and replacements have settled (i.e., the protocol part of heuristics), but also study the actual content file transfer while working on the case of sharing purpose (i.e., the impact on the content retrieval performance). Finally, we compared the steady-state outcome to what the optimization had predicted.

We have tried to gauge the effectiveness of optimization and heuristics not only by comparing one against the other, but also by running some variants of either approach with the aim of catching a glimpse of what we would stand to lose or gain, if we chose a simpler (or a more convolute) strategy than the ones outlined in the previous chapter on optimization model (Chapter 3) and distributed heuristics (Chapter 4).

As far as optimization was concerned we evaluated two different content placement strategies. The first strategy is the *joint* optimization method, described in Chapter 3, in which the friends who have the largest interest in the corresponding content item and the highest uplink bandwidth will be selected first (assuming their quota is not used up). The second strategy is a *bandwidth-based* optimization method, in which friends reachable through gateways with the highest bandwidth will be selected, regardless of their interests in the uploaded items. The optimal bandwidth-based placement is still obtained from Eq.(3.10) by changing the benefit definition into $w^{(k)}_{(U_f)} = C_{hi}$.

Concerning the heuristics, we considered three versions of GPA:

- GPA-j, which corresponds to the definitions outlined for Algorithm 1;
- GPA-b, where the benefit of a friend U_f on GW_h just depends on the uplink bandwidth from GW_h to GW_i (where U_j is located) , i.e., $w^{(k)}_{U_f} = C_{hi}$;
- GPA-r, where elements in Z_j are randomly sorted. So users randomly choose which friends to cache the content item, as long as enough quota is available, not considering any other factors.

Likewise, we evaluated two versions of RPA, RPA-ns and RPA-s differing by the sorting of B_i. The former corresponds to the definitions outlined for Algorithm 2. In the latter, the benefit is defined just as the sum of interests of the associated users who are also friends of the content owner, disregarding the uplink bandwidth to the owner gateway; This behaviour is “selfish”, hence RPA-s, because the receiving gateway only tries to maximize the interest of its own users.

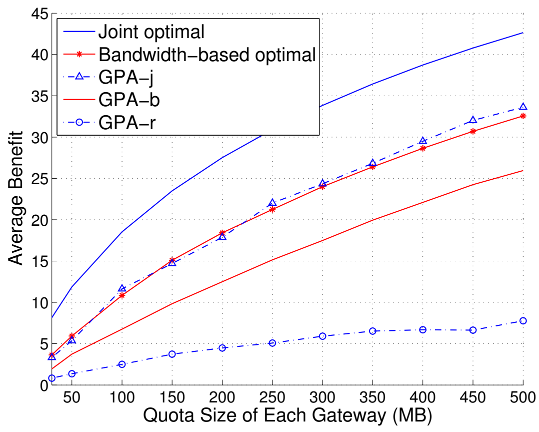
Another variant, which affects the GPA procedure, concerns the size of items to cache. At first, we assumed that all items have the same size. While not realistic per se, it could be meaningful if the implementation of the caching system were limited to a single class of items. In this case, tagged as “fixed size” in our results, we assumed all items to have a 10 MB size. Then, we considered items of any possible size within a bound. Such “variable size” case features random item sizes following a truncated exponential distribution with expected value of 10 MB and maximum size of 50 MB. While studying the latter case, though, we soon found out that allocation results were dangerously biased toward smaller items, so we introduced a *fair item size balancing* mechanism in GPA. Items were divided into size groups of 10 MB each and GPA was modified so that a caching request for one item was sent only if the total amount of data already cached in the item group did not exceed the total amount already cached for items of the first size group bigger than it. For instance, if a 15 MB item in the 10-20 MB size group increased the total amount of data already cached for that size group to 95 MB, and the total amount of the 20-30 MB size group were still 90MB, the request would be put on hold.

The previous variants are for the backup purpose, and the main change for the sharing purpose is just the change on the definition of the user benefit, which is defined in Chapter 3 already. However, two variant methods are not evaluated for the sharing purpose. One is the *bandwidth-based* methods (including *bandwidth-based* optimization method and GPA-b) which just consider the bandwidth resources, so there is no change in the user benefit definition for sharing purpose. The other missing one is RPA-s (“selfish” version), that’s due to in the case for sharing purpose, the residential gateways play in the collaborative way.

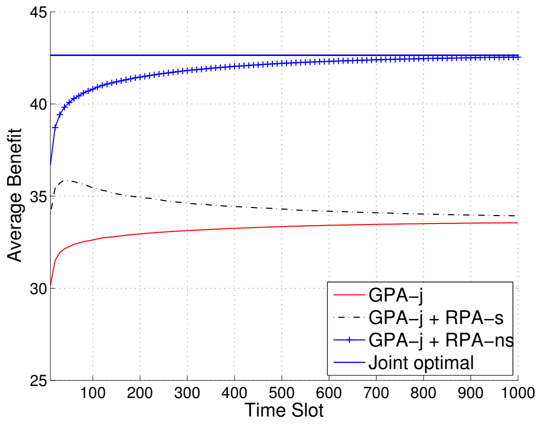
5.2.2 Performance Evaluation

Our first set of plots aims at comparing the optimization results, in their variants, with the corresponding variants of the distributed heuristics in which the GPA algorithm alone in employed for the backup purpose. The rationale of such comparison is to show the importance of the replacement management introduced by RPA (which is not used in these first results). For reason of space, we cannot show the whole possible parameter space, so we will just focus on a few representative cases to prove our point. Throughout the section, the number of gateways is $N = 1000$, the number of users is $M = 3000$ and the number of item types is $L = 10$. Results with $L = 50$ and $L = 100$ were qualitatively similar.

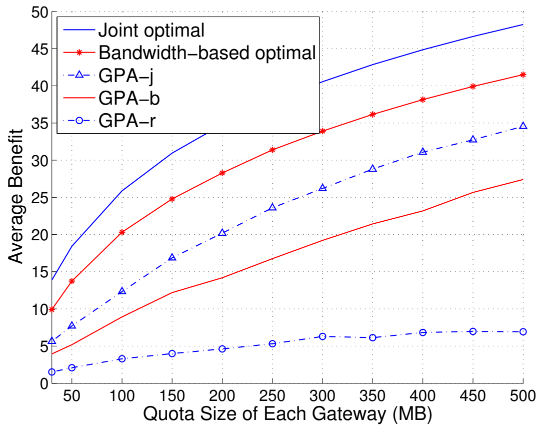
The plot in Fig. 5.6a clearly ranks the optimization and heuristics variants in



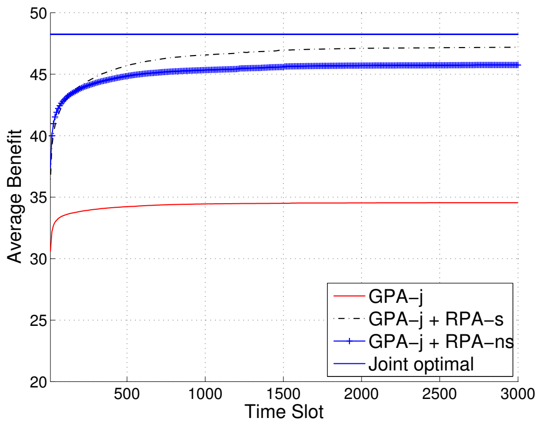
(a) Variable quota, fixed item size.



(b) Q_i=500MB, fixed item size.



(c) Variable quota, variable item size.

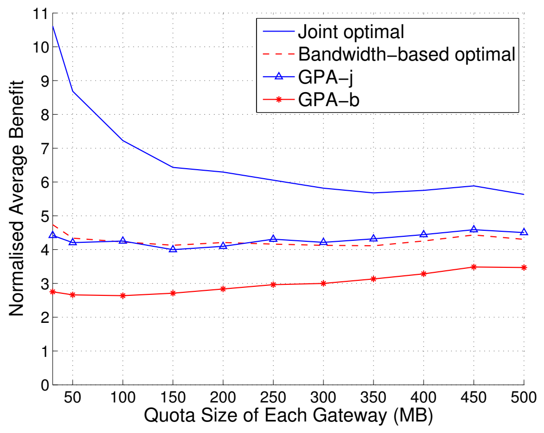


(d) Q_i=500MB, variable item size.

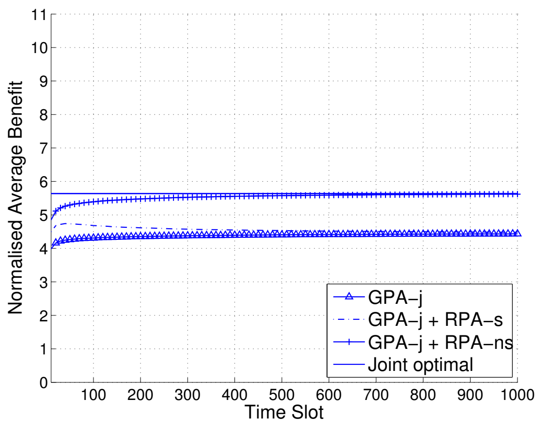
Figure 5.6: Average user benefit obtained by different methods for backup purpose under different cases.

terms of *average user benefit* as defined in Eq.(3.8) for backup purpose, for various quota constraints and fixed item size. The average benefit is the average value of user benefit which can be obtained by each user from the current used content placement method. It shows that jointly optimizing bandwidth and benefit is a clear winner. Even though the average benefit is low (due to that an average of three users per gateway sharing the same quota), the advantage of finding an optimal allocation for cached content depending on interest and bandwidth is clearly visible. Additionally, GPA heuristics alone do not match the joint optimization results, as expected. Similar conclusions can be drawn for the variable item size case in Fig. 5.6c, where GPA-j fares even worse.

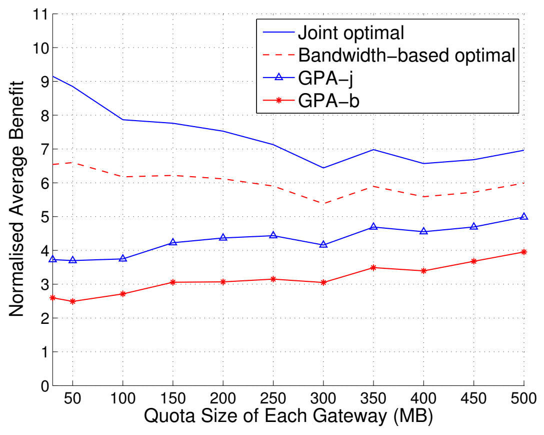
In the next set of results, we let receiving gateways run the replacement algorithm, RPA, in its two (selfish and non-selfish) versions. Fig. 5.6b plays out the $Q = 500MB$ quota case over time, in the same scenarios just examined. It shows that, given some time to converge, GPA-j with RPA-ns together yield an allocation that progressively corrects the initial uneven caching distribution provided by the distributed implementation of GPA-j alone. The selfish version of RPA, RPA-s, instead shows that if the owner and the storing gateway are not on the same page, as it were, when deciding what items are preferable to cache, the performance reverts to that of GPA-j alone. The selfishness of RPA, however, seems to have a lesser impact in the variable item size case, shown in Fig. 5.6d.



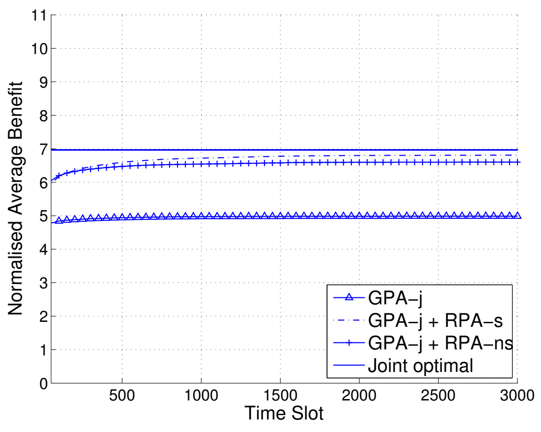
(a) Variable quota, fixed item size.



(b) Q_i=500MB, fixed item size.



(c) Variable quota, variable item size.



(d) Q_i=500MB, variable item size.

Figure 5.7: Normalized average user benefit obtained by different methods for backup purpose under different cases.

Furthermore, we normalized the average user benefit based on the value of average user benefit obtained by GPA-r, which is the random placement method, to emphasize the difference among these placement or replacement strategies under the same scenario settings. Fig. 5.7 indicates the same basic ideas with the Fig. 5.6. Also we can infer that for both cases of fixed and variable content item size, the difference between the optimization and heuristics would be enlarged, when the quota size on each friend gateway is decreased which means when the storage capacity becoming the critical resource. And GPA-j with RPA heuristics can progressively bridge the gap with the joint optimal method, shown in Fig. 5.7b and 5.7d.

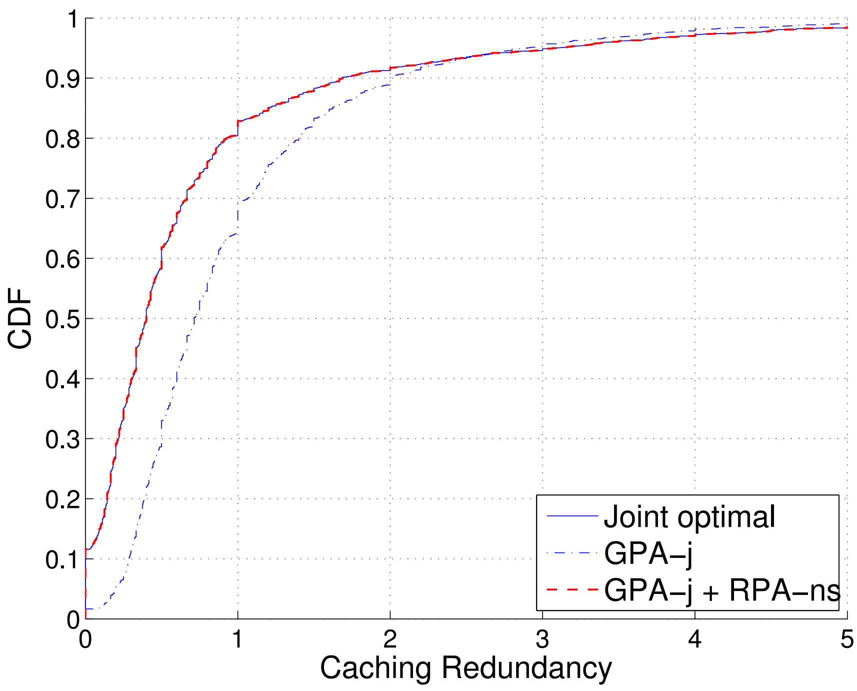


Figure 5.8: CDF of caching redundancy for backup purpose. (fixed item size and $Q_i = 500$MB)

We next plot the *caching redundancy*, i.e. the average number of a user's own items that have been replicated onto its friend gateways, upon reaching convergence of the distributed heuristics. Caching redundancy for one user is defined to be the ratio between the total number of replicas for all the items belonging to that user and the number of remote friend gateways. The CDF of the caching redundancy is useful to understand the thoroughness of the caching process. In the plot of Fig. 5.8, RPA-ns allows the remote gateway to replace the items (all of the same size) with smaller benefit and improve the storage thoroughness as much as the optimal method (the two curve indeed overlap).

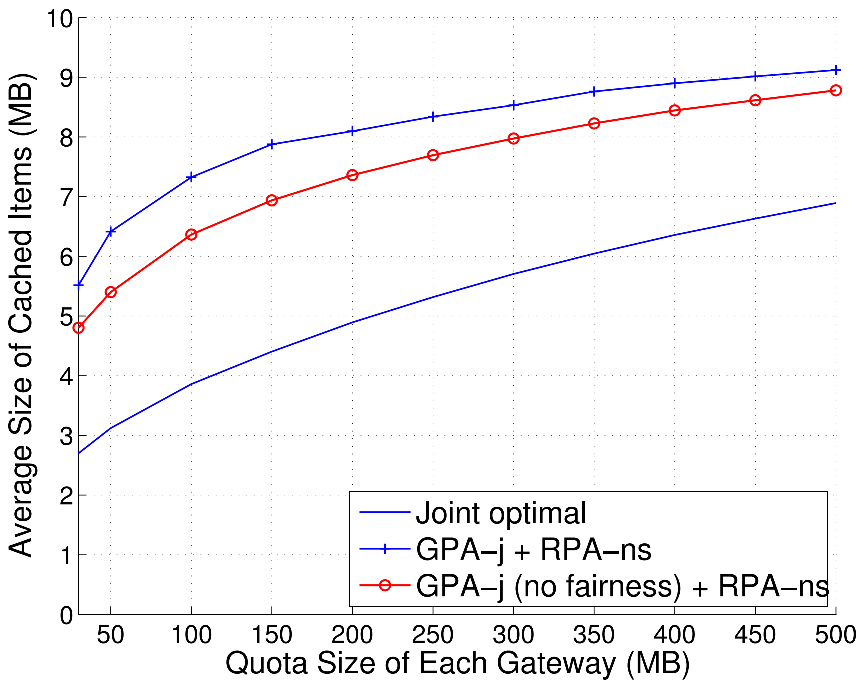
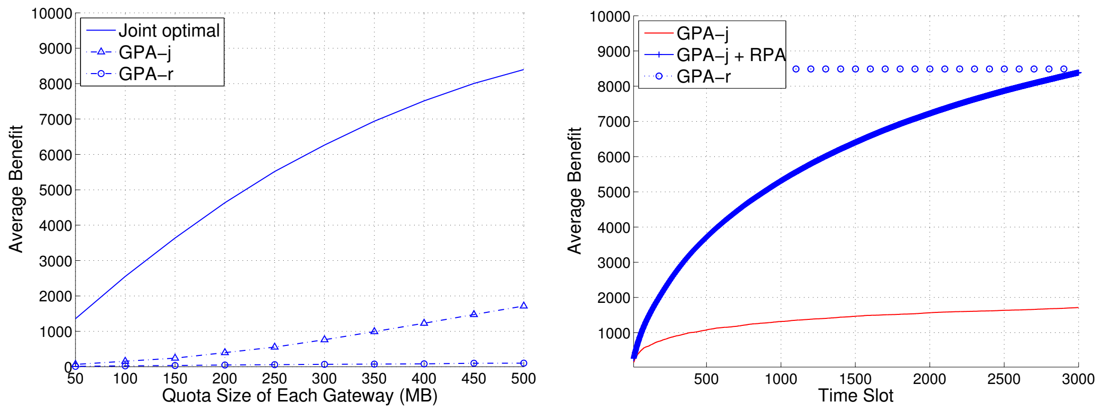


Figure 5.9: Average cached item size as a function of gateway quota for backup purpose. (variable item size)

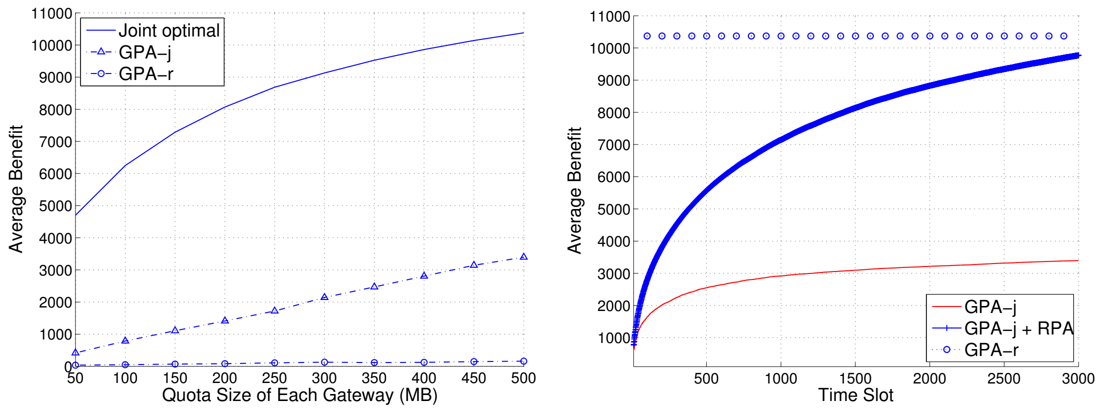
Finally, the plot of Fig. 5.9 reports the average size of items cached in the “friend” storage quota at remote gateways for backup purpose, when convergence is reached. Recalling that the average item size in the variable item case is 10MB, we can conclude that the loss of average user benefit with respect to the optimal case shown in Fig. 5.6d is offset by a fairer distribution of item sizes in the gateway storage across the federated network (i.e., the average size of cached item achieved by GPA-j with RPA-ns is just 10% smaller than the average item size in the system, as opposed to 30% smaller in the joint optimization case). Also, the use of fair item size balancing with GPA proves of some consequence to achieve this result.

Next, we turn our eye in the cases for the sharing purpose: the friend gateway caches the content from the friends not only for the backup of owner’s original copy, but also for providing another sources shared with the content owner’s other friends. Similarly, in terms of *average user benefit* as defined in Eq.(3.9) for sharing purpose, the plot in Fig. 5.10a compares the joint opimization and GPA heuristics alone variants, under various quota constraints and fixed item size. The jointly optimizing bandwidth and benefit is also a winner. Additionally, the gap between GPA heuristics alone and the joint optimization results is even larger in the sharing purpose than in the backup purpose, as it indicates the mistake in choosing the candidate friend gateway for caching would be magnified by the circle of friends for



(a) Variable quota, fixed item size.

(b) Q_i=500MB, fixed item size.



(c) Variable quota, variable item size.

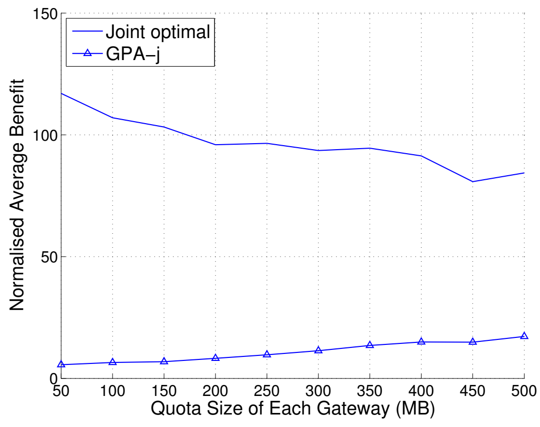
(d) Q_i=500MB, variable item size.

Figure 5.10: Average user benefit obtained by different methods for sharing purpose under different cases.

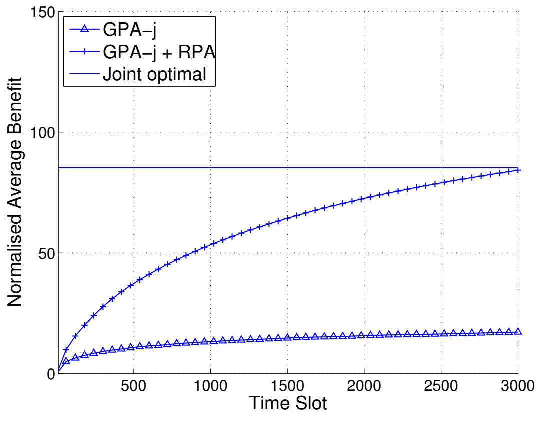
sharing purpose, in which the objective is to improve the benefit or utilities for all the friends of the content owner, not just the owner itself. Similar conclusions can be drawn for the variable item size case in Fig. 5.10c.

Moreover, we let receiving gateways run the replacement algorithm, RPA, only in the non-selfish mode, due to in the sharing case all the gateways run in the collaborative way. Fig. 5.10b plays out the $Q = 500MB$ quota case over time, in the same scenarios just examined. It shows that, given some time to converge, GPA-j with RPA together yield an allocation that progressively corrects the initial uneven caching distribution provided by the distributed implementation of GPA-j alone, under both cases of fixed and variable content size, which is shown in Fig. 5.10d.

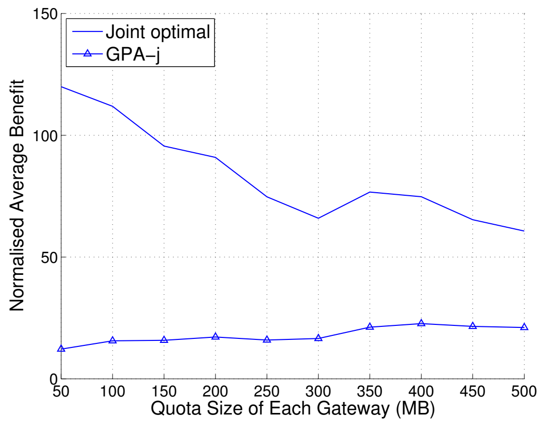
Due to the user benefit definition for sharing purpose considers not only the



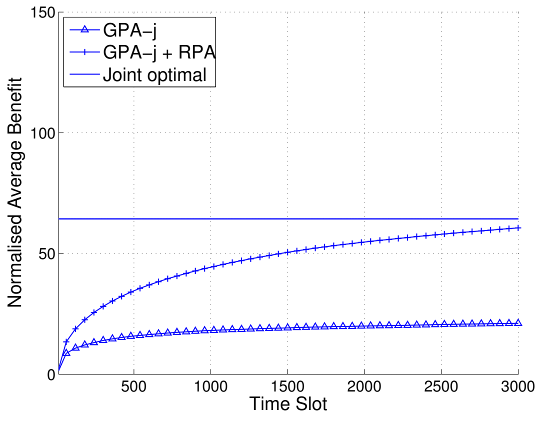
(a) Variable quota, fixed item size.



(b) Q_i=500MB, fixed item size.



(c) Variable quota, variable item size.



(d) Q_i=500MB, variable item size.

Figure 5.11: Normalized average user benefit obtained by different methods for sharing purpose under different cases.

content owner's interest and the bandwidth back to the owner, but also the interests of the owner's other friends and the bandwith towards them. The absolute value of average user benefit is much larger than for the backup purpose, shown in Fig. 5.10a. We also normalized the average user benefit based on the value of average user benefit obtained by GPA-r to emphasize the difference among these placement or replacement strategies under the same scenario settings. Fig. 5.11 indicates the same basic ideas with the Fig. 5.10. And GPA-j with RPA heuristics can progressively bridge the gap with the joint optimal method, shown in Fig. 5.11b and 5.11d, similarly with the backup purpose. So we can infer that GPA-j with RPA heuristics can progressively converge to the optimial solution under various friend quota size and any cases of variable content size, for both backup and sharing purpose.

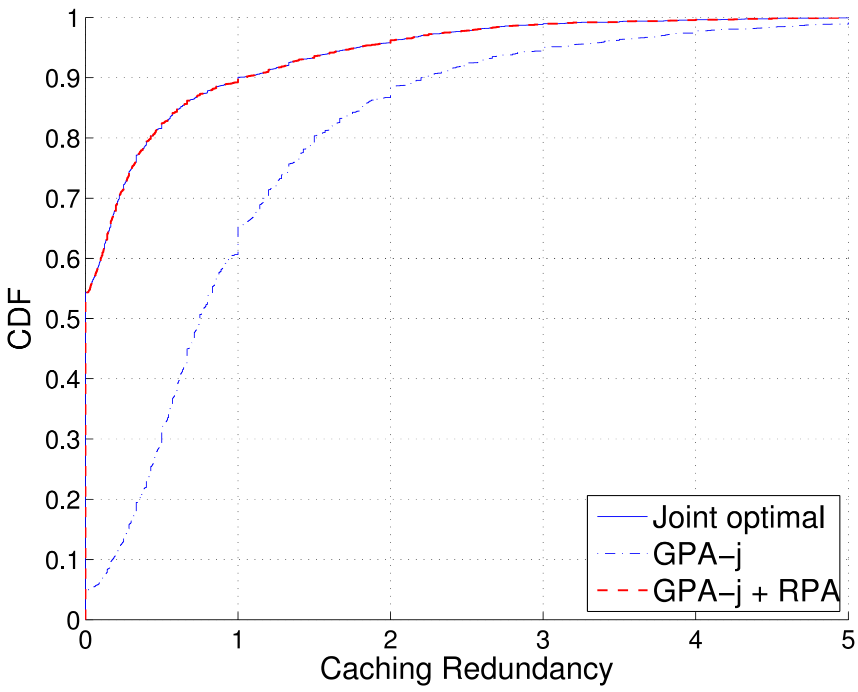


Figure 5.12: CDF of caching redundancy for sharing purpose. (fixed item size and $Q_i = 500\text{MB}$)

We next plot the *caching redundancy*, i.e. the average number of a user's own items that have been replicated onto its friend gateways, upon reaching convergence of the distributed heuristics, for the sharing purpose. The notable difference between the backup purpose and sharing case is, for backup purpose, the cache content item could only be accessible by the content owner. Otherwise, for sharing purpose, the cached content item would be shared among the circle of owner's friends. So the cached content might be quite popular among the friends, or plenty of friends might hold high value of interest in the content. In the plot of Fig. 5.12, RPA allows the remote gateway to replace the items (all of the same size) with smaller benefit and improve the storage thoroughness as much as the optimal method (the two curve indeed overlap). We also can see there are about half of users have no piece of item cached on their friend gateways, that is due to the realistic social graph used in the simulation scenario. With regarding to the CDF of friend degree in social graph shown in Fig. 5.1, about 50% of users have only less than around 13 friends, and 10% of users can have one copy cached for each own content item on average (shown in Fig. 5.12) because they have more than 50 friends.

As for the average cached item size for sharing purpose, the plot of Fig. 5.13

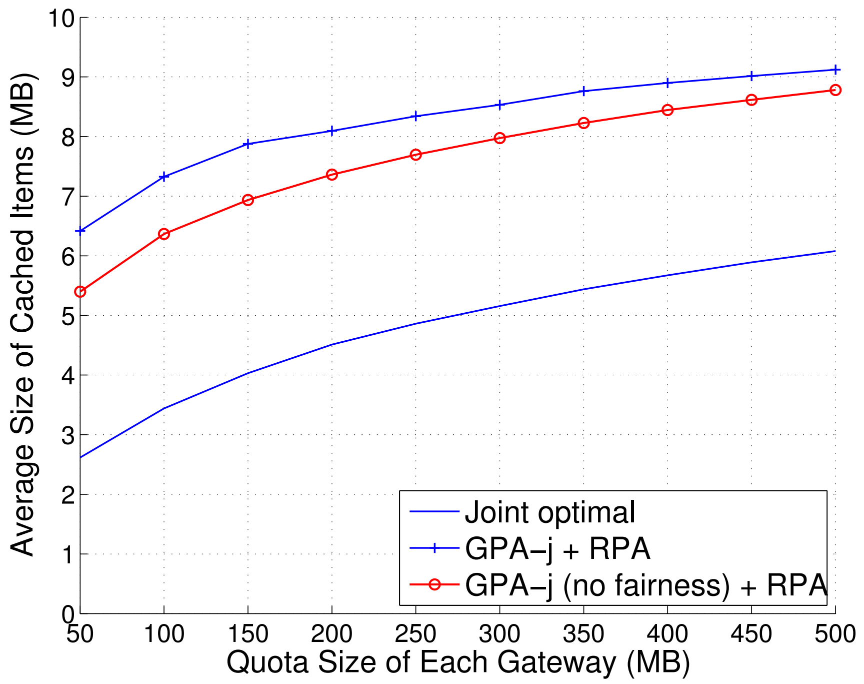


Figure 5.13: Average cached item size as a function of gateway quota for sharing purpose. (variable item size)

reports the similar results within the Fig. 5.9, achieved by GPA-j (selfish and non-selfish version) with RPA heuristics.

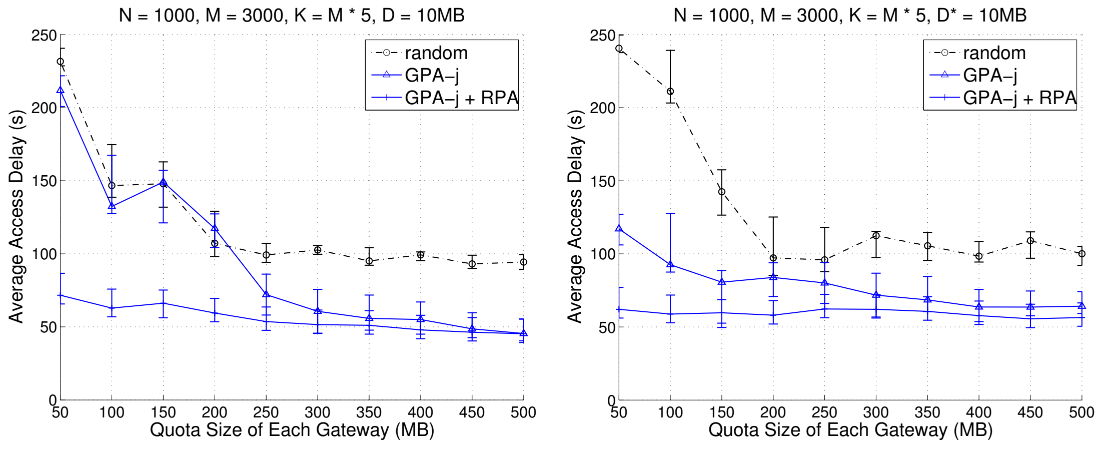
Especially, for sharing purpose we tend to study the impact of caching techniques adopted on residence gateways on the performance of content retrieval for the home users.

Firstly we need to simulate the content retrieval procedure for the home users: For each simulation run, first we start the content (re)placement algorithm for caching user's content on their friends' gateways; around the halfway (nearly to 3000s) we randomly pick up 100 users starting to retrieve/access the content owned by their friends till the end; they send the request to the geographically nearest replica gateway and the access rate to one certain type of content is according to the interest vector of the user itself.

Then we collect the statistics on the content retrieval duration, like the content access delay, hops, geographical distance and the consumed network bandwidth etc, when the simulation stops.

Finally we can compare the impact of three different caching algorithms (GPA-j heuristic alone, GPA-j with RPA heuristic and random placement) on the content retrieval performance. In the following are some interesting findings:

From the Fig. 5.14 showing the average access delay, which is defined as the



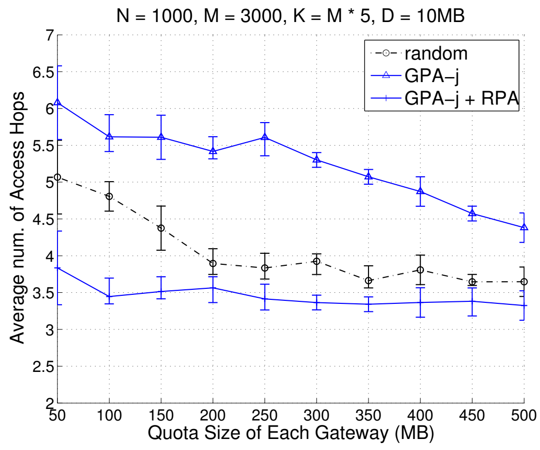
(a) fixed item size (b) variable item size

Figure 5.14: The comparison of content access performance under different content (re)placement strategies. (in terms of average access delay)

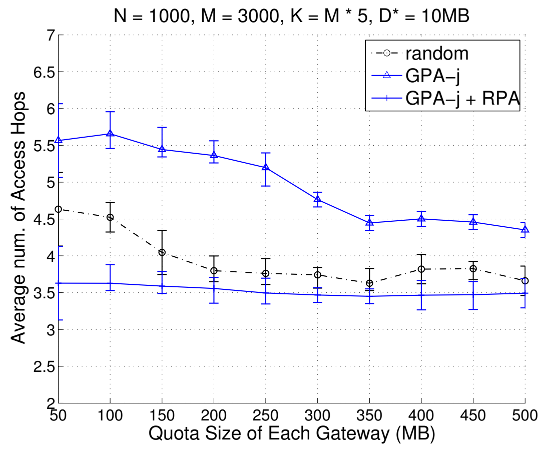
round-trip time starting from sending the content retrieval access to receiving the whole content successfully, under the fixed and variable item size cases both, we can infer that the GPA-j with RPA heuristic method defeats the other twos undoubtedly, the access delay is stable even when the quota size is extremely limited (50M case). The interesting point for random placement algorithm, is the access delay will turn to be stable after the quota size increasing up to 200MB. That is due to the storage resource is not the bottleneck anymore for the content retrieval performance when the friend quota size is larger than 200MB. And when the quota size is not a bottleneck anymore, the performance of GPA-j heuristic alone method can dramatically approximate to the replacement method.

In terms of hops (Fig. 5.15) or geographical distance (Fig. 5.16), which is defined as the network link hops or geographical distance between the requesting residential gateway and the replica gateway respectively, the user is always able to find the appropriate replica inside of the same city or even the local residence/federation (3-4hops) when using GPA-j with RPA heuristic method. Much larger number of hops or distance is required when using GPA-j heuristic alone than the random placement method. That's due to, according to the objectives of GPA-j, it just pick up the friend gateway with the higher bandwidth and the higher interest. On the other hand, the random placement method will reflect the distribution of the friendship probability on the geographical distance (most of friends are close to each other geographically), which is used in our realistic simulation scenario.

Previously we measure the content performance from the side of content requesting gateway; now we take the side of content replica gateway to assess how

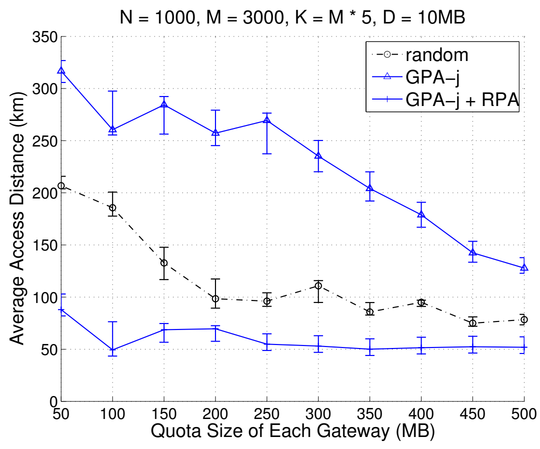


(a) fixed item size

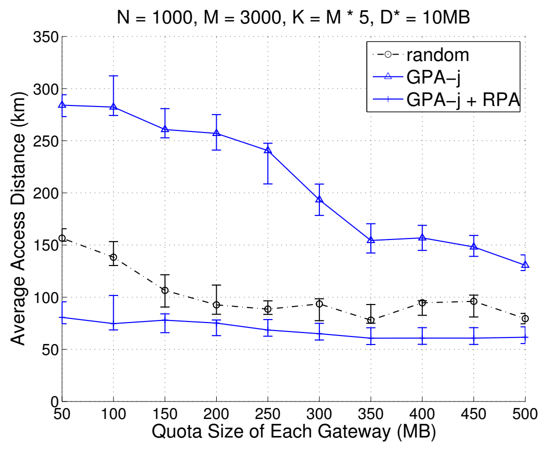


(b) variable item size

Figure 5.15: The comparison of content access performance under different content (re)placement strategies. (in terms of average number of access hops)



(a) fixed item size



(b) variable item size

Figure 5.16: The comparison of content access performance under different content (re)placement strategies. (in terms of average access distance)

the caching techniques adopted on residential gateways could improve the content retrieval performance. Fig. 5.17 depicts the difference in the percentage of content retrieval request could be met by the replica gateway inside of the local neighbourhood/federation under the random placement and GPA-j with RPA heuristic method. Using the random placement for caching, around 40% of content requests could be met within the same city and half of requests which are met in the same

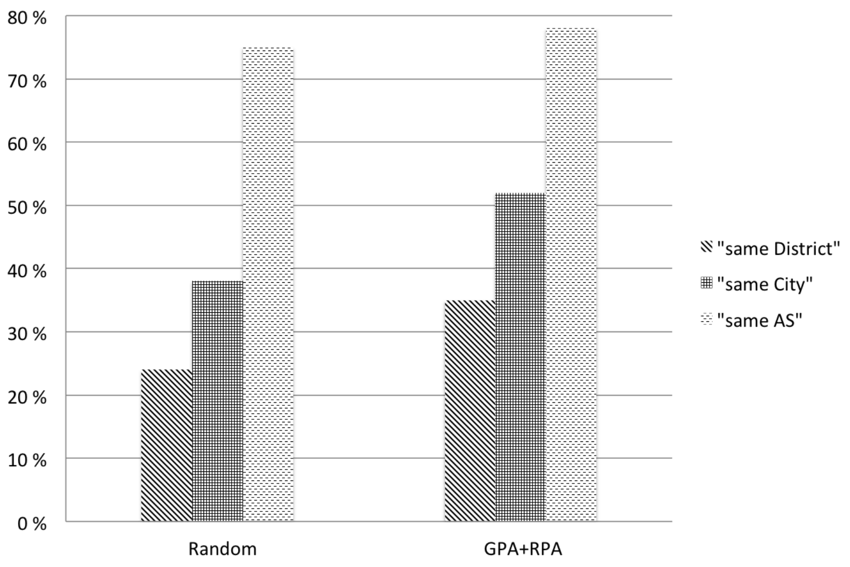


Figure 5.17: The percentage of retrieval request vs. distance.

city (about 20% in the absolute value) would be satisfied in the same district (which is simulated as the neighbourhood). This also reflects the distribution of the friendship probability on the geographical distance exactly. Under the GPA-j with RPA heuristic method, around half of requests can be satisfied within the same city and more than 30% could be met in the same district, that means using the GPA-j with RPA heuristic method, the content item could be cached on the residential gateways which are much closer to the potential interested home users. That's also the intention and value of our distributed heuristic caching algorithms.

Chapter 6

Implementation works

This chapter concerns the implementation and evaluation works to deploy the caching module on the real testbed environment. Firstly, the overview of gateway implementation architecture will be provided; followed by the design of a framework for caching module and the detailed description on the caching module implementation. Finally, we use Common Open Research Emulator (CORE) to emulate a realistic network environment for residential gateways to test and evaluate our content caching algorithms implementation.

6.1 Overview of gateway architecture

Firstly, we will provide the overview of gateway architecture providing home users (and external gateways) with a unified content management mechanism to access content, regardless of its location in the home network. The gateway content management, illustrated in Fig. 6.1, includes the following components:

- The unified content management view, mapping home network content into a unified view of representation.
- The DAM achieves access unification at content level, abstracting the location of the content.
- The caching module performs content caching either on local resources or on external, federated resources, leveraging the home users' social information.

And then proceed to describe each of them in details.

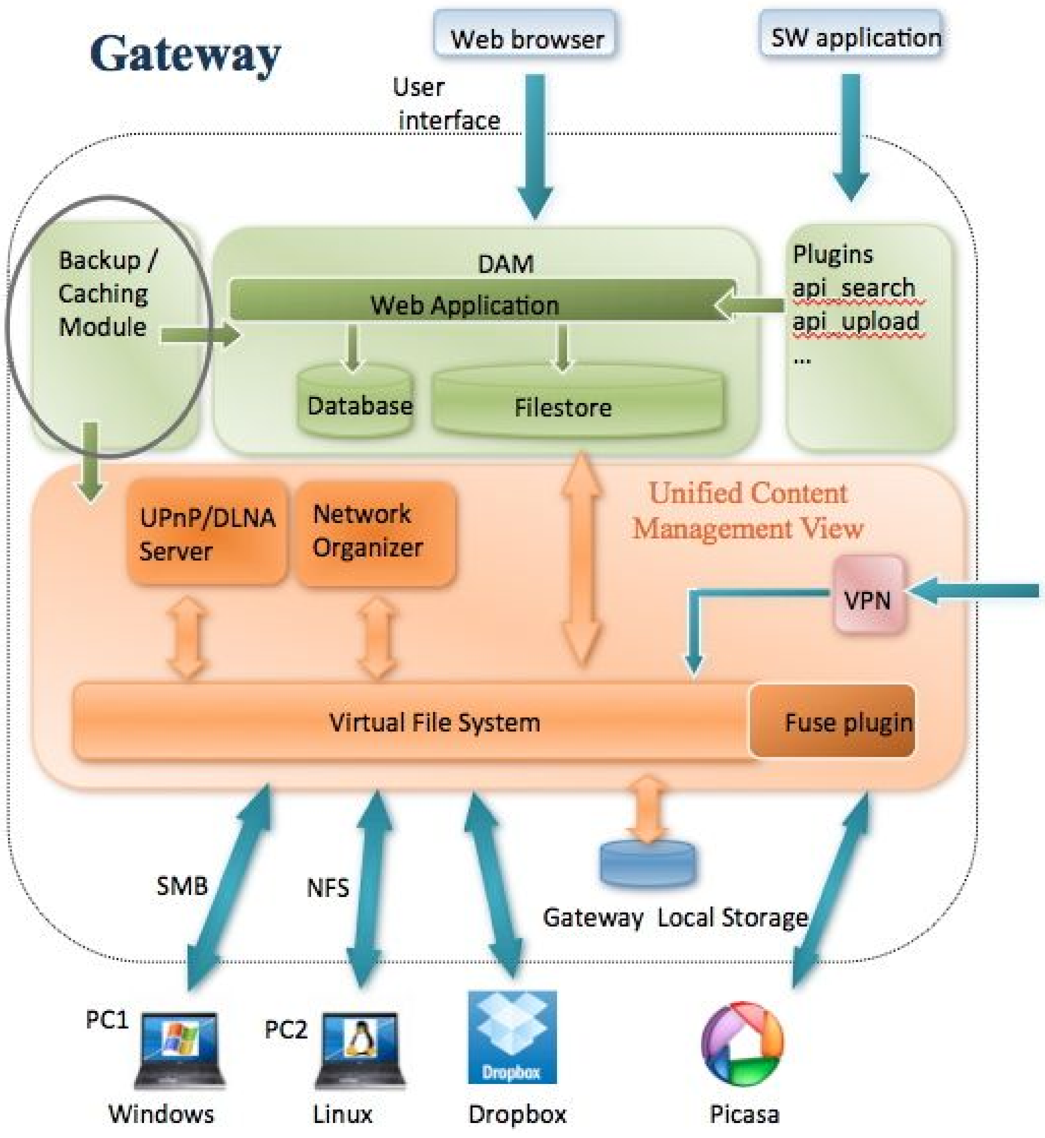


Figure 6.1: Architecture of the content management in the gateway.

6.1.1 Unified content management view

The unified content management view unifies the file systems and the network protocols. It offers a file system view of the home network. It is based on the Virtual File System (VFS) provided by the Linux operating system. It therefore represents the unified content management view of the residential network content as if it were a local file system interface. It also contains a network organizer module, which is in charge of managing the registration/mounting/unmounting operations and detecting which device leaves the residential network. The unified content management view mainly provides local file systems access. As such, mainly local devices (i.e., running on the same residential network), and properly configured via the network organizer can access the gateway at the network file system level. Remote access could also be provided at the same level using a VPN (Virtual Private Network) connection.

The unified content management view layer also includes a UPnP/DLNA media server, running on the content management view to expose all the residential network content towards UPnP control points and media renderers.

6.1.2 DAM: Digital asset management

The DAM is an application, running on the top of the unified content management view. It allows task such as content indexing and retrieval. It is based on a LAMP (Linux Apache MySQL PHP) software package [62]. It uses the file system interface provided by the VFS layer. Each resource has a unified resource ID, which is allocated by the DAM when a new resource is imported. It contains a local storage area, called *filestore* and a metadata database MySQL:

- Filestore: a directory tree containing the uploaded content. A file is stored (or a symbolic link is created) and renamed resource ID with file extension in the filestore. For example, picture01.jpg is stored in filestore as 152.jpg. These folders contain also icon and preview of the pictures.

- Database: the database contains metadata related to resource ID: file type (mp3 or jpg ...), tags like id3 or exif tags, the keywords related to the resource that can be used to find the resource.

The DAM also provides a set of API allowing control by remote software applications. and the DAM provides 2 types of access interfaces: the user interface and the software interface, which are described hereby.

User interface

The user has access to the system via a browser. Two types of front-ends are available in the DAM: one for PCs or laptops and one for smartphones with touch screens. The user access is protected via HTPP authentication. Therefore, the user needs to be registered in a DAM or gateway so as to access it. Once authenticated, the user can access the DAM either locally or remotely.

Software application interface

The DAM can be interfaced to another application using a HTTP interface. Assuming the application is properly authenticated, it can be interfaced to the DAM whether it runs locally on the same gateway. The application can reside on a remote device like PC or smartphone.

6.1.3 Caching module

The caching module interacts with the DAM in order to create copies of home content following users' directions and interests. This module will be in charge of disseminating multiple copies of the same content among neighbouring gateways for the purpose of having the content readily available when needed (though not in a reliable way, i.e., the content may need to be downloaded from other sources if not found in the caching system). Caching module can exploit the unified content access in order to manage all the resources in a transparent way, saving into the cached content all the information about user-generated and file-system-level metadata along with the resource themselves. It also query the DAM database to retrieve the user-generated tags associated to each resource, to be stored and saved as metadata. These metadata will then allow the selection of appropriate destination for the cached content (e.g., DAM-level user-generated tags such as "wildlife pictures" could trigger the caching of users' photos toward storage facilities shared by federated users with the similar interests).

This is the socially-aware caching mechanism which leverages the interaction between remote residential gateways in a social way, i.e., by exploiting the users' social networking information, so that caching recipients are those gateways whose users are most likely to be interested in accessing the shared content, as explained in [5].

6.2 Architecture of caching module

This section concerns the design of a framework for caching module. Before we start talking about architecture behind this module the entities that are part of the entire framework is to be introduced, and then followed by the detailed description of the caching module architecture.

The caching module operates on the top of the DAM layer as explained previously and is composed of a caching decision algorithm module which runs a background caching procedure by uploading selected content owned by home users to selected remote friend gateways, and a social information-gathering module.

In the caching decision module, a low-complexity, distributed heuristic algorithm will be included, to match remote friends' interests with the content to cache by leveraging typical social networks indicators, such as interests, hobbies and preferences of home users, and by having all personal digital data appropriately tagged.

The module takes two different viewpoints: that of content owners who have data to share or backup and that of remote gateways which provides their own storage space for their social friends. From the viewpoint of content owners, not only do they wish to cache or restore the data as fast as possible, but in the long run, also wish that the remote gateway keeps the content for as long as possible. Therefore, content owner are naturally disposed to choose friends from whose gateways they can retrieve the cached content more quickly. Also, they would like friends to be interested in the content they upload, because such friends are more inclined to store it for a long time. At the receiving end, the remote gateway will devote its storage space for friends to maximize the whole benefit accounting for both its users' interest and the bandwidth toward the content owner's gateway.

Although the caching decision algorithm module is running in the background, it still requires the home users to tag which content needs to be cached through the DAM interface. The module must also be able to gather the knowledge of content type tags by the DAM APIs.

The other module used by the socially-aware caching functions is the social information-gathering module, which allows the gateway to interact with social networks. Clearly, given the specific characteristics of each social network, more than one module will have to be designed and implemented so as to be tailored to online social networks.

Also, the socially-aware caching functionalities can be exploited by the backup module in the maintenance step of the process. Cached replicas of a specific content can be used for generating new parity blocks in case the owner of the content (i.e., the backup module running on the content owner's gateway) is offline, and, at the same time, the number of parity blocks decreases below a lower bound.

6.2.1 Entities and terminology

The entities of the architecture related with caching module are summarized with what names they are referred to as following:

- Lookup service (LS): a central logical entity providing global information lookup features. The lookup service acts as a database server and a resource manager. It stores authenticated users' credentials, the mapping between authenticated users and gateways. It has a comprehensive knowledge on the federation, and interfaces to the gateways' monitoring modules fro recording availability and bandwidth statistics and providing thus detailed information to caching module. Furthermore, it has an active part in knowing which gateways are online in a given moment, by polling them at regular intervals. Also, for the caching service, it will keep track of the placement of each content replica in the federation as well as the mapping between registered Figaro users and social network users.

- Device: each connected and registered device on a local gateway (e.g., laptops, smartphones ...).
- DAM: the Digital Asset Management. This is the main web application, running on each gateway and providing browsing functionalities of the currently available content in a residential network.
- Home storage: the gateway's storage exploited for storing cached copies (e.g., a network attached storage - NAS). We call it "home" as it is relative to a single home network.
- Gateway: a node in the federation serving a single home environment.
- Federation: a set of gateways organized in some way (e.g., neighbourhood, friendship ...).

6.2.2 Caching module and service

One of the drawbacks of personal or cloud backup approaches is the fact that data of potential interest of other users sits unused in a storage device. Let us consider the following example. George has a set of pictures of the latest family vacations and he wants to show them to his friend John, while, at the same time backing them up. George remotely uploads the pictures to John's NAS, where a storage quota is reserved for such purpose; John is then notified that a copy of the pictures now exists in his NAS and that he is welcome to have a look, while keeping it in its NAS as a backup. For fairness, a similar quota for John's backups should be set aside at George's. The example could be extended to a close group of friends, as defined within social networks, and the potential of such a scheme instantly become apparent. By leveraging typical social networks indicators, such as interests, hobbies and preferences, and by having all personal digital data appropriately tagged (at the DAM level), the matching of remote users and content to cache allows achieving social content sharing.

To efficiently speed up the access to user-generated content for end users in the federated residential network, the caching service provides the home gateway with information on what content to cache and what to discard, and for how long, based on inter-gateway network bandwidth measurement and user interest information learned and estimated from online social networks (OSNs).

The caching service is provided by the caching module, which operates on the top of the DAM level and runs a background caching procedure in charge of:

- Managing the home storage reserved for the caching service.
- Gathering the user's social information from OSNs.
- Processing and extracting metadata associated to resources on caching.
- Taking the cache placement and replacement decision.
- Managing retrieval of content cached on neighbouring gateways.

Assisting the caching module is the Social Data Manager Unit (SDMU), a helper module that could be deployed on a single, common server outside the residential network (e.g., one per neighbourhood, or one per provider). The SDMU must have a public address in order to interact with OSNs and does not permanently store single user information.

Decomposed caching module architecture

We now provide a description of the internal functions of the caching module. Fig. 6.2 shows the building blocks of the caching module.

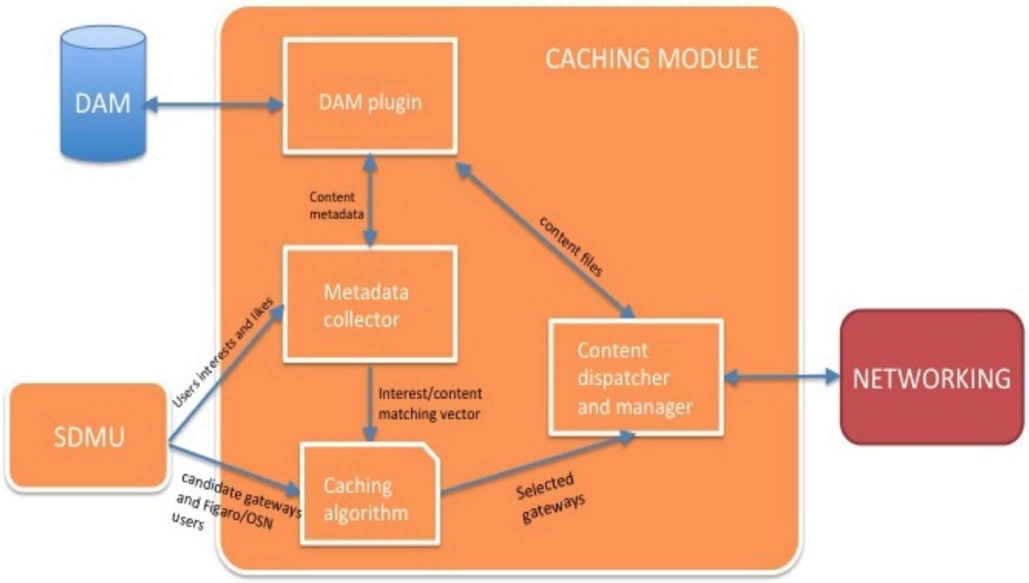

CACHING MODULE
DAM
DAM plugin
Content metadata
content files
Metadata collector
Users interests and likes
SDMU
Interest/content matching vector
Content dispatcher and manager
NETWORKING
candidate gateways and Figaro/OSN users
Caching algorithm
Selected gateways

Figure 6.2: Decomposed caching module architecture.

- *DAM plugin*: interacts with the DAM and provides access to content stored in the home environment; it also allows a user to tag content files scheduled for caching.

- *Metadata collector*: extracts metadata from content files and matches it according to a tree of predefined categories.
- *Caching algorithm*: ranks gateways in the preferred order to receive cached content (among a candidate list provided by the SDMU) according to a heuristics selection process that tries to maximize bandwidth availability and user interests, or "benefit" according to different purposes of user for backup or sharing.
- *Content dispatcher and manager*: manages signalling with remote gateways; transfers the content to the selected gateways; accepts/rejects content (to be cached locally) from remote gateways; tracks the location of cached content; dispatches locally-cached content information to requesting users (either local users or authorised neighbouring users).

Content caching procedure

The following steps, illustrated in shown in Fig. 6.3, are envisioned for the operation of the Caching module.

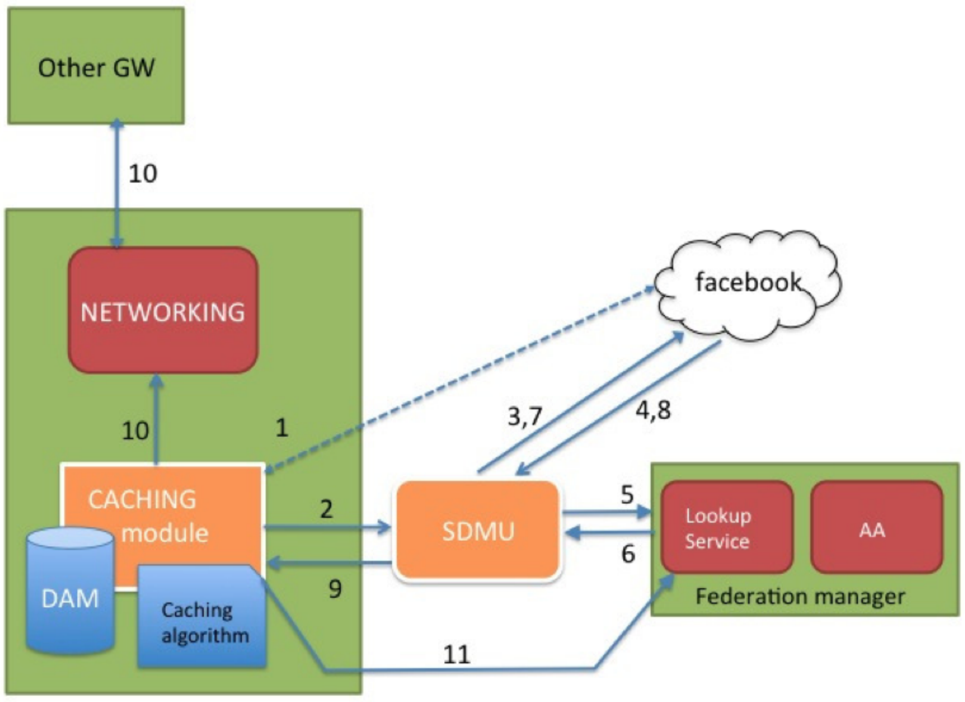


Figure 6.3: Content caching procedure.

- Step 1. Preliminaries.

  Establish the interface communication with an OSN (e.g., Facebook in the example).

- Step 2. Credentials access authorization.

  The caching module authorizes the Social Data Manager Unit (SDMU) to access the user's Facebook credentials and to send them to Facebook to access the user's profile.

- Step 3. Information request (1).

  The SDMU contracts Facebook and requests social information for the user (e.g., his/her interests, the posts that he/she "liked", the links he/she shared, as well as a list of his/her friends) via Graph API [63].

- Step 4. Information delivery.

  The SDMU receives the user's social info from Facebook.

- Step 5. Information parsing.

  The SDMU sends the list of OSN IDs of the user's friends to the Lookup Service (LS). Some interaction and authorization with the AA module are in order at this point.

- Step 6. Friend identification.

  The LS identifies the OSN IDs that corresponds to FIGARO users and returns to the SDMU, for each identified FIGARO/OSN friends: the IP address of the residential gateway where they reside and the bandwidth estimation information for such home gateway. It is assumed that the bandwidth information is either stored at the LS (through scheduled updates) or it is queried from the monitoring module of the involved gateways.

- Step 7. Information request (2).

  The SDMU contacts Facebook and requests social information for the FIGARO/OSN friends (a subset of the user's total friends).

- Step 8. Social data collection.

  Facebook returns the FIGARO/OSN friends info to the SDMU, which parses and formats it.

- Step 9. Response.

  The SDMU inputs to the caching algorithm module:

  - A list of candidate gateways and FIGARO/OSN users on those gateways, including their reachability information (IP address and uplink/downlink bandwidth estimation).
  - The interests (likes and links) of each FIGARO/OSN user identified.

- Step 10. Content placement in the federation.

  The caching module runs the caching algorithm detailed in Chapter 4 to determine where to place content replicas among the candidate gateways. Upon the selection of each gateway, the caching module interacts with the networking module to start the appropriate signalling with the remote gateway and request caching authorization. If accepted, the content will be transferred to the remote gateway, where it is stored in the remote caching storage partition. The remote DAM is triggered in order to index the new content.

- Step 11. LS update.

  If the transfer to the remote gateway is successful, the caching module reports to the LS the location (the gateway address) of the content just cached. The content is identified through a unique ID (e.g., an XML tag).

Content retrieval procedure

The cached content can be retrieved as show in Fig. 6.4. The user on GW1 starts the procedure by interacting with the caching module to identify which of his/her neighbours are caching content that is described by one or more keywords. The following steps are then executed:

- Step 1. Remote friend lookup.

  The caching module identifies the friend which caches content matching the keywords input by the user and asks the LS (or a local LS cache to speed up the process and relieve the burden on the LS proper) to provide the friend's gateway information. The content is identified through a unique ID.

- Step 2. Content locator request.

  Through the networking module, the remote gateway is contacted and a request for a specific content ID request is issued to its caching module.

- Step 3. Authorization verification.

  The remote caching module interacts with the AA module to establish whether the requesting gateway is authorised to receive content from it.

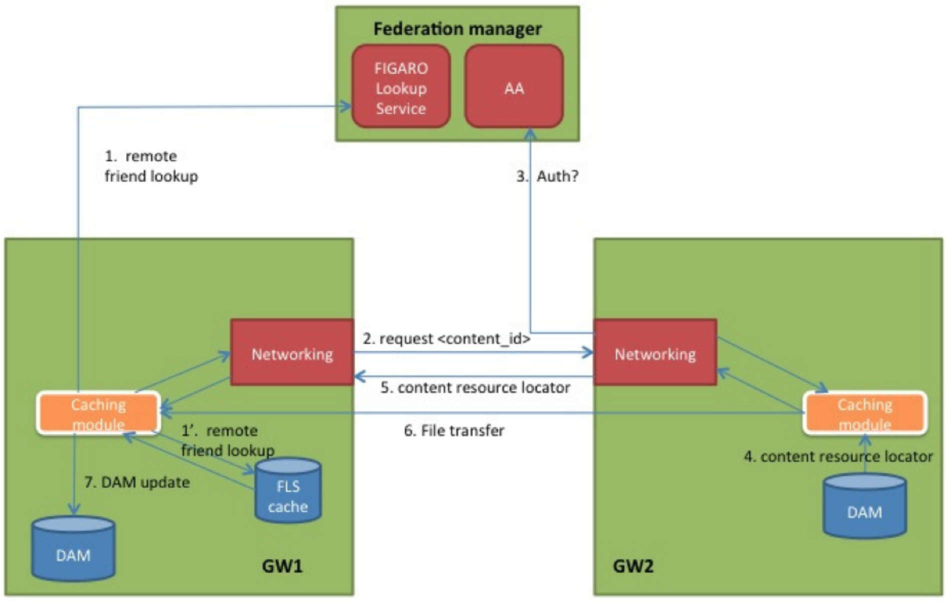


Figure 6.4: Content retrieval procedure.

- Step 4. Retrieval of content resource locator.

  The remote caching module identifies the content in its storage and retrieves the DAM handle (henceforth called “content resource locator”) to return to the requesting gateway.

- Step 5. Content locator response.

  The content resource locator is returned to the requesting caching module through the networking module.

- Step 6. Content downloading/file transfer.

  The caching module on GW1 starts a file transfer procedure to acquire a local copy of the content from GW2.

- Step 7. DAM update.

  Upon transfer completion, the caching module tells the DAM plugin to index the new content.

- Step 8. (optional) LS update

  Similarly to the content caching case in the previous paragraph, the caching module may choose (it will be implemented as user’s option) to inform the LS that a new copy is available and its location (GW1).

Local cache replacement

If remote gateways "passively" accepted all caching requests until their local storage is filled up, their collection of cached items would not match any optimum allocation, being strongly dependent on which "friend" started the caching procedure first, hence which requests they receive first in a greedy fashion. After all, the friends of the users associated to a gateway share the quota on this gateway by competing with each other.

In order to alleviate such unbalanced situation, a *replacement* algorithm is run by every gateway upon receiving a caching request and discovering that the quota is already filled up. In order to maximize the benefit of the users associated to the receiving gateway, the replacement strategy considers the removal of cached items with lower benefit for local users than the incoming item. Of course, if a gateway replaces a content item, a "cache-delete" message must be sent to inform the content owner that it needs to find a new gateway where to store such item.

6.3 Description of implementation

There are two well-defined stages in the execution of the caching module totally depicted in Fig. 6.5: the first phase is the preliminary configuration of the data structures modelling the initial placement of gateways, users, and related files, and the second one takes charge of the implementation of placement/replacement algorithms in client and server side both.

To start the caching module the information on the associated home users, the content items required to cache and the remote friends' gateway should be initialised at first. These interactive information should be obtained from the other modules continuously, like DAM plugin and federation lookup service etc. In this section focusing on the implementation of caching module alone, we simplify these information, which are obtained in advance, as inputed in the corresponding configuration files already. The first phase is to configure the current residential gateway through parsing the related information from the data structure formatted in the configuration files.

Configuration phase

As shown in Fig. 6.6, there are mainly two configuration files deployed on each gateway, which are usr.conf and lookup.conf. Next, these two configuration files will be introduced in details:

The first configuration file, usr.conf, describes the associated home users on the current gateway and the content items belonging to the corresponding user which are required to cache. This file is following a precise syntax to fill the data structures

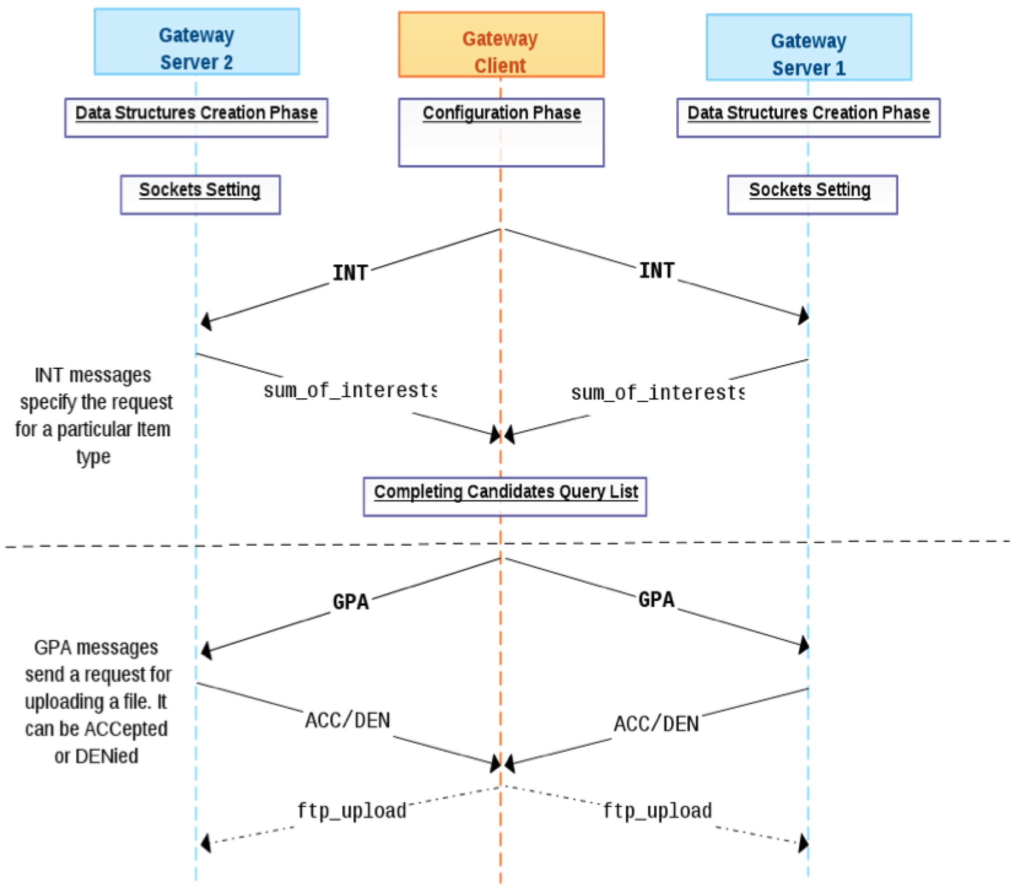

Gateway Server 2
Gateway Client
Gateway Server 1
Data Structures Creation Phase
Configuration Phase
Data Structures Creation Phase
Sockets Setting
Sockets Setting
INT
INT
INT messages specify the request for a particular Item type
sum_of_interests
sum_of_interests
Completing Candidates Query List
GPA
GPA
GPA messages send a request for uploading a file. It can be ACCepted or DENied
ACC/DEN
ACC/DEN
ftp_upload
ftp_upload

Figure 6.5: Temporal analysis of the two phases of configuration and caching request.

of the current residential gateway and respective home users. The first information specified is that the identification number of the gateway, followed by a description about each user, including its friends, interests and of course the set of items that the user confirms to cache.

The second file, lookup.conf, is used to acquire the location information of other remote friends in the federation. These are related to the friends of those home users present in the current gateway. In the real deployment, these information could be acquired from the lookup service, which is a DNS-like location service in the federation, when the multiple modules are incorporated together through interactive API. Moreover, the already-known location information of remote friends can be cached in this file. We could obtain the remote friend's associated gateway with IP address and port information through the lookup by the friend identifier.

After the parsing of the configuration files, using the information of user's friend list (from usr.conf file) and each friend's location (from lookup.conf file) we can

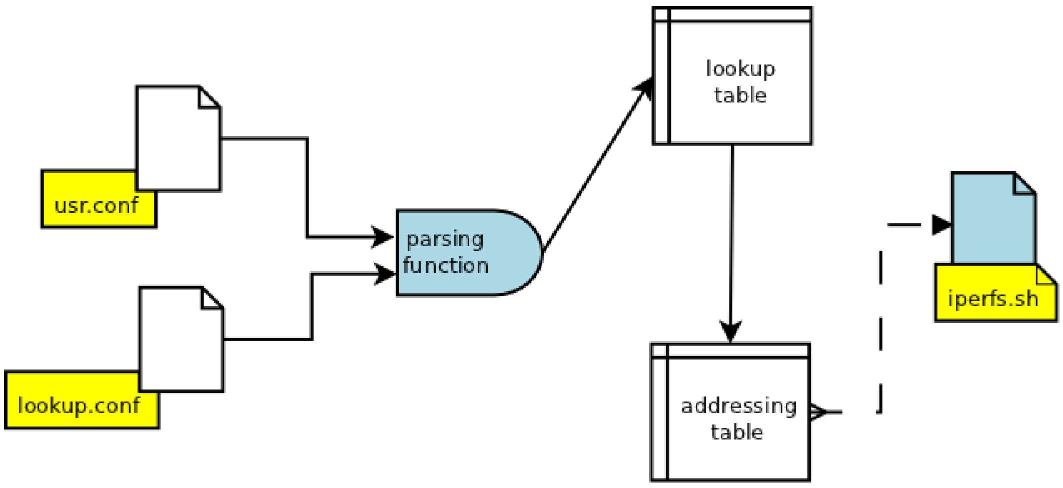


Figure 6.6: Configuration phrase on each gateway.

compute the list of remote friends for each local home user associated on the current gateway. The list of remote friends, including the basic information related with the associated gateway for caching the user's content potentially, would be stored in the table called as *lookup table*. In details, for each remote friend, the associated gateway identifier and the IP address with the port number for establishing network connection would be recorded in the pre-defined format.

Then the configuration procedure would continue to proceed to the formation of a new table that depends on the *lookup table*, named as *addressing table*. In the *lookup table*, it in fact includes all the remote friends in the federation, and the column of the associated gateway's identifier could have duplicates. That's due to the fact that multiple remote friends could possibly are associated on the same residential gateway. Moreover, the records with the same associated friend gateway identifier would be extracted from the *lookup table* and combined into one record. So for each local home user, the *addressing table* contains the list of remote friend gateways that would potentially provide the storage space for caching the content owned by the local users. Also the *addressing table* manages the network connections to the respective remote friend gateway.

However, the two important factors - network bandwidth and friend interest, on which our caching placement and replacement algorithms are based, are still missing. As for the network bandwidth metric, it can be acquired from the network monitoring module under FIGARO project. But here for simplicity, we use a famous bandwidth measuring tool, named *Iperf* [64], to discover the bandwidth between any two residential gateways. To obtain an average value for a better estimation of the available bandwidth, the client-side script iperfs.sh is performed multiple times to the specific address and port of a remote friend gateway. So the available bandwidth information of any specific remote friend gateway could be calculated before performing any type of request to that corresponding gateway. And the result of bandwidth information, which would be refreshed by calling the script iperfs.sh in a pre-defined frequency, is stored in the *addressing table* afterwards.

In order to get the interests of friends for each item, the current home gateway must request each remote friend gateway list in the *addressing table* for the total amount of the item owner's friends' interest in the corresponding item type.

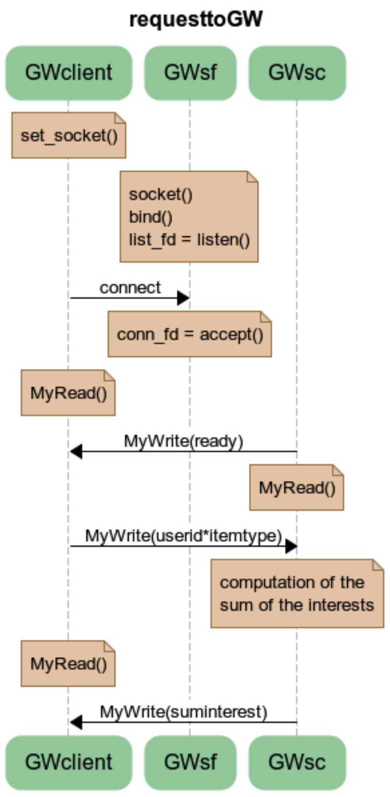


Figure 6.7: Representation of the request sending procedure and its implementation.

Each request to the remote gateway is done within the function requesttoGW() depicted in Fig. 6.7. The caching module on each residential gateway runs both two threads in two modes: client and server modes respectively. The request which is sent by the current gateway in client mode will be received by server thread. Once receiving the connection request, the server thread will create a child thread to serve it. In Fig. 6.7 we present an example of request for the interest value of remote friend gateway. We see the gateway making the request, GWclient, connects to the remote gateway GWsf, the main server thread, which redirects the request to the child GWsc, which responds directly to the client. In the request, the child server thread can identify two pieces of information: *userid* and *itemtype*. So the child thread will check all the local home users who are also the friends of the user with the specific *userid*, and return the sum of their interest value in the specific type of content item.

When the current residential gateway has made requests to all remote friend gateways' interest for each content item, a *query list* for each item could be reached to denote a sorted list of candidate remote friend gateways for caching the corresponding content item. Then the *query list* for all the items owned by the same user will be combined into one *query list* for that corresponding user. Furthermore, The *query list* is sorted by the benefit, based on the factors of network bandwidth and friends' interest just acquired, defined in the Chapter 3 on the optimization model. The remote friend gateway with the larger benefit will be requested earlier, that's due to caching the corresponding item on it would bring the larger benefit for the users involved. Finally, the completion of the *query list* for each user will lead to the next step, that is to start the caching placement request according to the algorithm GPA.

GPA and RPA implementation

The GPA (Greedy Placement Algorithm) algorithm is executed multiple times within a gateway until the query list for each user is empty. The basic idea of the algorithm is already explained in the Chapter 4 on heuristic methods. The flow chart of implementation details is illustrated in Fig. 6.8, in which it should be noticed that, the block of *GPA to GW* means to send the GPA request to remote friend gateway, which is depicted in the bottom half of Fig. 6.5. The GPA request contains the content owner's identifier and the information of the item to request including the type, size and meta data etc.

Once receiving the GPA request, the residential gateway activates the server thread to check whether there is enough storage space to cache the incoming content item. If the space is not enough, it will call the replacement algorithm, RPA, for whose details please refer to the Chapter 4 on heuristic methods. Also, the gateway sends the RPA request to notify the removed item's owner that the corresponding item's replica doesn't exist any more.

Implementation of transferring files

Another issue to be noticed for the implementation work is the transferring of the content files, when the response of GPA request is positive. To transfer the files between residential gateways we use two tools: one is a C library called libcurl [65] in the client side and the other is vsftpd [66] daemon program running for the server side. Both of two transfer the content files exploiting the file transfer protocol, FTP.

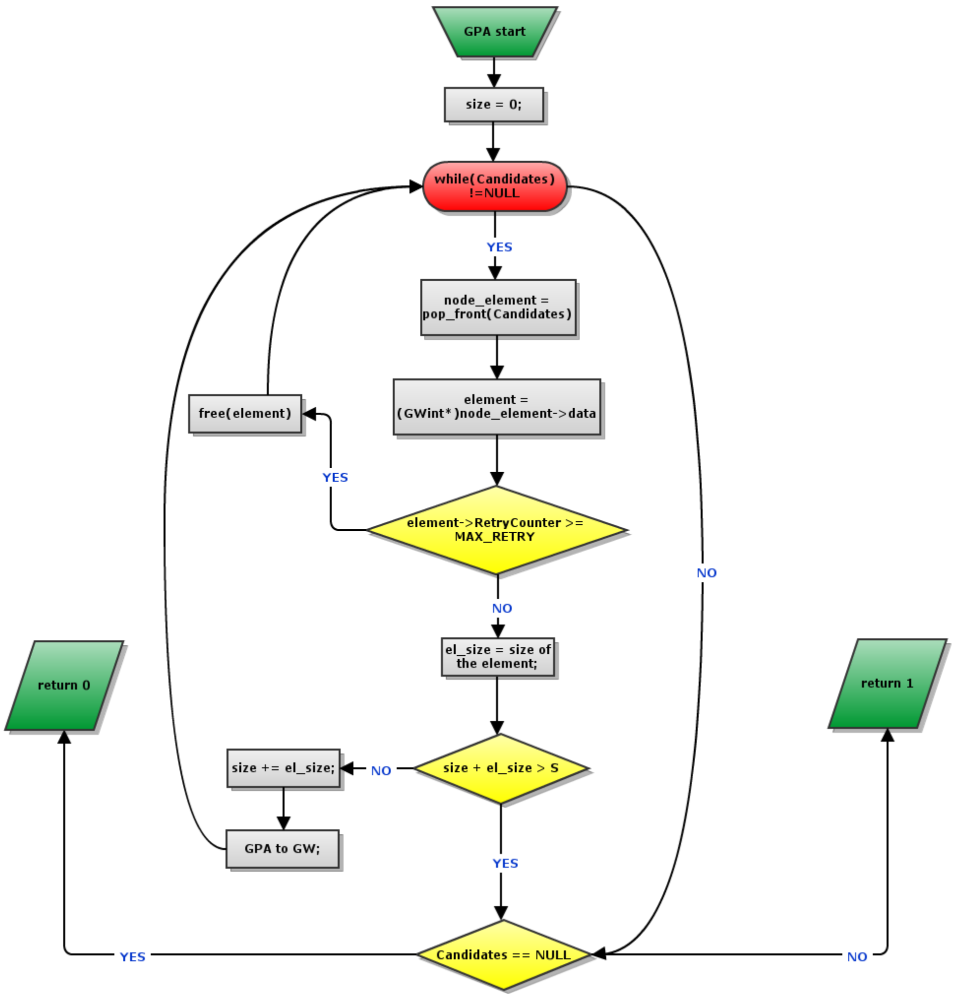


Figure 6.8: Flowchart depicting the algorithm of GPA.

The server-side tool vsftpd is a highly configurable and very versatile daemon that runs in background on Linux as FTP server. Specifically the configuration file, called vsftpd.conf, which is created during the installation, consists of options that can be enabled to customize the behaviour of the ftp server. Meanwhile, the client-side configuration for transferring files is also demanding as much as the server side. The client side of the gateway does not have to worry about managing connections for FTP file transfers, so the choice of this programming interface is almost inevitable. The library libcurl is a very powerful tool, although it uses C functions, one can safely say that the level of abstraction in the use of features can be very high. With this library you can play about 20 different network protocols, from the FTP to IMAP, SCP, HTTPS supporting SSL certificates and various modes for authentication.

6.4 Evaluation via emulation

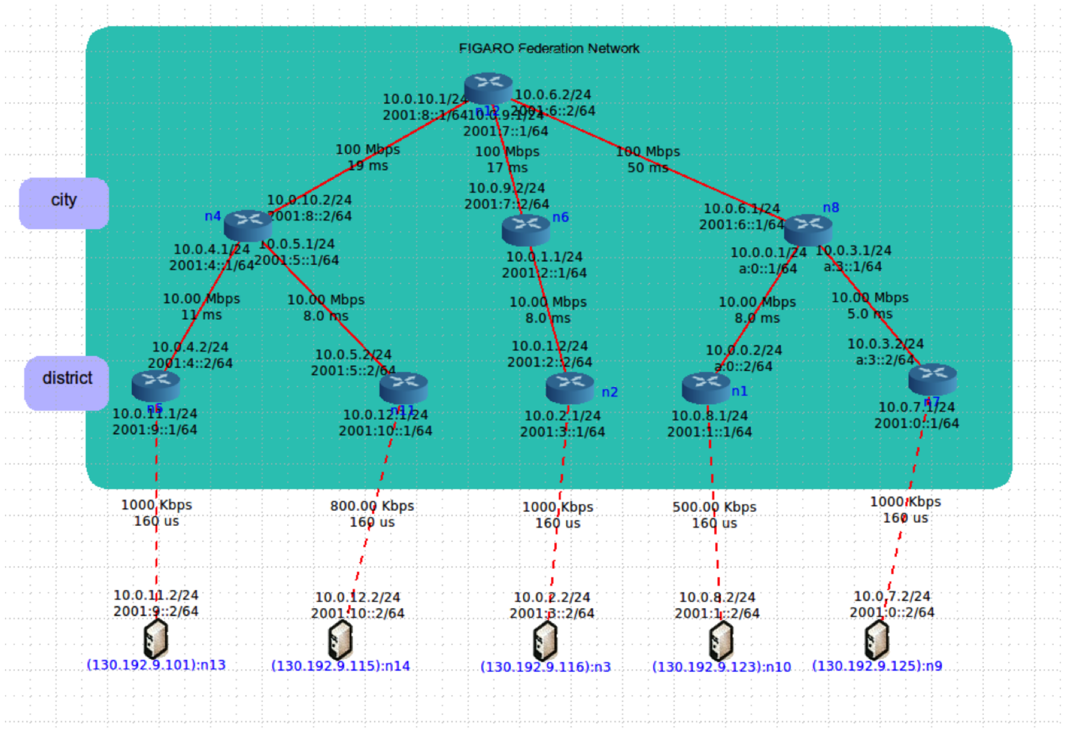


Figure 6.9: Emulation testbed.

To demonstrate the effectiveness and efficiency of the caching placement and replacement algorithms implemented and give reasons for their use within the architecture of FIGARO, it is desired to evaluation the impact of caching algorithms on the performance of content retrieval and the comparison of different caching algorithms implementation. The evaluation tests are divided into two portions, one implements a random caching approach when placing files, while the other solves the placement by making use of the caching algorithms GPA with RPA.

The evaluation has tried to recreate a realistic scenario in which multiple particular users wants to retrieve content from a remote residential gateway. We implement the caching algorithms as the daemon application running on multiple Linux servers located inside of main campus of Politecnico di Torino. The fact is these servers connect to each other through a switch in a subnet with the link of $1Gbps$ capacity. The data transfer among these servers would take place instantly and it can not demonstrate the difference between the two caching approaches under this network configuration. So to this end, we need an emulator to emulate the connection with much complex network topology among these servers. Also during the test the background traffic is inserted between them (e.g. listening to music from YouTube, measurement bandwidth, etc ...) to make a more realistic environment.

Fig. 6.9 illustrates the emulation network showed in the coloured box created using CORE [67] (Common Open Research Emulator) which is a tool for emulating networks on one or more machines. The emulation testbed is composed by a network of routers structured in levels. The first level starting from the top represents a possible Tier 2 Italian. The second level of the tree, however, reported a Tier 3 city (e.g., Turin, Milan, Rome), which have been suitably chosen latency. The final level is made up of set of district routers with which the residential gateway implemented by Linux server are connected through different types of connections provided by ISPs allowed.

The current emulated topology shown in Fig. 6.9 is only deployed by the preliminary experiments to demonstrate the use of caching algorithms GPA with RPA can proactively cache the content closer to the interested friends in the real testbed. The emulation tests with much more complex topology, much larger scale of gateway and more realistic content retrieval schema are already list in our ongoing work.

Chapter 7

Conclusion and perspectives

7.1 Conclusion

The goal of this work is to look beyond traditional approaches of content sharing and backup involving residential networks for home users. In particular, we believe the user's utility accounting for user's interests and network bandwidth can be maximized by placing the content "outside the home" in a cloud formed by other residential networks and exploiting the user's social networking information. In our vision, the user generated content could be cached on the friends' residential gateways which is located on the edge of Internet and could be tuned as a *stable caching layer* for delivering the content owned by home users. We formulated this optimization problem as a budgeted maximum coverage problem and solved it numerically in a synthetic social network. Then we evaluated and compared the performance of different content placement strategies across different home gateway quota cases and showed that the joint interest/bandwidth optimization strategy is superior to other one. Furthermore, distributed heuristic placement and replacement algorithms are proposed to approximate the joint optimization strategy, and implemented and evaluated under a realistic network environment.

The contribution of this thesis are summarized as following:

- We formulated this content placement problem as a Budgeted Maximum Coverage (BMC) problem which is NP-hard, and we used Gurobi solver [6], which runs a variant of the branch-and-cut algorithm, to numerically solve the optimization model and to obtain the optimal content placement solutions. We also compared it with two different content placement strategies for gateways with various quota sizes, under a realistic simulation scenario with synthetic social network and realistic network environment.

- We then devised and evaluated some low-complexity, distributed heuristic algorithms which can be implemented on federated residential gateways to realize a social caching strategy and use simulations in the same synthetic social network scenario to show the final content placement among "friendly" gateways well approximates the optimal solution under different network settings. And we evaluated the impact of different content caching strategies on the content retrieval performance in terms of average access delay, hops and distance using NS3 simulator.

- We also set up a synthetic social network that shares the basic common properties of real social networks, namely realistic degree distribution and the distribution of friends and their position/distance on the network topology, to make our simulation scenario more realistic. We chose Facebook as a target, since it is one of most popular and largest online social networking sites nowadays. In particular, we used the findings in other related works to characterize the network properties and to establish a relationship between geographical distance and friendship probability that matched the one that can be measured in Facebook.

- Last but not least, we proposed and implemented an implementation framework prototype to connect the residential gateway with the online social network (OSN) service through API services exposed by OSN (i.e. Facebook Graph API) in the implementation work of federated residential gateways. So each gateway can be allowed to collect its user's social networking data with the user's permission, and to obtain the interest information shared by their friends who also belong to the federation residential networks after checking with lookup service (which is a naming service set up by us to manage the information on the home users and their associated gateways). Meanwhile, we exploited Common Open Research Emulator (CORE) to emulate a realistic network environment for residential gateways to test and evaluate our content caching algorithms implementation.

7.2 Perspectives

Our ongoing work prominently includes the evaluation work of the caching module implementation with much more complex emulation topology, much larger scale of gateway and more realistic content retrieval schema.

Some issues which are beyond the focus within this thesis are already outlined in our working list. For example, the home user should not be required to explicitly manage and schedule caching. Instead caching should occur automatically and transparently to the user. An intelligent scheduling mechanism for content caching should need a second thought running the caching module in the background especially during the off hours, the weekends or the working hours while only 10% of available bandwidth is consumed allowing the home users to use the network for other tasks. And the integration work between the caching module and other components in the residential gateway architecture is under the discussion and ongoing.

Bibliography

[1] Internet world statistics. http://www.internetworldstats.com.

[2] 2009 consumer electronics storage report. http://www.tomcoughlin.com/techpapers.htm/.

[3] The FIGARO FP7 ICT project. http://www.ict-figaro.eu/.

[4] L. Backstrom, E. Sun, and C. Marlow. Find me if you can: improving geographical prediction with social and spatial proximity. In *Proceedings of the 19th international conference on World wide web*, pages 61–70. ACM, 2010.

[5] J. Jiang and C. Casetti. Socially-aware gateway-based content sharing and backup. In *Proceedings of the 2nd ACM SIGCOMM workshop on Home networks*, pages 61–66. ACM, 2011.

[6] Gurobi optimizer. http://www.gurobi.com/html/products.html.

[7] FTH Den Hartog, M. Balm, CM De Jong, and JJB Kwaaitaai. Convergence of residential gateway technology. *Communications Magazine, IEEE*, 42(5):138–143, 2004.

[8] Hgi-rd001-r2.01 home gateway technical requirements: Residential profile v1.01. http://www.homegateway.org/publis/RD-001-R2-01_HG-Tech-Req-Residential_V1-01.pdf, 2008.

[9] S. Defrance, R. Gendrot, J. Le Roux, G. Straub, and T. Tapie. Home networking as a distributed file system view. In *Proceedings of the 2nd ACM SIGCOMM workshop on Home networks*, pages 67–72. ACM, 2011.

[10] P. Budde. Australia - national broadband network - design and deployment strategies. http://www.budde.com.au/Research/Australia-National-Broadband-Network-Design-and-Deployment-Strategies.html.

[11] E.J.L. Lu and C.W. Chen. An enhanced edcg replica allocation method in ad hoc networks. In *e-Technology, e-Commerce and e-Service, 2004. EEE'04. 2004 IEEE International Conference on*, pages 465–472. IEEE, 2004.

[12] L. Yin and G. Cao. Supporting cooperative caching in ad hoc networks. *Mobile Computing, IEEE Transactions on*, 5(1):77–89, 2006.

[13] B. Tang, H. Gupta, and S.R. Das. Benefit-based data caching in ad hoc networks. *Mobile Computing, IEEE Transactions on*, 7(3):289–304, 2008.

[14] G. Pallis and A. Vakali. Insight and perspectives for content delivery networks. *Communications of the ACM*, 49(1):101–106, 2006.

[15] J. Zhang. A literature survey of cooperative caching in content distribution networks. *arXiv preprint arXiv:1210.0071*, 2012.

[16] K.L. Johnson, J.F. Carr, M.S. Day, and M.F. Kaashoek. The measured performance of content distribution networks. *Computer Communications*, 24(2):202–206, 2001.

[17] S. Triukose, Z. Wen, and M. Rabinovich. Measuring a commercial content delivery network. In *Proceedings of the 20th international conference on World wide web*, pages 467–476. ACM, 2011.

[18] G. Haßlinger and F. Hartleb. Content delivery and caching from a network provider's perspective. *Computer Networks*, 55(18):3991–4006, 2011.

[19] 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.

[20] W. Galuba. Friend-to-friend computing: Building the social web at the internet edges. Technical report, Citeseer, 2009.

[21] A. Derhab and N. Badache. Data replication protocols for mobile ad-hoc networks: a survey and taxonomy. *Communications Surveys & Tutorials, IEEE*, 11(2):33–51, 2009.

[22] W. Wu and J.C.S. Lui. Exploring the optimal replication strategy in p2p-vod systems: Characterization and evaluation. *Parallel and Distributed Systems, IEEE Transactions on*, 23(8):1492–1503, 2012.

[23] Z. Huang, E. Biersack, and Y. Peng. Reducing repair traffic in p2p backup systems: Exact regenerating codes on hierarchical codes. *ACM Transactions on Storage (TOS)*, 7(3):10, 2011.

[24] A. Duminuco and E.W. Biersack. Hierarchical codes: A flexible trade-off for erasure codes in peer-to-peer storage systems. *Peer-to-peer networking and applications*, 3(1):52–66, 2010.

[25] N. Belaramani, M. Dahlin, L. Gao, A. Nayate, A. Venkataramani, P. Yalagandula, and J. Zheng. Practi replication. In *Proc NSDI*, 2006.

[26] V. Ramasubramanian, T.L. Rodeheffer, D.B. Terry, M. Walraed-Sullivan, T. Wobber, C.C. Marshall, and A. Vahdat. Cimbiosys: A platform for content-based partial replication. In *Proceedings of the 6th USENIX symposium on Networked systems design and implementation*, pages 261–276. USENIX Association, 2009.

[27] A. Post, P. Kuznetsov, and P. Druschel. Podbase: Transparent storage management for personal devices. In *International Workshop on Peer-to-peer Systems*, volume 2008, page 14, 2008.

[28] H. Hayashi, T. Hara, and S. Nishio. On updated data dissemination exploiting an epidemic model in ad hoc networks. *Biologically Inspired Approaches to Advanced Information Technology*, pages 306–321, 2006.

[29] M. Hauspie and A. Panier. Localized probabilistic and dominating set based algorithm for efficient information dissemination in ad hoc networks. In *Mobile Ad-hoc and Sensor Systems, 2004 IEEE International Conference on*, pages 11–20. IEEE, 2004.

[30] J. Luo, J.P. Hubaux, and P.T. Eugster. Pan: Providing reliable storage in mobile ad hoc networks with probabilistic quorum systems. In *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, pages 1–12. ACM, 2003.

[31] H. Yu, P. Martin, and H. Hassanein. Cluster-based replication for large-scale mobile ad-hoc networks. In *Wireless Networks, Communications and Mobile Computing, 2005 International Conference on*, volume 1, pages 552–557. IEEE, 2005.

[32] Z. Zhang, Q. Lian, S. Lin, W. Chen, Y. Chen, and C. Jin. Bitvault: A highly reliable distributed data retention platform. *ACM SIGOPS Operating Systems Review*, 41(2):27–36, 2007.

[33] H. Weatherspoon. *Design and evaluation of distributed wide-area on-line archival storage systems*. PhD thesis, Citeseer, 2006.

[34] L.P. Cox, C.D. Murray, and B.D. Noble. Pastiche: Making backup cheap and easy. *ACM SIGOPS Operating Systems Review*, 36(SI):285–298, 2002.

[35] M. Vrable, S. Savage, and G.M. Voelker. Cumulus: Filesystem backup to the cloud. *ACM Transactions on Storage (TOS)*, 5(4):14, 2009.

[36] C. Boldrini, M. Conti, and A. Passarella. Contentplace: social-aware data dissemination in opportunistic networks. In *Proceedings of the 11th international symposium on Modeling, analysis and simulation of wireless and mobile systems*, pages 203–210. ACM, 2008.

[37] E. Jaho and I. Stavrakakis. Joint interest-and locality-aware content dissemination in social networks. In *Wireless On-Demand Network Systems and Services, 2009. WONS 2009. Sixth International Conference on*, pages 173–180. IEEE, 2009.

[38] P. Pantazopoulos, I. Stavrakakis, A. Passarella, and M. Conti. Efficient social-aware content placement in opportunistic networks. In *Wireless On-demand Network Systems and Services (WONS), 2010 Seventh International Conference on*, pages 17–24. IEEE, 2010.

[39] A. Miklas, K. Gollu, K. Chan, S. Saroiu, K. Gummadi, and E. De Lara. Exploiting social interactions in mobile systems. *UbiComp 2007: Ubiquitous Computing*, pages 409–428, 2007.

[40] J.A. Pouwelse, P. Garbacki, J. Wang, A. Bakker, J. Yang, A. Iosup, D.H.J. Epema, M. Reinders, M.R. Van Steen, and H.J. Sips. Tribler: a social-based

peer-to-peer system. *Concurrency and Computation: Practice and Experience*, 20(2):127–138, 2008.

[41] Z. Anwar, W. Yurcik, V. Pandey, A. Shankar, I. Gupta, and R.H. Campbell. Leveraging Social-Network Infrastructure to Improve Peer-to-Peer Overlay Performance: Results from Orkut. *Arxiv preprint cs/0509095*, 2005.

[42] L. Zaczek and A. Datta. Mapping social networks into p2p directory service. In *Social Informatics, 2009. SOCINFO'09. International Workshop on*, pages 10–15. IEEE, 2009.

[43] L. Han, B. Nath, L. Iftode, and S. Muthukrishnan. Social butterfly: Social caches for distributed social networks. In *Privacy, security, risk and trust (passat), 2011 ieee third international conference on and 2011 ieee third international conference on social computing (socialcom)*, pages 81–86. IEEE, 2011.

[44] L. Han, M. Punceva, B. Nath, S. Muthukrishnan, and L. Iftode. Socialcdn: Caching techniques for distributed social networks.

[45] D. Quercia and L. Capra. FriendSensing: recommending friends using mobile phones. In *Proceedings of the third ACM conference on Recommender systems*, pages 273–276. ACM, 2009.

[46] M. Safaei, M. Sahan, and M. Ilkan. Social Graph Generation & Forecasting Using Social Network Mining. In *2009 33rd Annual IEEE International Computer Software and Applications Conference*, pages 31–35. IEEE, 2009.

[47] 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*, page 42. ACM, 2007.

[48] C. Wilson, B. Boe, A. Sala, K.P.N. Puttaswamy, and B.Y. Zhao. User interactions in social networks and their implications. In *Proceedings of the 4th ACM European conference on Computer systems*, pages 205–218. ACM, 2009.

[49] A. Clauset. Finding local community structure in networks. *Physical Review E*, 72(2):26132, 2005.

[50] Y. Beyene, M. Faloutsos, D.H.P. Chau, and C. Faloutsos. The eBay Graph: How do online auction users interact? In *of: Computer Communications Workshops*, pages 1–6, 2008.

[51] D.S. Hochba. Approximation algorithms for np-hard problems. *ACM SIGACT News*, 28(2):40–52, 1997.

[52] S. Khullera, A. Mossb, and J. Naor. The budgeted maximum coverage problem. *Information Processing Letters*, 70:39–45, 1999.

[53] N. Megiddo, E. Zemel, and S.L. Hakimi. The maximum coverage location problem. *SIAM Journal on Algebraic Discrete Methods*, 4(2):253–261, 1983.

[54] O. Berman, D. Bertsimas, and R.C. Larson. Locating discretionary service facilities, ii: maximizing market size, minimizing inconvenience. *Operations Research*, 43(4):623–632, 1995.

[55] O. Berman, R.C. Larson, and N. Fouska. Optimal location of discretionary service facilities. *Transportation Science*, 26(3):201–211, 1992.

[56] R. Cohen and L. Katzir. The generalized maximum coverage problem. *Information Processing Letters*, 108(1):15–22, 2008.

[57] M. Gjoka, M. Kurant, C.T. Butts, and A. Markopoulou. Walking in facebook: A case study of unbiased sampling of osns. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9. IEEE, 2010.

[58] J. Jiang and C. Casetti. Distributed content backup and sharing using social information. *NETWORKING 2012*, pages 68–81, 2012.

[59] J. Illenberger, G. Flöteröd, M. Kowald, K. Nagel, K. Nagel, and K. Nagel. *A model for spatially embedded social networks*. ETH, Eidgenössische Technische Hochschule Zürich, IVT, Institut für Verkehrsplanung und Transportsysteme, 2009.

[60] M. Piatek, T. Isdal, T. Anderson, A. Krishnamurthy, and A. Venkataramani. Do incentives build robustness in bittorrent. In *Proc. of NSDI*, volume 7, 2007.

[61] The network simulator ns-3. `http://www.nsnam.org/`.

[62] LAMP (software bundle) wiki. `http://en.wikipedia.org/wiki/LAMP_(software_bundle)`.

[63] Facebook Graph API. `https://developers.facebook.com/docs/reference/api/`.

[64] Iperf. `http://sourceforge.net/projects/iperf/`.

[65] Libcurl. `http://curl.haxx.se/libcurl/`.

[66] Vsftpd. `https://security.appspot.com/vsftpd.html/`.

[67] Common open research emulator (CORE). `http://cs.itd.nrl.navy.mil/work/core/`.