Transparent Bandwidth Aggregation for Residential Access Networks

Yufeng Duan*, Paolo Giaccone*, Pino Castrogiovanni†, Dario Mana†, Claudio Borean† and Claudio Rossi‡
*Department of Electronics and Telecommunications, Politecnico di Torino, Turin, Italy
†Telecom Italia, Torino, Italy
‡Microsoft Innovation Center, ISMB, Torino, Italy
{yufeng.duan,paolo.giaccone}@polito.it, {pino.castrogiovanni,dario.mana,claudio.borean}@telecomitalia.it, rossi@ismb.it

***Abstract*—In this paper we propose, implement and evaluate a bandwidth aggregation service for residential users that enhances the throughput of their Internet broadband connection through the aggregation of available capacity at neighboring broadband links. Network resources are aggregated by the residential access gateway using the 802.11 radio interface to simultaneously serve home users and to share the broadband connectivity with neighboring access gateways. Differently from previous works, our aggregation scheme is transparent both for local users, who are not required to modify their applications or device drivers, and for neighboring users, who do not experience any meaningful performance degradation. The proposed approach aims at a commercial deployment, leveraging on existing access gateways and ADSL-based access networks.**

Keywords*—*Wireless access networks, access point aggregation, ADSL, traffic scheduling.

I. Introduction

The growing popularity of high-speed Wi-Fi technologies (as IEEE 802.11n) deployed in residential networks has exacerbated the inequality between the bandwidth available in local domestic networks and the bandwidth available to access the Internet Service Provider (ISP) network. Notwithstanding EU plans for fast broadband foresee 100% coverage of 20 Mbps (or more) access connections for EU citizens by 2020, the majority of member states are still on the way to support basic broadband access connections, based on ADSL technology [1]. Basic ADSL provides an aggregate capacity (around 1-10 Mbps) typically lower than local area networks based on Ethernet and Wi-Fi, and constitutes often the performance bottleneck for residential users accessing Internet-based applications (as cloud computing and storage). Consequently, several methods have been recently proposed to increase the performance perceived by the domestic users by aggregating the available bandwidth of neighboring Wi-Fi Access Points (APs). This approach is practically enabled by the high density of APs in residential areas, which provides overlapping radio coverage. A recent study [2] showed that, in a metropolitan area, a generic AP can see up to 52 neighbors, with a median value around 7. Furthermore, despite the strong correlation of the residential daily Internet traffic at the aggregated level, which follows nicely a day-night sinusoidal traffic shape, the traffic correlation among neighboring users is typically small due to different habits and behaviors of each person. Thus the instantaneous traffic of neighboring ADSL connections is often uncorrelated, enabling statistical multiplexing of the traffic among neighboring users. State-of-art solutions for access bandwidth aggregation require modifications at client side, such as custom drivers or specific applications to be installed on users' devices. This makes their deployment not commercially viable due to chipsets variety, to operating systems diversity, and to additional constraints imposed by smartphones and tablets development environments.

In this paper we present Beyond One's Bandwidth (BOB), a distributed gateway-centric system exploiting the collaboration between multiple Access Gateways (AGs) to provide a higher Internet connection speed to residential users without any software or hardware modification at the client side. The AG is a standard access device that integrates a broadband modem, a network-layer router, and a Wi-Fi AP. Residential devices such as laptops, smart phones and TVs can access the Internet by associating to the AP or by connecting through Ethernet. We design BOB to meet the transparency requirements of a real commercial deployment. Each AG in BOB is responsible of constructing a dedicated wireless topology with other nearby AGs, forwarding portion of traffic to neighbors while guaranteeing that each user is able to fully exploit his own broadband connection bandwidth, without observing any meaningful performance degradation due to the cooperative sharing scheme. We implement and test BOB in an operational scenario with several typical clients. The results show that BOB can provide a substantial increase of the throughput with negligible overhead under synthetic traffic and real applications.

The paper is organized as follows. In Sec. II we discuss the relevant previous work. In Sec. III we describe the whole architecture, whose detailed implementation is discussed in Sec. IV. Finally, we evaluate experimentally the performance of BOB in Sec. V and draw our conclusions in Sec. VI.

II. Related Work

In recent years, many works have been focused on the bandwidth aggregation problem in the context of the access network. FatVAP [3] is one of the first works on aggregating the access broadband bandwidth of multiple APs. FatVAP has two main contributions: an 802.11 driver that enables a client to connect to multiple APs using a single radio wireless chip; a scheduler that enables a client to decide to which APs to connect to maximize the throughput. Unfortunately, the proposed approach is not suitable for domestic users of ISPs, since it is not transparent for the user, who is indeed required to install a new driver that is not integrated in an off-the-shelf operating system and maybe not supported by the

available chipset of the wireless card. In addition to that, [3] does not involve an authentication procedure when connecting to an AP and for this reason it cannot be adopted in a residential access network. The work in [4] extends [3] and addresses specifically the security issue when connecting to multiple APs. It proposes a fast authentication mechanism, but it is not compatible with legacy 802.11 security protocols. Furthermore, clients connect directly to each other using a virtual interface connected in ad-hoc mode, limiting severely the portability of the approach. Finally, whenever a client shuts down, it cannot share anymore the backhaul bandwidth to others, even if its access gateway is still working. In [5] a new scheduling scheme is proposed that guarantees a fair access among multiple clients, which overcomes the drawback of FatVAP that only maximizes the performance of a single client, neglecting the potential unfairness among all the clients. Finally, [6] proposes to aggregate the access bandwidth by exploiting the transmission on overlapping channels, but it relies completely on a non-standard communication technology. In [7] a new opportunistic approach is proposed to aggregate the backhaul bandwidth. The main idea is that the client sends a packet in broadcast towards all the APs and each AP runs a scheduler that decides whether to forward this packet. As a result, it requires a strong cooperation among the clients, APs, and the servers, only achievable with customized devices. Furthermore, broadcast communications in 802.11 are inefficient in terms of bandwidth, occurring always at the minimum data rate.

One of the most relevant work to ours is SmartAP [8]. Evolving from [5], SmartAP develops an AP-based scheduling algorithm that tries to maximize the overall throughput of all the clients. SmartAP is transparent for the user, since it does not require modification on the applications, the wireless card driver, or the operating system. An optimization algorithm coordinates the traffic flows among neighboring APs in order to maximize the bandwidth. To achieve some transparent behavior for the local user of an AP, each AP must estimate its own available bandwidth and communicate to a central controller running the optimization algorithm. Due to the latencies for the bandwidth estimation and the centralized control, the approach slowly reacts to fast varying loads. Unlike SmartAP, our approach is completely distributed, since each Access Gateway runs a traffic control scheme independently from the others that also guarantees to preserve the bandwidth for the local traffic. Another relevant work is 3GOL [9], which shares the same motivation of our work, i.e. boosting the speed of ADSL users. Instead of aggregating the bandwidth of multiple APs, [9] proposes to exploit a parallel cellular connection to increase the speed of a home user. The proposed approach is not transparent for the user, which requires to install a dedicated scheduler in his clients.

III. Cooperative Bandwidth Aggregation

We consider a residential access network of an ISP, in which each household is equipped with an AG. The AG is connected through an ADSL line to the ISP's POP and provides connectivity to users' devices through an 802.11 interface. Fig. 1 shows a basic example of 3 AGs that cooperate to implement BOB bandwidth aggregation scheme. The wireless interface of each AG is connected to the *local* devices but also to the *neighboring* AGs. In the example,

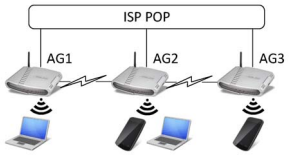

Fig. 1: Example of a scenario with three cooperating AGs running BOB

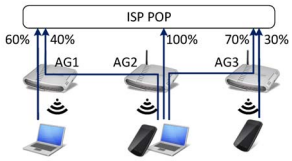

Fig. 2: Example of a bandwidth sharing scenario achieved by BOB

AG2 is connected to both AG1 and AG3. BOB is designed to exploit the unused ADSL bandwidth of the neighboring AGs to boost the performance of the local devices. Fig. 2 shows an example in which the local devices of AG1 and AG3 are currently exploiting 60% and 30% of their own ADSL upload bandwidth, respectively. Thanks to the cooperation of the two neighboring AGs, the local devices of AG2 exploit not only their local ADSL bandwidth, but also the unused ADSL bandwidth of AG2 and AG3 to/from the POP. Assuming all the ADSL link rates being the same, the overall uplink bandwidth gain would be 210%.

We designed the overall bandwidth aggregation system by considering the following constraints. *(1) Transparent performance*: the users' QoS should not be affected negatively by the sharing scheme. This means that the local user should be able to fully exploit his local ADSL bandwidth, independently from the neighbors' behavior. If some performance degradation is experienced, it must be negligible and not notable. *(2) Transparent deployment*: the system should not require any modification of the applications and of the Wi-Fi interface drivers at the users' devices, and should be compatible with the most common existing Internet transport protocols and with standard IP routing. *(3) Single wireless interface*: the AG should exploit a single Wi-Fi physical interface to connect both the local users and the neighboring AGs. *(4) Self-configuring scheme*: the cooperation scheme must be enabled in a distributed way without the need of a central control. All these constraints are desired by any ISP who wishes to scale the approach to a large population of users by providing a proprietary low-cost AG to its subscribers.

To meet all the previous design constraints, we combined many techniques. First, to achieve transparent performance, a traffic scheduler running in the AG regulates the traffic flows contending for the ADSL bandwidth. This guarantees that most of the ADSL bandwidth is devoted to the local devices, whereas a small bandwidth (negligible for the local user) is given to the neighboring AGs to avoid the starvation of active TCP flows. Sec. III-C will be devoted to describe the details of such traffic scheduler. Second, to

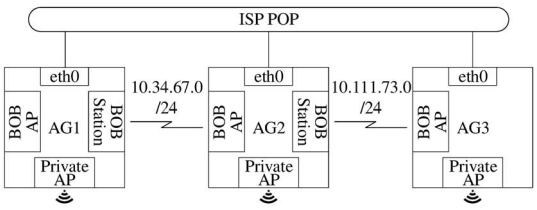

Fig. 3: Layer-2 topology highlighting the role of each kind of virtual interface.

achieve transparent deployment, the AG is entirely responsible to route the traffic flows. An internal flow-based load balancing scheme, described in Sec. III-B, route part of the incoming TCP/UDP flows to the neighboring AGs, seamlessly for the local devices and for the whole ISP network. Thus, neither hardware or software modification is required in the users' devices and the bandwidth sharing scheme can work with all the different 802.11 standards currently available. We wish to emphasize that such *transparent deployment* is the main difference of our approach with respect to the state of art [3], [4], [5]. The constraint about the single wireless interface poses some natural limitation regarding the scalability of the approach, since a common channel must be shared across all the cooperating AGs. Finally, to allow the cooperation among the AGs, we build a logical interconnection topology among the neighboring AGs, as detailed in the following Sec. III-A.

A. Communication topology

The topology connecting the AGs and their local devices requires multiple connections at MAC layer for each AG. Since the network interface is single, this configuration can be achieved by defining many virtual interfaces on the Wi-Fi physical interface. One virtual interface, referred as "*private AP*", is devoted to the local user and acts as a standard 802.11 AP establishing a BSS for the local devices. Another virtual interface, denoted as "*public BOB AP*", provides the connectivity to the neighboring AGs acting as access point. Finally, one or more virtual interfaces, referred as "*BOB station*", allows the AG to connect to the public BOB AP interfaces of other cooperating AGs. Each virtual interface is configured with a MAC address chosen automatically in increasing order with respect to the original MAC of the physical interface, to avoid conflicts at MAC layer. Fig. 3 shows an example of a topology achievable by the three virtual interfaces present in each AG.

The formation of the layer-two topology is achieved by a simple distributed algorithm, as follows. Each AG periodically scans for public BOB AP interfaces. Whenever it finds a new one, if not yet connected to it, it associates to it and establishes the bidirectional cooperation among the two AGs. To enable such process, each public BOB AP is identified by a BSSID composed by combining the unique identifier of the AG (obtained by the native public AP BSSID) and the string "BOB". This allows to understand whether one AG is already associated to another AG and to avoid double associations (as when a single AG acts as both public AP and station towards another AG).

In addition to the above association scheme, the cooperation between two AGs is actually established only if the RSSI is above a given threshold. This guarantees that only neighboring AGs cooperate when they have a reasonable good connectivity.

As final comment, note that this algorithm does not prevent the formation of layer-2 topologies with loops. This is not actually a problem, since the flow balancer can only route the traffic to the local ADSL connection or to the neighboring AGs. Whenever a neighboring AG receive this traffic, it sends the traffic directly to its own ADSL connection. In this way, a multi-hop wireless communication is never established, simplifying completely the routing problem and avoiding routing loops. After establishing the layer-two connectivity among the AGs, the public AP runs a DHCP server to provide the IP address to the BOB-station interfaces. To avoid conflicts of IP network addresses, the public AP is set in the range 10.X.Y.0/24, where X.Y are chosen according to a 16-bit hash function applied to the string of native AG BSSID.

B. Flow-level balancing

All the incoming traffic on the private AP interface is processed by a flow balancer, to eventually distribute the traffic across different neighboring AGs. The balancer works at transport layer and identifies the traffic based on the standard pair of transport ports and network addresses. When the first packet of a new flow reaches the balancer, a load balancing algorithm selects the local destination, that is either the local interface of the ADSL connection, or the BOB interfaces (BOB-station or BOB-AP). Note that, since all the traffic is routed through a NAT to reach Internet, the first packet of a flow is always generated by a local device. Furthermore, given that an unique public IP address is associated to each AG, all the following packets of the same flow need to be forwarded along the same path of the initial packet. Note that this choice also avoids out-of-sequence packets within a flow, which are very poorly tolerated by TCP/UDP protocols, and it is compatible with standard IP routing also for the backward path.

We have explored two possible flow-balancing schemes.

Weighted Round Robin (WRR) is the basic algorithm in which the number of flows sent on each BOB-interface and to the ADSL connection is proportional to the maximum bandwidth available in each interface. Such value can be estimated approximatively by the association data rate of each wireless interface and by the sync rate of the ADSL link. The AG must maintain an almost-static Interface Rate Table (IRT) with the updated data rates of each interface.

Pending Flow Balancing (PFB) is designed to distribute each new flow to the specific local interface (either BOB or ADSL) with the minimum number of pending active flows. A TCP flow is defined as "pending active" during the period between the initial SYN handshake and the final FIN handshake. An UDP flow is similarly defined after the initial packet and until a timeout expiration after the last observed packet. The AG must maintain a Pending Flow Table (PFT) that keeps track of the numbers of active flows on each interface.

If we assume equal flow sizes, WRR tends to balance the load across the BOB interfaces obliviously of the actual band-

width that different paths would experience, which depends on the local congestion in each neighboring AG. In a worst-case scenario, all elephant flows will be directed to lower bandwidth paths whereas mice flows to the higher bandwidth paths, with severe throughput degradation. Instead, PFB self-adapts the load on each path according to its actual available bandwidth, since it concentrates the flows on the paths with the higher throughput, on which the number of pending flows tends to decrease faster. Whenever a sudden reduction of the available bandwidth is experienced (due for traffic fluctuations and to the implemented traffic scheduler), the corresponding flows will start to experience some temporary starvation and the number of pending flows along the path will not decrease, thus raising the probability that new flows will be routed along alternative paths with better available bandwidth. Thus, PFB is also preferred for asymmetric links, as it balances the load according to the actual available bandwidth of each link. In Sec. V-B we will show an experimental comparison between the performance of the two algorithms, and discuss the performance for asymmetric links.

Note that a flow-level balancing could be inefficient in the case of very few concurrent flows. This is true in general, but in practice the number of active flows by a client is quite large, since many bandwidth-hungry applications open more than one flows, as described later in the evaluation part of the paper (see Sec. V). An alternative solution to optimize the routing would be to balance the traffic at packet level. In this case, the only option would be to implement a MPTCP [10] proxy within the AG, but this approach requires to install a reverse MPTCP proxy within the POP. Thus, this option is not transparent for the ISP and violates one of our design constraints.

C. Traffic Scheduling

A work-conserving traffic scheduler regulates the access to the uplink ADSL interface, and manage the set of output queues associated to such interface. This scheme is crucial to guarantee transparent performance for the local devices of an AG. The flows are scheduled at a fine granularity, i.e. at packet level, according to a Hierarchical Token Bucket (HTB) [11], [12]. HTB can be described by a multilayer tree, in which each node corresponds to a traffic class and in particular the root node corresponds to the main interface towards which all the traffic is sent. Each class is controlled by an internal token bucket and the multilayer topology is used to aggregate multiple classes (i.e. each node defines a new class aggregating all the children classes) and to specify detailed scheduling rules on such aggregation. More in details, each class is assigned with a pair of parameters $(r_{\min}, r_{\max})$, where $r_{\min}$ is the minimum average rate and $r_{\max}$ is the maximum rate that cannot be exceed by the traffic within the class. In a nutshell, when the corresponding queues are backlogged, the class traffic will receive at least $r_{\min}$ bandwidth, but no more than $r_{\max}$. The hierarchy allows one child class to increase its rate by borrowing the unused bandwidth from the ancestor class, providing high flexibility to set the minimum and maximum rates for single and groups of traffic classes.

Fig. 4 shows the hierarchy among the traffic classes defined in BOB. Consider a generic AG to which $k_1 + k_2$ neighboring AGs are associated; k_1 are associated as stations to the public BOB-AP interface, whereas k_2 are associated through multiple

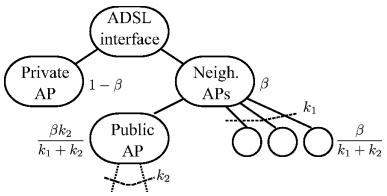

Fig. 4: Traffic classes in HTB scheduler with corresponding minimum rates $r_{\min}$, normalized to the ADSL link speed

BOB-station interfaces. We set $r_{\max} = 1$ (normalized to the ADSL link speed) for all the traffic classes, in order to exploit all the available bandwidth: the scheduler is indeed fully work-conserving. This choice guarantees also that the unused bandwidth of the local ADSL connection can be fully exploited by the neighboring nodes. In addition to this, we imposed that $1-\beta$ (with a small value of $\beta > 0$) fraction of the local ADSL bandwidth must be guaranteed to the local devices, whenever they are actively sending packets. A small β fraction of the ADSL bandwidth is instead devoted to guarantee that the flows from the neighboring nodes will not starve. This minimum bandwidth is divided evenly across all the $k_1 + k_2$ neighboring AGs. Note that this allocation provides a reasonable level of fairness among neighboring AGs that use different data rates to connect wirelessly to the local AG.

IV. Implementation

We implement BOB on Linux and test it on several laptops with several kernels, namely from 2.6.38 to 3.16. The wireless card we use is an Atheros AR9285 for which Linux provides a built-in driver (ath9k). We have also successfully imported BOB into Arduino YUN with RTL8188CUS wireless card with the default driver provided by Realtek. In the following sections, we describe some relevant design issues for the Linux implementation.

A. Communication Topology

We implement the topology formation using a Python script that uses the subprocess module to call external Linux commands. In particular, to create multiple virtual interfaces we used `iw` tool.

B. Flow-level Balancing

We implement the algorithm for the flow-level balancer in Python and run it as a daemon. It chooses the path for each flow, according to the WRR or the PFB scheme. When using the PFB scheme, we use the `conntrack` tool to get the information of the active pending flows.

The main implementation issue to address is how to route in a transparent way the packets according to the flow-balancer decision. To solve this, we adopt the sequence of operations described in Fig. 5. Whenever a packet is received by the AG through the local private AP interface, we have two cases. (1) If the packet is the first of a TCP/UDP flow, the flow-level balancer chooses the path (i.e. the local broadband connection or one of the neighboring AG) to route the packet. The packet is marked through an appropriate `iptables` rule based on the chosen path and this is stored in a marking table. The

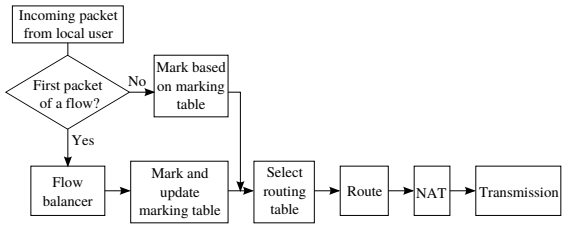

Fig. 5: Procedure followed by the flow-level balancer.

marking allows to use a specific routing table when exploiting a neighboring AG, according to which the default gateway has been set equal to the IP address of the BOB interface present in the neighboring AG. In addition to this, NAT modifies the source IP address to support the routing back on the reverse direction. (2) If the packet is not the first one of a flow, it is marked according to the marking table to choose the same path of the first packet of the flow, and thus the packet is processed by the desired routing table.

C. Traffic Scheduler

We implement the traffic scheduler using the native HTB scheduler available in the traffic control `tc` tool available in Linux. Referring to Fig. 4, we set $\beta = 0.01$ to reserve just 1% of the bandwidth for the flows arriving from the neighboring AGs. To be able to classify correctly the traffic entering the scheduler, we exploit the same marking capabilities described before to mark a packet based on the incoming interface.

V. Experimental Evaluation

We setup a testbed with 3 AGs, each of them running on a laptop. Two laptops are ASUS 1015BX and one is an HP EliteBook 8570w. Each AG is connected to an independent ADSL modem. We used a desktop PC with an Edimax USB wireless card (with RTL8188 chipset) as the client, which is associated to its home AG. We run all tests at night in order to minimize the interference from campus Wi-Fi and other networks. The evaluation mainly focus on: (i) evaluating the maximum gains achieved by our system with real applications, (ii) comparing the two algorithms for flow-level balancing described in Sec. III-B, (iii) and assessing the performance of HTB scheduler described in Sec. III-C.

A. Performance for Standard Cloud Storage Applications

We select Google Drive [13], OneDrive [14] and Dropbox [15], which are some of the most popular Cloud Storage services, as test applications. To evaluate the performance, we run several tests varying the number of activated AGs and the type of application, setting the rate of all the ADSL links to 2 Mbit/s. In each experiment, we upload 10 files through the application, with average file size of 50 MB. We capture the traffic at the ADSL interface of the local AG and of neighboring AGs, considering just the packets directed to the specific IP addresses adopted by the applications. Since all the considered applications adopt TCP as transport protocol, we were able to evaluate the upload throughput by considering the ACK and sequence numbers at TCP level.

TABLE I: Performance achieved by BOB during file upload of cloud storage services

Scenario	Throughput gain		
	GoogleDrive	OneDrive	Dropbox
2 AGs	1.98	1.99	1.00
3 AGs	2.96	2.97	1.00

TABLE II: Time in seconds to send 100 files with different load balancing schemes

Test	ADSL 1	ADSL 2	Minimum	PFB	RR
1	2 Mbit/s	2 Mbit/s	235.1	264.4	283.2
2	1.5 Mbit/s	2.5 Mbit/s	235.1	272.3	357.7

Table I shows the throughput gain, i.e. the ratio between the actual bandwidth obtained when BOB is enabled and the one when BOB is not enabled. The results for WRR and PFB flow-balancing were exactly the same. For GoogleDrive and OneDrive, BOB increases the throughput by a factor equal to the number of cooperative AGs. This large gain is due to the fact that GoogleDrive and OneDrive open multiple TCP connections [16] while uploading files and the flow-level balancer is able to exploit fully the available aggregate bandwidth. On the contrary, Dropbox opens only one TCP connection to upload files [17]. Since BOB balances the traffic on per-flow basis, it cannot exploit the available bandwidth at the neighboring AG. This highlights the main weakness of our flow-level balancer, which is effective only when the number of relevant data flows is larger than one.

B. Performance of Flow-level Balancing

We utilize two AGs to compare the performance of WRR and PFB algorithms described in Sec. III-B. In this test, the client sends 100 files to a server and we repeat the experiment 10 times. We set WRR weights to balance equally the traffic across the two possible paths. The size of each file is uniformly distributed between 250 kB and 2.5 MB, and the client opens one TCP connection per file. The client starts to send each file according to a Poisson process at rate 0.4 file/s. We exploited a rate limiter to vary the ADSL capacity of the two links to investigate the impact on the performance of the two load balancing schemes. In the first test, the two ADSL links have the same bandwidth, which is equal to 2 Mbit/s, while in the second test the rate of one link is set to 1.5 Mbit/s and the other one is set to 2.5 Mbit/s.

Table II shows the upload time to send the 100 files with WRR and PFB. The minimum value is calculated theoretically by dividing the aggregate file size by the sum of the two ADSL capacities and provides a loose lower bound to the upload time. In the first test, with two equal ADSL links, PFB and WRR perform almost the same. WRR assigns to each link the same number of files and the performance degradation with respect to PFB is due to the randomness of the file sizes, that do not allow a perfect balancing. On the contrary, in the second test, PFB greatly outperforms RR due to the ability to adapt to the available bandwidth, by considering the number of pending flows during the flow allocation. Since the available bandwidth of a real BOB system is expected to frequently change, PFB is the best choice to pursue in practice.

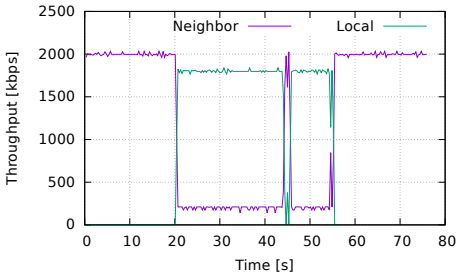

Fig. 6: The throughout for the local and neighboring user measured at the Local AG

C. Performance of the Traffic Scheduler

We test the behavior of HTB scheduler with 2 AGs, naming the first one local AG and the second one neighboring AG. We let one client to associate to each AG, referred as the local and the neighbor user, respectively. We set both ADSL capacities equal to 2 Mbit/s. We set β in HTB equal to 0.1, thus assuring 1.8 Mbit/s to the local user and 0.2 Mbit/s to the neighboring user. We let the neighbor user to continuously upload multiple files using OneDrive. As described in Sec. V-A, OneDrive opens multiple TCP connections and BOB is able to fully exploit the ADSL link of the local AG. After a while, we let the local user initiate a short upload with Dropbox.

Fig. 6 shows the throughput of the two users measured at the local AG. Initially, the local AG correctly shares 100% of its ADSL capacity with its neighbor. Upon starting the uploading, at time 20s, the local user is able to instantaneously get the allocated bandwidth of 1.8 Mbit/s, while the throughput of the neighboring user simultaneously drops to 200 kbit/s. After the local user finishes the upload, the neighbor fully regains the available bandwidth of the local AG. The result shows that the bandwidth is allocated as expected, i.e., according to the HTB settings. Besides this, it shows how fast HTB reacts when the local user starts to generate traffic. This clearly shows that the traffic scheduler is able to guarantee a transparent behavior for the local user, which is affected by the neighboring users by a small throughput degradation, that can be easily controlled by the β parameter.

VI. Conclusion

In this paper, we have proposed BOB, i.e. Beyond One's Bandwidth, a distributed system composed of cooperative Access Gateways, which can increase the Internet connection speed of residential users. Unlike the previous solutions, our system is totally transparent to the clients. We presented the basic architecture and the communication topology of the system. We designed a load balancer that distributes the traffic on flow level according to two schemes: Weighted Round Robin and Pending Flow Balancing. We also designed a work-conserving traffic scheduler that exploits the HTB algorithm and allows to control finely the small performance degradation due to the bandwidth sharing.

To validate our design, we have then implemented a prototype of BOB on Linux machines and tested the performance of our approach with real applications. Notably, we have shown that with some standard cloud storage services we can boost the performance and fully exploit the available bandwidth of the neighboring AGs. We have also shown the performance gain achievable by PFB flow-level scheduling when the available bandwidth is not evenly distributed across the cooperative AGs. Finally, we have shown the fast reactivity of the proposed scheme in preserving the local bandwidth to the local user of an AG.

The main weakness of our approach in terms of performance is due to the traffic balancer, which works flow-by-flow to be transparent for the ISP in terms of routing. We leave for future work the investigation of the effectiveness of such an approach in more practical scenarios in which many clients are simultaneously connected to the AGs controlled by BOB and each client is running a large set of heterogeneous applications. Intuitively, BOB efficiency should improve, but we need to prove it.

VII. Acknowledgments

This work has been partially supported by project EIT-ICT Lab "SDN at the Edges".

References

[1] EU Commission, "Commission staff working document on the implementation of national broadband plans," 2012, http://ec.europa.eu/information_society/newsroom/cf/itemdetail.cfm?item_id=7948.

[2] L. DiCioccio, R. Teixeira, and C. Rosenberg, "Measuring home networks with homenet profiler," in *PAM*. ACM, 2013.

[3] S. Kandula, K. C.-J. Lin, T. Badirkhanli, and D. Katabi, "FatVAP: Aggregating AP backhaul capacity to maximize throughput." in *NSDI*, 2008.

[4] X. Xing, S. Mishra, and X. Liu, "ARBOR: hang together rather than hang separately in 802.11 wifi networks," in *Proc.of IEEE INFOCOM*, 2010.

[5] D. Giustiniano, E. Goma, A. Lopez Toledo, I. Dangerfield, J. Morillo, and P. Rodriguez, "Fair WLAN backhaul aggregation," in *Proc. of ACM MobiCom*, 2010.

[6] E. Chai and K. G. Shin, "Sidekick: AP aggregation over partially overlapping channels," in *IEEE ICNP*, 2011.

[7] S. Jakubczak, D. G. Andersen, M. Kaminsky, K. Papagiannaki, and S. Seshan, "Link-alike: using wireless to share network resources in a neighborhood," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 12, no. 4, pp. 1–14, 2009.

[8] E. Goma and D. Giustiniano, "SmartAP: Practical WLAN backhaul aggregation," in *Wireless Days (WD), 2013 IFIP*. IEEE, 2013.

[9] C. Rossi, N. Vallina-Rodriguez, V. Erramilli, Y. Grunenberger, L. Gyarmati, N. Laoutaris, R. Stanojevic, K. Papagiannaki, and P. Rodriguez, "3GOL: Power-boosting ADSL using 3G OnLoading," in *Proc. of ACM CoNEXT*, 2013.

[10] H. Han, S. Shakkottai, C. V. Hollot, R. Srikant, and D. Towsley, "Multi-path tcp: a joint congestion control and routing scheme to exploit path diversity in the internet," vol. 14, no. 6, pp. 1260–1271, 2006.

[11] "Hierarchical Token Bucket theory," http://luxik.cdi.cz/~devik/qos/htb/manual/theory.htm, 2015.

[12] R. Garroppo, S. Giordano, S. Lucetti, and E. Valori, "The wireless hierarchical token bucket: a channel aware scheduler for 802.11 networks," in *IEEE WoWMoM*, June 2005.

[13] "Google drive," https://www.google.com/drive/, 2015.

[14] "Onedrive," https://onedrive.live.com/, 2015.

[15] "Dropbox," https://www.dropbox.com/, 2015.

[16] I. Drago, E. Bocchi, M. Mellia, H. Slatman, and A. Pras, "Benchmarking personal cloud storage," in *Proc.of ACM IMC*, 2013.

[17] I. Drago, M. Mellia, M. M Munafo, A. Sperotto, R. Sadre, and A. Pras, "Inside dropbox: understanding personal cloud storage services," in *Proc. of ACM IMC*, 2012.