Internet Sharing in Community Networks

Emmanouil Dimogerontakis

a dissertation
presented to the
Universitat Politècnica de Catalunya
in candidacy for the degree
of Doctor of Philosophy

recommended for acceptance
by the Department of
Computer Architecture

Advisors: Dr. Leandro Navarro & Dr. Luís Veiga &
Dr. Roc Meseguer

March 2017

Abstract

The majority of the world's population does not have any or adequate Internet access. This implies that the Internet cannot provide service to the general public, to reach anyone without discrimination. Global access to the Internet for all requires a dramatic reduction in Internet access costs especially in geographies and populations with low penetration. In response to this problem, various local communities build their own network infrastructures, Community Networks(CN),and provide affordable inter-networking with the Internet, based mostly on wireless technologies. Sharing resources, such as infrastructure or Internet access, is encouraged at all levels to lower the cost of network infrastructures and services. Communities can develop their own network infrastructures as a commons, using several interconnected Wireless Mesh Networks (WMN) given its sheer size, and sharing several Internet gateways among their participants. These Internet gateways are typically web proxies for Web access, the most popular traffic, and a small fraction using IP tunnels.

Access to the Internet through web proxy gateways relies on users or organisations sharing the full or spare capacity of its Internet connection with other users. However these gateway nodes may be overloaded by the demand, and their Internet capacity may experience problems under lack of regulation. The purpose of this thesis is to analyze the behaviour of already existing Internet sharing mechanisms used by communities and propose and evaluate a framework that allows users to access and share Internet bandwidth achieving fairness across the users and best effort utilization of the available resources.

Keywords

community networks; Internet access; resource sharing;

Contents

1 Introduction

Internet access has become a requirement to participate in society; for instance, to access public services, education material, social media and also to support everyday work of millions of organizations. However, the majority of the world's population is not online [28] yet, far from the vision of "universal service". Global access to the Internet for all requires a dramatic reduction in Internet access costs especially in geographies and populations with low penetration [23]. This situation enhances the digital divide between several communities/regions/countries, and the rest of the world. Therefore, while the Internet is for everyone [16], as Vint Cerf says: *"it won't be if it isn't affordable by all that wish to partake of its services, so we must dedicate ourselves to making the Internet as affordable as other infrastructures so critical to our well-being"*.

As a way to mitigate this challenge, in many regions worldwide the citizens self-organize in order to explore alternative models for getting Internet access under reasonable conditions. An example of it are the Community Networks (CN) [52], that are crowdsourced data network infrastructures built by citizens

and organisations, who pool their resources and coordinate their efforts [6] to provide an Internet community access service to their members, including the deepest rural communities worldwide [43].

For instance guifi.net, probably the largest community network in the world has more than 30,000 network nodes. It is organised as an inter-network with several local WMN. 12,500 registered users can use any of the 356 web proxies (May 2016). The network links between nodes are contributed and managed by the participants. Therefore paths between nodes, such as client to proxy may not be reliable [3] or guaranteed, especially when compared to commercial offerings from centrally managed ISPs. Access to the Internet through web proxy gateways relies on users or organisations sharing the full or spare capacity of its Internet connection with other guifi.net users.

1.1 Community Networks

The community networks are quite new, and they represent an alternative paradigm for developing network infrastructures and services in a broad sense. Communities can propose locally adapted self-organized cooperative schemes for developing self-provided data networking solutions, sharing wireless links and spectrum, optical fibre, and Internet gateways; and even sharing Internet connectivity with other members of the community.

These communities communities usually describe themselves as *open*, *free*, and *neutral*. They are open since everyone has the right to know how they are built. They are free because the network access is driven by the non-discriminatory principle; thus, they are universal. Moreover, they are neutral in terms of technology solutions to extend the network, and neutral for supporting data transfers.

When these fundamental principles are applied to an infrastructure, they often result in networks that are *collective goods*, *socially produced*, and governed as *common-pool resources* (CPR). Natural CPR, also called commons (such as, communal pastures, fisheries, forests), were studied in depth by E.

Ostrom [42]. According to that we use the term *network infrastructure commons* [39].

These infrastructures developed cooperatively become regional IP networks that enable inexpensive interaction and access to local digital content and services. In addition, there exists the issue of access to the global Internet, that can be reached through Internet Service Providers (ISP) in these regional network infrastructures.

There are many examples of community networks that can fit in this scheme. In [40] we outline 18 cases, with 9 described in detail, and 267 potential cases in 41 countries. There are also several studies that consider structural [15, 37, 51], technological [7, 36, 52] and organisational [6, 35, 40] points of view of these networks.

1.2 guifi.net Web Proxy Service

Guifi.net is an open, free, and neutral network built by its members: citizens and organisations pooling their resources to build and operate a local network infrastructure, governed as a common pool resource [6]. The network infrastructure is mostly wireless [52] with a fiber backbone. Participants can extend the network to reach new locations and use the network to reach intranet services such as the web proxy service.

The most popular application in community networks is web access and guifi.net is no exception. Web proxy nodes connected both to guifi.net and an ISP act as free gateways to the Internet to the community network users. Proxies run on simple servers and take advantage of individuals or organisations (like libraries or municipalities) offering their Internet access to other guifi.net users. Using web proxies, public entities can provide free Internet access without infringing telecom market competence regulations.While some of the web proxies are kept as a private service, 356 out of the 477 registered web proxy servers in the network (May 2016) are shared with all the network registered participants (12,500). A registered member is allowed to use any

proxy of their convenience, although recommended to use one nearby. Users can select or change its choice based on quality of experience. Therefore, while some proxies may become popular and highly used, others may remain underused.

Without access to one of these proxies or a guifi.net connected ISP, community members can still share contents and access applications within the same community network, but not to external resources. In order to get Web access, the clients manually specify a list of proxies, by starting with the main proxy and following with the secondary ones. Proxy access is performed through federated authentication credentials. In case a proxy does not respond (timeout) or rejects the connection, the client automatically switches to the next proxy in the list. The choice of proxies is manual and the list usually comes from acquaintances in the community or personal experience.

Internet access through web proxies is clearly a limited service compared to an IP tunnel, as the service is usually restricted to a set of protocols/ports; however, it can enhance privacy as the origin IP addresses are hidden. The most popular application in community networks is Web access. Many citizens, private and public organizations involved in community networks, such as freifunk.net or guifi.net, have chosen to provide that service within their community network. Using Web proxies through local networking infrastructures (e.g., Community Networks) that provides local/regional connectivity, the citizens can reach Internet content and services at no additional cost.

1.3 Spare Internet Capacity

We define *spare Internet capacity* as the network traffic that can be moved to and from the Internet by secondary users with no performance degradation or cost penalty for primary users. Secondary traffic can have short term effects for the primary in packet queueing, resulting in service degradation, such as packet delay, loss and reduced throughput. That affects data transport (TCP) generating a lower throughput with longer and more variable down-

load times, and also an overall degradation of quality of the user experience. Therefore, the secondary traffic should be unnoticed by the primary user, both when the sum of primary and secondary is below the capacity of the Internet access link, and when exceeded. Peak usage is an extreme case, where the secondary traffic may need to be blocked to avoid an impact on cost under the common 95-percentile pricing schemes [25] used by transit ISPs to charge according to peak demand.

There is a rich body of work focused on reducing the cost and increasing the coverage of Internet in several scenarios. For instance, the Lowest Cost Denominator Networking (LCD-Net) [44] explores resource pooling Internet technologies to support benevolence in the Internet. Some of these ideas are illustrated by WiFi sharing schemes, community-led (PAWS) or commercially-run (FON), where home broadband subscribers donate their controlled (but for free) broadband Internet spare capacity to fellow citizens. This is done by sharing a fixed portion of throughput [1]. In contrast, this work considers not just local access to a shared WiFi hotspot, but also remote access to the shared resource over a community network that can use any network technology, such as, wired or wireless meshes. This research also takes advantage of all spare capacity, with little or no visible impact on the primary user. This means secondary users can get from all to nothing, depending on the demand of primary use.

1.4 Problem Statement

Studying the Internet access based on web proxies in guifi.net we observed that some proxies may be overloaded and, therefore, offer degraded or unusable performance, while others may remain underused, due to bad or manual choice. Users of overloaded proxies, or that use congested links to reach their proxy, experience degraded quality of experience (QoE) in Internet access. It is also interesting to note that the set of overloaded and underutilized proxies varies according to the access patterns of the users. Moreover, we observed

that this challenge is an instance of the more general problem where a WMN inter-network community accesses the Internet using a pool of shared Web proxies in different WMN nodes. We decompose this general problem in two major components. First, the relation between users-proxies, which concerns how in a macroscopic level the demand of the users can meet in an efficient way the offered resources. Second, the relation between each user who offers his resources (primary) and the consumers of his resources (secondary), which concerns the preservation of the experience of the primary users, since they are a critical factor of the service producing as low as possible overhead to the secondaries.

In the users-proxies relation, the challenge is that clients in any WMN node should select the right proxy according to the performance of the internal network path and the load of the web proxies. The net effect is that a large population of C clients can browse the web taking advantage of the aggregated capacity of a pool of P web proxies, with $C \gg P$, over a WMN infrastructure, at a fraction of the cost of C Internet connections.

The designed solution must be:

- Incremental and backwards-compatible: should be able to be deployed incrementally, so it should work fine for both baseline or enhanced clients.
- Dynamic: Users can and should switch proxies wisely to maximize their QoE. This can be the result of changes in network topology, path load, or proxy performance.
- Decentralized: should not require any central component.
- Scalable: could scale up to the current number of users and proxies and beyond.
- Routing-agnostic: should not depend on the transport and routing algorithms, or on specific network features.

Concerning the primary-secondary users relation, we consider N citizens or organizations sharing their unused Internet access capacity benevolently with M neighbors and members of their community, through a local or regional community network. In order to provide such a service without negatively affecting the quality of access of the primary users, we propose utilizing gateways to aggregate the primary traffic (i.e., that of the Internet access donors) from the one of the beneficiaries (i.e., the secondary traffic) fairly.

Each of the M beneficiary nodes selects one or a few of the N Internet gateways, where they send their traffic. The gateways receive the IP traffic or HTTP requests from these secondary nodes and try to provide them with a solution. Although this traffic uses the spare capacity of the Internet access, it may compete with the primary source traffic, hinder its performance, and also increase its cost. Therefore, only making this sharing process innocuous for the donors, will allow this mechanism be sustainable over time. However, keeping under control this aspect of the traffic represents a major challenge for the managers of community networks.

2

Contributions

2.1 List of Publications

Accepted

P1. Dimogerontakis, E., Meseguer, R. & Navarro, L. *Internet Access for All: Assessing a Crowdsourced Web Proxy Service in a Community Network* in *Passive and Active Measurement Conference* (CORE2014 Rank B) (2017).

P2. Dimogerontakis, E., Neto, J., Meseguer, R. & Navarro, L. *Client-Side Routing-Agnostic Gateway Selection for heterogeneous Wireless Mesh Networks* in *IFIP/IEEE International Symposium on Integrated Network Management* (CORE2014 Rank A) (2017).

Pending Review

The following papers have been submitted for review.

P4. Dimogerontakis, E., Meseguer, R., Navarro, L., Ochoa, S. & Veiga, L. *Community Sharing of Spare Network Capacity* in *IEEE International Conference on Networking, Sensing and Control* (ERA2010 Rank C),(under review) (2017).

P3. Dimogerontakis, E., Meseguer, R., Navarro, L., Ochoa, S. & Veiga, L. *Design Trade-offs of Crowdsourced Web Access in Community Networks* in *IEEE 21st International Conference on Computer Supported Cooperative Work in Design* (CORE2014 Rank B),(Under Review) (2017).

Under Preparation

P5. Dimogerontakis, E., Braem, B., Meseguer, R. & Navarro, L. *Socio-Economic Experiences, Challenges and Lessons in Community Networks around the world* in – (2017).

Other Publications

The background research to this thesis has led to the following publications:

1. Millan, P., Molina, C., Dimogerontakis, E., Navarro, L., Meseguer, R., Braem, B. & Blondia, C. *Tracking and Predicting End-to-End Quality in Wireless Community Networks* in *2015 3rd International Conference on Future Internet of Things and Cloud* (Aug. 2015), 794–799.
2. Selimi, M., Khan, A. M., Dimogerontakis, E., Freitag, F. & Centelles, R. P. Cloud services in the Guifi.net community network. *Computer Networks* **93, Part 2.** Community Networks, 373–388 (2015).
3. Escrich, P., Baig, R., Dimogerontakis, E., Carbó, E., Neumann, A., Fonseca, A., Freitag, F. & Navarro, L. *WiBed, a platform for commodity wireless testbeds* in *2014 IEEE 10th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)* (Oct. 2014), 85–91.
4. Dimogerontakis, E., Vilata, I. & Navarro, L. *Software Defined Networking for community network testbeds* in *2013 IEEE 9th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)* (Oct. 2013), 111–118.

2.2 Contributions

Analysis of a crowdsourced web proxy service Global access to the Internet for all requires a dramatic reduction in Internet access costs particularly in developing areas. This access is often achieved through several shared web proxy gateways in commercially and community driven local or regional access networks. In an effort to understand the functionality and performance of this shared Internet access we performed a measurement study of a crowdsourced Internet proxy service in the guifi.net community network that provides free Internet access to a large community with a high ratio of users to proxies. Our study focus on a representative subset of the whole network with about 900 nodes and roughly 470 users of the web proxy service. We analyze the service from three viewpoints: web content traffic from users, performance of proxies and influence of the access network. We observed that CNs can and are being used for providing a basic Internet access, by the means of Web proxies. Nevertheless, we observed the necessity of a regulation mechanism that would enable the fair and efficient usage of the resources.
The main results related with this contribution are presented in Chapter 3 and were originally reported in [P1].

Internet Gateway selection mechanism In order to facilitate the fair usage of the Web Proxy service we start by investigating and building the users-proxies regulation mechanism. We developed a client-side distributed system that optimizes the client-gateway mapping, agnostic to underlying infrastructure and protocols, requiring no modification of proxies or the underlying network [P2]. Clients choose proxies considering network congestion, proxy load and proxy performance, without requiring a minimum number of participating nodes. Our proposal was evaluated experimentally with clients and proxies deployed in guifi.net. Our selection mechanism avoids proxies with heavy load and slow internal network paths, while achieved a network overhead linear to the number of clients and proxies. Moreover, we demonstrated

that the trade-offs between informed proxy selection and admission control in proxies, could alleviate imbalances and uncertainty, and also improve the service with little additional burden [P3]. Nevertheless, the Internet sharing process can negatively affect the service received by the users sharing their connections, thus jeopardizing the continuity of this community service.
The main results related with this contribution are presented in Chapter 4 and Chapter 5 and were originally reported in [P3, P2] correspondingly.

Secondary usage of spare Internet capacity In the studied model of proxy usage, we argue that it is important to differentiate between primary users, who share their Internet connection, and the secondary users. To address the possible performance degradation of the users sharing their connections we proposed a middlebox that separates the traffic of the primary users from that of the secondary users. We analysed the impact and behaviour of several mechanisms for using this gateway, in order to determine how to maximize network utilization, use of the excess network capacity, and minimize the impact on the primary traffic. Finally, we presented a set recommendations to achieve the best performance isolation for the primary user, while the secondary user obtains the spare capacity equivalent to non-differentiated best effort.
The main results related with this contribution are presented in Chapter 6 and were originally reported in [P4].

3

Current State in CN Usage for Basic Internet Access

3.1 Introduction

guifi.net exemplifies how regional communities can develop their own network infrastructures, using wired and wireless links to create a regional IP network, and sharing several Internet gateways among all their participants. These gateways are usually web proxies for Web access, the most popular traffic, but can accommodate other traffic through HTTP CONNECT, SOCKS or tunneling. Proxies, not exempt from the drawbacks of middleboxes, have also additional advantages: some content and DNS resolution can be shared in caches, and most important, proxies can protect the privacy of end users if they trust the proxy provider.

In this chapter we present an study of the existing Web proxy service in a guifi.net zone We first describe the collected datasets in Section 3.2. Then we analyse the service from three viewpoints: 1) service usage by end-users: patterns of usage and content in Section 3.3, 2) the proxy, Section 3.4, in

terms of caching, users, performance and variability, and 3) the local network, Section 3.5, in terms of topology and usage. Our measurements describe the effectiveness of a simple setup of a regional network sharing a set of Web proxies in delivering free basic Web access to a large population.

3.2 Data Collection

For our analysis we choose to study the guifi.net zone Llucanes, a region in the Osona county of Catalunya, Spain. As explained in [22], this zone is representative of other rural guifi.net networks. Furthermore, Llucanes is the only guifi.net zone with published anonymized logs for all (four) involved operational proxies. Even-day proxy log entries anoymise the client IP address and show information about the requested URLs, while odd-day proxy logs show the opposite. We assisted in the preparation and publication of these logs[*]. The logs combined with openly accessible information about network topology, network links and network traffic information, provide a consistent and complete view of this regional network.

3.3 Service Usage Viewpoint

The behaviour of the users and the service can be described at macro-level as a set of time series concerning metrics that can be extracted from the monthly logs, namely bytes per request, number of requests and number of users. Figure 3.1 illustrates the traffic time series for the aggregate set of proxies showing a daily repetitive pattern, but also strong aperiodic negative spikes, which were statistically verified as a dominant period of 1 day, and the second largest peak at 12 hours.

[*]Logs in Squid format: `http://dsg.ac.upc.edu/anon_guifi_proxy_logs`

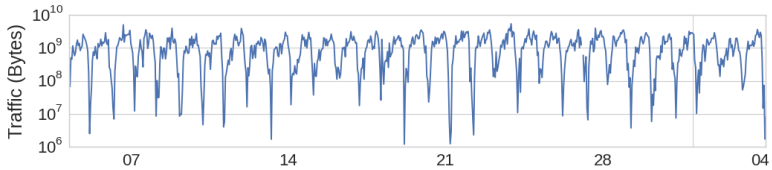


Figure 3.1: Web proxy time series (days in April 2016)

Service Usage: The majority of the traffic is due to a relatively small number of large requests (20% of the requests produce 97% of the traffic), while the rest of the requests present little variation in size. Additionally, as expected, the majority of the traffic (90%) is created by 15% of the users, but in contrast to the distribution of request size, the distribution of traffic and number of requests per user varies exponentially across users.

For the analysis of the service processing rate we calculate the **request processing throughput** as the bits per time elapsed for each request, depicted in fig. 3.2, ranging from less than 10^7 for the worst 10% to at least 10^8 for more than 80% of requests.

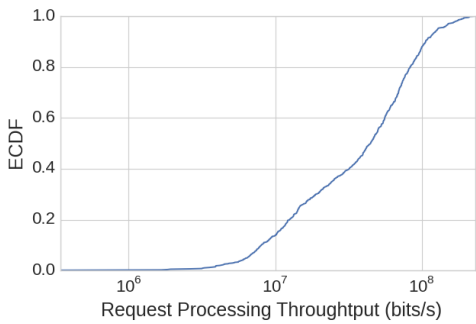


Figure 3.2: Processing rate per request

Domain	Traffic Fraction
googlevideo	27.85%
mega	16.73%
fbcdn	5.40%
rncdn3	2.80%
nflxvideo	2.70%
xvideos	2.60%
tv3	2.54%
level3	2.51%
google	1.96%
apple	1.78%

Table 3.1: Top Domains by traffic

Content analysis: Using the even-day proxy logs we looked at request types and target URL of users' requests. The majority of the traffic, almost 50%, consists of HTTP CONNECT requests, which is the method to establish TCP tunnels over HTTP, including HTTPS which is indisputably the main usage appearing in the logs. While for HTTP CONNECT we cannot know the corresponding content type, the most common type for the rest of the

requests is the generic *application/** with 23%, followed by video (19%) and image (5.5%).

The traffic for all analysed proxies in Table 3.1, including HTTP CONNECT, shows that the top video portal traffic occupies 36% of the traffic, which is an impressive large amount. For completeness, we mention that this is not reflected in the number of requests, therefore it is attributed on the size of the objects requested. Since video is by far the HTTP type with most traffic, it is not surprising to find that 4 out of 10 top domains are video portals. The distribution of web traffic per URL we found that it can roughly approximate a Zipf distribution, equivalent to results in [38] with domestic Internet connections.

3.4 The Proxy Viewpoint

In this section we investigate the capabilities and influence of the proxy servers involved. Our dataset concerns the only 4 proxies operating in the Llucanes zone. Table 3.2 shows the CPU and RAM characteristics of the proxy servers, as well as the nominal maximum throughput of the Internet connection they offer. They are very diverse, with great differences in Internet throughput (4–80Mbps). We also observe that proxy 11252 has the slowest combined characteristics. Despite that these servers provide other services, e.g. SNMP, the interference caused by other services is expected to be negligible.

Id	CPU	RAM	Max Throughput
3982	Intel amd64 2-core 2.6GHz	2GB	80Mbps
10473	Intel x86 2-core 2.6GHz	0.5GB	6Mbps
11252	AMD Athlon(tm) XP 1700+	0.5GB	4Mbps
18202	Intel amd64 2-core 2.7	2GB	8MBps

Table 3.2: Description of Proxies

The analysis of logs for the four proxies is summarized in Table 3.3. The values are averages for each proxy over a month of daily logs. The first group of columns (Different data) shows a data object storage perspective, with

	Different Data (MB)			Data transferred (MB)				Ratio (/All transfrd)		
Proxy	All	Repetd	Cached	All	Repetd	Cached	Connect	Repetd	Cached	Connect
10473	606	37	9.2	1481	95	14.3	943	6.4%	0.9%	63.7%
11252	3572	1234	28	15352	5512	99	7578	35.9%	0.6%	49.4%
18202	6384	1498	151	15963	3039	253	9274	19.0%	1.6%	58.1%
3982	2542	435	55	6019	855	96	3128	14.2%	1.6%	52.0%
Avg	3276	801	61	9704	2376	115	5231	18.9%	1.2%	55.8%

Table 3.3: Average volume of data in four proxies and ratios in a month of logs

the amount of different data objects requested (disregarding the number of requests for each). The second group (Data transferred) shows a data transfer perspective, with the amount of traffic in each category. The third group shows data transfer ratios to the total transferred. We distinguish between "All" content, seen or transferred by the proxy, content requested repeatedly (same URL, cacheable or not), content served from the cache (checked or not against the server), and content that is invisible (Connect method, typically HTTPS, passed through blindly).

Cache effectiveness: As introduced before, the passed through content (HTTPS) represents the majority of the proxy traffic (49.4–64%). Although URLs repeat significantly (6.4–36% of proxy traffic), the content successfully served from the cache (after validation or not) only represents a negligible amount (1–1.6%). Considering number of requests instead of the amount of data, despite URLs repeat often (20–41%), the content does not seem cache friendly, as cache hits only represent a very small portion (3–10%). The analysis in number of requests compared to byte count indicates that cached content usually corresponds to small objects. Bad cache performance can be attributed to characteristics of the proxy service, such as small cache size, small number of concurrent users per proxy, or to increasingly non-cacheable served content. We next look at how these apply to our scenario, claiming that that non-cacheable content is the main factor affecting cache performance.

Cache size: As far as the cache size, the default allocated cache size in guifi.net proxy settings is 10GB of secondary storage, while in some proxies caching is not enabled. However, we found out that cached content that results in cache hits only accounts for a maximum of 151 MB (if all repeated

URLs were cacheable) and an average of 61 MB (based on HITs) of data per day. In the extreme case where all content as cacheable and discounting the transparent CONNECT/HTTPS data, the amount of daily data seen (i.e. all content for all URL seen) accounts for a maximum of 1.5 GB and 801 MB on average, easily achievable with RAM-based caches.

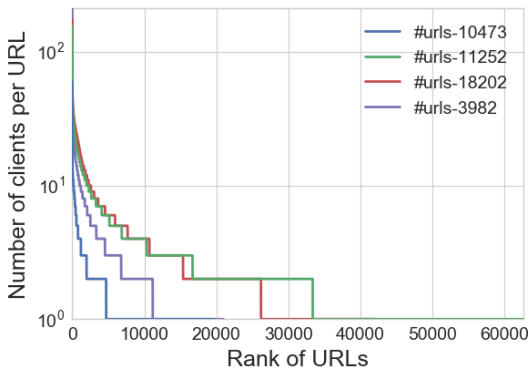


Figure 3.3: Rank of URLs by number of clients requesting them, by proxy

Sharing across clients: Proxies can provide the benefit of sharing network resources reusing not only HTTP content, but also reusing DNS resolution data as client web browsers delegate that to the proxy, or even reusing established TCP connections among multiple clients. Figure 3.3 shows the popularity of URLs across different clients in each proxy over a month, with top values between 60 to 212 different clients accessing the same URL. The number is related to the structure of the service, with many decentralized proxies and few users each and no inter-cache cooperation, which limits the potential of sharing cached content across more users.

Proxy selection: Users are instructed to check the public list of nearby proxies (in their network zone) in the network management directory[†] with shows a list of nearby proxies, including status and availability ratio, or follow the advice of trusted neighbours with previous usage experience. Therefore

[†]Llucanes: `https://guifi.net/en/node/8346/view/services`

the choice is influenced by social factors and the reputation of the service, but in most cases the first choice is the nearest operational proxy with acceptable availability or reputation. Typically several nearby Web proxy services are configured in client Web browsers. As all federated proxies use the same authentication service, users are free to choose whatever proxy they prefer. The choice of proxy is rather fixed and prioritized, only switching to lower choice proxies when the first fails.

Users and proxies: Figure 3.4 presents the distribution of the average number of users per hour. The different proxies show similar distributions, though we observe that proxy 10473 has a differentiated demand, with 40% of time without any user and a maximum of 10 users per hour. For the rest of proxies, the majority of time (60%) have an almost linear distribution between 5 and 25 users, with near equally distributed values, and an average of around 17 users per hour for proxies 11252 and 18202, and an average of 12 users for proxy 3982. The difference in distribution among proxies comes as a result of preference for proximity and manual selection. To complete the picture, we found an average of 10 users in periods of 10 secs, an average of 76 different users per proxy and day, and a maximum of 254 in a month.

The user's distribution among proxies has a clear impact in the distribution of the number of requests in figure 3.5. The ordering of proxies with respect to the number of users remains visible in the distribution of requests. Also, there is close-linear behaviour between 20% and 60% for all proxies except 10473. For proxies 11272 and 18202 the number of requests per hour is typically between 1K and 10K requests, with a mean of 8187 and 6716 respectively. In proxy 3982 typical values are between 500 and 1K requests per hour.

Regarding the number of clients seen by a proxy every day, the values (min, average, max) range from the lowest in proxy 10473 (14, 20, 27) to the highest in proxy 3982 (59, 82, 101). These numbers reflects the spirit of a highly decentralized service with many small capacity local proxies.

Internet connection and processing performance: Figure 3.6 provides the distribution of the Internet connection usage per proxy, cal-

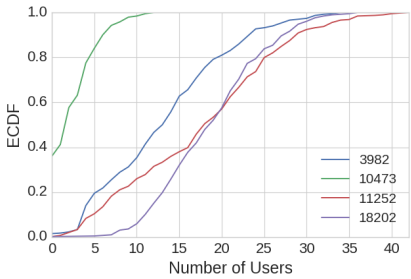


Figure 3.4: Hourly average number of users per proxy

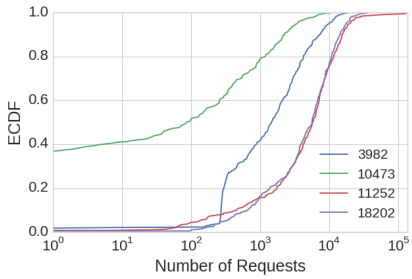


Figure 3.5: Hourly average number of requests per proxy

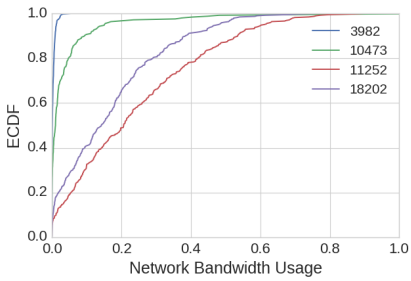


Figure 3.6: Network usage per Proxy

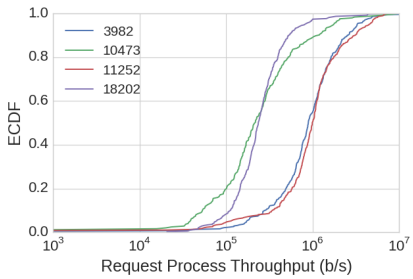


Figure 3.7: Hourly average request processing throughput per Proxy

culated as the approximate instant connection throughput of each proxy normalized by its maximum Internet throughput as provided in Table 3.2. All proxies show low utilization of their network resources, being approximately less than 0.3 (30%) for all the proxies for 80% of the time. Nevertheless, proxies 11252 and 18202 have significantly higher traffic.

Figure 3.7 shows the distribution of the request processing throughput, as defined in Table 3.2. We observe that all proxies have almost identical distribution but around different mean values, depending on the individual characteristics of the proxy. Moreover, we can see that a significant percentage (>60%) of the time proxies serve at a very narrow range of processing throughput, meaning they can offer a stable service. Even in the worst cases, the ser-

vice does not suffer from extreme degradation, while remaining higher than 100Kbps 80% of the time. We also observe that for proxies 3982 and 11252, the processing throughput distribution resembles the number of requests distribution in Figure 3.5 possibly indicating, as before, that the proxies are not saturated.

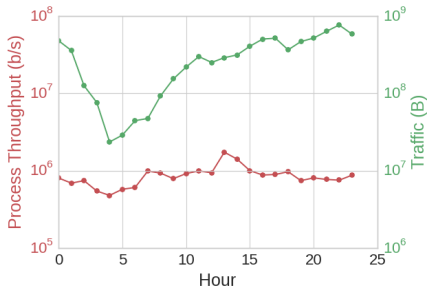


Figure 3.8: Daily average request processing throughput compared to traffic

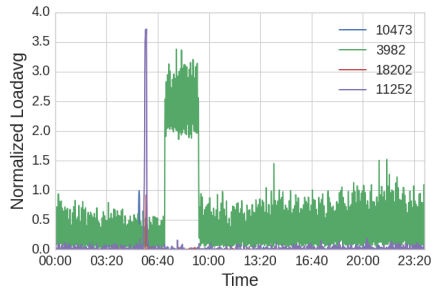


Figure 3.9: Daily Median Loadavg per proxy normalized by #CPUs

To gain a more complete perspective we also studied the daily aggregates of the traffic, users and requests clearly observing not only the expected human daily pattern but also a clear effect of the different way each proxy receives and serves request as a result of the users' manual proxy selection. Moreover, studying the mean daily patterns, we noticed that, as seen in fig. 3.8, that the processing throughput presents very small variations implying a stable service behaviour. Furthermore, the traffic volume varies more than 1.5 orders of magnitude. The fact that the processing throughput is not affected by the traffic size confirms our observation that the servers are not saturated. Additionally, in order to verify that the processing capabilities of proxies are not a bottleneck for the service, we monitored the proxies' CPU using the *loadvg* Linux metric. The results, showing a strong daily cyclic pattern, are summarized in Figure 3.9 that shows the daily median of the per-minute loadavg for each proxy normalized by the number of CPUs. Except from 3982, affected by other co-located network services, the proxies are not overloaded. The brief daily peak in each proxy is due to the daily restart of the proxy that includes a cache reindexing.

graph	nodes	edges	degree max/mean /min	diameter
base-graph	902	914	98/2.04/1	11
proxy-clients-graph	463	472	60/2.04/1	10
backbone-graph	47	56	10/2.38/1	9

Table 3.4: Summary of Llucanes network graph

Even at that small scale, we observed the daily cycle of human activity with preference for evenings and really reduced traffic during the first hours of the day. The pattern is visible in all the described metrics in different degrees.

From all the above we can conclude that the proxies are able to offer a stable service, with respect to the traffic load, allowing them to be used as an alternative domestic Internet connection. Moreover, in our concrete scenario, the network capacity of the proxies is underutilized assuming that no other services co-located in the host of the proxy are heavily using the Internet network capacity.

3.5 The Local Network Viewpoint

The local network infrastructure has also an influence in the final user experience. For the analysis we used information extracted from odd day logs that provide these details while hiding URL destinations.

Network structure: For the local network we considered all operational nodes and links of the Llucanes guifi.net zone[‡]. We refer to the entire zone network as the base-graph. Moreover, we refer as Proxy-Clients graph to the part of the Llucanes network including only the nodes (clients, routers, proxies) that participate in the proxy service. More information concerning the network structure, hardware characteristics, and protocols used in guifi.net can be found in [14].

Similarly to the rest of Osona county zones, and in general to many rural community network deployments, the network consists of a small set of in-

[‡]More information on the Llucanes zone `https://guifi.net/en/node/8346/`

terconnected routers, the backbone graph, where each router is connected with a large number of end nodes, most of all wireless links, mainly 802.11b Wlan connections [14]. Users access the entire guifi.net network from the end nodes. Some of the routers act also as hosts for various guifi.net services, including the proxy service. Table 3.4 describes the main characteristics of the aforementioned graphs. We notice that the mean degree of the base-graph and of the proxy-clients-graph is very low since the end-nodes with degree 1 dominate the distribution of degrees. The low mean degree value in the backbone-graph is more interesting though, since it implies that the majority of the routers have only two neighbours. Figures 3.10 and 3.11 provide a view of the Proxy-Clients graph and the backbone-graph. The colors of the participating nodes and routers indicate that they are using the proxy with the same color. Moreover, in fig. 3.11 the darkness of the link color denotes the cost in latency for a byte to cross this link, therefore the darker the color the more expensive is the link to use.

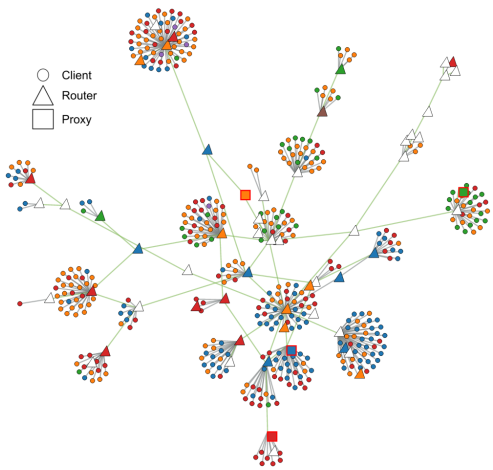


Figure 3.10: Llucanes Proxy/Clients

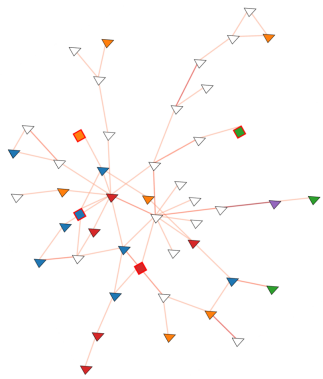


Figure 3.11: Llucanes Backbone

Network usage: Since the selection among proxies is static (manual configuration), the analysis of local network usage can show the effect of selection

on local network usage and the perceived user experience. Towards that end, we first analyse metrics of distance between the users and the proxies. Figure 3.12 shows the distribution of the number of hops between the users and the selected proxies. The distribution is almost uniform for 95% of the users with values between 1 and 6 hops. The remaining 5% is split between 7 and 8 hops. Nevertheless, we observe that manual choices result to a slight increase in number of hops, therefore possibly introducing small unnecessary overheads. The latency involved, depicted in fig. 3.13, shows a different behaviour. Almost 80% of the users experience an average latency smaller than 15ms to reach their proxy. The remaining 20% lies between 20ms to 35ms. Despite the almost uniform distribution of hops, latency values vary much less, implying that during normal network conditions, the distance between the users and proxies is not significantly deteriorating the user experience for web services.

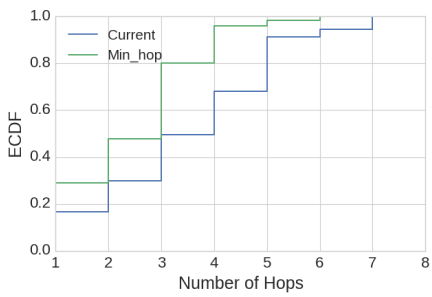


Figure 3.12: Number of network hops between users and their selected proxies

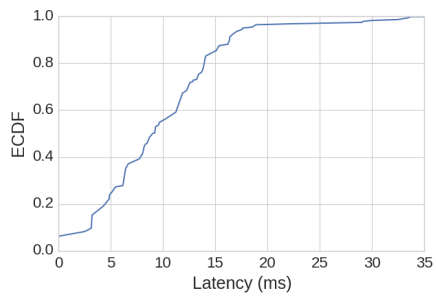


Figure 3.13: Average latency between users and their selected proxies

Download throughput: As we described earlier, the request processing throughput is calculated in the proxy based on the request elapsed time, which includes the time the proxy requires until sending the last byte of the web object to the client. Therefore, any significant local network deterioration affects the throughput behaviour. Based on this observation we can utilize the request processing throughput metric for objects larger than 1MB, in order to estimate significant deterioration on the user experience. Including smaller

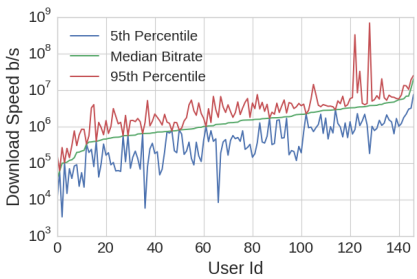


Figure 3.14: Estimation of user experience throughput with objects >1MB

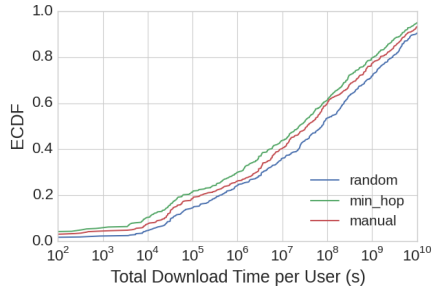


Figure 3.15: User cost as sum of download times (1 month)

objects would give unreliable throughput results due to the major influence of network buffering in the proxy, DNS caching and network latency variations for short connections. Figure 3.14 illustrates the individual user experience in throughput. We estimated from proxy logs the download speed for objects larger than 1 MB. A simplifying assumption is that users focus on few or a single large object at a time. If so, our measures could be taken as a lower bound for the experienced individual download throughput. Median values of download throughput appear quite stable with median values ranging from 0.1Mbps to 10Mbps for different users. Quite good result for the many users of a free crowdsourced service.

Furthermore, in order to show the margin for improvement in the user experience using other proxy selection strategies, we simulated the traffic of the users using a min_hop and a random strategy taking into account local link latencies. As seen in Figure 3.15, the total download time of each user throughout the month in the manual selection is asymptotically better than the random selection while asymptotically worse than the min_hop selection. Considering that the proxies are not the bottleneck, this result shows that a proxy selection mechanism would improve user experience of the proxy service. Nevertheless, we plan to extend our simulations taking into account the proxies processing and download speed.

3.6 Related Work

Most work on wireless networks focuses on usage traffic patterns, link level characteristics and topologies, but not user experience, e.g. MadMesh [39], Google WiFi [31] and Meraki [8] networks. In these studies, Internet access is direct instead of using proxies, and these wireless networks are homogeneous. Thus, measurement results cannot easily be compared with this. In the Google WiFi and MadMesh transfer rates are limited to 1 Mbps, but 80% getting less than 80Kbps in Google WiFi. In MadMesh 80% get less than 1Mbps with 85% of the clients connected within 3 hops to Internet, comparable with our results that achieve higher speed but more hops to a web proxy.

The evaluation of Facebook's Free Basic Service [4] shows comparable performance (80-600Kbps for FB vs. 0.1-10 Mbps median speeds) better in our case, despite significant differences: in clients (mobile devices vs. any device), access network (cellular mobile carrier vs. wireless fixed community network), web proxies (centralized large servers vs. distributed small servers with network locality), and web service and content providers (redesigned and optimized vs. unmodified content).

The web proxy business has changed significantly over the years. The percentage of cacheable content has been decreasing, coupled with a dramatic increase of HTTPS traffic. The performance of web proxies is not only about high-level metrics such as hit rates. Low-level details such as HTTP cookies, aborted connections, and persistent connections between clients and proxies as well as between proxies and servers have a visible impact on performance, particularly in heterogeneous bandwidth environments [50]. In [26], authors analyse a mobile network topology with a two level cache hierarchy. Their claim that a caching system can be efficient when only 5.1% of traffic is suitable for caching, what shows that caching in our case with lower rates may not be that beneficial.

Wireless network user experience has been characterized previously. The first [29] focuses on web traffic and the use of proxies to access Internet con-

tent in rural areas. Five years ago, using a single high latency and slower VSAT Internet connection (64-128Kbps) obtained RTTs sometimes over 10 secs, closer to a DTN case, and cache hit rates of 43%. There are complementary lessons, about security or that content from CDN is usually not cacheable, but the scenarios are too different. The second study [28] looks at web traffic patterns and content caching. They mention the decreasing cache hit rates over previous studies, even lower in our study 5 years later with a dramatic increase of HTTPS traffic.

3.7 Summary of Lessons Learned

The analysis of the guifi.net proxy service describes a crowdsourced, social solidarity driven, free basic Internet service built from many small proxy servers spread across a regional community network, contributed by locals for locals. These proxies act as gateways to Web content and DNS, that can be cached and shared among clients or act as middleboxes for HTTPS transfers, the majority of traffic. Being in the middle can also help protect the privacy of clients.

The analysis confirms the trend to non-cacheable content, small cacheable objects, and therefore small object caches that can even fit in RAM. Proxies have a small number of clients, ranging from 14 to 101 per day. Moreover, there is a good balance of traffic and number of clients per proxy despite the manual proxy selection, driven by locality (same zone), client choice and advice from neighbors. The system is simple and resilient since each proxy is independent and clients just switch to their next choice in case of failure of their proxy.

The service has satisfactory performance (0.1-10 Mbps, good client-proxy latency), with no perceived Internet, access network or service congestion, despite the typical daily patterns of usage. That can be attributed to the use of small servers spread over the regional access network, close to end-users with locality preference. Nevertheless, scaling or coordination between

services in different zones does not seem trivial.

4

A Web Proxies regulation mechanism for CNs

4.1 Introduction

As a consequence of the lack of regulation, presented Chapter 3,and despite being a critical service for the community, current proxy gateway services are quite fragile. As described in 1.4 one of the problem components concerns the relation between the users' and the proxies. More specifically, in this chapter we present and evaluate a passive user-side distributed system that optimizes the client-gateway mapping. Moreover, considering that, CNs and WMNs consist of heterogeneous technologies and combine diverse routing protocols, agnostic to underlying infrastructure and protocols, requiring no modification of proxies or the underlying network. To our knowledge, Network-aware state-of-art proxy selection schemes for WMNs do not work in this heterogeneous environment. Our selection mechanism avoids proxies with heavy load and slow internal network paths. The overhead is linear to the number of clients and proxies.

For the performance estimation we propose two metrics, one to estimate proxy service latency (see Sec. 4.4) and another client-proxy path (see Sec. 4.3) latency. An extended Vivaldi mechanism is used to indicate client-proxy path performance and the Time-To-First-Byte (TTFB) moving average of their HTTP requests to indicate proxy performance. Second, we propose a mechanism where clients use these metrics to rank proxies and use these indicators to select the top ones in terms of QoE, or to switch to the next best proxy when performance degrades. This mechanism is client-side (see Sec. 4.2), it avoids hotspots (see Sec. 4.6) and has a low overhead (see Sec. 4.5).

The metrics and the client selection mechanism were instantiated in the Community-Lab.net experimental testbed in nodes acting as clients inside guifi.net interacting with a set of guifi.net web proxies.The result from experiments show that our procedure is sound: our method is able to provide good measures of client-proxy and proxy-Internet latencies and follow its variability. We found out that our client selection mechanism is cost-effective in finding out proxies that result in good web performance and QoE for clients. Our results improve in cost-benefit over other quick-to-measure alternatives (such as Vivaldi-only and minimum hops) and less costly in traffic and delay than slower performance-oriented measures.

The rest of this chapter is structured as follows: Section 4.2 describes the approach, system model, experimental environment. The measurement of network performance is discussed in Section 4.3 and proxy performance in Section 4.4. Proxy selection is presented in Section 4.6. Section 4.5 provides an analysis of overhead. In § 4.7 we discuss the related work and we finally conclude in Section 4.8.

4.2 Overview

Our goal is to design a practical, non-optimal but best-effort, scheme where clients can select a proxy using network and proxy performance metrics that would not require the modification of any network components and that could

function in a heterogeneous environment. To this end, we implemented an estimation-based framework, where clients cooperate sharing their network and proxy performance estimations in order to prioritize their list of known proxies, being able to make an informed proxy selection. Unlike other proposals, the framework does not try to find an optimal client-proxy assignment, but helps clients to avoid bad choices that would significantly degrade their service experience. The non-optimality is the price we have to pay in order to achieve a scalable solution that can be applied in real heterogeneous WMN preserving a low overhead. More specifically, we present a proxy selection framework, which using information from a network performance estimator and a proxy performance estimator as shown in § 4.6, can select good proxies and can manage to avoid proxies that are overloaded, or have very slow Internet connections, or are located behind very slow internal mesh paths.

The network performance estimator provides estimates of client-client and client-proxy network latency. It is a Vivaldi network coordinates system based on [45], extended similarly to [41] in order to estimate the round-trip latency of nodes that are not part of the Vivaldi network – the proxies. All the clients of the proxy selection system participate in the Vivaldi network and thus, exchanging a small amount of messages periodically they maintain an updated view of the latencies across them. Moreover, each client periodically has to monitor one of the proxies and share this information with the rest of the clients. As we demonstrate in § 4.3, these measurements are sufficient to allow the clients to create a preference list ordering the proxies according to their network latency.

The proxy performance estimator provides estimates of the load of the proxy, concerning the quality of the service currently provided. It is based on the widely used practical assumption that the TTFB of an HTTP request can reflect the service performance [10, 20]. In our framework, each client is passively calculating the TTFB of the HTTP replies that he receives from his proxy. Then the client can use this value to estimate the load of his proxy and share it with his Vivaldi neighbours. As we present in § 4.4,

this mechanism allows clients to avoid proxies with heavy load or high delay Internet connections.

4.2.1 System Model

For the description of the model we assume a static topology in a wireless mesh network. We make no assumptions about the quality of the mesh network, and we allow dynamic link conditions (a very slow link is indistinguishable from a very congested link). We use latency as our metric of load, both for links and proxies.

Let C denote the set of clients, and P denote the set of proxies. For every request that a client $c \in C$ is sending to a proxy $p \in P$ the experienced latency is:

$$t_{lat} \approx t_{request\_c\_p} + t_{proxy\_p} + t_{response\_c\_p} \tag{4.1}$$

where $t_{request\_c\_p}$ represents the time required by client c to connect to proxy p and send the request. It is proportional to the round-trip time between c and p, $t_{mesh\_rtt\_c\_p}$:

$$t_{request\_c\_p} \approx A * t_{mesh\_rtt\_c\_p} \tag{4.2}$$

The $t_{mesh\_rtt\_c\_p}$ latency depends on the network conditions of the chosen path between client c and proxy p. For the rest of this chapter we will assume that A equals to 2, which corresponds to the client-proxy TCP handshake and the HTTP request.

The t_{proxy_p} latency represents the total time that proxy p needs to process the request until he initiates the request to the remote server. This includes the time that the request is waiting before starting to be served, which is a good indicator of the load of proxy p, as it correlates directly with the number of outstanding proxy requests yet to be served. We assume that at a given point in time different clients experience the same t_{proxy_p} if they use proxy p, independently of who is measuring it - § 4.4 validates our assumption.

Finally, $t_{response\_c\_p}$ is the time that proxy p takes to complete the HTTP request. This time depends on the load and capacity of the proxy's Internet connection and on the latency to access and retrieve the content, related to the distance from the content and content availability. From all the above, we deduce that the request latency can be approximated by:

$$t_{lat} \approx 2 * t_{mesh\_rtt\_c\_p} + t_{proxy\_p} + t_{response\_c\_p} \tag{4.3}$$

We argue that t_{mesh_rtt} and t_{proxy} can be used together to provide clients with a good preference indicator allowing him to avoid loaded proxies and proxies located behind slow paths. In § 4.3 we describe how we use Vivaldi to estimate t_{mesh_rtt}. In § 4.4 we elaborate on how TTFB can be used to estimate t_{proxy}.

4.2.2 Experimental Environment

In order to assess our decisions, we experimented separately with each component of our solution. Following the practical approach of our work, we decided to perform our experiments in guifi.net, under real mesh network conditions. For the experiments, we were given access to 5 end-nodes across different guifi.net mesh networks and 3 proxies that are also being used by the guifi.net users. The nodes and the proxies are distributed in various locations in Catalonia, Spain. Despite the small scale of our experiments, we are still able to assess the behaviour of the presented components.

As explained in § 4.3, proxies do not actively participate in the measurements, they nevertheless need to respond to UDP pings allowing clients to estimate their round-trip latency. Therefore it is worth mentioning that for the results presented here we used a UDP echo server in the proxies, obstacle which can be practically overcome with tools such as [47].

4.3 Network Performance

In this section we describe and demonstrate how an extended version of Vivaldi [45] can be used to estimate the current performance of the mesh network, expressed as a latency metric, helping the clients to avoid overloaded paths.

Each client in our system participates in a Vivaldi network to estimate his round-trip latencies to the other clients. Moreover, implementing the ideas described in [41] modified for providing more accurate estimates, we allow Vivaldi to monitor nodes external to the Vivaldi network. We show that this allows the clients to maintain an updated estimation about their latency towards each of the proxies, excluding though the proxies from the Vivaldi network. Although Vivaldi was designed to predict latency between hosts in the Internet (mostly wired), we show that it can also coverage and be used to predict latencies in WMNs despite the RTT variations caused by the wireless environment.

Vivaldi estimates RTT by performing ping between nodes. Each Vivaldi node maintains a list of $C + R$ *neighbors*: C that are estimated to be closest, and R other random nodes, located anywhere in the network. The algorithm works in *rounds*, which are triggered every T seconds. In every round, a node randomly selects a neighbor from the list, performs N UDP pings to him, and asks him to send back its own and its neighbors' coordinates. The variables C and R can be tuned depending on the size of the network and the topology in order to increase random/remote node discovery or create strong local clusters. The variable N affects the accuracy of the prediction in exchange for the ping traffic overhead.

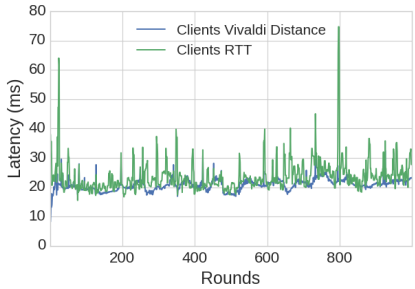
In addition, as mentioned, we can also satisfactorily predict round-trip latency from a Vivaldi node to each proxy, even if they do not actively participate in the network coordinates system. To achieve this each Vivaldi node maintains coordinates that represent $C + R$ proxies, as described above. In every round, a node performs N UDP pings to a proxy p, selected in a similar manner that he selects neighbours. Then, he updates the coordinates

he maintains for p and shares the measured latency with his selected neighbor for this round. Then, the neighbour updates the coordinates he maintains for proxy p as described in [41].

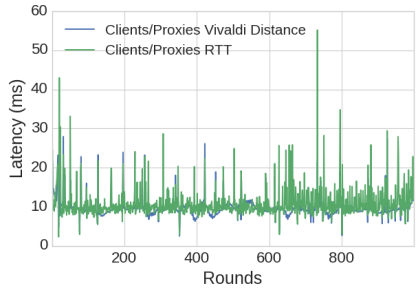
4.3.1 Measuring network performance

Similarly to [45], we define the error of a path as the absolute difference between the predicted RTT for the path (using the coordinates for the two nodes at the ends of the link) and the actual RTT. We define the error of a node as the median of the path errors for paths involving that node. We define the error of the system as the median of the node errors for all nodes.

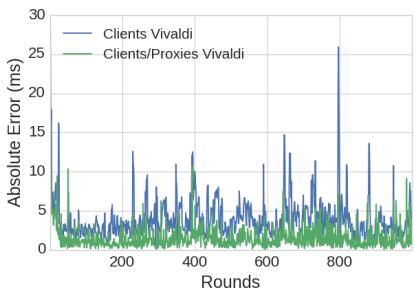
We experimented, using the described environment, in order to characterize the behaviour of the Vivaldi coordinates in a wireless mesh network. First, we performed an experiment where clients are using Vivaldi to estimate the latencies between them and the extended version of Vivaldi to estimate their RTT to the proxies. This way we can understand the predictive potential of the selected algorithms. It is worth mentioning that our experiment was executed in nodes that participate in a real network and therefore were processing real network traffic and using shared mesh links. Figures 4.1a and 4.1b show the real and predicted latency between clients throughout the experiment. The median latency between the clients was 22.29 ms, while the median predicted was 20.82 ms. The median latency between clients and proxies was 9.8 ms while the corresponding median predicted was 9.36 ms. Figure 4.1c depicts the absolute prediction error of the Vivaldi estimation between clients as well as the one between clients and proxies. We observe that the error of the latency prediction between clients and proxies is lower. This fact can be attributed to the smaller variation of the real latency between clients and proxies, but also to our described improvements in [41] . The empirical cumulative distribution function of the prediction absolute errors, as seen in fig. 4.1d, helps us observe that the median absolute error of the predicted latency between clients is 3.37 ms while 80% of the experiment time the nodes present a median error of



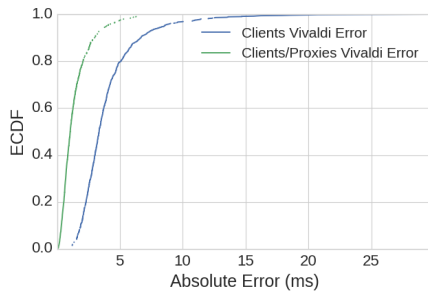
(a) Comparison of median RTT and median Vivaldi estimated latency between clients.



(b) Comparison of median RTT and median Vivaldi estimated latency between clients.



(c) Absolute error in estimated RTT for Vivaldi nodes and proxies.

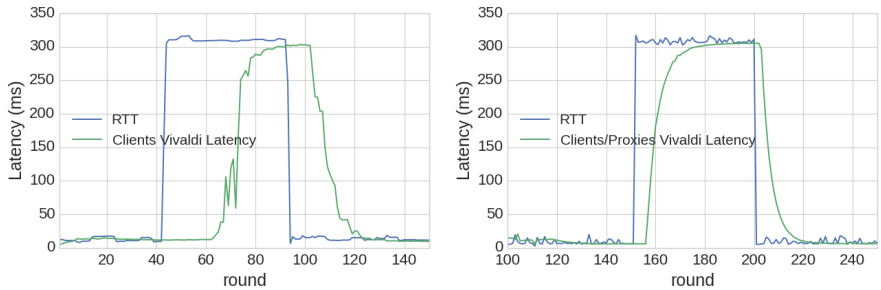


(d) Empirical cumulative distribution function for the absolute error in the estimated RTT for Vivaldi nodes and proxies.

Figure 4.1: RTT and Absolute Error

less than 5 ms. As far as client-proxy Vivaldi latency prediction is concerned, the median absolute prediction error is 1.07 ms while 80% of the experiment time the nodes present a median error of less than 2.5 ms.

In our second experiment we tested the ability of Vivaldi, extended version as well, to adapt to network changes. Figure 4.2a shows that there is some delay in Vivaldi adapting to latency changes between the clients, taking around 30 rounds to adjust its estimates to be over 200 ms. However, as seen in fig. 4.2b, proxy estimates are much faster to adapt, taking around 12 rounds to re-adjust the estimates.



(a) Timeline showing the changes in Predicted RTT reflecting the changes in real RTT for clients.

(b) Timeline showing the changes in Predicted RTT reflecting the changes in real RTT for proxies.

Figure 4.2: Delay Proxies and clients

We show that our system can estimate the round-trip times between clients, as well as between clients and proxies with error less than 5 ms and 2.5 ms respectively, under real mesh network conditions. These low prediction and triangulation errors (median relative error in the range of 10%) are comparable to the original Vivaldi on the Internet. Moreover, we demonstrated that our estimation can eventually trace serious anomalies in the latency of paths. Therefore, we argue that these estimates are satisfactory in order to prioritize paths from clients to proxies that present differences in latency higher than 5 ms and avoid highly loaded paths.

4.4 Measuring Proxy Performance

In this section we describe and show how TTFB can be used to estimate the current performance of the proxy, expressed as a latency metric, helping the clients to rank choices, avoiding overloaded proxies and proxies with Internet connection that exhibits high delays.

TTFB has been widely used in real deployments but also in recent Internet measurement research [10, 20] to indicate the responsiveness of a web service since it combines the TCP connection time and the remote server processing

time. TTFB is a useful web performance estimator since it is measured passively on the client-side, leveraging information from the already existing client traffic. Nevertheless, our scenario is more complicated, since we aspire to utilize TTFB measurements on the client-side to estimate the performance of the proxy that mediates between the client and the requested content.

Assuming that t_{proxy_ttfb} is the time the proxy needs to receive the first byte of response from the remote server then $t_{response}$ from eq. (4.3) can also be expressed as:

$$t_{response\_c\_p} \approx t_{proxy\_ttfb} + t_{transport\_response} \tag{4.4}$$

where $t_{transport\_response}$ is the time until the client has received the complete response. Both t_{proxy_ttfb} and $t_{transport\_response}$ depend on the available bandwidth of the Internet connection of the proxy, and the delays in the path from the proxy to the destination server, as well as the responsiveness of the remote end-server. Additionally, $t_{transport\_response}$ depends on the performance of the path between the client and the proxy. Considering eq. (4.3), the TTFB as measured on the client-side can be expressed as:

$$t_{ttfb\_c\_p} \approx 2 * t_{mesh\_rtt\_c\_p} + t_{proxy\_p} + t_{proxy\_ttfb} \tag{4.5}$$

t_{proxy_ttfb} differs depending on the proxy, the remote server and the requested content. The analysis of the variability of different t_{proxy_ttfb} latencies, related to how well the proxies are connected to specific remote servers, lies beyond the scope of this work. Therefore, in our current work we choose not to study t_{proxy_ttfb} and assume it is stationary for each proxy, representing the delays in the proxy's Internet connection. Nevertheless, as part of our future work we plan to investigate whether and how it is possible to create an estimation model, where each client will be able to use his current and previous HTTP connections to various remote servers in order to identify how this

metric affects the measured TTFB. For the rest of this chapter we are assuming that all the clients are trying to access the same content that is always available, located in remote servers in similar distance from all the proxies and all the proxies have the same Internet connection bandwidth capacity.

Therefore, based on eqs. (4.3) and (4.5), the latency incurred by the proxy could be expressed as:

$$t_{proxy\_p} \approx t_{ttfb\_c\_p} - 2 * t_{mesh\_rtt\_c\_p} \tag{4.6}$$

t_{proxy_p} can provide us with an estimation of the proxy performance, calculated by eq. (4.6) with the measured TTFB on the client-side and the network. However, the TTFB measurements can be very noisy (sometimes packets are significantly delayed due the proxy or network load, or proxies may complete a request quickly despite heavy load). To minimize the effect of noise in our estimation, we filter the obtained t_{proxy_p} values with an exponential moving average which can be tuned by a parameter α. If the value of α is too high, the effect of noise in the measurements leaks into the filtered value, while if α is too low, the filtered values adapt slower to the measured real values, smoothing the peaks and valleys. Moreover, measuring periodically the TTFB of HTTP requests, we have to handle delays that are higher than the measurement period. To this end, we developed a penalty scheme, assuming that the request will eventually be completed, our scheme is based on the simple idea that the TTFB will be at least as high as the time that the client waited for it. Thus, if a client has not received the first byte for longer than the last t_{proxy_p} value then the estimated value keeps increasing in every measurement period until it is received.

Clients periodically exchange the calculated t_{proxy_p}, thus reducing the need for probing, as the value indicates how good a proxy is at serving requests for any client. These messages are forwarded through the Vivaldi network. Currently, we assume that the client is performing HTTP requests sequentially. However, this is not a realistic assumption, since in a typical

scenario a browser is generating multiple parallel HTTP requests which can target different servers. As part of our future work we plan to investigate how to choose or combine measures from multiple HTTP transfers to estimate a TTFB value for each proxy.

4.4.1 Measuring Proxy Load And Internet Connection Delays

In our first experiment we evaluate the relation between the t_{proxy_p} and the proxy load. The proxy load is represented by various variables monitored on the proxy, including the CPU and the number of incoming and outgoing packets per second in the internal and the external interfaces. Figure 4.3 allows the comparison between the normalized median of the proxy variables compared to the estimation and the extended estimation of t_{proxy_p}. The proxy is loaded with external requests 150 seconds after the beginning of the experiment and t_{proxy_p} starts presenting high peaks while the extended estimator presents a more clear relation to the load behaviour. Figure 4.4 presents another perspective of the relation between the proxy load and the extended estimator, including the plot of the Principal Component Analysis which demonstrates that the higher the load values are the higher the values of the extended estimator. As a result, we can argue that our extended estimator is behaving similarly to the proxy load, and therefore we can claim it can be used to detect heavily loaded servers.

The goal of our second experiment was to evaluate how our estimator responds to proxies with Internet connections that have significant delays. To achieve that, we introduce artificial network delay in the external network interface of the proxy. As seen in fig. 4.5, both the simple and the extended estimator successfully measure the introduced delay. Nevertheless, the extended estimator appears to need more time to return to the normal levels, as expected. Therefore we verify that our estimators are responsive to the proxies' Internet connection delays.

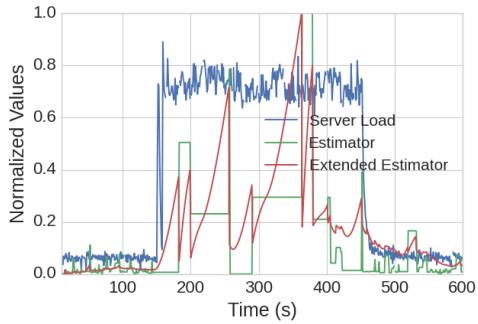


Figure 4.3: Median comparison of the normalized proxy load metrics with a clients' normalized Extended TTFB (α 0.05)

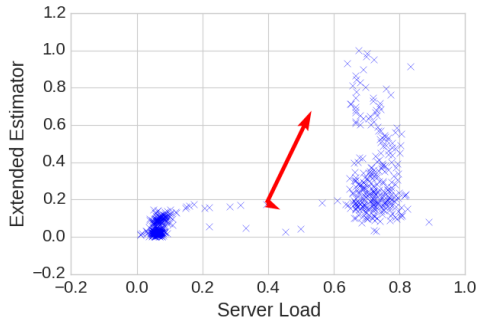


Figure 4.4: Principal Component Analysis of the median of the proxy load metrics compared to the Extended TTFB (α 0.05)

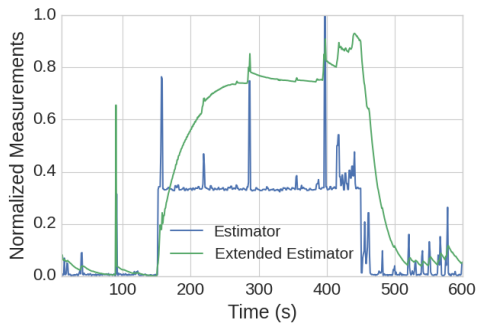


Figure 4.5: Median TTFB and Extended TTFB (α 0.05)

4.4.2 Sharing TTFB

Despite the fact that our estimators behave similarly with the proxy load, we need to verify that the estimator measured from client c for proxy p can be

useful for other clients as well. To investigate this issue, we performed an experiment where one single proxy was used that was serving all the nodes. Figure 4.6 represents the Spearman's rank correlation coefficient[52] between the extended estimators of the different clients throughout the experiment. Spearman's rank correlation coefficient targets to identify correlations that can be expressed by a monotonic function, thus resulting in high values, as we observe in our result, when both of the compared sets ascend or descend similarly.

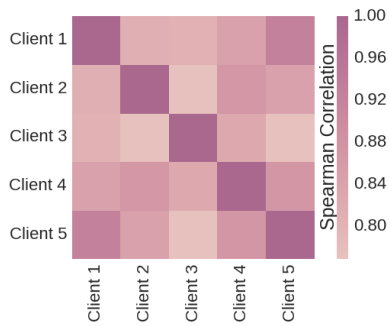


Figure 4.6: Spearman's rank correlation heatmap of the various clients Extended TTFB estimator (α=0.05)

The described proxy performance estimator is not an accurate estimator in terms of absolute values, but has a behaviour similar to the proxy load, enabling the client to rank choices and avoid saturated proxies. Moreover, the extended estimator calculated by one client behaves similarly throughout the different clients and can, thus, be disseminated across them reducing the overhead and allowing clients to have updated information concerning proxies they are not currently using.

4.5 Overhead Analysis

The two performance estimation components of our system function in parallel. Thus, the total overhead is:

$$overhead = overhead_{vivaldi} + overhead_{ttfb} \tag{4.7}$$

According to the challenges that Vivaldi [45] faces by design, a network coordinates system should produce a minimal amount of overhead traffic when probing. The overhead network traffic generated by Vivaldi is, in bytes per second:

$$overhead_{vivaldi} = (2 * ping_{size} * ping_{freq} + data) * n \tag{4.8}$$

$$data_{vivaldi} = (n_p * 160 + n_n * 160 + 10)/round_{period} \tag{4.9}$$

$$ping_{freq} = round_{pings}/round_{period} \tag{4.10}$$

In the formulas above, n is the number of nodes in the Vivaldi system and n_n and p_n are, correspondingly, the maximum number of known neighbours and proxies. We can see that the overhead of Vivaldi increases linearly with the amount of participants. Vivaldi works in rounds: every round each node sends a few pings to each of its neighbors, and rounds occur every few seconds. In our deployment we use 8 pings per round, with a round starting every 10 seconds. Moreover, in our case it corresponds to one neighbour plus one proxy, and the maximum number of neighbours and proxies is 8. That equates to 436 bytes per second per client, which is acceptable even in a wireless mesh network environment. For example, assuming all the 30,000 nodes of guifi.net were clients, the overhead would be approximately 1.5 MB/s distributed all over the network, which sums up to be 1.6% of the average daily incoming Internet traffic [6](data from 2015).

The gathering of TTFB metrics is passive for the proxy currently selected by the client, and then shared between the nodes of the system. Nevertheless,

we may ping a proxy if we haven't had any metrics for a a certain time period as described in § 4.6. The network overhead of the proxy TTFB protocol is, in bytes per second:

$$overhead_{ttfb} = O(proxies) * payload/timeout \tag{4.11}$$

$$payload = payload_{request} + payload_{response} \tag{4.12}$$

$$timeout = m_1 * proxy_{distance} + m_2 * num\_closer + b \tag{4.13}$$

Whenever a client overcomes the personalized *timeout* in the proxy information, we query it. If we set the m factor too low, the information won't have time to propagate and many nodes will query the proxy. However, if we set the m factor too high, it may take a long time until a node is finally queried.

Due to the randomly selected neighbors, let us make the assumption that any node may be connected to any other node. Let us assume a single node pings the proxy. Then, in the next round (assuming a synchronous model), any of the other $N-1$ nodes may query this *knowledgeable* node with probability $1/(N-1)$, pulling the desired proxy information.

The number of nodes learning the desired information at a given round can be modelled through a binomial distribution with $p = k/N$, where k is the number of nodes that possess said information. The expected value is k – we expect k nodes learning the information at each round. This means that we expect all nodes, on average, to learn the information after $log_2(N)$ rounds. For the $30,000$ nodes currently registered in guifi.net, that equates to 15 rounds.

It is worth noting Equation (4.11) assumes m and b parameters are correctly tuned so that the proxy is contacted by a very low number of nodes with high probability.

4.6 Proxy Selection

After describing our approach for measuring the performance of the network and the proxies, in this section we describe how clients are able to select prox-

ies informed by the presented metrics. Moreover, we present an experiment where clients, adopting our solution, manage to avoid overloaded proxies, very slow internal paths and very slow Internet connections, where as if a minimum hop or minimum delay approach was to be adopted the clients would not be able to avoid service deterioration.

On top of our performance estimation tools we built an application-level proxy selection platform. Each client maintains a *proxy selection table*, similarly to a routing table, where each line corresponds to a known proxy and contains the estimated distance, as described in § 4.3, the extended estimation of the proxy latency, as described in § 4.4 and the number of hops to that proxy. Based on this information various proxy selection strategies can be implemented. Nevertheless, the implementations need to take into account the described sensitivity of the provided estimators.

We used the provided estimators to implement a proxy selection strategy suitable to their rationale, aiming to avoid saturated proxies, proxies with saturated Internet connection as well as proxies behind saturated paths. To achieve that, the selection strategy orders the proxies according to the sum of the network latency estimation and the proxy latency estimation, selecting the lowest value. Our implementation avoids unnecessary oscillations by defining a minimum threshold which should be overcome in order to change the selected proxy. Additionally, we implemented a recovery mechanism for situations where a proxy is not being used by any client for a significant amount of time, therefore his current performance estimation value is unknown. In order to prevent all the clients from querying the proxy at the same time, the clients maintain a personalized timeout that depends on a global recovery time, the locally last known measurement of the proxy and their personal mesh distance to that proxy. If the timeout is reached without receiving any updates the client is actively probing the proxy to learn its know TTFB value. This way, we manage to make clients that are close to the proxy in charge of querying it and then propagate the information to the other nodes.

In order to evaluate our minimum load selection strategy we implemented

two simple proxy selection strategies based on the minimum hop metric and the minimum network delay metric, that were used to compare to the minimum load solution. Under the minimum hop strategy each client selects the closest proxy in terms of hops while in the minimum network delay strategy the clients select the proxy that has the smallest Vivaldi latency estimator.

The objective of the evaluation experiment was to describe how the different strategies of the clients deal with the disruptions of the provided service. The clients use the proxies selected by the routing strategies to repeatedly download files of 1 Mb from the same remote server choosing every 10 seconds, our Vivaldi period, a new proxy if necessary. The value of 1Mb was chosen because in normal conditions a client needs less than the period of 10 seconds to download the file, therefore we can evaluate more accurately the selection alterations. We adopt as evaluation metric the download time experienced by the clients. The experiment lasted 1600 seconds and was repeated for each strategy. Between 50 and 350 seconds we introduce a high amount of requests in one of the proxies. Between 550 and 850 seconds we simulate in one of the proxies an external Internet connection with high delays. Between 1050 and 1350 seconds we simulate a slow mesh path in one of the proxies.

Figure 4.7 depicts the median clients' download time per strategy. We observe that our strategy leads the clients to experience a very small amount of download time peaks, especially compared to the static *min_hops* solution. The y axis of the plot is limited to 2 seconds in order to allow easier comparison, nevertheless the overall distribution of the values can be seen in fig. 4.8. As depicted, *min_hop* and *min_delay* present higher average values compared to *min_load* (0.76s,0.71s and 0.48s respectively). Most importantly, related to avoiding overloaded options, *min_hop* and *min_delay* have many more and significantly higher peaks - than *min_load* (maximum 6.33s,4.89s and 1.23 respectively), especially *min_hop* that is a static strategy. *min_load* manages to minimize the number of peaks, confirming our argument that it succeeds to avoid the loaded options. The manner in which *min_load* is avoiding the loaded options is also shown in fig. 4.9, where we observe that

clients avoid the loaded by requests *proxy_3* between 150 and 350 seconds, as well as *proxy_2* in the ranges of 550-650 seconds and 1050-1350 seconds where we simulate the mesh path delay and Internet connection delay respectively. It is also worth pointing out that in the performed experiment *min_delay* and *min_hop* do not appear to be affected by some overloaded conditions introduced, but this is a result of the specific experiment conditions (network latencies and distances) and not of their ability to avoid it.

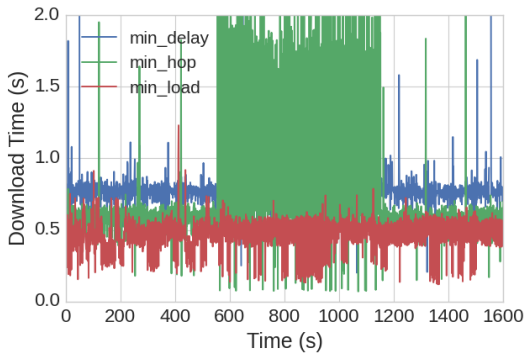


Figure 4.7: Median client download time for 1Mb per strategy

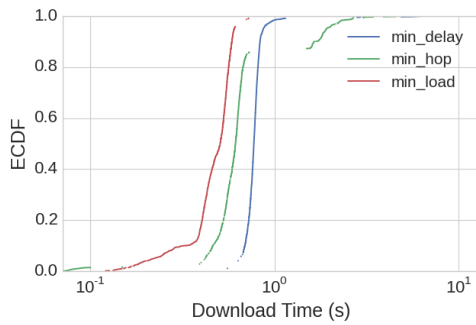


Figure 4.8: Empirical Cumulative Distribution Function comparison between the median time to download 1Mb per strategy

The results we presented in this section verify how the performance estimators presented in the previous sections can be used by clients to rank and make informed choices from a large set of proxy Internet gateways, avoiding

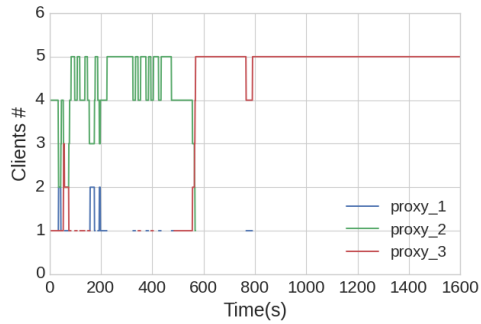


Figure 4.9: Number of clients per proxy for `min_load` strategy

proxies that would deteriorate their user experience.

4.7 Related Work

Proxy selection in wireless mesh networks is strongly related with the topic of gateway selection in wireless mesh networks which has been extensively studied in the past. The solution presented here for web proxy gateways and HTTP requests is applicable to IP tunnel gateways and IP flows. The works presented in [25, 32] fail to function in heterogeneous environments and to avoid infrastructure modifications since they present solutions that operate in the mesh routing layer which are inherently prohibitive for heterogeneous environments, while they require modifications in the infrastructure routers. [34, 40] require additional software in the side of the gateways. All the works mentioned, despite the fact the presented solutions are interesting, they lack practical implementation and/or testing in real environment. An exception to the above, and closer to our work is [17], where the clients cooperate to probe the gateways and then use the results to select a proxy. Furthermore, while conceptually [17] can function in heterogeneous environments, in practice it needs modification of the existing underlying routing protocols.

Concerning mesh network performance measurements, the majority of the solutions for wireless mesh networks propose solutions based on active

monitoring of network metrics, such as path delay in [17, 42], estimated link quality in [25, 34], link interference in [25, 34] and path packet loss rate [17]. All these approaches would entail a high monitoring overhead, except [17], where monitoring is done cooperatively to reduce the overhead.

As far as Internet gateway performance measurement is concerned all the above proposals use active measurements to evaluate its performance. More specifically [42] uses a congestion delay function, [32] monitors the unused Internet Connection (available capacity). [25, 34] force the gateways to participate in the monitoring process by measuring the queue length of their Internet interface, while [17] is performing active probes. Contrary to these approaches, our solution is totally passive, implying though less accuracy.

Finally, while we used the Vivaldi[45] system for estimating network performance there are various network coordinates approaches that allow nodes to estimate the latency between them while reducing the measurement overhead. From an abstract perspective, network coordinates are a virtual positioning system where nodes gather information about the network to position themselves and other nodes in a coordinate space and are used to estimate the inter-node latency Vivaldi[45] is a fully distributed network coordinates system that functions based on the idea of placing nodes in a two-dimensional euclidean space. The measured *ping* latency between the nodes is used to position them in the euclidean space. In addition to the probing, Vivaldi also uses spring-relaxation to *nudge* nodes in the Euclidean space to minimize prediction errors. While there have been proposals for updates of the Vivaldi algorithm the original algorithm is performing fine compared to the improvements[27]. Moreover, the state of the art of network coordinates includes more sophisticated and more accurate systems, which nevertheless are not fully distributed since they are based on the idea of the external landmarks, like Pharos[35]. As a result, we argue that the Vivaldi algorithm is a satisfactory option for our goal.

4.8 Conclusions

This chapter introduces reliable and inexpensive latency-based metrics capable of predicting and triangulating performance indicators, and a client-side proxy selection mechanism that combines these metrics to make good choices in terms of QoE or performance, taking into account the contribution of the local network, proxy gateways and their Internet connection. This mechanism avoids proxies with heavy load and slow internal network paths. The overhead is linear to the number of the clients and proxies.

5

Exploiting Traffic Patterns and Network Locality

5.1 Introduction

The users-proxies selection regulation mechanism described in 4 succeeds to dynamically assign users to proxies in a best effort manner, without though considering information that is related with the user traffic and the network infrastructure. In this chapter we present how this kind of information can be leveraged that could lead to a more informed proxy selection from the users, improving the final user experience as well as the overall service performance. This study considers several data inputs; e.g., the patterns of usage from service logs, the design choices and implications (considering client and proxy choices) according to patterns of usage, and the relative location of users and proxies in the network topology. The results show the key metrics, the design space for cooperative choices, the involved trade-offs, and the effects on the service cost and performance.

Section 5.2 looks at the behavior and clustering of users according to con-

tent and network locality, and it also analyzes the impact on the criteria for proxy selection. We present an analysis of the current scenario, limitations and potential for improvement from the perspective of the access network in Section 5.3, proxies in Section 5.4 and users in Section 5.5. Section 5.6 presents the conclusions and the future work.

5.2 Clustering of Users

We started the study exploring data concerning the service usage, in order to group users according to their behavior. Then, we identified the graph communities that exist in the network to analyze the factor of network locality. Thus, we tried to understand the trade-offs of grouping users according to similarities in their behavior and/or according to their location in the network.

5.2.1 Clustering according to usage

For the analysis of service usage according to patterns of data traffic, and based on [43], we considered four different types of clustering algorithms: K-means, suitable for generic applications, DBSCAN and Ward's hierarchical clustering (HC) that can trace complex patterns. The input used by the algorithms was the total data transferred per user in bytes, as well as the corresponding amount of traffic for contents that constitute a large amount of the total service traffic, like video (20%), image (6%) and HTML (2%). We experimented with various cluster sizes for K-means and Ward's HC, including well-known empirical estimation methods like the 'elbow method', as well as many parameters for DBSCAN. Table 5.1 presents the optimal results for each method in terms of cluster validation. For the validation we used the coefficient Shilouette score that has values in [-1,1]. As described in Table 5.1, for all the cases there is a big cluster of 450-480 users with a Shilouette score of 0.9, indicating a very strong cluster density. Nonetheless, the rest of the users belong to overlapping clusters, with scores close to 0. After manually reviewing other results of the algorithms for getting a better insight, since it

Table 5.1: Results from clustering algorithms on usage

Method	Clusters #	Clusters Size	Clusters Shilouettes
DBSCAN	2	7, 499	0.03, 0.90
Ward's	2	33, 473	-0.04, 0.89
	3	4, 29, 473	0.30, 0.01, 0.87
K-Means	2	21, 485	0.09, 0.89
	3	10, 44, 452	0.07, -0.04, 0.88

is the standard process in these cases, we chose Ward's HC method with 3 clusters, that partitions the users in one large consistent cluster and two small overlapping clusters, minimizing thus overlapping elements.

Table 5.2: Users Behavior Clusters Description (Ward's)

ID	Size	Shilouette	Characteristics	Alias
1	473	0.87	Low total traffic	Light
2	29	0.01	Medium total and video/images traffic	Medium
3	4	0.30	High total and video/images traffic	Heavy

Table 5.2 presents the characteristics of the clusters, as formed using Ward's HC for 3 clusters. We find two consistent clusters of users with distinct properties. The fig. 5.1 depicts the comparison of the clusters in terms of traffic and size (number of users). Cluster 1 of *light* users, includes the majority of users and their profile consists of generating very low traffic, as low as 1% of the maximum noticed per user value, mostly HTML browsing. Cluster 3, *heavy* users, consists of only 4 users and it is characterized by high total traffic, where most of it is spent on downloading video and images. Cluster 2, *medium* users, presents an intermediate behavior; nevertheless, following the patterns of the *heavy* users. *Medium* users create a significant portion of the total traffic, around 20% of the maximum value, which they consume mostly on videos and images. This cluster has low consistency, with users presenting

a behavior similar to cluster 3, but with traffic level close to cluster 1.

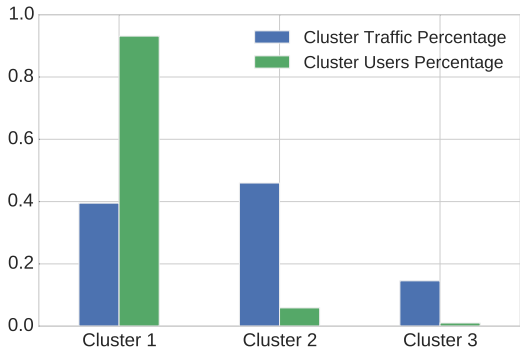


Figure 5.1: Traffic and Users Percentage per Cluster

5.2.2 Clustering according to Network Locality

For the analysis of user groups according to network locality, we use graph community detection techniques. Based on [36], we choose three of the most prominent detection algorithms: Spinglass, Multilevel and Infomap. The data input for the algorithms is the backbone graph, consisting of 48 nodes. Moreover, since the studied guifi.net zone has a small well-connected backbone, with many clients connected to the routers of the backbone, we used the number of clients using those routers to establish the graph weight for the InfoMap algorithm. The weight for each link is defined as the average time to transfer a single byte according to our topology dataset. The results of the different algorithms can be seen in Table 5.3. We compare the algorithms using the modularity score, which lies in the range [-1/2, 1), where the higher the value, the more consistent the community. Experimenting with the algorithms we noticed that the node size argument of the Infomap does not affect significantly the output, thus Infomap does not offer any additional information. Therefore, we choose the Multi-level Algorithm that has the highest modularity score and smaller number of clusters, considering the

small backbone.

Table 5.3: Comparison of Community Detection Algorithms

	Infomap	Multilevel	Spinglass
Modularity	0.699	0.712	0.702
Clusters	12	9	15

Figure 5.2 shows the resulting graph for the Multi-level algorithm. The squares represent the routers that operate also as proxies. As depicted, the proxies are not well positioned relatively to the network clusters, considering that most of the clusters have no proxies, while one of the clusters has two proxies. Additionally, we observe that there are clusters poorly connected to their neighbouring clusters, resulting in an infrastructure far from ideal. For the rest of this work we assume that all the clients of a router belong to the cluster of that router.

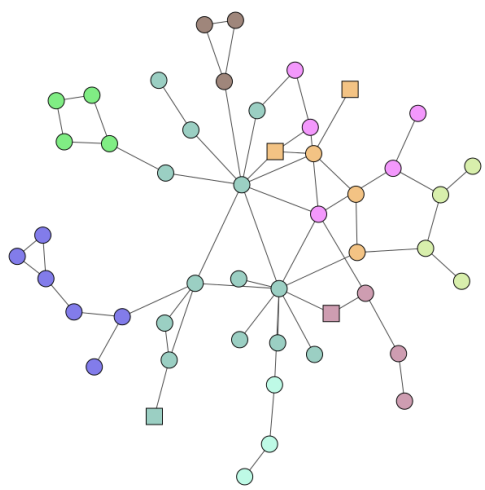


Figure 5.2: Multi-level Community Detection for the backbone network (colors)

5.2.3 Influence of the criteria for proxy selection

According to our clustering analysis, we present simulations that exploit the two clustering techniques in algorithms for proxy selection, in order to provide alternatives to the current manual proxy selection. The objective is to demonstrate the impact of network locality and user traffic behavior on the performance of the proxy service and user experience. Thus, we show how they can be used to inform the design of an improved service.

Next, we present an initial evaluation of the mentioned techniques under the perspectives of the network, the proxies and the users. It is important to clarify that our algorithms implement one of several ways to use the information from user behavior clustering and community detection. The first algorithm we implemented, referred as `data_cluster`, uses the clustering of user behavior to assign equivalent user load to each proxy by equally distributing the users of each cluster. In the cases where a new user has to be assigned to a proxy and all existing assignments from the clusters are equally balanced, the algorithm selects a proxy randomly. The second algorithm, referred as `network_cluster`, uses graph community detection to assign users to proxies according to the proximity of their community. For instance, a user with an available proxy in his community will be assigned to this proxy, while in the opposite case, it will be assigned to the proxy that is located in the closest community. In case of equal proximity, the proxy selection is random. Finally, we implemented an algorithm that combines both solutions in one of the possible ways. The algorithm `data+network` is mainly based on the `data_cluster` algorithm, but in case it encounters equal assignments, it uses the `network_cluster` algorithm to decide. All these algorithms are compared with the `manual` manual service selection.

5.3 Network perspective

The impact on the network is studied according to the total bytes transferred through each link during the simulation. We do not take into account possible

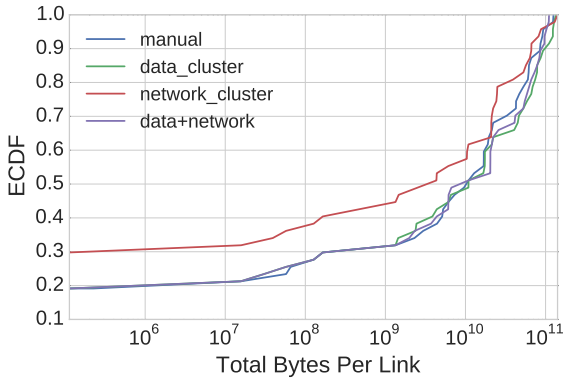


Figure 5.3: Comparison of Total Links Bytes Per Strategy ECDF

retransmissions, and we assume that the links cannot be saturated and have always the same performance, even across different links.

As shown in Figure 5.3, the `network_cluster` algorithm outperforms significantly the other algorithms in distributing the load in the links. It maintains the total traffic of 50% of the links, one order of magnitude lower than the other algorithms without compensating that by overloading a few links, as we would expect for the links that connect the clusters. The other algorithms present a similar, but shifted, distribution. Moreover, considering that each algorithm is using different number of links to send the traffic, it is worth mentioning that `network_cluster` transfers the lowest total amount of bytes, 1 Terabyte, while `data_cluster` is the most expensive transferring 1.7 Terabytes. We also find that `data+network` lies between `network_cluster` and `data_cluster`, with 1.4 Terabytes, while `manual` transfers 1.3 Terabytes.

Overall, we observe that network locality plays a significant role in distributing the load on the network. Even in the case of existing communities without proxies, like the studied case, a locality-aware service can reduce its network impact.

5.4 Proxy perspective

From the perspective of the proxies, it is important for both the service performance and the user's experience to distribute the load according to the capacity and performance of each proxy.

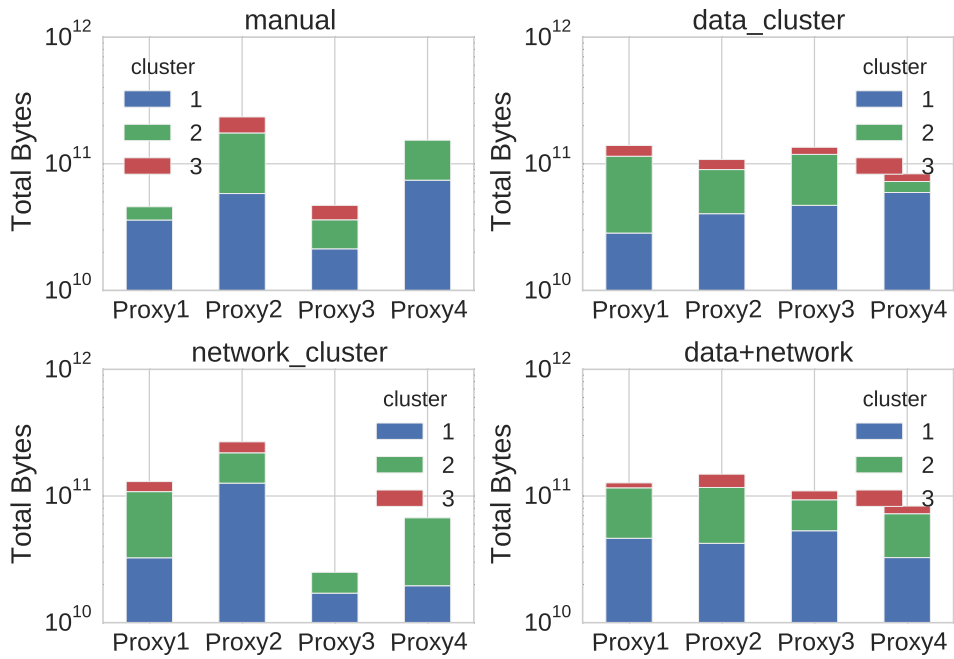


Figure 5.4: Comparison of Strategies: Traffic per Proxy and Clustering

In our simulations we start by assuming that all proxies have infinite capacity and the same processing performance (i.e., unlimited throughput). We evaluate the different algorithms by the total amount of bytes sent to each proxy per strategy, with information of the corresponding clusters, as seen in fig. 5.4. We initially observed that the *heavy* users occupy an important percentage of the traffic, even though they are nearly the 1% of the total users. Nonetheless, *light* users generate the majority of traffic despite the fact that each of them use the service comparatively much less. Therefore,

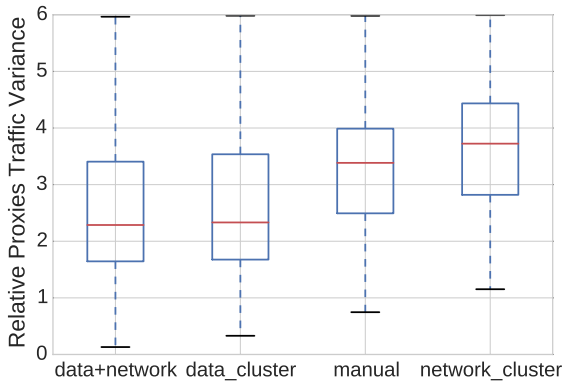


Figure 5.5: Proxies per second Relative Traffic Variance

as a result of manual selection the proxy load is very unbalanced, but the `data_cluster` and `data+network` algorithms succeed in balancing the traffic. The `network_cluster` approach can result to an imbalance in the load among proxies, due to sub-networks with an uneven number of clients and the proxies inconveniently placed with respect to the clients. It is worth noting that, from the proxy perspective, the `data+network` algorithm achieves its goal very successfully since it is mainly based on the `data_cluster` algorithm; however, it also achieves a better performance than `data_cluster` from the network perspective. Moreover, as shown in fig. 5.5, the sum of distances of traffic values for each proxy to the mean at each instant is clearly smaller for the `data_cluster` or `data+network`. This small variability implies that these algorithms work well over both short and long term periods. We can therefore deduce that an algorithm that combines both, user clustering and network graph community detection, can be used for tuning the trade-off of impact of uneven proxy load and excessive network impact, due to long network paths. This lesson is applicable to server selection in a decentralized service.

If we take into consideration the limited capacity and throughput in proxies, then balancing the traffic across them according to the capacity of each proxy becomes a key issue. For example, in the case of a large number of users

the clustering information could be used to perform admission control and therefore congestion control in the proxy.

In the current scenario proxies have a rough admission control based exhaustion of limits, and they do not on congestion control according to load or performance. Proxies take new requests based on a maximum number of concurrent clients, even when the proxy service is already under-performing for ongoing responses. This results in poor performance during peaks of large requests that cause congestion or a service timeout. In our decentralized scheme, clients have a list of several proxy choices. Clients make an initial choice, proxies can reject connections, and clients can just make a new local choice, transparently retry and continue from there, with no major visible effect to the user. The combination of clients using a list of proxy choices, proxy admission control, and network routing choices results in a simple, decentralized and cooperative regulation scheme that requires little coordination.

Admission control is important in large user populations, e.g., wide-area networks with many proxies, since proxies have a limited Internet access capacity. Any hotspot or imbalance in a massive system can easily lead to congestion, either in the access network, any proxy or the Internet access, resulting in a dramatic reduction of service throughput for many users of that proxy.

In addition to the local choices at each client and proxy, there is potential for a global optimization in balancing global choices, across all proxies, by combining the user traffic behaviour, user proxy choices, and proxy capacities. Thus, we can help avoid globally imbalanced scenarios, where a proxy is saturated or providing low throughput, while at the same time another proxy is underutilized.

5.5 Users perspective

The evaluation of impact on service performance from the user perspective is the most complex, as users have different metrics to assess their service

according to their diverse usage habits. While exploring these metrics is future work, here we present a first simple cost model to estimate how users perceive the impact of the presented algorithms. We assume that users try to minimize the transfer time in the local network, combined with the processing time in the proxy server.

As far as the network is concerned, we define as c_{l} the cost of the link l, in terms of time, to transfer one byte, assuming that the links have infinite capacity, although we plan to study more sophisticated models in the future.

For each user we calculate the total cost of the network transfer as $\sum_{l=0}^{\mathrm{n}} c_{\mathrm{l}} * b_{\mathrm{u}}, l \epsilon L_{\mathrm{u}}$, where L_{u} is the set of links and b_{u} the total number of bytes attributed to user u.

The users' perception of the proxy performance is modeled similarly to the network performance. We define c_{p} as the cost of proxy p to process one byte, from the time it receives the request from the user, until it sends the last byte. We calculate the cost c_p of each proxy p separately for every strategy as $t / \sum b_{\mathrm{u}}, u \epsilon U_{\mathrm{p}}$, where t is the total measurement time, b_{u} the total number of bytes sent by user u and U_{p} the set of users of proxy p. Based on that, the proxy perceived cost for each user is: $c_{\mathrm{p}} * b_{\mathrm{u}}$.

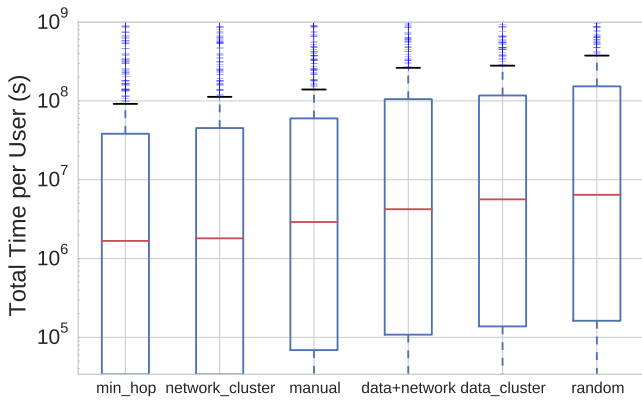


Figure 5.6: Cost per user ECDF

Considering that the costs are linear and independent, we can assume that the overall cost perceived by a user u is: $C_{\mathrm{u}} = \sum_{l=0}^{\mathrm{n}} c_{\mathrm{l}} * b_{\mathrm{u}} + c_{\mathrm{p}} * t_{\mathrm{u}}, l \epsilon L_{\mathrm{u}}$. Hence, the objective of user u would be to minimize C_{u}. Figure 5.6 presents the distribution of the users' costs for each of the presented strategies.

While the distributions have very similar behavior, we can observe that for 80% of the users, the network community detection strategy performs slightly better than the current situation, and the rest of the strategies follow. The community strategy achieves equivalent results to a $min - hop$ strategy, only differing when proxies are not in the center of its zone. The *random* strategy achieves equivalent results to cluster, as the latter only cares about contents and none about infrastructural aspects.

The network efficiency of community-based proxy selection, and therefore the impact of network locality, appears as an important factor. Studying the individual costs we observe that the network transfer time cost is in average significantly higher than the proxy processing cost, a fact that explains why the community solution performs better overall, even though it is an inefficient option for load distribution in the proxies. The (clustering according to) user behavior appears to have an influence on the user perceived performance (cost), since it presents a differentiated behavior from the current situation (manual proxy selection). However, the simplicity of the model does not allow us to draw more conclusions.

In contrast, the current situation is that clients (Web browsers) have a list of proxy servers manually defined or adjusted. The initial configuration is based on hints from other nearby users, or by downloading the list from a local guifi.net forum. The adjustments come from similar sources, personal usage experience, hints from other users or news about new proxies being offered. Web browsers switch to another proxy server just when a proxy fails to respond and do not provide load balancing, or more effective choices considering to degradation, congestion signals or relative performance. These models enables us to design a service selection algorithm that takes into account the characteristics of the users and the local network, confronting

thus the inefficiencies caused in the service and the user experience by the manual static proxy selection.

5.6 Conclusions

The analysis of service logs shows patterns of usage and network topology grouping users and proxies, that influence of the criteria for proxy selection. The currently manual and not well-informed choice of proxies by clients work rather well for its users, but it result in inefficiencies that affect the service cost and shows episodes of degraded performance. Considering that situation, this chapter explores alternatives for cost reduction and service improvement when going from a simple but rigid mapping between users and proxies, towards coordinated informed choices based on several metrics. Design trade-offs lie in considering infrastructural aspects (e.g., reduce network cost, avoid network and proxy congestion) and service aspects (e.g., good response time or QoE).

The combination of server alternatives in clients, finer grain proxy admission control, and the underlying network routing decisions result in a decentralized cooperative regulation scheme that can provide a crowdsourced proxy service, with good performance and requiring little coordination. Moreover, that scheme allows the network scaling up to larger sizes.

6

Sharing Only The Exceeding Bandwidth

6.1 Introduction

Chapters 4 and 5 describe our approach on how the users-proxies selection regulation mechanism can be improved, without introducing significant overhead. In this chapter we focus on the second component of the Internet sharing problem, as described in § 1.4. More specifically we look at the cost reductions resulting from N citizens or organizations sharing their unused Internet access capacity benevolently with M neighbors and members of their community, through a local or regional community network. In order to provide such a service without negatively affecting the quality of access of the primary users, we propose utilizing gateways to separate the primary traffic (i.e., that of the Internet access donors) from the one of the beneficiaries (i.e., the secondary traffic).

Each of the M beneficiary nodes selects one or a few of the N Internet gateways, where they send their traffic. The gateways receive the IP traffic or HTTP requests from these secondary nodes and try to provide them with a solution. Although this traffic uses the spare capacity of the Internet access,

it may compete with the primary source traffic, hinder its performance, and also increase its cost. Therefore, only making this sharing process innocuous for the donors, will allow this mechanism be sustainable over time. However, keeping under control this aspect of the traffic represents a major challenge for the managers of community networks.

In order to help address this challenge, we analyze several of the mechanisms for sharing the spare Internet capacity among third parties in guifi.net, the ways to provide it, and the performance implications of connectivity sharing at no additional economic cost. Based on the obtained results, we present a set of lessons learned that can help make suitable and sustainable this sharing process.

The already introduced premise of no additional cost implies that in 95-percentile pricing schemes [46], secondary users cannot top-up the primary traffic or it will incur in an additional cost. Of course, other cost models come with different limits, or none at all for most fixed domestic broadband Internet with unlimited traffic and flat cost. Some previous works [49] [37] have shown that water-filling (taking advantage of already-paid-for off-peak bandwidth resulting from diurnal traffic patterns and percentile pricing), allows delay tolerant asynchronous bulk data to be transferred effectively at no transmission cost to the ISP. In a scenario with multiple Internet gateways available to users, while one could stop serving secondaries to avoid extra traffic charges, clients could switch to another available proxy, as seen in Chapter 4.

Section 6.2 describes the experimental framework, and it shows the evaluation results in Section 6.3. Section 6.4 presents the lessons learned and conclusions.

6.2 Experimental framework

The system model and the scenario for experimental evaluation represent a gateway middlebox that separates the primary traffic from the secondary

coming from a number of nodes in the local access network, the wired or wireless community network. All primary and secondary traffic go to the Internet. In our experiments we assume the traffic is Web-like, where primary and secondary traffic comes from clients that make Web requests that result in downloading Web objects. We also assume that all clients interact with a single server that provides content to both primary and secondary clients. Figure 6.1 shows the nodes participating in the testbed: (1) primary and secondary clients, (2) the gateway that routes traffic from both types of clients, and that interact with a server on the Internet (3). The gateway node manages both primary and secondary traffic, and it applies different techniques for each class, considering the limited capacity of the available Internet access, while trying to assess and minimize the impact of secondary traffic on the primary one.

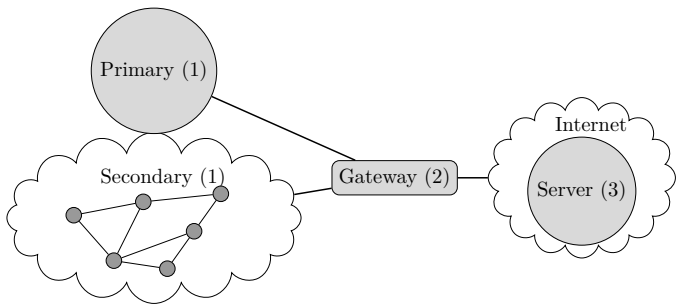


Figure 6.1: Physical architecture of the testbed

Model for Internet access with primary and secondary users Internet access is modelled at the gateway using the traffic control and queueing disciplines tools available in Linux. The Client-Gateway connection is 100 Mbps. The Gateway-Server link uses values obtained from the Measurement-Lab testbed of Telefonica [33], the largest ISP in Spain: 1.72 download throughput (Mbps), analysed in [9]. The bottleneck is at the gateway. In order to validate our experimental setup we used the Network Diagnostic Test, which is the same tool used to characterise real ISPs. In addition to the modeled

values, we have validated that the modeled access behaves as we expect.

Traffic modeling In the scenario of community networks, and specifically in the guifi.net [14], gateways act as Web proxies and therefore the traffic will be HTTP. We have used the wrk2 tool to generate customer traffic. This allows performing realistic HTTP benchmarking removing the effects of "co-ordinated omission" [21] from the measurements.

Metrics The goal of this study is to compare the different mechanisms to share excess bandwidth: i) with the primary only (Prim_only), and ii) with the primary and secondary traffic without any specific mechanism (Best_effort). These two experiments are the best case for Prim_only, and the case that we want to improve in the Best_effort. To evaluate the behaviour of the tests mechanisms we utilize two metrics: 1. co-inflicted delay on the service time of HTTP requests for each mechanism (it is normalized to the mean delay throughout the best-effort primary and secondary traffic), and 2. the network throughput.

6.2.1 Traffic sharing between primary and secondary

The mechanisms of "traffic engineering" have to: a) act only at the gateway not requiring end-to-end changes, b) be transparent to clients and servers, and c) be innocuous (i.e., to have no impact or cost) when the gateway is not congested.

We have experimented with three types of mechanisms based on: i) traffic shaping, ii) Active Queue Management (AQM) and iii) tunnelling. In the first one the gateway monitors traffic and discards non-compliant packets. In AQM the gateway does not use a FIFO strategy for packets, and it tries to prioritize packages by type or flow. In the last case we replace the congestion control of the end-to-end transport protocol to that of the tunnel.

Figure 6.2 shows the location of these three types of mechanisms in the network stack, indicating the types of test applied to the primary traffic (column a) and the secondary one (column b). In dark it marks the layer in which the

mechanism acts. Next, we explain the mechanisms considered in this study.

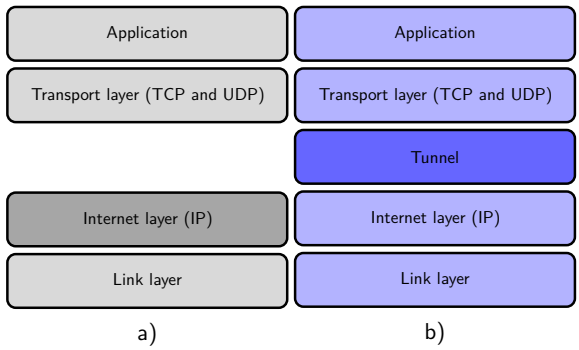


Figure 6.2: Types of tests applied to primary (left column) and secondary (right) traffic

Based on traffic shaping In this case we used a Borrowing strategy, in which the primary and the secondary traffic have a guaranteed minimum throughput. Moreover, the borrowed unused throughput can be utilized with priority to the primary traffic.

Based on active queue management Here we used Stochastic Fairness Queueing (SFQ) and CODEL mechanisms. The first one is used by several ISP [15], since it tries to order the packets more fairly (a packet from each flow). The second mechanism has been successfully used to significantly mitigate the bufferbloat phenomenon [23].

Based on tunneling In this case we used three strategies: TCP_Cubic, TCP_Vegas and TCP_LP. In the first case we use the tunnel TCP congestion control algorithm to manage the secondary traffic. In the second case, the secondary traffic was managed through a TCP Vegas tunnel, and the congestion avoidance algorithm emphasized packet delay (Round-trip time) rather than packet loss. In the last case (TCP_LP), the secondary traffic was managed through a TCP type low-priority tunnel, with the idea of controlling congestion [30]. It would give less priority to secondary traffic than the best effort, and its main goal is to utilize only the excess network bandwidth.

6.3 Results

The experiments considered in this study are intended to evaluate the impact of secondary traffic in the primary traffic, considering the Prim_only and Best_effort cases as reference. Moreover, the experimentation intended to determine the impact of the different techniques when the gateway is not overloaded, the impact of the different techniques under overload, the sensitivity to the characteristics of the traffic, the overhead cost of tunneling, and the overhead of using WiFi links, typical of access networks such as community networks. In order to make service time results comparable across different experiments, where possible, the results were normalized to the overall best_effort service time mean of each experiment.

6.3.1 Gateway not overloaded

The client traffic model to represent the case of a not overloaded gateway consists on a single primary user with two concurrent connections each, and four to five secondary users with ten concurrent connections each. All HTTP requests involve to objects weighting 0.1MB. There is a random time between HTTP requests that ranges from 10 to 50 ms. The Internet connection model corresponds to 1.72 Mbps of download throughput. The resulting total traffic (primary + secondary) does not exceed on average the maximum throughput of the connection. Although both traffic compete, there is sufficient throughput for both of them.

From the results shown in Figures 6.3 and 6.4 we can conclude that the difference of service time for the primary traffic (between Prim_only and Best_effort) is relevant. Moreover, the service time achieved by the secondary traffic has a large impact on the service time of the primary one. On the other hand, the Best_effort strategy is already usable, it will not require major improvements. Looking at the service time of the primary traffic, TCP_LP offers values very close to Prim_only. However, if we consider the service time of the secondary traffic, the TCP_LP mechanism offers comparable values to

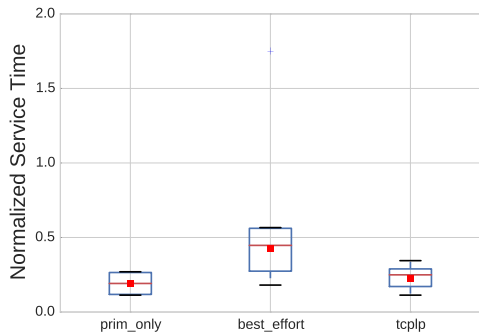


Figure 6.3: Normalized service time of primary traffic with underutilized Internet Connection

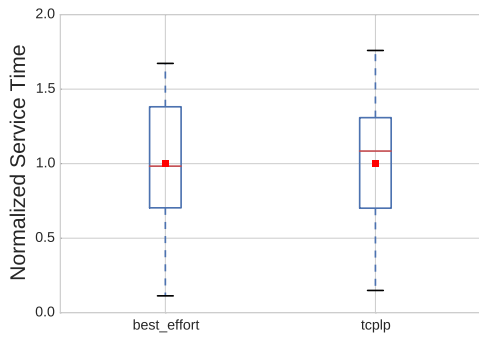


Figure 6.4: Normalized service time of secondary traffic with underutilized Internet Connection

Best_effort. Therefore, it significantly improves the primary traffic without penalizing the secondary one.

Applying the TCP_LP mechanism has no significant effect on the primary traffic when the gateway is not overloaded. Therefore, from now on we will focus only on cases in which the gateway is nearly or fully overloaded.

6.3.2 Gateway is overloaded

In order to simulate and reproduce an overloaded Internet connection, we use the following HTTP traffic generation model: the primary traffic (represents

one user) is generated with a rate of 5 requests per second, while the secondary one (represents 5 users) is generated with a rate of 25 requests per second. All the HTTP requested objects have a fixed size of 12.5 KB, except if explicitly stated otherwise. Moreover, the primary traffic is generated with a random *user think time* in the range of 10-50ms between every request. Additionally, we limit the throughput of the Internet connection to 1.72 Mbps for downloading and 0.54 Mbps for uploading to provide a more realistic experimental environment. Throughout all the experiments the total traffic is generated with a rate greater than 1.72 Mbps to achieve the saturation of the Internet connection. As a result, the primary and the secondary traffic have to compete to access the Internet connection. The results are shown in Figures 6.5 to 6.8.

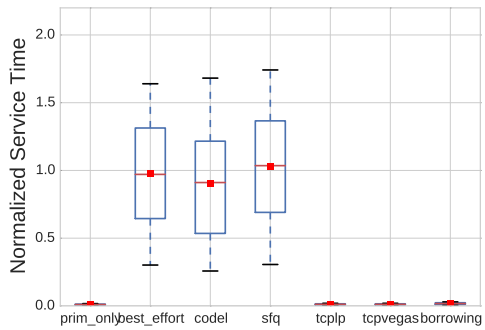


Figure 6.5: Comparison of the effect of strategies on Service Time of the primary traffic under saturated Internet Connection

CODEL and SFQ are applied to all traffic without differentiating primary and secondary. From the results shown in Figures 6.5 and 6.6 we can conclude that the difference of service time between Prim_only and Best_effort (for the primary traffic) is very important, which means that the secondary traffic has a very large impact on the first one, even with no comparatively relevant improvement on the secondary's traffic, meaning poorer utilization overall. Moreover, the service time of the primary traffic (TCP_LP, TCP_Vegas and Borrowing) offers good values, very close to Prim_only. In case of the secondary traffic these values are somewhat higher than Best_effort. They penalize

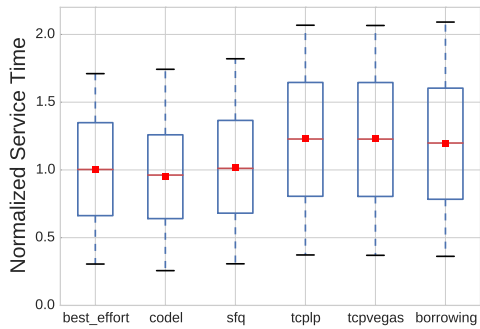


Figure 6.6: Comparison of the effect of the strategies on Service Time of the secondary traffic under saturated Internet Connection

the secondary traffic by a small factor around 20%. In case of CODEL, it even brings a slight improvement to the service time compared to Best_effort for both primary and secondary traffic. SFQ does not seem to bring any significant improvement to any of them.

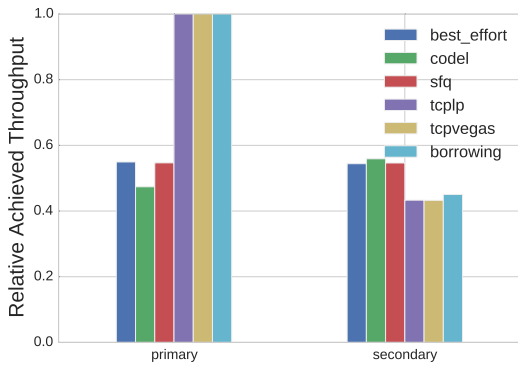


Figure 6.7: Throughput comparison under saturated Internet Connection

The effect on TCP is illustrated by Figure 6.8. Based on these results we can conclude that the number of retransmissions is very low compared to the number of HTTP requests performed in each experiment. Here CODEL makes a difference over the rest. The greater number of retransmissions for

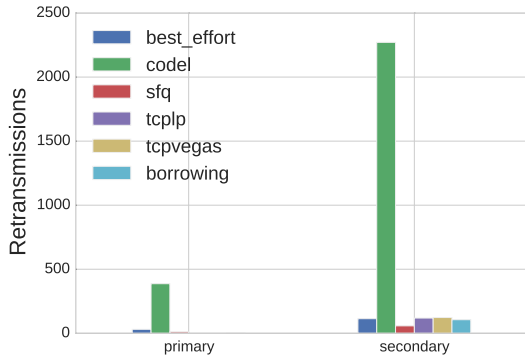


Figure 6.8: Retransmission comparison under saturated Internet Connection

the secondary traffic reduces the throughput, compared to the Best_effort.

The secondary traffic has a very large impact on the service time and throughput of the primary traffic, especially when competing by the access in an overloaded gateway. TCP_LP, TCP_Vegas and Borrowing are good candidates, since their values for the primary traffic are very close to Prim_only. They penalize the secondary traffic, with latency and throughput values slightly worse than Best_effort. CODEL and SFQ do not show any significant difference compared to the Best_effort approach; however, they could be applied at the same time with the previous ones; e.g., TCP_LP + CODEL.

6.3.3 Sensitivity analysis

Here we analyze, both the sensitivity to object size, and to the distribution of concurrent requests of the different techniques. Additionally, we show the behavior of the techniques in experiments on a setup with extreme values for several parameters.

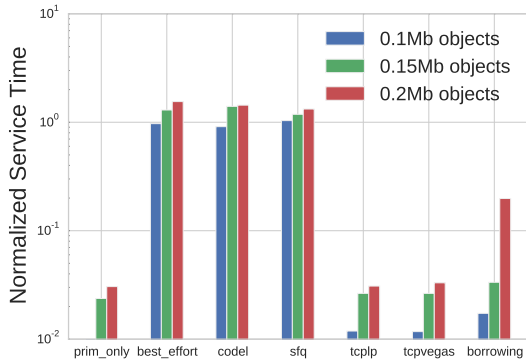


Figure 6.9: Primary traffic Service time sensitivity on Object Size of the strategies

Object Size

In this case the results are normalized based on the mean throughout the primary and secondary service time results for 0.1 Mb. Considering the results shown in Figure 6.9 we can conclude that there is a clear relationship between service time and object size. Increasing the object size results into an increasing service time, which is expected since we increase as well the data request rate. Moreover, as far as the primary traffic is concerned, CODEL and SFQ present a behaviour close to the Best_Effort, but slightly varying based on the object size; TCP_LP and TCP_Vegas are the solutions with a behaviour closest to Prim_only.

On the other hand, while the Borrowing strategy seems to have a performance similar to the Prim_only, it appears to be more sensitive to the the size of requested objects. When the object size is increased (by increasing the stress on the server), the service time for the primary deviates significantly from Prim_only. There is a very similar pattern of increasing service time while increasing object size for all secondaries.

Proportion of requests primary-secondary

We vary the proportion of requests between primary and secondary traffic, while keeping the total throughput and the total number of requests. Results are normalized based on the mean throughout the primary and secondary service time results for the pair of 5/25 reqs/s. As expected, there is no visible difference between CODEL and SFQ when they are applied without differentiating primary and secondary traffic, and the aggregation of primary and secondary connections is kept at 30 concurrent connections. Therefore, Figure 6.10 only presents strategies affected by the changes, omitting CODEL and SFQ.

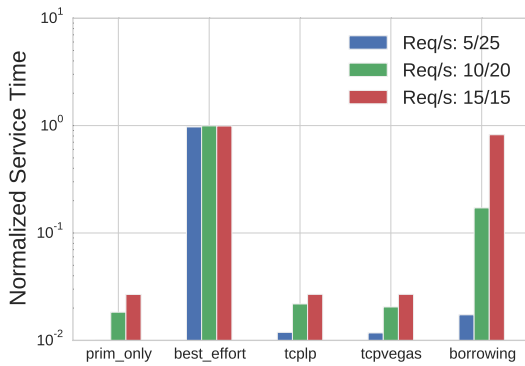


Figure 6.10: Primary traffic service time sensitivity on requests rate

Considering the service time of the primary traffic, TCP_LP and TCP_Vegas still behave very similar to Prim_only. However, Borrowing presents again a differentiated behaviour. It is very sensitive to the proportion of connections between the primary and secondary nodes. In any case where the primary or secondary traffic exceeds its configured upper bound data rate, the service time will increase accordingly. Regarding the secondary traffic, all service times are more or less close to the Best_effort service time.

"Edge Cases" for object size

The objective of this experiment is to show extreme cases while keeping the overall throughput with a very small object size, which fits in a single TCP segment, compared to a large object size. In this case, the results are normalized based on the mean throughout the primary and secondary service time for the pair of 0.01Mb objects - 150 reqs/s. Analyzing the results in Figures 6.11 and 6.12 we can conclude that they seem to make sense with previous ones. SFQ and CODEL work well, but they only provide small improvements mostly when there are many requests. In the primary, TCP_LP and TCP_Vegas behave almost like the Prim_only, with an advantage for TCP_LP with small objects. Moreover, the Borrowing strategy seems to make the situation worse when there are a lot of requests, while it improves (for both to primary and secondary traffic) with larger objects.

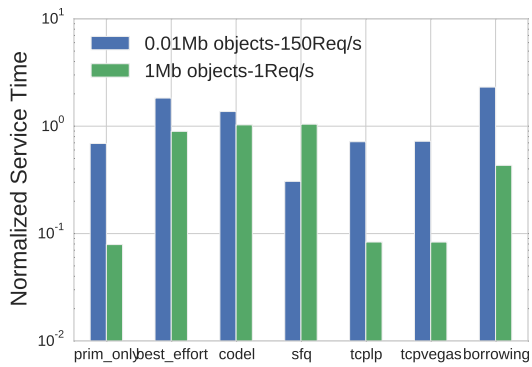


Figure 6.11: Primary traffic Service time comparison on edge scenarios

Our main lessons here are as follows. TCP_Vegas and Borrowing are very sensitive to Object size. In a real scenario we do not know the size of objects. They are not the best candidates for selection. Borrowing is very sensitive to the distribution of the number of connections. In a real scenario we do not know the number of concurrent connections of the primary or the secondary. It is not the best candidate for selection. Even in extreme cases,

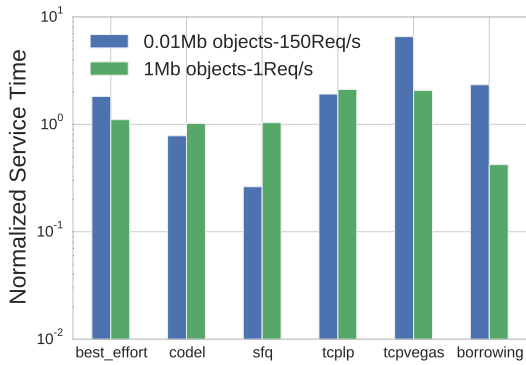


Figure 6.12: Secondary traffic Service time comparison on edge scenarios

in the primary traffic, TCP_LP behaves very close to Single. So far it ranks as the best candidate.

Other factors

We have compared the differences in behaviour among different types of TCP tunnels. Comparing the experiment results we observed that the IP in IP tunnel has the same behavior as Best_effort in the primary traffic, with or without delay, with relative differences less than 0.04%. Therefore, we can conclude that tunnel based techniques do not add any penalty. Additionally, we compared the behaviour of TCP_LP and TCP_Vegas used in the secondary tunnels against TCP_Cubic that was used for the the primary traffic and as transport under the tunnels. We observed that the aggressiveness of TCP_Cubic is the reason that TCP_Vegas behaves similarly to TCP_LP in our experiments. Substituting TCP_Cubic in the primary tunnels for other TCP algorthim we expect to obtain similar results for TCP_LP tunnels, but TCP_Vegas tunnels would deteriorate the primary service time while improving the secondary service time.

We have evaluated the overhead of wireless (WiFi-based) links for secondary users, as this is quite common in access networks, such as community

networks. We evaluated an ad-hoc (IBSS) network in channel 4 of the 2.4 GHz band. The results show equivalent results to a wired Ethernet connection, just with a slight improvement for CODEL and SFQ both, for the the primary and secondary traffic.

Finally, we want to mention that there is other important traffic in the network that may be affected by the secondary traffic and by the overload; for instance, some important IP packets such as DNS, ICMP or SYN. The results in Figure 6.13 show that even in the case of an overloaded gateway, CODEL and SFQ contribute to improve the behavior of TCP_LP. Flows with very few packets, ICMP ping in the figure, no longer suffer from "starvation" by virtue of not being trapped in a FIFO queue. The same effect is achieved for the secondary traffic, and also also in the primary and secondary traffic for a non-overloaded gateway.

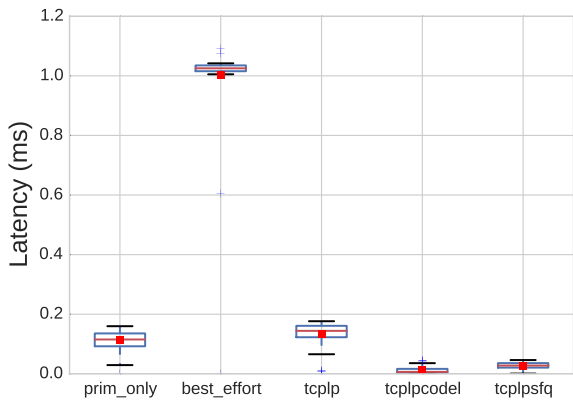


Figure 6.13: Primary client ping latency combining AQM and TCP_LP

6.4 Conclusions

In this chapter, we look at cost reduction of the Internet access that benevolently citizens share with others through a local community network. The analysis of several mechanisms for sharing spare Internet capacity shows the

performance and drawbacks for primary and secondary users. In summary, TCP_LP appears as the most promising option, regardless of whether the gateway is overloaded or not. The primary traffic is apparently not affected by the secondary one (it behaves like Prim_only). The secondary traffic achieves to get the spare capacity (it behaves like the non-differentiated case, best effort, with a limited penalty around 20%). Combined with complementary queueing techniques (e.g., TCP_LP + CODEL or TCP_LP + SFQ) instead of just a FIFO queue, it allows to "treat well" other small, but very important packets for the user experience, such as DNS or ICMP. We believe that combining multiple shared Internet connections at no additional penalty in performance and cost over a local or regional community network is a valuable method to accelerate the expansion to Internet for everybody.

7 Conclusion

Communities of citizens develop network infrastructures cooperatively based on heterogeneous Wireless Mesh Networks. They can achieve global Internet or Web access using a pool of web proxy gateways shared across many participants in the local community network. This affordable Internet access requires a simple but effective mechanism to arbitrate the client-proxy choice to ensure a good quality of experience and avoid degraded service.

Analyzing part of a community network we observed that the system is simple and resilient since each proxy is independent and clients just switch to their next choice in case of failure of their proxy. While the system shows a satisfactory performance in a small scale, the manual proxy selection of the users and the traffic patterns would not allow the Web access service to scale properly. In our effort to address this problem we decompose it in two major components: users-proxies relation and primary-secondary users relation.

Concerning the relation between the users and the proxies we presented two reliable and inexpensive latency-based metrics capable of predicting and triangulating performance indicators, and a client-side proxy selection mech-

anism that combines these metrics to make good choices in terms of QoE or performance, taking into account the contribution of the local network, proxy gateways and their Internet connection. This mechanism avoids proxies with heavy load and slow internal network paths. The overhead is linear to the number of the clients and proxies. Additionally, we analyzed how the currently manual and not well-informed choice of proxies by clients work rather well for its users, but it results in inefficiencies that affect the service cost and shows episodes of degraded performance. Considering that situation, this paper explores alternatives for cost reduction and service improvement when going from a simple but rigid mapping between users and proxies, towards coordinated informed choices based on several metrics.

Concerning the relation between primary (who share their Internet connection) and secondary (the beneficiaries) users we studied several mechanisms for sharing spare Internet capacity shows the performance and drawbacks for primary and secondary users. We show that with a middlebox between the gateway and users the Internet access experience of the primary users can be retained, undermine the experience of the secondaries. Studying various strategies, we proposed a combination of congestion-level tunneling and queuing techniques that can be used for this middlebox that guarantee the user experience of the primary when the shared Internet connection is saturated, without introducing overhead the rest of the time.

Overall, we believe that we have carved the path for a way to use already existing Community Network infrastructures in order to provide basic Internet access.

Bibliography

1. GAIA WG. *Global Access to the Internet for All Research Group* [Online; accessed 14-September-2016]. 2016 (cit. on p. 1).
2. Navarro, L., Freitag, F., Baig, R. & Roca, R. in (ed on Community Connectivity (DC3), D. C.) 25–71 (Internet Governance Forum, 2016) (cit. on p. 3).
3. netCommons. *Deliverable D1.2, Report on the Existing CNs and their Organization (v2)* Tech. Rep. D1.2. Sept. 2016 (cit. on p. 3).
4. Sen, R. *et al. On the Free Bridge Across the Digital Divide: Assessing the Quality of Facebook's Free Basics Service* in *Proceedings of the 2016 ACM on Internet Measurement Conference* (2016), 127–133 (cit. on p. 26).
5. Abujoda, A., Dietrich, D., Papadimitriou, P. & Sathiaseelan, A. Software-defined wireless mesh networks for internet access sharing. *Computer Networks* **93, Part 2,** 359–372 (2015) (cit. on p. 5).
6. Baig, R., Roca, R., Freitag, F. & Navarro, L. guifi. net, a crowdsourced network infrastructure held in common. *Computer Networks* **90,** 150–165 (2015) (cit. on pp. 2, 3, 43).
7. Baldesi, L., Maccari, L. & Lo Cigno, R. Improving P2P streaming in Wireless Community Networks. *Computer Networks* **93, Part 2,** 389–403 (2015) (cit. on p. 3).
8. Biswas, S. *et al.* Large-scale measurements of wireless network behavior. *ACM SIGCOMM Computer Communication Review* **45,** 153–165 (2015) (cit. on p. 26).
9. Braem, B., Bergs, J., Blondia, C., Navarro, L. & Wittevrongel, S. *Analysis of End-User QoE in Community Networks* in *Computing for Development (ACM-DEV)* (London, UK, 2015), 159–166 (cit. on p. 67).
10. Chen, F., Sitaraman, R. K. & Torres, M. End-user mapping: Next generation request routing for content delivery. *ACM SIGCOMM Computer Communication Review* **45,** 167–181 (2015) (cit. on pp. 31, 37).

11. Internet Society. *Global Internet Report 2015* Oct. 2015 (cit. on p. 1).

12. Maccari, L., Baldesi, L., Lo Cigno, R., Forconi, J. & Caiazza, A. *Live Video Streaming for Community Networks, Experimenting with PeerStreamer on the Ninux Community* in *Proc. Workshop on Do-it-yourself Networking: An Interdisciplinary Approach (DIYNetworking)* (Florence, Italy, 2015), 1–6 (cit. on p. 3).

13. Maccari, L. & Lo Cigno, R. A week in the life of three large Wireless Community Networks. *Ad Hoc Networks* **24, Part B,** 175–190 (2015) (cit. on p. 3).

14. Vega, D., Baig, R., Cerdà-Alabern, L., Medina, E., Meseguer, R. & Navarro, L. A technological overview of the guifi.net community network. *Computer Networks* **93,** 260–278 (2015) (cit. on pp. 1, 3, 22, 23, 68).

15. Gong, Y., Rossi, D., Testa, C., Valenti, S. & Täht, M. D. Fighting the bufferbloat: on the coexistence of AQM and low priority congestion control. *Computer Networks* **65,** 255–267 (2014) (cit. on p. 69).

16. Lo Cigno, R. & Maccari, L. *Urban Wireless Community Networks: Challenges and Solutions for Smart City Communications* in *ACM Int. Workshop Wireless and Mobile Technologies for Smart Cities (WiMobCity), part of MobiHoc* (Philadelphia, PA, US, 2014), 49–54 (cit. on p. 3).

17. Ko, B. J., Liu, S., Zafer, M., Wong, H. Y. S. & Lee, K. W. *Gateway selection in hybrid wireless networks through cooperative probing* in *International Symposium on Integrated Network Management (IM)* (IEEE, 2013), 352–360 (cit. on pp. 48, 49).

18. Rey-Moreno, C., Roro, Z., Tucker, W. D., Siya, M. J., Bidwell, N. J. & Simo-Reigadas, J. *Experiences, challenges and lessons from rolling out a rural WiFi mesh network* in *ACM Computing for Development (ACM-DEV)* (2013), 11 (cit. on p. 2).

19. Sathiaseelan, A. & Crowcroft, J. LCD-Net: lowest cost denominator networking. *ACM SIGCOMM Computer Communication Review* **43,** 52–57 (2013) (cit. on p. 5).

20. Sundaresan, S., Feamster, N., Teixeira, R. & Magharei, N. *Community Contribution Award – Measuring and Mitigating Web Performance Bottlenecks in Broadband Access Networks* in *Internet Measurement Conference (IMC)* (ACM, 2013), 213–226 (cit. on pp. 31, 37).

21. Tene, G. *How not to measure latency* in *Low Latency Summit* (2013) (cit. on p. 68).

22. Cerda-Alabern, L. *On the Topology Characterization of Guifi.Net* in *Proc. Int. Conf. Wireless and Mobile Computing, Networking and Communications (WiMob)* (Barcelona, Spain, 2012), 389–396 (cit. on pp. 3, 14).

23. Nichols, K. & Jacobson, V. Controlling queue delay. *Communications of the ACM* **55,** 42–50 (2012) (cit. on p. 69).

24. Vega, D., Cerda-Alabern, L., Navarro, L. & Meseguer, R. *Topology patterns of a community network: Guifi.net* in *Proc. Int. Conf. Wireless and Mobile Computing, Networking and Communications (WiMob)* (Barcelona, Spain, 2012), 612–619 (cit. on p. 3).

25. Boushaba, M. & Hafid, A. *Best path to best gateway scheme for multichannel multi-interface wireless mesh networks* in *Wireless Communications and Networking Conference (WCNC)* (IEEE, 2011), 689–694 (cit. on pp. 48, 49).

26. Catrein, D. *et al. An Analysis of Web Caching in Current Mobile Broadband Scenarios* in *New Technologies, Mobility and Security* (2011), 1–5 (cit. on p. 26).

27. Chen, Y., Wang, X., Shi, C., Lua, E. K., Fu, X., Deng, B. & Li, X. Phoenix: A Weight-Based Network Coordinate System Using Matrix Factorization. *IEEE Transactions on Network and Service Management* **8,** 334–347 (2011) (cit. on p. 49).

28. Ihm, S. & Pai, V. S. *Towards understanding modern web traffic* in *Internet measurement conference* (2011), 295–312 (cit. on p. 27).

29. Johnson, D. L., Pejovic, V., Belding, E. M. & van Stam, G. *Traffic characterization and internet usage in rural Africa* in *World Wide web* (2011), 493–502 (cit. on p. 26).

30. Khare, R. & Lawrence, S. *A Survey of Lower-than-Best-Effort Transport Protocols* RFC 6297. 2011 (cit. on p. 69).

31. Afanasyev, M., Chen, T., Voelker, G. & Snoeren, A. Usage patterns in an urban WiFi network. *IEEE/ACM Transactions on Networking* **18,** 1359–1372 (2010) (cit. on p. 26).

32. Ancillotti, E., Bruno, R. & Conti, M. *Load-balanced routing and gateway selection in wireless mesh networks: Design, implementation and experimentation* in *World of Wireless Mobile and Multimedia Networks (WoWMoM)* (2010), 1–7 (cit. on pp. 48, 49).

33. Dovrolis, C., Gummadi, K., Kuzmanovic, A. & Meinrath, S. D. Measurement Lab: Overview and an Invitation to the Research Community. *ACM SIGCOMM Computer Communication Review* **40,** 53–56 (2010) (cit. on p. 67).

34. Ashraf, U., Abdellatif, S. & Juanole, G. *Gateway selection in backbone wireless mesh networks* in *Wireless Communications and Networking Conference, (WCNC)* (IEEE, 2009) (cit. on pp. 48, 49).

35. Chen, Y., Xiong, Y., Shi, X., Zhu, J., Deng, B. & Li, X. Pharos: accurate and decentralised network coordinate system. *IET Communications* **3,** 539–548 (2009) (cit. on p. 49).

36. Lancichinetti, A. & Fortunato, S. Community detection algorithms: A comparative analysis. *Phys. Rev. E* **80,** 056117 (Nov. 2009) (cit. on p. 54).

37. Laoutaris, N., Smaragdakis, G., Rodriguez, P. & Sundaram, R. *Delay tolerant bulk data transfers on the internet* in *ACM SIGMETRICS Performance Evaluation Review* **37** (2009), 229–238 (cit. on p. 66).

38. Maier, G. *et al. On Dominant Characteristics of Residential Broadband Internet Traffic* in *Internet Measurement Conference* (2009), 90–102 (cit. on p. 16).

39. Brik, V. *et al. A measurement study of a commercial-grade urban wifi mesh* in *Internet measurement conference* (2008), 111–124 (cit. on p. 26).

40. Galvez, J. J., Ruiz, P. M. & Skarmeta, A. F. G. *A distributed algorithm for gateway load-balancing in Wireless Mesh Networks* in *Wireless Days* (2008), 1–5 (cit. on p. 48).

41. Ledlie, J., Seltzer, M. & Pietzuch, P. Proxy network coordinates. *Target* **22,** 25 (2008) (cit. on pp. 31, 34, 35).

42. Bortnikov, E., Cidon, I. & Keidar, I. in *Distributed Computing* 77–91 (Springer Berlin Heidelberg, Berlin, Heidelberg, 2007) (cit. on p. 49).

43. Berkhin, P. in (eds Kogan, J., Nicholas, C. & Teboulle, M.) 25–71 (Springer, 2006) (cit. on p. 52).

44. Akyildiz, I. F., Wang, X. & Wang, W. Wireless Mesh Networks: A Survey. *Computer networks* **47,** 445–487 (2005) (cit. on p. 2).

45. Dabek, F., Cox, R., Kaashoek, F. & Morris, R. Vivaldi: A decentralized network coordinate system. *ACM SIGCOMM Computer Communication Review* **34,** 15–26 (2004) (cit. on pp. 31, 34, 35, 43, 49).

46. Goldenberg, D. K., Qiuy, L., Xie, H., Yang, Y. R. & Zhang, Y. Optimizing cost and performance for multihoming. *ACM SIGCOMM Computer Communication Review* **34,** 79–92 (2004) (cit. on pp. 5, 66).

47. Spring, N., Wetherall, D. & Anderson, T. *Scriptroute: A Public Internet Measurement Facility* in *USENIX Symposium on Internet Technologies and Systems (USITS)* **4** (USENIX Association, 2003), 17–17 (cit. on p. 33).

48. Cerf, V. *The Internet is for everyone* RFC 3271. 2002 (cit. on p. 1).

49. Shalunov, S. & Teitelbaum., B. *QBone Scavenger Service (QBSS) Definition. Internet2 Technical Report, Proposed Service Definition, Internet2 QoS Working Group Document* tech. rep. (2001) (cit. on p. 66).

50. Feldmann, A. *et al. Performance of web proxy caching in heterogeneous bandwidth environments* in *INFOCOM* (1999), 107–116 (cit. on p. 26).

51. Ostrom, E. *Governing the commons: the evolution of institutions for collective action* (Cambridge University Press, Nov. 1990) (cit. on p. 3).

52. Spearman, C. The proof and measurement of association between two things. *The American journal of psychology* **15,** 72–101 (1904) (cit. on p. 42).