Third-Party Customization of Residential Internet Sharing using SDN

Hassan Habibi Gharakheili, Luke Exton, Vijay Sivaraman
University of New South Wales, Sydney, Australia
Emails: {h.habibi@, l.exton@student., vijay@}unsw.edu.au

John Matthews, Craig Russell
CSIRO, Sydney, Australia
Emails: {john.matthews, craig.russell}@csiro.au

***Abstract*—Today's residential Internet service is bundled and shared by a multiplicity of household devices and members, causing several performance problems. Customizing broadband sharing to the needs and usage patterns of each individual house has hitherto been difficult for ISPs and home router vendors. In this paper we design, implement, and evaluate a system that allows a third-party to create new services by which subscribers can easily customize Internet sharing within their household. Our specific contributions are three-fold: (1) We develop an over-the-top architecture that enables residential Internet customization, and propose new APIs to facilitate service innovation. (2) We identify several use-cases where subscribers benefit from the customization, including: prioritizing quality-of-experience amongst family members; monitoring individual usage volumes in relation to the household quota; and filtering age-appropriate content for selected users. (3) We develop a fully-functional prototype of our system leveraging open-source SDN platforms, deploy it in selected households, and evaluate its usability and performance benefits to demonstrate feasibility and utility in the real world.**

I. Introduction

Residential networks are becoming increasingly complex [1]. Whereas a typical home a few years ago had but a few PCs/laptops, today's home additionally has tablets, smart-phones, smart-TVs, gaming consoles. Indeed Cisco VNI [2] predicts that the average number of connected household devices globally will rise from 4.65 in 2012 to 7.09 in 2017, representing an annual compound growth rate of 8.8%. The growing number of connected residential devices, bundled over a common Internet connection to access a range of services, poses new challenges for households that were not encountered before, as illustrated with a realistic scenario next.

Consider a family of four living in a suburban house - the father often takes work-related teleconferences from home, the mother likes watching Internet-TV, the son is a keen on-line gamer, and the daughter spends a lot of time on Facebook. Typical issues confronting this household might be: (1) Often in the evenings, the father experiences poor quality on his teleconferences; unsure if this is caused by others in the house concurrently consuming bandwidth, he tries to get his kids to stop their online gaming or social-networking activity at those times, often to no avail. Having the ability to prioritise his teleconference over other sessions would allow him to work much more effectively from home. (2) Every month the household exceeds the usage quota on its Internet plan, and the father wonders if this is because of his work teleconferences, his wife's video downloads, or the kids doing excessive online gaming/social-networking. Visibility into the volume of data consumed by each household device, and indeed being able to set per-device monthly limits, would allow the subscriber to better manage the sharing of the Internet plan within the household. (3) With kids spending more time online, parents are increasingly concerned about ease-of-access to adult/violent content, and constant distractions from online social networks. Having a means to block access at the network-level would provide additional safeguards to those implemented at the individual client devices.

The problems mentioned above can (at least partly) be solved today, but in ways that are cumbersome and demanding on the user. Several home gateways offer QoS control features, but are not easy to configure even for the technically literate, and usually prioritize traffic in the upstream (rather than downstream) direction of the broadband link. Data download volumes can be extracted from devices, but require substantial effort to harvest. Parental shield software can be installed on clients, but require per-device management, and can be circumvented by savvy kids. There is a dire need for a solution that coherently and comprehensively addresses these problems.

The residential market is very price competitive and low-margin, and ISPs tend to view "managed residential services" as not being lucrative enough. Further, "managed" services have typically meant manual provisioning and support, which is cost intensive and unscalable. Similarly, home router vendors have to-date developed proprietary and piece-meal solutions embedded into their devices, which rarely get upgraded as technologies evolve, and require high technical sophistication from the user for effective use. We believe that software defined networking (SDN) has the potential to address these challenges – it allows configurations at the network-level to be automated, while the capabilities can be exposed via carefully-crafted APIs to allow a third-party to develop more complex value-add services, that are then exposed to end-users via easy-to-use GUIs (web-portals or mobile apps).

Our specific contributions are as follows. First, we develop an "over-the-top" architecture that best enables innovation in residential Internet customization; it comprises SDN home routers, APIs built on top of SDN controllers, and portal/app-based user interfaces. Our second contribution is to identify three use-cases of residential Internet sharing (related to QoE, parental filters, and usage control) that are poorly addressed today, and to show how the underlying APIs can be composed to build new tools to dynamically control the sharing in a simple way. Lastly, we prototype our system, including the front-end portal, the back-end orchestrator, the SDN controller modules and OVS network elements. We perform a limited trial at a small number of households to validate its feasibility.

The rest of this paper is organized as follows: §II describes

our architecture and APIs. In §III we discuss use-cases to which our framework is applied. Our prototype implementation is described in §IV. In §V we evaluate an "over-the-top" deployment in a small number of houses. Relevant prior work is discussed in §VI, and the paper concluded in §VII.

II. System Architecture

In spite of the growing need for home-users to customize their household Internet sharing, the two entities best positioned to address this gap have not risen to the challenge – ISPs are contending with a highly competitive fixed-line broadband market that has dissuaded them from innovating in this segment, while home router vendors have to-date constrained themselves to software embedded within their devices, exposed via poor user-interfaces and rarely upgraded over their lifetime. We believe that it is worth trying a different approach, one that unbundles the responsibility and creates the right incentives for each entity to participate in a way best aligned with their interests and business models.

A. Entities, Roles, Flow of Events

Historically, neither ISPs nor home-router vendors have been adept at consumer-facing software. We therefore introduce a new over-the-top entity, called the *Service Management Provider (SMP)*, that undertakes development and operation of the customization services proposed in this paper. The job of the SMP is to exercise (limited) configuration control over home-router on behalf of the consumer, without being directly on the data path. Fig. 1 shows that the SMP interacts with home router equipment via standard SDN OpenFlow protocol, and with home users via easy-to-use GUIs. This architecture enables the SMP to serve subscribers of multiple ISPs.

SMP role/benefits: The SMP provides customization interfaces (portals/apps) to users (described in §III), translating these into network-level operations invoked via APIs (described in §II-B). We intentionally decouple the SMP from the infrastructure vendor so that multiple entities can compete for this role – an ISP or home router vendor may of course develop the SMP capabilities in-house, bundling it with their offerings to increase retention and revenue; a content provider (e.g. Google, Netflix) or cloud service operator (e.g. Amazon, Apple) may also have an interest in this role so it can improve delivery of its own services; or a new entrant may take up this role with a view towards greater visibility and analytics of home network usage.

Home-router vendor role/benefits: Today's home-routers (much like commercial routers) are vertically integrated, with diverse feature sets and management-interfaces bundled onto the device at production time. Since this market is fragmented and competes on price, user-experience becomes a secondary consideration, and feature sets to support emerging applications are obsoleted quickly. Our architecture encourages such vendors to forego user-interface development, and instead focus on supporting APIs that allow an external entity (the SMP) to configure network behavior. This reduces the development burden on vendors, allowing them to focus on their competitive advantage, while the cloud-based control model can give them better feedback on feature-usage on their devices.

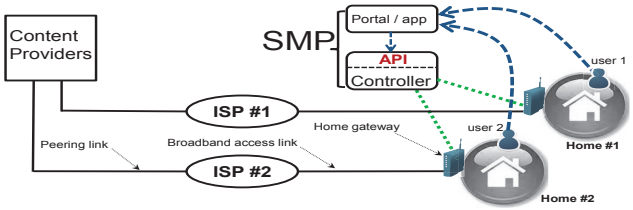


Fig. 1. High level architecture

Consumer role/benefits: The consumer's need for customizing their Internet sharing is more likely to be met by an SMP specialized in the task, than by a generalist ISP or router vendor selling a bundled product. User preferences can be learnt, stored in the cloud, and restored even if the subscriber changes ISP or the home-router. Features and look-and-feel can be personalized from the cloud, and configuration options updated as technologies and use-cases evolve.

Flow of events: The flow of events starts with a consumer signing up with an SMP, and getting access to the portals/apps through which they can manage their Internet sharing. The SMP's cloud-based controller in turn takes over the control of home router to manage services using SDN. The SMP maintains all state information pertinent to the subscriber (their devices, preferences, statistics, etc.), and translates user-requests from the portal/app to appropriate API calls into the home router, as described next.

B. APIs Exposed by the Network

We propose and justify a set of API functions below, arguing how they can be realized using SDN capability. The authentication mechanisms needed to prevent illegitimate use of the API and for logging purposes are beyond the scope of this work, as is the pricing model associated with use of these APIs.

Our API design is inspired by the approach in PANE [3] that defines three categories of interactions: applications issue *requests* to affect the state of the network, *queries* to probe the network state, and *hints* to help improve performance. In this work we restrict ourselves to the first two categories.

[Query] *Device discovery*: A function *get_devices (subscriber_id)* allows the SMP to query the network controller for a list of devices that belong to the subscriber, obtained from the home gateway. Note that the *subscriber_id* represents the identity of home router and is known to the SMP, and the *device_id* is a unique number assigned to each device belonging to that subscriber.

[Query] *Device presence*: The function *last_seen (subscriber_id, device_id)* enables the SMP to query for the last time a user device was seen in its home network, allowing it to (re)construct context relevant to security settings. A subscriber wishing to keep such information private may elect not to take up the SMP's services.

[Query] *Usage statistics*: A function *get_byte_count (subscriber_id, queue_id)* returns the downstream byte-count pertaining to a specific queue for the subscriber (a similar function can be defined for upstream traffic). The accounting capability is built into most of today's SDN platforms.

[Request] ***Bandwidth provisioning*****:** In order to manage QoE for household devices and applications, we require the home router to expose bandwidth provisioning primitives via an API. We assume that a default queue (say queue-0) at the home router initially carries all traffic destined to a household. The function *create_queue (subscriber_id)* creates a new queue on the downlink to the subscriber, and returns the id of the newly created queue (recall that queue-0 is the default queue). The function *set_queue_params (subscriber_id, queue_id, queue_rate, queue_size)* can be used by the SMP to set parameters, such as minimum service rate, maximum buffer size, etc. for this queue. The function *map_device_to_queue (subscriber_id, queue_id, device_id)* maps a user device to an existing queue (note that multiple devices can be mapped to one queue for aggregation purposes, and that a device maps to default queue-0 unless specified otherwise).

[Request] ***DNS redirection*****:** We debated whether we should create an API to request the home router to redirect an arbitrary subset of user traffic, and decided that it has potential for abuse by a malicious or careless SMP. We therefore limit the API to *dns_redirect (subscriber_id, device_id, dns_service)*, which forces all DNS requests (destination UDP port 53) from a specified device of the subscriber to be redirected to a specified server. Note that the DNS resolution server is a parameter, allowing for flexibility and customization by the SMP.

The SMP can build powerful value-add services by composing these simple network APIs exposed by the controller. The APIs can be enriched over time if this model gains traction and as new use-cases emerge.

III. Customizing Internet Sharing

Our intent in this section is to identify and elaborate on use-cases wherein subscribers can benefit from customization of their residential Internet sharing. While the use-cases are largely based on our own experiences and anecdotal evidence, we have done a limited corroboration via a survey of 100 anonymized participants (33% from USA, 38% from Canada, and 29% from Europe/Australia) recruited using the Amazon Mechanical Turk (AMT) crowdsourcing platform (ethics approval 08/2014/34 was obtained from UNSW Human Research Ethic Panel H for conducting this survey). We asked users about specific problems they faced in each of the areas described next.

A. Quality of Experience (QoE)

Among the participants surveyed, more than 20% reported that they experienced degraded online QoE frequently or very frequently, with another 40% reporting quality problems sometimes. Interestingly, users who reported a higher frequency of concurrent online usage from the house also reported more frequent degradations in quality, indicating that sharing of Internet bandwidth by household members/devices is a likely determinant of QoE. In many households today, QoE degradation is prevalent, will likely worsen in years to come, and is inadequately addressed by home router vendors. We believe that our architecture can facilitate QoE provisioning in an effective and standardized way (using SDN), and can be exposed to users in a simple-to-use manner, as will be illustrated via our prototype in the next section.

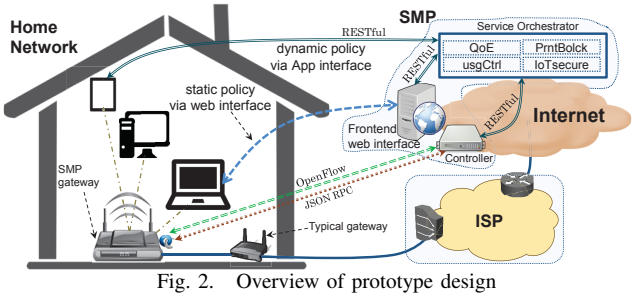


Fig. 2. Overview of prototype design

B. Parental Filters

Studies show that 72% of kids aged 0-8 in the US use smart mobile devices [4], and kids aged 9-18 average several hours online daily [5]. Unsupervised online time risks (intended or unintended) exposure to inappropriate (e.g. sexual or violent) content – prior surveys [6] have reported 70% of teens hiding their online activity from parents, and our survey found 52% of households with one or more children to express moderate to high concern about their kids' online activities. In theory, a plethora of client-side tools are available to shield kids from inappropriate content, including child-safe DNS resolvers, search engines, browser filters, operating system modes, and free/paid software suites. In practise, however, their uptake seems to be poor: 86% of users in our survey did not use any such tools (only 8% of users said they enforced safe-search, 4% managed parental control software, 2% used password/lock-protection). We believe that existing client-level solutions can be complemented with network-level blocks that cannot be easily circumvented.

C. Usage Control

Many ISPs impose limits on monthly data downloads as part of the Internet plan. While subscribers today have the ability to monitor their aggregate usage (e.g. via the ISP portal), they have little visibility into data consumption on a per-device basis. For example, knowing if the bulk of the consumption is arising from work-related teleconferences or kids' online videos might help the subscriber determine how to adapt usage pattern in the house, or how to apportion broadband charges between work and personal use. In our survey, 45% of participants had a moderate to high interest in knowing per-device data consumption in their household, yet only 17% had tried some tool that could give them this data. Client-side tools require effort from the user to install, operate, and harvest from a multitude of devices with diverse operating systems. Our architecture exposes these statistics via a clean API that allows the SMP to harvest, store, and present them to the user in a multitude of ways.

IV. Prototype Implementation

We have implemented a fully functional prototype of our system that uses our proposed architecture and APIs to provide the above customization capabilities to subscribers. Our over-the-top system includes the access switch (OVS) enhancements for home-routers, network controller (FloodLight) modules, the service orchestrator (Ruby on Rails) and

web-GUI (Javascript/ HTML) operated by the SMP. Our network controller operates in our University data-center, while the rest run in the Amazon cloud. Our implementation is currently deployed in a small number of houses (discussed in §V). Our implemented design is depicted in Fig. 2, and http://api.sdnho.me/ shows our user-interface live.

SMP Gateway: We installed OpenWrt firmware (v12.09) and OVS (v1.9.0) on a TP-Link WR1043ND gateway, and connected it at layer-2 (via the LAN interface) to the existing home gateway (so the household can fail-over to its legacy network if needed). As shown in Fig. 2, it exposes both standard OpenFlow APIs as well as JSON RPCs for queue management (explained below). We found that dynamic QoS management APIs were lacking in OVS, so we wrote our own module in C++ called ***qJump*** that bypasses OVS and exposes JSON RPC APIs to the SDN controller for queue creation and modification in the Linux kernel using `tc` (traffic controller). We enhanced our ***qJump*** module to initiate and maintain an outbound connection to port 8081 on our controller.

Network controller: We used the Floodlight (v0.9) OpenFlow controller for operating the home network, and developed Java modules to implement the APIs presented in §II-B.Successful API calls result in appropriate actions (e.g. flow table rules and queue settings) at the respective home-router (with OVS bridge).

1) *RedirectDNS:* redirects all DNS queries from a selected device to the specified DNS service. This module inserts a default rule in the access switch to hijack all DNS requests (destined to UDP port 53) from this host, and send them to the controller. Once the controller learns the IP address of the DNS server to which the original request was sent, it inserts a pair of (higher priority) rules in the OVS that can respectively replace the IP address in both directions (i.e. request and reply). This ensures that subsequent DNS requests do not need to be forwarded to the controller, and further that the DNS hijacking is transparent to the client.

2) *statsCollect*: returns transmit bytes for the chosen subscriber's queue. The subscriber id is mapped to the appropriate OVS bridge, and existing core functionality in FloodLight is invoked to extract the queue-level statistics for that bridge.

3) *discDev*: returns the id of all devices connected to the bridge associated with this subscriber. We use the device MAC address as the id (we operate at Layer-2), and obtain the MAC list per household from FloodLight's database.

4) *bandwidthManager*: manages QoE by controlling queues, their rates, and flow-rule-to-queue mappings across the access switches. This module supports queue creation and rate setting by invoking the ***qJump*** module in the appropriate switch (corresponding to the subscriber) via JSON RPC. It then updates flow rules in the flow table of the switch so that the chosen device maps to the appropriate queue.

Service Orchestrator: We implemented a service orchestrator in Ruby-on-Rails that holds the state and the logic needed by the SMP to manage services for the subscriber. It interacts on one side with the controller via the aforementioned APIs, and on the other side with the front-end portal and user apps (described next) via RESTful APIs, as shown in Fig. 2. It uses a MySQL database with tables for subscribers, devices, queues, policies, user preferences and statistics. It acts upon REST commands from the user portal/apps by retrieving the appropriate state information corresponding to the subscriber's command, and calling the appropriate sequence of controller's APIs, as discussed for each functionality next.

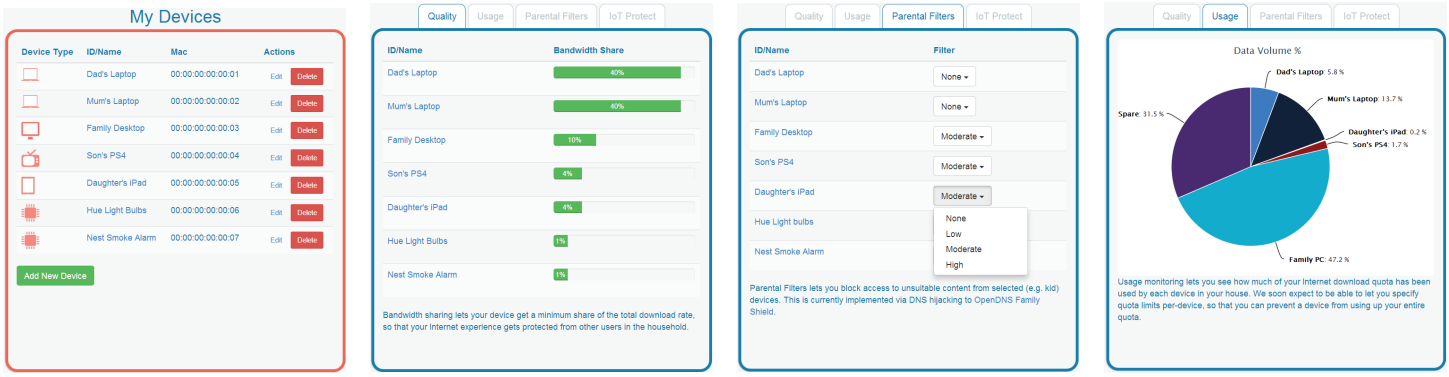
Web-based portal: provides the front-end for users to customize their services, and is implemented in Javascript and HTML. Snapshots are shown in Fig. 3, and we encourage the reader to see it live at http://api.sdnho.me/. Upon signing in, the user sees their household devices listed in the left panel, while the right panel shows tabs for each service. Fig. 3(a) shows 7 devices for this user served by TPG (the subject of the experiments described in §V), comprising laptops, desktop, iPad, TV, and IoT devices. Each service tab is described next.

The ***Quality*** tab (Fig. 3(b)) gives the user a slider bar to set a download bandwidth share for each device; in this example the father's laptop is set to get at least 40%, the kid's iPad to 4%, etc. When the bandwidth share is changed via the slider, the portal calls the REST API "`POST /qos/subsID {"mac":<mac>, "bw":<bw>}`" to the service orchestrator, which checks its internal mappings of user device to queue id, and calls the controller's API to set the bandwidth for the appropriate queue, first creating the queue if needed.

The ***Parental Filters*** tab (Fig. 3(c)) allows the user to select a filtering level for each device in the house. Our implementation currently uses the standard settings provided by OpenDNS – the "moderate" setting blocks all sites that have nudity, alcohol/drug, gambling, etc., while "high" additionally blocks messaging, social networking, and photo sharing. The user's choice of filtering level for their kid device is conveyed to the service orchestrator via the REST API call `POST /pc/subsID {"mac":<mac>, "dns":<filter-level>}` where the filter levels are the ones supported by OpenDNS for now, and can be extended to arbitrary custom lists in the future. The service orchestrator in turn maps this call to the controller's API for DNS redirection. The `GET` and `DEL` methods are also implemented so the UI can list/delete filter settings.

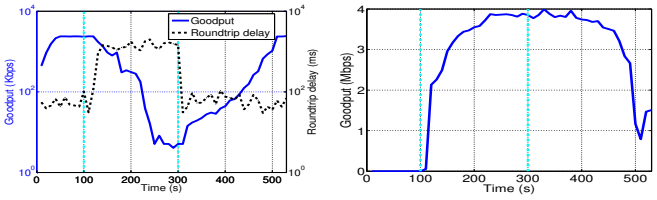
The ***Usage*** tab (Fig. 3(d)) shows usage statistics (e.g. download volume) for each device on a daily, weekly, or monthly basis. The web interface makes a `GET /usage/subsID {"mac":<mac>, "since":<time>}` call to the service orchestrator to obtain the data downloaded by a device since a given time. The service orchestrator obtains the current byte count for the corresponding queue via the *get_byte_count()* API; subtracting the byte count value at the value of parameter `time` (stored in its state tables) yields the data volume downloaded by the device since the specified time, which is then displayed as a pie-chart in the portal.

Additional to the portal (which requires proactive setting by the user), we have also developed two customized iOS applications, ***Skype+*** and ***YouTube+***, similar to the ones reported in [7], that allow the user to react to poor QoE by dynamically dilating bandwidth. Our apps provide the user with a "boost"



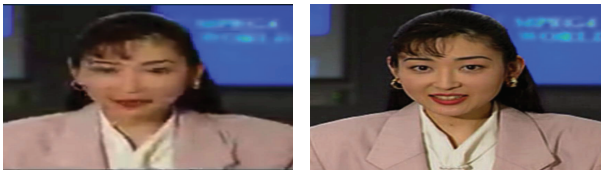
(a) Home network devices (b) QoE Control (c) Parental Filters (d) Usage Statistics

Fig. 3. Web interface showing (a) devices, (b) bandwidth, (c) filters, and (d) usage.



(a) Skype (b) IDM

Fig. 4. Skype and IDM performance at home



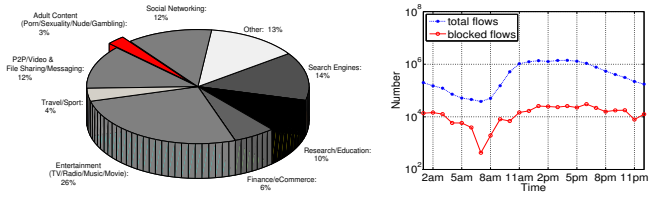
(a) 180p (b) 720p

Fig. 5. Skype quality (a) without and (b) with "boost"

button that signals the SMP (using the same REST APIs to the service orchestrator as used by the portal) to increase the bandwidth share for the device – for Skype+ we reserve 2 Mbps for HD quality, and for YouTube+ we hardcode a static mapping of video resolution to bitrate.

V. Residential Experimental Results

We have deployed a limited trial of our system in 5 houses (including some of the authors of this paper), covering the four major ISPs in Australia (Telstra, Optus, iiNet and TPG). Our SMP controller (FloodLight augmented with our modules) runs on a VM in the University data center. For the experiments described in this section, QoE control, was problematic, since our OVS gateway sits *downstream* from the bottleneck link (from the ISP to the home), and hence cannot directly control sharing of bandwidth at the bottleneck. To overcome this we came up with a crude solution – we modified our ***qJump*** module to artificially create a bottleneck within the home network; for example, if the broadband link has 5Mbps downstream capacity, we throttle it to 4Mbps within the home, and then do downstream queue management to partition this capacity amongst the home devices in the desired fraction. This throttling forces TCP to react by adjusting its rate, which after several RTTs converges to the desired rates. While this is not ideal (since it wastes some broadband capacity), it achieves the desired effect, as demonstrated next.



(a) Skype (b) IDM

Fig. 6. (a) Domain tagging of our trace, (b) Measure of "Parental Filter"

A. Quality of Experience

Our experimentation of QoE control reported here was done in the house served by iiNet. The download capacity was throttled to 4Mbps within the house, and two devices were connected to our OVS gateway: one running Skype and the other downloading a large file using IDM. Fig. 4 shows how Skype and IDM progress with time. Initially, Skype operates by itself and experiences perfect quality (average goodput 2.3Mbps and roundtrip delay below $100ms$). At 100s, IDM starts, and Skype bit-rate drops steadily to about 4Kbps and RTT rises above 1.5s, while IDM consumes nearly all the bandwidth available. The Skype video resolution drops to 180p, with pixelation as shown in Fig. 5(a). At 300s, the user signals the SMP by pressing the "boost" button on our ***Skype+*** app, requesting a minimum rate of 2Mbps. The SMP triggers the Skype traffic to be isolated in a separate queue on the OVS home gateway and given 2Mbps. The resulting throttling causes IDM to gradually reduce its rate (Fig. 4(b)), allowing Skype bandwidth to steadily increase, as shown in Fig. 4(a).

The "Parental Filters" tab in the SMP portal (Fig. 3(c)) allows the user to select a filtering level for each household device. To illustrate its potential value, we collected a 24-hour trace of flow-level activity from the University cache-log system, comprising 13.92 million flows accessing $87,794$ unique domains. We wrote a Python script to query OpenDNS for the tags associated with each of these domains (OpenDNS categorizes over 2 million domains with 57 tag values). OpenDNS successfully returned tags for 91.2% of the domains we queried – in Fig. 6(a) we show a pie-chart of the proportion of sites corresponding to the various tags (collapsed to a small number for convenience of depiction) for a one-hour section of our trace. We observed that 3% (i.e. $15,776$) of all accessed

sites were tagged as having adult content, while 12% were categorized as social networking.

To validate that a parent can filter such content out, we nominated one laptop in the home (served by TPG) as a child device, and wrote a Python script that replays the entire campus trace data of 13.92 million flows above. Using the portal, we set this device to have a "Moderate" level of filtering, which we map to the OpenDNS FamilyShield DNS server (`208.67.222.123`). The service blocks adult content, associated with tags "Pornography", "Tasteless" and "Sexuality" (in addition to also blocking phishing and other malware sites). Fig. 6(b) shows (on log scale) about 4.5% of flows to be blocked hourly, returning to the user a default page stating that the service is blocked. If the filtering level is set to "high", social networking sites also get blocked. This demonstrates the ease with which a subscriber can protect their child from inappropriate web content, and how an SMP can empower the consumer to arbitrarily customize this service.

B. Usage Control

Fig. 3(d) shows a pie-chart depicting data consumption by each device on a selected day (the reader can also see this in the "Usage" tab at http://api.sdnho.me/). In all, the devices downloaded 4,571MB of data, with the family PC dominating. Assuming a monthly quota of 200GB, i.e. 6.67GB per-day, the house (served by TPG) nominally has about 31.5% spare for the day. The take-away message is that depicting such data (be it daily, weekly, or monthly) in various ways is more easily achieved from the cloud, than from a home-router. The interface can easily be augmented to allow per-device caps on download volume consumption, without requiring any upgrade in the network or at the user premises.

VI. Related Work

HCI research has captured the growing complexity of managing home networks [8], and surveys of existing router/OS-based tools have revealed usability problems as a major impediment [9]. We are by no means the first to propose new tools and architectures for the home network – Kermit [10] gives visibility into network speeds and usage for household devices; HomeVisor [11] offers a home network management tool enabling remote administration and troubleshooting via high-level network policies; [12] dynamically prioritizes home network traffic by monitoring application windows in focus; [7] presents interfaces and apps similar to ours for the user to interact with the underlying network to control quality for different applications; and our own earlier work in [13] develops a client-hosted application for QoE control, which is a precursor to the cloud-based approach taken in the current paper. While all the above works are relevant, the present work distinguishes itself by focusing on broadband sharing and customization (rather than network set-up or trouble-shooting), and develops a three-party architecture (and interfaces) with a specialized entity that develops holistic and easy-to-use cloud-based utilities.

Several broader frameworks developed for enterprise, WAN and data-center networks are also relevant to this work: [14] allows QoS control in the enterprise; [3] inspires some of our APIs for application-network interaction; Procera [15] develops a framework for network service creation and coordination; Jingling [16] out-sources enterprise network features to external providers; while our own framework in [17] develops APIs for content provider negotiation with an ISP. Tools similar to the ones we propose are also starting to emerge in the market: HP offers SDN apps for improving performance or security in enterprise networks [18], VeloCloud [19] offers cloud-based WAN management for branch offices, and LinkSys has recently introduced a cloud-managed smart WiFi router [20]. These parallel efforts corroborate that SDN and cloud-based tools are likely to gain traction in years to come, and our work facilitates adaption of enterprise/WAN models to the home environment.

VII. Conclusions

We have argued that residential customers need better ways to manage Internet sharing in the house. ISPs and home-router manufacturers have not to-date met this need, due to a combination of business and technology reasons. We have proposed an over-the-top architecture that can help overcome the business obstacles, and developed new APIs that leverage emerging SDN technology. We identified use-cases directly relevant to homes today, and evaluated our solution via real deployment in homes. When shown our user-interface, all survey participants expressed interest in trialling it, and 37% stated that they were willing to pay for such a solution.

References

[1] A. Sabia and F. Elizalde, " Market Trends: Home Networks Will Drive Success of the Connected Home," Gartner, Report, Mar. 2013.
[2] Cisco VNI. (2012) Service Adoption Forecast for 2012-2017.
[3] A. Ferguson et al, "Participatory Networking: An API for Application Control of SDNs," in *Proc. ACM SIGCOMM*, Hong Kong, Aug 2013.
[4] Common-Sense-Media. (2013) Zero to Eight: Children's Media Use in America.
[5] iKeepSafe. (2010) Too Much Time Online. http://www.ikeepsafe.org/.
[6] CNN. (2012) Survey: 70% of teens hide online behavior from parents. http://goo.gl/vf2w0m.
[7] Y. Yiakoumis et al., "Putting home users in charge of their network," in *Proc. ACM UbiComp*, New York, NY, Sep. 2012.
[8] R.E. Grinter et al., "The Ins and Outs of Home Networking: The Case for Useful and Usable Domestic Networking," *ACM Transactions on Computer-Human Interaction*, vol. 16, no. 2, pp. 8:1–26, Jun 2009.
[9] J. Yang et al., "A Study on Network Management Tools of Householders," in *Proc. ACM HomeNets*, New Delhi, India, Sep 2010.
[10] M. Chetty et al., "Why is my Internet Slow?: Making Network Speeds Visible," in *Proc. CHI*, Vancouver, BC, Canada, May 2011.
[11] T. Fratczak et al., "Homevisor: Adapting home network environments," in *PRoc. EWSDN*, Oct 2013.
[12] J. Martin et al., "User-driven dynamic traffic prioritization for home networks," in *Proc. ACM SIGCOMM W-MUST*, aug 2012.
[13] H. Kumar et al., "User control of quality of experience in home networks using SDN," in *Proc. IEEE ANTS*, Dec. 2013.
[14] W. Kim et al., "Automated and scalable QoS control for network convergence," in *Proc. USENIX INM/WREN*, College Park, MD, USA, Apr 2010.
[15] H. Kim et al., "Improving network management with software defined networking," *Comm. Magazine, IEEE*, vol. 51, no. 2, pp. 114–119, 2013.
[16] G. Gibb et al., "Outsourcing network functionality," in *Proc. ACM SIGCOMM HotSDN workshop*, Aug. 2012.
[17] V. Sivaraman et al., "Virtualizing the Access Network via Open APIs," in *Proc. ACM CoNEXT*, Dec 2013.
[18] HP. App Store. http://www.hp.com/go/sdnapps.
[19] VeloCloud. Cloud-Delivered WAN. http://www.velocloud.com.
[20] LinkSys. Smart WiFi Router. http://www.linksys.com/en-us/smartwifi.