Community Sharing of Spare Network Capacity

Emmanouil Dimogerontakis*‡, Roc Meseguer*, Leandro Navarro*, Sergio F. Ochoa†, Luís Veiga‡

* Universitat Politècnica de Catalunya
Barcelona, Spain
{edimoger,meseguer,leandro}@ac.upc.edu

† Universidad de Chile
Santiago, Chile
sochoa@dcc.uchile.cl

‡Tecnico Lisboa/INESC-ID Lisboa
Lisboa, Portugal
luis.veiga@inesc-id.pt

***Abstract*—In several community scenarios, people share benevolently their spare broadband Internet access connectivity with other people who cannot afford it. Although laudable, this sharing process can negatively affect the service received by the primary users, thus jeopardizing the continuity of this community service. In this paper we propose the use of a gateway that separates the traffic of the primary users from that of the secondary users, the beneficiaries of this sharing. We analyze the impact and behavior of several mechanisms for using this gateway, to determine how to maximize network utilization, use of the excess network capacity, and minimize the impact on the primary traffic. As a result we present a set of lessons learned and recommendations. Particularly, some strategies that use tunneling for managing the primary and secondary traffic achieve the best performance isolation for the primary user, while the secondary user obtains the spare capacity equivalent to non-differentiated best effort, with a limited penalty (around 20%). Combined with complementary queueing techniques (instead of FIFO), other important flows for the user experience (such as DNS or ICMP) can be practically unaffected.**

Keywords*—*Internet access sharing, community network, user experience, proxy service.

I. Introduction

Internet is for everyone [1], as Vint Cerf says *"it won't be if it isn't affordable by all that wish to partake of its services"*. Global access to Internet for everybody requires not only the increase of the service availability, but also a dramatic reduction of its cost, especially in geographies and populations with low penetration (e.g., rural and underserved communities, and also retired people) [2]. In this paper 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. 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 of the donors from the one of the beneficiaries, 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 serve them adequately. 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, in case of data volume or 95-percentile pricing schemes. This can be a strong demotivating factor for the donors of Internet access resources, therefore, making this sharing process innocuous for them is critical to make the Web sharing service sustainable over time. However, keeping under control this aspect of the traffic represents a major challenge for the administration of community networks. To help address this challenge, we analyze several of the mechanisms for sharing the spare Internet capacity among third parties in a community network (guifi.net [3]), 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 to make this sharing process suitable and sustainable.

The next section presents the background and an analysis of the study scenario. Section III describes the experimental framework, and it shows the evaluation results in Section IV. Section V presents the lessons learned and conclusions.

II. Background

Our study scenario is defined by M citizens connected to guifi.net [3], a self-managed regional network infrastructure that provides low cost regional connectivity. A community network of N citizens offers its spare Internet capacity through a gateway device. These gateways can offer IP or HTTP access: IP tunnels, Web proxies and TCP over CONNECT HTTP method [4].

We define *spare Internet capacity* as the network traffic that can be transferred to and from the Internet by secondary users (using Internet connectivity provided by others) with no significant performance degradation or cost penalty for primary users (sharing their Internet connectivity). The secondary traffic may have short-term impact on the primary in terms of packet queueing, resulting in service degradation due to packet delays, loss and reduced throughput. This affects data transport generating a lower throughput with longer and more variable download times, and also an overall degradation of the user experience quality.

Given that Internet connectivity may be given to primary users according to a cost model (e.g. 95% percentile, data volume), the secondary traffic might result in a cost penalty that should be considered and ideally avoided. For instance, the secondary traffic may need to be blocked to avoid having an impact on cost under the common 95-percentile pricing schemes [5] used by transit Internet Service Provider (ISP) to charge according to peak demand. 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. Therefore, the goal of any regulation would be that the secondary traffic does not affect the primary users in terms of cost and performance. Achieving this goal depends on the total traffic, primary and secondary, and should apply both when the Internet connection is above and below the saturation point. Some previous works [6] [7] have shown that water-filling (taking advantage of already-paid-for off-peak bandwidth

978-1-5090-4429-0/17/$31.00

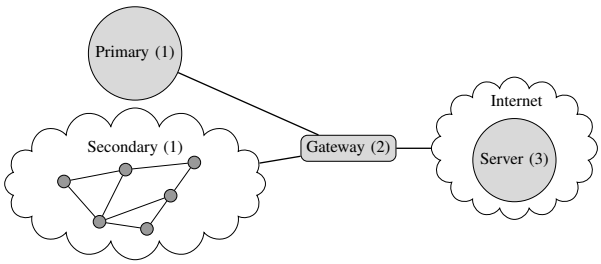

Fig. 1. Physical architecture of the testbed.

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. This work schema was proposed in [8] for a heterogeneous wireless mesh network, but it is applicable to other types of community networks.

There is a rich body of work focused on reducing the cost and increasing Internet coverage under several scenarios. For instance, the Lowest Cost Denominator Networking (LCD-Net) [9] explores resource pooling Internet technologies to support benevolence on the Internet. Some of these ideas are illustrated by WiFi sharing schemes, community-led (PAWS [10]) or commercially-run (FON [11]), 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 [12]. 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. Our research also focuses on 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 capacity occupied by the primary.

III. Experimental framework

The system model, as well as the scenario for experimental evaluation, consists of a gateway middlebox that separates the primary traffic from the secondary that originates from a number of nodes of the local access network, part of a wired or wireless community network. All primary and secondary traffic is destined to servers on the Internet. For our experiments it was assumed that 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 assumed that all clients interact with a single server that provides content to both primary and secondary clients. Figure 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 connects them with servers on the Internet (3). The gateway node manages both primary and secondary traffic, and applies different techniques to each traffic, considering the limited capacity of the available Internet access uplink, while trying to assess and minimize the impact of secondary traffic on the primary one. All the experiments were performed on the testbed described above, which was deployed in a laboratory type of environment.

A. Internet Access Model with Primary and Secondary Users

The characteristics of Internet access described here are enforced on the gateway using the traffic control and queueing discipline tools available on Linux. The Clients-Gateway connection throughput is 100Mbps. The Gateway-Server download throughput is limited to 1.72Mbps, based on the values presented in [13] for Telefonica, the largest ISP in Spain. We validated that these values were imposed correctly on the testbed using the Network Diagnostic Test, a tool that is used to characterize real ISPs. In addition to the modeled values, we validated that the modeled access behaves as expected. In the described model the bottleneck is the gateway, who bridges the fast local and the slow external connections.

B. Traffic modeling

In the scenario of community networks, and specifically in the guifi.net [3], gateways act as Web proxies and therefore the traffic will be HTTP. We have used the *wrk2*[1] tool to generate customer traffic. This allows us to perform realistic HTTP benchmarking removing the effects of "coordinated omission" [14] from the measurements.

C. Metrics

The goal of this study was to compare the different mechanisms to share spare capacity. As best case point of reference we define the case where there is only traffic originating from the primary (*prim_only*). The baseline case we want to improve is the best-effort case where there is primary and secondary traffic competing for the Internet capacity without any regulation mechanism (*best_effort*). To evaluate the behavior of the tests mechanisms proposed later we utilize two metrics: 1) the co-inflicted delay on the service time of HTTP requests for each mechanism - normalized to the mean latency across the best-effort primary and secondary traffic measured values - and 2) the network throughput.

D. Traffic sharing between primary and secondary

The mechanisms of "traffic engineering" have to act only on the gateway without requiring end-to-end changes, be transparent to clients and servers, and be innocuous (i.e., to have no impact or cost) when the gateway is not congested.

We experimented with three types of mechanisms based on traffic shaping (TS), Active Queue Management (AQM) and tunneling. For the mechanism based on traffic shaping, the gateway monitors traffic and discards non-compliant packets according to the spare capacity. In the case of AQM, the gateway does not use a FIFO strategy for packets, but tries to prioritize packages by type or flow. Finally, in the last case we used tunneling to replace the congestion control of the end-to-end transport protocol to that of the tunnel. All these mechanisms are implemented on the gateway and applied on HTTP traffic send by the clients to the gateway.

Figure 2 shows the location of these three types of mechanisms on the network stack, indicating the types of test applied to the primary traffic and secondary one. Next, we explain the mechanisms considered in more detail.

[1] https://github.com/giltene/wrk2

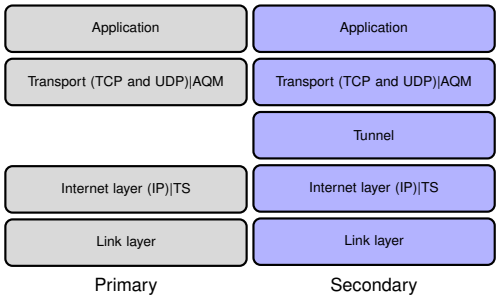

Fig. 2. Types of tests applied to primary and secondary traffic.

1) Based on traffic shaping: We adopted a *borrowing* strategy, according to which the primary and the secondary traffic have a guaranteed minimum throughput. The unused throughput is considered as borrowed by the secondary since the primary has priority in utilizing it.

2) Based on active queue management: Here we used the Stochastic Fairness Queueing (*sfq*) [15] and *CoDel* mechanisms (*codel*) [16]. The former one is used by several ISPs [17], since it tries to order the packets in a more fair manner (a packet from each TCP flow). The latter mechanism has been successfully used to significantly mitigate the bufferbloat phenomenon [16].

3) Based on tunneling: In this case we used three strategies: *TCP Cubic* (*tcpcubic*) [18], *TCP Vegas* (*tcpvegas*) [19] and *TCP LP* (*tcplp*) [20]. In the first case we used the tunnel's TCP congestion control algorithm (*tcpcubic*) to manage the secondary traffic. In the second case, the secondary traffic was managed through a *tcpvegas* tunnel, and the congestion avoidance algorithm emphasized packet delay (Rourd-Trip Time or RTT) rather than packet loss. In the *tcplp* case, the secondary traffic was managed through a TCP type low-priority tunnel, with the idea of controlling congestion [21]. This approach gives less priority to the secondary traffic than the best effort approach, with its main goal to utilize only the spare network bandwidth.

IV. Results

The experiments in this study are intended to evaluate the impact of secondary traffic on the primary traffic, considering the *prim_only* and *best_effort* cases as reference. The impact of the different techniques on the user experience is measured both when the gateway is not overloaded, and when the gateway is saturated. Additionally, we explored the sensitivity of the studied mechanisms to the characteristics of the traffic, the overhead of tunneling, and the overhead of using WiFi links, typical for access networks such as community networks. 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.

A. Gateway not overloaded

As a first step, we studied the scenario of a not overloaded gateway, where the traffic model consists of a single primary user with two concurrent connections each, and four to five secondary users with ten concurrent connections each. All HTTP requests involve 0.1MB objects. There is a random user think time between HTTP requests that ranges from 10 to 50 ms. As described earlier, the Internet connection is modeled to have

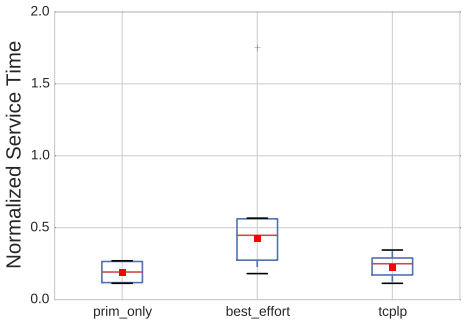

Fig. 3. Service time of primary traffic with underutilized Internet connection.

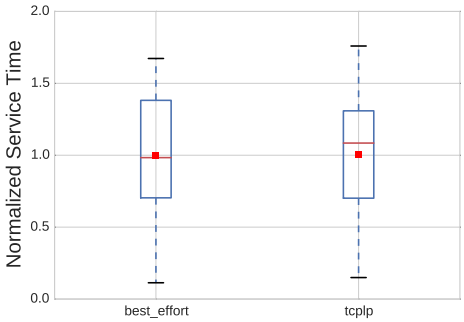

Fig. 4. Service time of secondary traffic with underutilized Internet connection.

a maximum download throughput of 1.72Mbps. The resulting total traffic (primary + secondary) does not exceed on average the maximum throughput of the connection. Although both traffics compete, there is sufficient throughput for both of them.

From the results shown in Figures 3 and 4 we observe that the difference of service time for the primary traffic (between *prim_only* and *best_effort*) is noticeable but not significant considering that the absolute latency experienced by the user when downloading 0.1MB is not very high. Thus, in this scenario, the *best_effort* strategy is already usable from the perspective of the primary user, without requiring major improvements. Moreover, it can be seen that the service time achieved by the secondary traffic has a large impact on the service time of the primary. Looking at the service time of the primary traffic, *tcplp* offers values very close to *prim_only*. However, if we consider the service time of the secondary traffic, the *tcplp* mechanism offers comparable values to *best_effort*. Therefore, it manages to improve the primary traffic without penalizing the secondary.

As shown, when the gateway is not saturated, applying a regulation mechanism, like *tcplp*, is not necessary and has no significant effect on the primary traffic. Therefore, the rest of this work will focus only on cases where the gateway is overloaded.

B. Gateway is overloaded

To simulate and reproduce an overloaded Internet connection, we used the following HTTP traffic generation model: the primary traffic (represents one user) was generated at the rate of 5 requests per second, while the secondary one (represents 5 users) was generated at the rate of 25 requests per second. All the HTTP requested objects have a fixed size of 12.5KB, except if explicitly stated otherwise. Moreover, the primary traffic was generated with a random *user think time* in the range

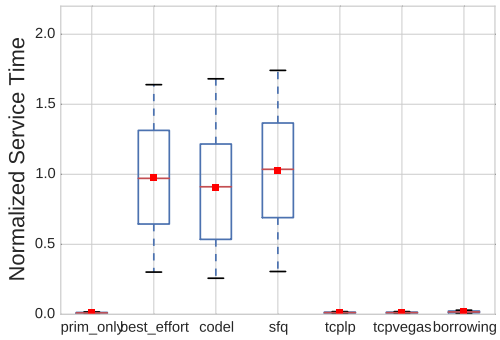

Fig. 5. Effect on primary service time under saturated Internet connection.

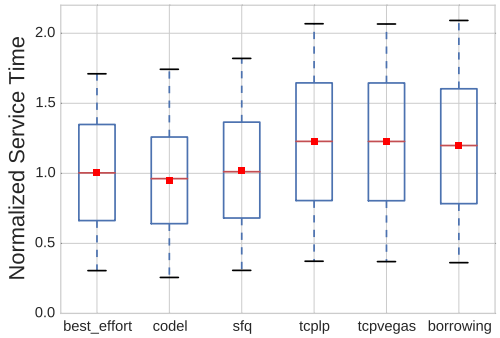

Fig. 6. Effect on secondary service time under saturated Internet connection.

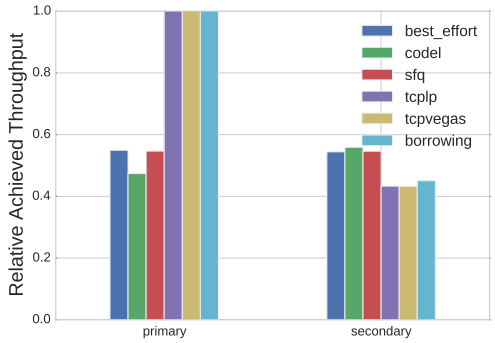

Fig. 7. Throughput comparison under saturated Internet connection.

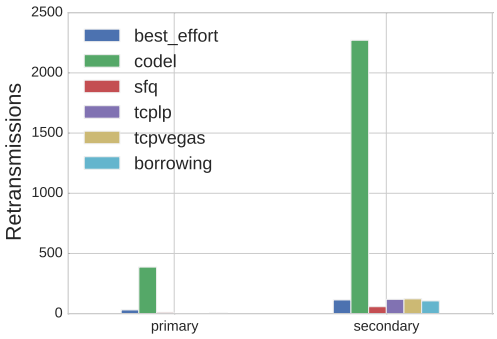

Fig. 8. Retransmission comparison under saturated Internet connection.

of 10-50ms between every request. As before, the download throughput of the Internet connection is limited to 1.72Mbps, to provide a more realistic experimental environment. Across all the experiments presented in this section, the total traffic was generated with a rate greater than 1.72Mbps to achieve saturation of the Internet connection. As a result, the primary and the secondary traffic had to compete to access the Internet. *codel* and *sfq* were applied to all traffic without differentiating primary and secondary. The results are shown in Figures 5 to 8.

Figures 5 and 6 show that the difference of service time between *prim_only* and *best_effort* (for the primary traffic) is substantial. The secondary traffic has both a very large impact on the primary service time, and achieves a service time much worse than *prim_only*, resulting to poorer utilization for all the users. Using *tcplp*, *tcpvegas* and *borrowing*, the primary user experiences a very good service time, while the secondary traffic service time is much less than the corresponding primary's gain (∼20%). *Codel* achieves only a very slight improvement to both the primary and secondary service time compared to *best_effort*, while *sfq* does not show significant improvement. As far as throughput is concerned, the success of the mechanisms to prioritize the primary traffic, without significantly deteriorating the experience of the secondary users, follows exactly the same patterns as latency, and can be seen in Figure 7.

The effect on the user's TCP flows is illustrated by Figure 8. We observe that the number of retransmissions is very low compared to the number of HTTP requests performed in each experiment with the exception of *codel* that shows its sensitivity to maximum number of flows that can be served.

As described earlier, the secondary traffic has a very significant impact on the service time and throughput of the primary traffic when competing for access on an overloaded gateway. *tcplp*, *tcpvegas* and *borrowing* seem to able to prioritize primary. Additionally, they penalize the secondary traffic, with latency and throughput values slightly worse than *best_effort*.

C. Sensitivity analysis

Investigating the sensitivity of the proposed mechanisms to characteristics of the traffic and the environment, we analyzed their relation of the service time, to the distribution of concurrent requests and to the object size.

1) Object Size: For object size experiment the results are normalized based on the mean across the primary and secondary service time results for 0.1 Mb. Figure 9 show that there is a clear relationship between service time and object size. As expected, increasing the object size results in an increased service time, the amount of data per second requested is increased as well. Moreover, as far as the primary traffic is concerned, *codel* and *sfq* present a behavior close to the *best_effort* mechanism, but slightly varying based on the object size. *tcplp* and *tcpvegas* are the solutions with behavior closest to *prim_only*. The *borrowing* strategy seems to have a performance similar to *prim_only*, but 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*. Additionally, we observe that there is a very similar pattern of increasing service time while increasing object size for all secondaries.

2) Proportion of requests primary-secondary: In this experiment we varied the proportion of requests between primary and secondary traffic, while keeping the total throughput and the total number of requests. The results are normalized based on the mean throughout the primary and secondary service

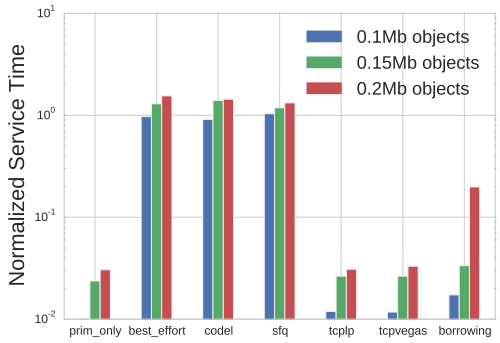

Fig. 9. Primary traffic service time sensitivity on Object Size of the strategies.

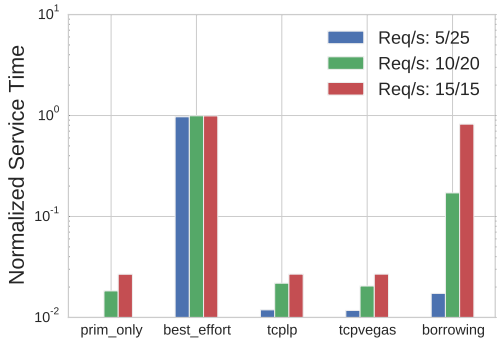

Fig. 10. Primary traffic service time sensitivity on requests rate.

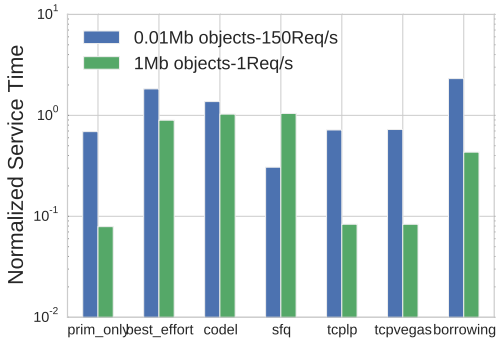

Fig. 11. Primary traffic Service time comparison on edge scenarios.

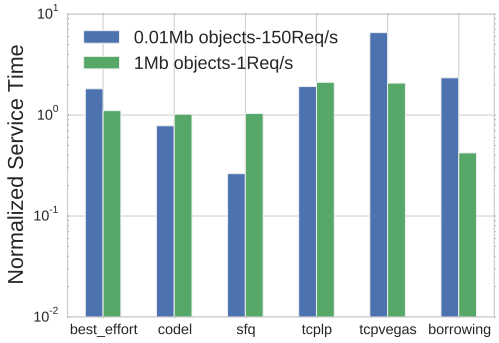

Fig. 12. Secondary traffic Service time comparison on edge scenarios.

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 10 only presents strategies affected by those changes, omitting *codel* and *sfq*.

Considering the service time of the primary traffic, *tcplp* and *tcpvegas* still behave very similar to *prim_only*. However, *borrowing* presents again a differentiated behavior, being 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 increases accordingly. Regarding the secondary traffic, all service times are close to the *best_effort* service time.

3) "Edge Cases" for Object Size/Requests Proportion: This experiment shows how the different strategies function in extreme cases while maintaining a fixed overall throughput. More specifically it compares the case of many req/s requesting very small objects, which fit in a single TCP frame, to the case of 1 req/s with a large object. For this experiment, the results were normalized based on the mean across the primary and secondary service time for the pair of 0.01Mb objects - 150 reqs/s. From Figures 11 and 12 we can observe that these scenarios are consistent with the previous ones. *sfq* and *codel* provide only small improvements, mostly when there are many requests. As far as the primary traffic is concerned, *tcplp* and *tcpvegas* behave almost like *prim_only*, with an advantage for *tcplp* with small objects. The *borrowing* strategy seems to aggravate the situation when there are many requests, while it improves (for both primary and secondary traffic) with larger objects.

4) Other Factors: We compared the differences in behavior among different types of tunnels. Comparing the experiment results we observed that the IP-over-IP tunnel has the same behavior as *best_effort* on 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 behavior of *tcplp* and *tcpvegas* used in the secondary tunnels against *tcpcubic* that was used for the the primary traffic and as transport under the tunnels. We observed that the aggressiveness of *tcpcubic* is the reason that *tcpvegas* behaves similarly to *tcplp* in our experiments. Substituting *tcpcubic* in the primary tunnels for other TCP algorithm we expect to obtain similar results for *tcplp* tunnels, while *tcpvegas* tunnels would deteriorate the primary and improve the secondary service time. This behaviour of *tcpvegas* sets *tcplp* as our best mechanism for sharing the spare Internet capacity so far.

Moreover, we evaluated the overhead of wireless (WiFi-based) links for secondary users, as it is quite common in access networks, such as community networks. We used an ad-hoc (IBSS) network in channel 4 of the 2.4 GHz band. The experiments show equivalent results to a wired Ethernet connection, except a slight improvement for *codel* and *sfq* both for the primary and secondary traffic.

From the results obtained we reach to the following lessons learned. The *sfq* and *codel* mechanisms provide a behaviour very similar to the *best_effort* scenario. The *borrowing* mechanism is sensitive to object size and to the distribution of the number of concurrent connections. Therefore these three mechanisms cannot be considered as good options for real-environment application. Furthermore, *tcplp* and *tcpvegas* behave very close to *prim_only*, as far as the primary traffic is concerned, even in edge scenarios, without creating great

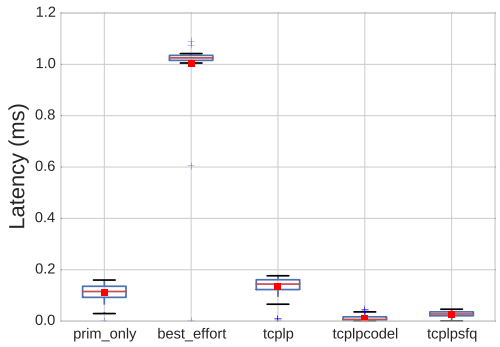

Fig. 13. Primary client ping latency combining *AQM* and *tcplp*.

extra overhead to the secondary. Between these two options, *tcplp* appeared to penalize less the secondary traffic, hence, it ranks so far as the best candidate.

Finally, we want to mention that there are other important types of traffic in a network that may be affected by the secondary traffic and the saturation of the Internet connection; for instance, some important protocols such as DNS, ICMP or packets like SYN. The results in Figure 13 show that even in the case of an overloaded gateway, *codel* and *sfq* contribute to improve the behavior of *tcplp*. Flows with very few packets, such as ICMP ping in the figure, no longer suffer from "starvation" by virtue of not being trapped in a FIFO queue. Additionally, we observe that utilizing existing AQM techniques like *codel* and *sfq*, that are already available and implemented, we can achieve improved primary responsiveness even compared to *prim_only*. While this can depend on the traffic type, it shows potential for a more mainstream adoption of these techniques as part of the default TCP/IP stack. The same effect is achieved for the secondary traffic, as well as in the primary and secondary traffic for a non-overloaded gateway.

V. Conclusions

In this paper, we investigate the cost reduction in Internet access by citizens that benevolently share with others, through a local community network, studying techniques that can guarantee Web experience of the donors. We evaluated the performance and drawbacks for the primary and secondary users of several mechanisms for sharing spare Internet. In summary, *tcplp* appears to be the most promising option, regardless of whether the gateway is saturated or not. The primary traffic is apparently not affected by the secondary one, behaving like if the primary client was performing request without competing for the Internet capacity. The secondary traffic achieves to utilize the spare capacity, behaving like the non-differentiated case, best effort, with a limited penalty around 20% on the latency. Combined with complementary queueing techniques (e.g., *tcplp* + *codel* or *tcplp* + *sfq*) instead of just a FIFO queue, it allows to "treat well" other small, but important for the user experience, traffic types, 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.

Acknowledgements

This work was partially supported by the Erasmus Mundus Joint Doctorate in Distributed Computing (EMJD-DC) funded by the European Commission (FPA-2012-0030), the H2020 project netCommons (H2020-688768), the Spanish government (TIN2016-77836-C2-2-R), and Portuguese funds through Fundação para a Ciência e a Tecnologia (UID/CEC/50021/2013).

References

[1] V. Cerf, "The internet is for everyone," RFC 3271, 2002. [Online]. Available: https://tools.ietf.org/html/rfc3271

[2] G. WG, "Global access to the internet for all research group," 2016, [Online; accessed 14-September-2016]. [Online]. Available: https://irtf.org/gaia

[3] D. Vega, R. Baig, L. Cerdà-Alabern, E. Medina, R. Meseguer, and L. Navarro, "A technological overview of the guifi.net community network," *Computer Networks*, vol. 93, pp. 260–278, 2015.

[4] M. Welzl, S. Gjessing, and N. Khademi, "Less-than-best-effort service for community wireless networks: Challenges at three layers," in *Wireless On-demand Network Systems and Services (WONS)*, Obergurgl, Austria, 2014, pp. 148–153.

[5] D. K. Goldenberg, L. Qiuy, H. Xie, Y. R. Yang, and Y. Zhang, "Optimizing cost and performance for multihoming," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 4, pp. 79–92, 2004.

[6] S. Shalunov and B. Teitelbaum., "Qbone scavenger service (qbss) definition. internet2 technical report, proposed service definition, internet2 qos working group document," Tech. Rep., 2001. [Online]. Available: http://qbone.internet2.edu/qbss/

[7] N. Laoutaris, G. Smaragdakis, P. Rodriguez, and R. Sundaram, "Delay tolerant bulk data transfers on the internet," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 37, no. 1, 2009, pp. 229–238.

[8] E. Dimogerontakis, J. Neto, R. Meseguer, L. Navarro, and L. Veiga, "Client-side routing-agnostic gateway selection for heterogeneous wireless mesh networks," in *IFIP/IEEE Int. Symp. Integrated Network Management (IM)*, Lisboa, Portugal, 2017, pp. 1–8.

[9] A. Sathiaseelan and J. Crowcroft, "Lcd-net: lowest cost denominator networking," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 2, pp. 52–57, 2013.

[10] A. Sathiaseelan, J. Crowcroft, M. Goulden, C. Greiffenhagen, R. Mortier, G. Fairhurst, and D. McAuley, "Paws: Public access wifi service," in *The Third Digital Economy All-hands Meeting: Digital Engagement (DE), Aberdeen, Scotland/United Kingdom*, 2012.

[11] Fon. (2016) Fon corporation. [Online; accessed 10-February-2017]. [Online]. Available: https://fon.com

[12] A. Abujoda, D. Dietrich, P. Papadimitriou, and A. Sathiaseelan, "Software-defined wireless mesh networks for internet access sharing," *Computer Networks*, vol. 93, Part 2, pp. 359–372, 2015.

[13] B. Braem, J. Bergs, C. Blondia, L. Navarro, and S. Wittevrongel, "Analysis of end-user qoe in community networks," in *Computing for Development (ACM-DEV)*, London, UK, 2015, pp. 159–166.

[14] G. Tene, "How not to measure latency," in *Low Latency Summit*, 2013.

[15] P. E. McKenney, "Stochastic fairness queueing," in *INFOCOM '90, Ninth Annual Joint Conference of the IEEE Computer and Communication Societies. The Multiple Facets of Integration. Proceedings, IEEE*, Jun 1990, pp. 733–740 vol.2.

[16] K. Nichols and V. Jacobson, "Controlling queue delay," *Communications of the ACM*, vol. 55, no. 7, pp. 42–50, Jul. 2012.

[17] Y. Gong, D. Rossi, C. Testa, S. Valenti, and M. D. Täht, "Fighting the bufferbloat: on the coexistence of aqm and low priority congestion control," *Computer Networks*, vol. 65, pp. 255–267, 2014.

[18] S. Ha, I. Rhee, and L. Xu, "Cubic: A new tcp-friendly high-speed tcp variant," *SIGOPS Oper. Syst. Rev.*, vol. 42, no. 5, pp. 64–74, Jul. 2008.

[19] S. Low, L. Peterson, and L. Wang, "Understanding TCP vegas: Theory and practice," Tech. Rep., 2000. [Online]. Available: ftp://ftp.cs.princeton.edu/techreports/2000/616.pdf

[20] A. Kuzmanovic and E. W. Knightly, "Tcp-lp: Low-priority service via end-point congestion control," *IEEE/ACM Transactions on Networking*, vol. 14, no. 4, pp. 739–752, Aug. 2006.

[21] R. Khare and S. Lawrence, "A survey of lower-than-best-effort transport protocols," RFC 6297, 2011. [Online]. Available: https://tools.ietf.org/html/rfc6297