Personalizing the Home Network Experience using Cloud-Based SDN

Hassan Habibi Gharakheili[†], Jacob Bass[∗], Luke Exton[†], Vijay Sivaraman[†]
† School of Electrical Engineering & Telecommunications, UNSW, Sydney, Australia.
∗ Newtworks, Sydney, Australia.
Emails: {*h.habibi@unsw.edu.au, jacob@newtworks.net.au, l.exton@student.unsw.edu.au, vijay@unsw.edu.au*}

***Abstract*—Home networks are becoming increasingly rich in devices and applications, but continue to share the broadband link in a neutral way. We believe the time is ripe to personalize the home network experience, allowing a household to differentiate its users (e.g. father's laptop prioritized over kid's iPad) and services (e.g. video streaming prioritized over downloading). In this paper we argue that SDN provides a way to automate self-customization by households, while cloud-based delivery simplifies subscriber management. We develop an architecture comprising a cloud-based front-end portal and SDN-based back-end APIs, and show how these can be used by the subscriber to improve streaming-video (YouTube) quality and video conferencing (Skype) experience, and to permit device-specific parental controls (e.g. Facebook access). We prototype and validate our solution in a platform comprising the Floodlight controller and OVS switches. Lastly, we evaluate our solutions via experiments of realistic scenarios to quantify the benefits in terms of improved quality of experience and new features for the user.**

I. Introduction

The average number of Internet-capable household devices is steadily rising [1] – a typical house today has desktop PCs, laptops, tablets, smart-phones, and smart TVs, with IoT appliances slated to add to the deluge in the near future. These devices (and the services accessed via them) are of varying utility to a user – for example, a video streaming session may be more critical than a software update. Further, households are composite entities of multiple users, each valuing services in a different way – for example, the father may value his work-related teleconference session more than his kid's video streaming session. The complex requirements in the house demand customization of how the broadband link is shared by the users, devices, and services in the home, but unfortunately no practical solution exists today, for reasons outlined next.

While some home gateways come equipped with features that allow user customization (e.g. prioritizing voice or blocking selected web-pages), they do not provide a scalable solution for several reasons: (a) they require higher level of user motivation and sophistication, posing a significant barrier to uptake, (b) they vary across vendors, making it difficult to come up with uniform solutions, and (c) they do not address the problem at the bottleneck, which is the access link coming into the house. Indeed, the ISP, who controls the access link is best positioned to provide customization for the subscriber. However, this has not happened to-date due to significant challenges: (a) managed services typically require (expensive) manual configuration, which is difficult to justify given the low margins in residential broadband, (b) traffic discrimination can impinge on "network neutrality", which remains a contentious issue, and (c) ISPs visibility into the home network is typically blocked by the home gateway (due to NAT).

We believe that the emergence of SDN technology, coupled with growth of cloud-based services, makes it timely to consider new solutions to the above problem. An ISP can use SDN to let users customize their service themselves, eliminating costs associated with manual network provisioning. The capability can be exposed to users through cloud-based portals, eliminating per-subscriber software management overheads. Net neutrality can be left to the individual subscriber to decide for their own household, rather than being imposed across the board. As we will show in this paper, users can get value from customization, while ISPs can offer it as a self-managed service at little extra cost. Our specific contributions in this paper are:

- We develop an architecture, comprising a cloud-based front-end user interface, and SDN-based APIs in the back-end, by which an ISP can allow users to customize their home network experience,
- We demonstrate YouTube streaming video quality, Skype video-conferencing experience, and Facebook parental control access, as innovative use-cases wherein subscribers benefit from customization, and
- We prototype our system using the Floodlight SDN controller and OVS switches, and evaluate the performance benefits in realistic scenarios.

The rest of the paper is organized as follows: Relevant prior work is summarized in §II. Section §III explains the use-cases considered in this paper, and §IV gives our system architecture and trade-offs. In §V we describe our prototype implementation including the GUI, the APIs, and traffic models. Section §VI describes our experimental results, and the paper concludes in §VII.

II. Related Work

Improving the service quality (QoS/QoE) has extensively been explored by industry and academia over past two decades, with limited success in practice. However, the use of SDN in addressing this problem holds promise [2]–[5], and home networking has been gaining increasing attention in the research community [6]–[12].

[2] and [4] describe QoS control frameworks which allow multiple applications to automatically interact with the network and to set the low-level quality related configurations using a set of programmable interfaces. There is an increasing number

of works focusing on QoE-driven Internet video delivery [3], proposing models and metrics towards enhanced user experience. The architecture we proposed in [5] suggests that quality control be exposed by ISPs to the content provider via open APIs. Using an automated per-flow negotiation can also allow the ISP to monetize the network infrastructure.

In [7], the home broadband access network is virtualized by the ISP in a way that multiple content providers such as video provider and smart grid utility, operate on isolated and independent controlled slices of network capacity. Similar to [13], HomeVisor [11] offers an SDN-based management tool for home network enabling remote administration and troubleshooting via a high-level network policies. [8] presents an intuitive interaction of user with underlying network to have control over home network to support quality of different applications. Improving home user experience using a dynamic traffic prioritization is studied in [10]. This work attempts to actively identify traffic flow of user interest by monitoring the application window in focus and then signaling the home router to serve it with a higher priority.

Works such as [6] and [9] address network security and troubleshooting by leveraging an off-site and third-party operated controller so that home users need not have sophisticated technical skills. In our earlier work [12], we developed a client hosted application with a simple and easy-to-use GUI by which the user can prioritise their devices and applications. The GUI then signals the ISP network via open APIs to configure the network accordingly. The present work differs in moving the application to the cloud to enable easier management, and demonstrates new capability not shown before.

III. Use Cases

The set of opportunities that a home user can benefit from customized network service is large and diverse: bandwidth assurance for streaming videos, low latency and low loss performance for gaming and voice applications, securing the private health-related or banking transactions, and so on. In this paper we start with three application use-cases: *video streaming*, chosen due to its large and steady growth of volume, real-time *teleconferencing*, chosen for its wide adoption by users, and *parental control*, chosen for its high value to users. The northbound APIs we develop and demonstrate for these use-cases will help illustrate the value of our approach, and can be extended in future work for other application types.

A. Streaming video

Streaming video, driven by providers such as NetFlix and YouTube, represents an increasingly large share of downstream Internet traffic. However, maintaining quality of experience in online video viewing over today's best-effort networks remains a challenge. Content providers routinely use techniques such as playback buffers, content caching and adaptive coding to improve delivery quality and to satisfy users. Multiple concurrent use of an access link can cause variable bandwidth available to any network application. Studies have shown this results in service degradation such as start-up delays and rebuffering events which ultimately lowers the user engagement and retention. A better technical solution would be for the network to explicitly assure the quality needs of such applications by slicing the access link. YouTube demands a minimum downland bandwidth of 500Kbps for a modest quality 480p video delivery while 1Mbps is recommended. This number increases to a level of about 4Mbps for HD 1080p streaming. Such assurance requires to be provisioned by ISP for downstream direction. Thus, as mentioned earlier this entails user's involvement to signal the ISP via proper API. Explicit control exposure would increase user satisfaction and ISP monetization opportunities.

B. Videoconferencing

As workplaces become more decentralised, a capable system for communicating is vital. There is no doubt that Internet has enormously changed our lifestyle specially in daily communication. Video teleconferencing has increasingly become popular means of communication besides emails and short text messages. Skype and Google Hangouts are arguably the most popular video chat service applications among home users. However, compared to one-way video streaming a successful video call requires more deterministic network service and quality assurance such as guaranteed bandwidth and less contention. To sustain a reasonable quality of video communication, Skype recommends upper limits of 0.2%, 200ms and 10ms for packet loss, latency and jitter respectively. A typical Skype video call consumes 128Kbps of bandwidth downstream and 1.2Mbps for HD quality. Skype application monitors the network condition and automatically adjusts the encoding and video quality. A "sliced" network controlled by end-user would assure a certain bandwidth and secure the Skype flow from contention over the network.

C. Parental Control

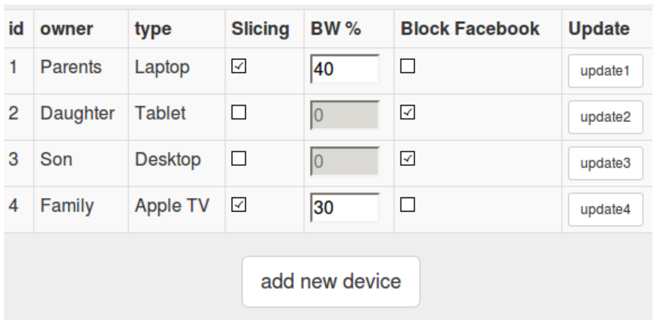
There is growing concern amongst parents, communities, and governments about the content kids are being exposed to on the Internet. A one-size-fits-all approach does not work, since different families may have different needs based on their circumstances (age of kids), values, and priorities. For example, parents may want to apply dynamic and customized policy for web access while kids are studying at home, or be more restrictive in the elementary years (e.g. not allowing use of social networking) and lifting the restrictions as they get older. There are currently parental control applications and tools that are installed on end-host device or home routers. The former solutions are often easily bypassed by even kids. Where the latter filters inappropriate sites on all the household's Internet-aware devices and it doesn't offer a dynamic and detailed control that a software solution can. Therefore, a fine grained, dynamic, and customizable control is needed, that requires visibility at a per-flow level.

IV. System Architecture

In this section, the high level architecture of the solution is described. We propose a system architecture of user-driven and QoE-aware in the home network. We first outline the major architectural choices and trade-offs (§IV-A), then describe the operational scenario (§IV-B), and the APIs (§IV-C).

A. Architectural Choices and Trade-Offs

The overall architecture is comprised of two main parts, the front-end user portal hosted on the cloud and the back-end software-defined controller hosted in the ISP network. The user-facing layer communicates with ISP network via open APIs of our devising, or via APIs already implemented by our choice of system components. This will allow us to



id	owner	type	Slicing	BW %	Block Facebook	Update
1	Parents	Laptop	☑	40	☐	update1
2	Daughter	Tablet	☐	0	☑	update2
3	Son	Desktop	☐	0	☑	update3
4	Family	Apple TV	☑	30	☐	update4

Fig. 1. User Interface of network management

add features or change functionality easily without requiring complete overhauls. The structure of our designed API calls follows the JSON-RPC specification facilitating simple design and easy debugging. Note that all of home connected devices are known by ISP network through user input via cloud-based portal as depicted in Fig. 1. However, this requires the user's technical expertise, in the future this can be resolved via automatic discovery of devices in the future. In order for this model to function, the ISP requires direct visibility of the devices inside the household which necessitates the home gateway operate over layer 2 with no NAT functionality. There is an alternative network architecture called "Virtual Home Gateway", where some of the current home gateway functions is pulled into the operator network, extending the home network next to the ISP infrastructure. In this paper we focus on customizing service delivery only over the broadband access link connecting the ISP local exchange and the home gateway, rather than an end-to-end approach, since the latter requires coordination across network domains that is deemed a long-term goal; moreover, network bottlenecks often lie in the access and not at the interconnects between networks.

B. *Operational Scenario*

Holistically, the system employs a typical call and response mechanism between the home super-user and the ISP network. A motivating example of the system's functionality is as follows. The user high-level demand such as resource allocation and turning on/off a restriction feature, for specific device is obtained via a simple graphical interface, translated into low-level semantics and then communicated to network controller via ISP exposed open APIs. For example, a user desires better performance for a streaming video application during movie viewing on the internet TV. He/She logins the ISP provided web portal and specifies the minimum portion of link capacity to assure seamless playback of video streaming. The controller thereupon reconfigures the switching hardware associated with the access link of that user's home by applying the requested policy, reporting back to the user interface, storing the successful change in its local database and maintaining the state of home network.

C. *The APIs*

There are two main APIs necessary for the architecture described here to function. The first allows the home user to specify which of connected devices need to be dynamically controlled by the ISP network. The second exposes the cloud-based customised configuration. As mentioned earlier, each API call is in the form specified by the JSON-RPC schema comprising: (a) *Method*: the type of API call , (b) *Parameters*: the specific instructions or requests of the API call , and (c) *Id*: a debugging identifier to assist in troubleshooting through log inspection. In what follows we describe the chosen "parameter" for each API in more detail.

1) Add-device: In order to provide a customized service on a per-device basis, the user specifies in the web portal which devices inside the home network are nominated for a preferred policy. This request does not make any changes to the network state. This API call is identified by the method: "add" and a single parameter is passed through, the MAC address of the device being added. Upon arrival of this API call, a separate queue is created on the switch prior to any bandwidth allocation being requested.

2) Net-Policy: Once a device is added into the portal then the user specifies his/her preference, choosing customized services offered by the ISP, otherwise the device will be served like an ordinary client on the network with no special treatment. The method portion of this API is set by "policy" and the parameters passed are: (a) an identifier for the device being updated, in this case the device MAC address, (b) the new "state" the device is requesting, and (c) the minimum fraction of total link capacity to be allocated to the device in percentage. The state parameter is a set of binary flags corresponding to different functionalities such as bandwidth assurance and parental control. Turning on/off each option will toggle the flag in the request, updating the state of the device. In the case of bandwidth assurance, Net-Policy API call changes the minimum assured bandwidth of the device to the percent specified in the call, activates the respective queue on the switch, and sets the flow table accordingly. In the case of the parental control web-site blocker, it will create static flows that drop all traffic originating from the pre-defined range of IP addresses destined to the related device.

V. Prototype Implementation

Our cloud-based customization tool is developed on a test-bed emulating a single home. In this prototype we assume SDN is already implemented by the ISP network and we require no modification of user equipment. This tool is made up of two parts, a "Home Agent" that communicates home user requests to the ISP, and an "ISP Agent" that receives these requests and re-configures the ISP network to provision the requested functionality.

A. *Network Topology*

We now briefly describe the topology of our experimental setup shown in Fig. 2.

1) Inside the home: Several computers representing home users are connected to an access point over Ethernet and Wi-Fi. This access point acts as a network gateway to an emulated WAN link from the ISP. These devices are used to emulate home user traffic. Home users communicate their customized service requirements over the Home Agent, which runs on a remote web-server.

2) The ISP Network: The ISP access link is provided by an Openflow switch running Open vSwitch (OVS) [14], connected directly to the home gateway. This access link is rate-limited to 5Mbps, representing an average residential broadband link capacity. This switch is controlled by an instance of the Floodlight controller, running locally inside the ISP network.

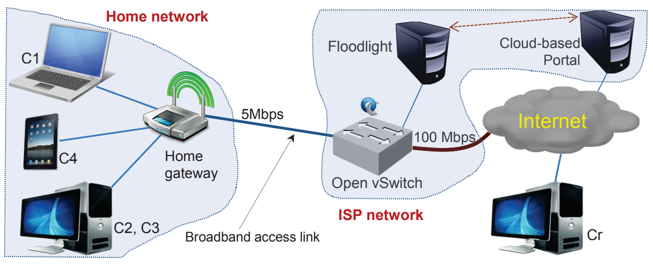


Fig. 2. Prototype architecture

The Openflow switch is a DELL PowerEdge R620 server with 8 Gigabit Ethernet ports, running the `Open vSwitch 1.10.0`. OVS does not implement QoS itself. Instead, it can configure some, but not all, of the QoS features built into the Linux kernel such as Hierarchical Token Buckets (HTBs) that assure minimum bandwidth for certain queues, providing the slicing function. This enables ISP to slice the broadband access link to provision a minimum guaranteed bandwidth on demand. Additionally, the API exposed by OVS empowers user to set and unset configuration options, allowing the switch to be controlled by a separate application, as opposed to patching the program.

The Floodlight controller [15] runs as an instance on a standard desktop computer running Linux. This controller was fundamentally chosen for its ability to set/unset static flow rules that can be controlled using a REST API. This allows us to place traffic into queues, slicing the downstream traffic. This also allows us to design our tool without needing to patch and maintain our version of the controller. Note that the ISP network is configured by the ISP Agent upon a request from the home.

B. Our Tool

1) ISP Agent: The ISP agent runs as a java program on an internal ISP server, communicating over HTTP with the Home agent, enabling and disabling the services the ISP develops, deploys and offers as a cloud-based solution. Upon receiving a legal request from the household, it changes the settings on relevant Openflow controller and switch.

2) Home Agent: The Home agent is the web portal managed by the ISP, used to expose the APIs to the user allowing them to manage their network service. It runs as a standard HTML web-site, and is served by an off-site web-server run by the ISP. The portal allows the user to add devices to the policy list, to allocate certain amount of bandwidth for devices and to request additional services offered by the ISP. At present, the service also has the ability to block access to a specific web-site, `www.facebook.com`, from any specified device on the network. Fig. 1 shows the home page the portal uses to communicate the current state of the network. For example, this screen shows four household devices and the super-user (say the father) who operates this control panel can allocate a bandwidth fraction, in percent, for each device and enact restrictive parental controls regarding Facebook access for the children's devices. 70% of total available bandwidth is sliced for the Apple TV (shared by the household members) and work-related laptop (belonging to the parents) and the remaining 30% is left for best-effort service to other devices. At the same time, the parental control feature is activated for daughter's tablet and son's desktop.

VI. Experimental Evaluation

To evaluate the efficacy of our tool, we conducted three suites of experiment measuring application performance perceived by user under the same condition of network background load. We present results from test of Skype video call, Pytomo tool (open-source Youtube crawler and analyzer), and also MOS measurement for an HD video (1080p) stream. For each of test suites, five different network conditions are experimented: **(a) No load**: Only the traffic of test-suite is running on the network with no background load. **(b) Light load**: Parallel to the test suite traffic, a single background download is destined to another device (C2 in Fig. 2). **(c) Medium load**: Parallel to the test suite traffic, multiple downloads are destined to another device. **(d) Heavy load**: Parallel to the test suite traffic, multiple multi-threaded download operations are active on other two devices (C2 and C3 in Fig. 2). **(e) Heavy load + Slicing**: "Heavy load" state with the slicing tool (quality assurance) is enabled by home user.

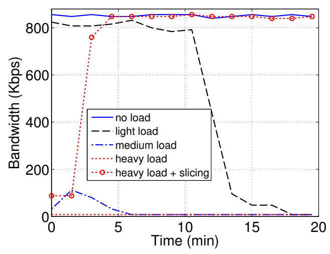
Except the last state mentioned above, access network serve all traffic flows by single default queue in a best-effort fashion. In our experiments, to emulate each download, a 50MB ISO file is transferred from an Internet-hosted FTP server. In single-threaded downloads the `wget` tool transfers the remote file and the IDM download manager is used for multi-threaded case.

A. Skype Quality Test

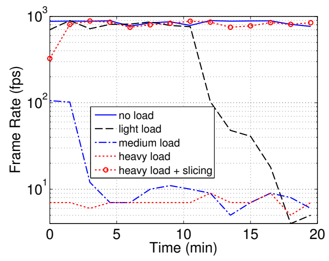
In this test we study the effect of user control on the quality of Skype video call. A Skype video session is established between the home client C1 and a remote client Cr on the internet (Fig. 2). To evaluate a consistent and repeatable video call experiment, we inject a 10 second long high quality video sequence into Skype at remote client Cr via a software-emulated webcam, looping the same video sequence. The chosen video is a standard news broadcast sequence ("Akiyo" by Joint Video Team) with little motion of head, mouth and shoulder representing a typical video call. We run this experiment for about 20 minutes for each of network conditions in order to capture the stationary state of Skype.

To measure the call performance, we rely on real-time "Call Technical Info" reported in Skype user interface. The home client C1 tracks the performance of the Skype session by recording the screen. In this way, we capture technical metrics such as download/upload bandwidth, frame rate, and Round-trip Time as well as visually perceived quality of video.

1) Available bandwidth: Fig. 3(a) shows that Skype application consumes 851 kbps of average bandwidth for a delivery of video at reasonable quality while no background load is present on the network (solid curve). As the network load grows the available bandwidth for Skype video call starts falling drastically (dashed curves), receiving small amount of bandwidth less than 1 kbps under heavily congested network situation. However, once the home user signals the ISP (at time 1.5m) for a preferred slicing of broadband access link, the allocated bandwidth to C1 (Skype traffic flow) shortly ramps up to the almost same level of "no load" case (dashed circle curve). As soon as the user request arrives to the network controller, the desired slice is created and made available accordingly. However, there is a gap of about 1.5 minutes for the user demand to be executed. That is because of negotiation



(a) Bandwidth (download)



(b) Frame rate

Fig. 3. Skype performance metrics: (a) Bandwidth, (b) Frame rate

No load	Light load	Medium load	Heavy load	Heavy load + Slicing
3 ms	86 ms	686 ms	1331 ms	2 ms

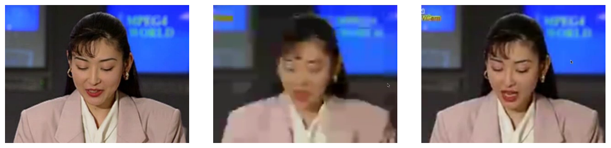
TABLE I. SKYPE PERFORMANCE: AVERAGE ROUND-TRIP TIME

signals cross the home Internet link which is already heavily congested by download traffic.

2) Video frame rate: Skype naturally adapts its sending frame rate to packet loss, packet delay and also available bandwidth [16]. Fig. 3(b) shows the automatic quality compensation performed by the Skype application if the available bandwidth decreases. When the network is under-utilized a perfect average frame rate of 852 is realized and this number drops by 45% when a light load starts contention and causes 470 frames to be received per second at client C1. Under an increasing network load, the video call quality significantly decreases that instant frame rate of below 20 is often observed (the two lower dashed curves). Moreover, a profound reduction of average frame per second (fps) is achieved in medium and heavy load conditions (22 and 7 respectively). However, use of slicing tool provisions the required bandwidth to the Skype call raising the average fps to acceptable level of 787. Needless to say, this QoE improvement comes at the cost of slowing down the other unimportant downloads.

3) Latency: Round-trip time (RTT) is an important metric for interactive real-time applications such as tele-conferencing and reflects a call responsiveness. RTT varies by several factors in the network such as buffering, and available bandwidth. Due to RTT rapid variations captured during the course of experiments, we report the average value of this metric. Table I shows that the more the network is congested, the less bandwidth is available, and the higher average RTT (as high as 1330ms for heavy load) is observed which directly translates into poorer quality. However, the high QoE (i.e. short average RTT of 1.857 ms) is sustained when the user configures the network with a guaranteed bandwidth serving C1.

4) Visual Quality: A decrease in network service quality causes the call to continue with a lower codec and nearly identical responsiveness. To demonstrate the effect of the bandwidth deficit, the received video frames at almost similar time of different experiments are presented side by side. Load is added from left to right while slicing activated in Fig. 4 (for space sake, all the screen shots are not illustrated). As load increases, the artifacts and errors increase that eventually makes the video unwatchable in Fig. 4(b). This shows how drastically Skype does reduce the codec quality to cope with network congestion, and how effectively our slicing tool can instead provide the necessary bandwidth to prevent this fall.



(a) (b) (c)

Fig. 4. Skype visual snapshot: (a) No load, (b) Heavy load, (c) Heavy load + Slicing

B. Pytomo Test for video streaming

Pytomo [17] is a python based tomography tool emulating user behaviour in video streaming, and performs analysis of several metrics pertaining to user experience. Pytomo fetches the most popular videos in youtube over HTTP and measures the quality metrics while loading and playing them. We used this tool to measure the performance of the most popular video service under different network situations, and then to examine the effect of our slicing tool on user QoE. The tests were each run for approximately 60 minutes, during which as many videos as possible were accessed (Increased network congestion makes less bandwidth available to the streaming service. This causes the video to stall or rebuffer and results in fewer videos played within the given time frame). In this test suite, we examine the effect of network load on delivery-related quality metrics such as video startup delay, buffering, initial bitrate and number of playback interruptions which altogether affect the user engagement and quality of experience [3]. The results of this experiment presented in Table II, are the average value of the metrics taken over all videos accessed.

Startup delay is a measure of video service responsiveness indicating the amount of time taken for a particular video to start playing at client after user initiates a request. Under no load condition the average latency is 4.6ms. However, with multiple downloads present on the network, the average startup delay grows by two orders of magnitude, before the profound reduction to 0.646ms is observed while home user specifies a desired slicing.

Buffering also frustrates viewers [18] and increases the rate of abandonment. In our Pytomo test, no buffering was experienced by any videos played when the slicing tool allocates a minimum bandwidth for the device streaming the video, allowing it to perform the same as in the no load situation. However heavy contentions raised by multiple concurrent flows, causes an average 1s and standard deviation 2.8s of buffering duration per video while some videos experience about 17s of buffering.

Initial bitrate gives the mean data rate during the initial buffering period of a viewing session. A higher initial rate leads to a lower or to no buferring period. If this rate is low, the video will initially buffer for longer, and will likely need to rebuffer throughout. Pytomo streams receives as high as 4955 Kbps on average (nearly full link capacity) while no other traffic flow crossing the network. As heavy load is introduced to the network and the available bandwidth for video traffic is reduced, the initial rate can only get 1149 Kbps, dropping by 75% compared to the no load network state. Once slicing is enabled while the heavy load is still present, the initial rate climbs back 3924 Kbps close to the original level.

Metric	No load	Heavy load	Heavy load + Slicing
Average playback duration (s)	203.7	128.9	172.1
Average startup delay (ms)	4.6	511	0.646
Average buffering duration (s)	0	0.994	0
Average initial bitrate (kbps)	4955	1149	3924
Number of interruptions	0	11	0

TABLE II. PYTOMO EXPERIMENT RESULTS

Load	Mean	Standard deviation
No load	3.310	0.000
Light load	2.660	0.393
Medium load	2.750	0.450
Heavy load	2.500	0.043
Heavy load + Slicing	3.310	0.000

TABLE III. MOS EXPERIMENT RESULTS

Interruptions: a poor connectivity will causes the video stream to be rebuffered, interrupting the viewing and decrease user satisfaction. As depicted in the last row of Table II, under no contentious load, there is no interruption, but once a heavy load is offered, 11 interruptions are seen by the streaming flow. With the user assuring a minimum bandwidth for his device, the number of interruptions returns to 0.

C. MOS Test for Video Streaming

In this test, we developed a Powershell script to emulate home users behavior online and to automatically evaluate the video streaming quality by traditional metric of Mean Opinion Score (MOS). This script streams an HD video of 1080p resolution from Youtube with random duration and then goes to idle state. The test suite is run for an hour with the script collecting data from at least 6 viewing periods, and calculating the MOS [19]. The computed mean and standard deviation of MOS value are presented in Table III. It is clear that, prior to any additional network load the video has a perfect average MOS of 3.31 with a standard deviation of 0. The MOS value gradually drops while more loads are introduced to the network showing a poor QoE when downloads interfere with video streaming. The last row demonstrates how the user can benefit from a cloud-based slicing tool and controlling the quality to realize similar performance as to when there is no network load.

D. Parental Control

To demonstrate this feature, we used the publicly available block of IP addresses provided by Facebook to populate a blacklist. This blacklist is then used to enable or disable static flows instructing the switch to drop the associated flows. Once enabled, the blocking takes place immediately, causing any packet from an address on the blacklist destined to the subjected device to be dropped, resulting in a timeout. This feature can also be extended to function based on time of day.

VII. CONCLUSIONS

As more devices are connected and new applications emerge within households, service customization is becoming more important for the users of such complex home network. This paper is an attempt to encourage ISPs to tap into the service tailoring dimension for user control of quality discrimination and new monetization opportunities. We have shown that it is possible to provide the subscribers with a simple and self-service portal by which the users can manage their Internet experience and differentiate their devices and applications. The cloud-based GUI communicates the user high-level preferences to the ISP back-end agent and network controller via SDN-based APIs, which can then dynamically adjust the access link slices to maintain the user's expectations of quality. We prototyped our system by building a standard cloud-based GUI, implementing the model APIs using Open-vSwitch and FloodLight controller, and writing client scripts that generate real user traffic. Our experiments show that use of our tool lets users improve their real-time video conferencing and video-streaming performance by stretching the non-time-critical traffic, and also restrict social networking access for children's devices. We believe our tool is a step towards more sophisticated and comprehensive home network management tools that include features for QoE, security, and much more.

REFERENCES

[1] A. Sabia and F. Elizalde, " Market Trends: Home Networks Will Drive Success of the Connected Home," Gartner, Report, Mar. 2013.

[2] W. Kim, P. Sharma, J. Lee, S. Banerjee, J. Tourrilhes, S.-J. Lee, and P. Yalagandula, "Automated and scalable QoS control for network convergence," in *Proc. USENIX INM/WREN*, College Park, MD, USA, Apr 2010.

[3] A. Balachandran, V. Sekar, A. Akella, S. Seshan, I. Stoica, and H. Zhang, "Developing a predictive model of quality of experience for internet video," in *Proc. ACM SIGCOMM*, Hong Kong, China, Aug 2013.

[4] A. Ferguson, A. Guha, J. Place, R. Fonseca, and S. Krishnamurthi, "Participatory Networking: An API for Application Control of SDNs," in *Proc. ACM SIGCOMM*, Hong Kong, China, Aug 2013.

[5] V. Sivaraman, T. Moors, H. Habibi Gharakheili, D. Ong, J. Matthews, and C. Russell, "Virtualizing the access network via open APIs," in *Proc. ACM CoNEXT*, Santa Barbara, CA, USA, Dec. 2013.

[6] N. Feamster, "Outsourcing Home Network Security," in *Proc. ACM SIGCOMM HomeNets*, New Delhi, India, Sep. 2010.

[7] Y. Yiakoumis, K.-K. Yap, S. Katti, G. Parulkar, and N. McKeown, "Slicing home networks," in *Proc. of ACM SIGCOMM workshop on HomeNets*, Toronto, ON, Canada, Aug. 2011.

[8] Y. Yiakoumis, S. Katti, T.-Y. Huang, N. McKeown, K.-K. Yap, and R. Johari, "Putting home users in charge of their network," in *Proc. ACM UbiComp*, New York, NY, USA, Sep. 2012.

[9] K. L. Calvert, W. K. Edwards, N. Feamster, R. E. Grinter, Y. Deng, and X. Zhou, "Instrumenting Home Networks," *SIGCOMM CCR*, vol. 41, no. 1, pp. 84–89, Jan. 2011.

[10] J. Martin and N. Feamster, "User-driven dynamic traffic prioritization for home networks," in *Proc. ACM SIGCOMM W-MUST*, Helsinki, Finland, Aug 2012.

[11] T. Fratczak, M. Broadbent, P. Georgopoulos, and N. Race, "Homevisor: Adapting home network environments," in *Proc. EWSDN*, Berlin, Germany, Oct 2013.

[12] H. Kumar, H. Habibi Gharakheili, and V. Sivaraman, "User control of quality of experience in home networks using SDN," in *Proc. IEEE ANTS*, Dec. 2013.

[13] H. Kim and N. Feamster, "Improving network management with software defined networking," *Communications Magazine, IEEE*, vol. 51, no. 2, pp. 114–119, 2013.

[14] Nicira Networks, "Open vSwitch," http://openvswitch.org/.

[15] Big Switch Networks, "Project Floodlight," http://www.projectfloodlight.org/.

[16] X. Zhang, Y. Xu, H. Hu, Y. Liu, Z. Guo, and Y. Wang, "Profiling skype video calls: Rate control and video quality," in *Proc. IEEE INFOCOM*, Orlando, Florida, USA, March 2012.

[17] Louis Plissonneau et al., "Project Pytomo," http://code.google.com/p/pytomo/.

[18] S. Krishnan and R. Sitaraman, "Video Stream Quality Impacts Viewer Behavior: Inferring Causality Using Quasi-Experimental Designs," in *Proc. ACM IMC*, Boston, MA, USA, Nov. 2012.

[19] R. K. Mok, E. W. Chan, X. Luo, and R. K. Chang, "Inferring the QoE of HTTP video streaming from user-viewing activities," in *Proc. ACM SIGCOMM W-MUST*, Toronto, ON, Canada, Aug. 2011.