SDN APIs and Models for Two-Sided Resource Management in Broadband Access Networks

Hassan Habibi Gharakheili†, Vijay Sivaraman†, *Member, IEEE*, Arun Vishwanath*, *Senior Member, IEEE*, Luke Exton, John Matthews⋆, Craig Russell⋆
†University of New South Wales, *IBM Research-Australia, ⋆CSIRO Data61, Sydney, Australia
{h.habibi, vijay}@unsw.edu.au, arvishwa@au.ibm.com,
luke.exton@gmail.com, {john.matthews, craig.russell}@data61.csiro.au

***Abstract*—Access networks, largely based on DSL or cable links, continue to be the bandwidth bottleneck between device-rich households and high-speed core networks, causing frustration for both end-users and content service providers (CSPs). In this paper we advocate that the scarce bandwidth resource on the access link be managed jointly, under software control, by the ISP, consumer, and CSP. Our first contribution is to develop SDN APIs for bandwidth control at fine-grain (per-flow) by the CSP and at coarse-grain (per-device) by the consumer, and highlight the benefits of such an architecture for all entities. Secondly, we develop an economic model to guide the ISP in determining bandwidth allocation that balances the needs of the CSP against those of the consumer, and demonstrate its utility via simulation of trace data comprising over 10 million flows. Lastly, we prototype our system using commodity home routers and open-source SDN platforms, and conduct experiments in a campus-scale network to demonstrate how our scheme permits proactive and reactive improvement in end-user experience.**

***Index Terms*—Resource management, video quality, broadband fast-lanes, software defined networking.**

I. Introduction

Video traffic continues its inexorable growth over the Internet, expected to triple in volume over the next 5 years to constitute nearly 80% of global Internet traffic. Internet Service Providers (ISPs) are grappling with the problem of investment needed for their access network infrastructure to carry this growing video traffic volume, particularly since the economic gains are bypassing them and flowing to over-the-top (OTT) video content service providers (CSPs) such as Netflix and YouTube. To gain a foothold in the business ecosystem of video delivery, ISPs have been experimenting with the concept of "fast-lanes", whereby certain traffic is given higher priority over others in the network [2], [3]. This was best exemplified by the revelation in 2014 that Netflix's paid-peering arrangement with Comcast led to significant improvement in Netflix performance for Comcast subscribers [4]. ISPs have over the past few years argued in favor of fast-lanes, funded by deals with CSPs, as a means of sustaining and upgrading their access network to cope with growing traffic volumes [5], [6], [7], without increasing costs for consumers. However, policy-makers and consumer activists are circumspect that such deal-making between ISPs and CSPs can be detrimental to the best interests of the consumer, who is not consulted in whether and which traffic streams get access to the fast-lanes.

Consumer sentiment seems to be strongly in favor of "network neutrality" that prohibits the ISP from treating (positively or negatively) any traffic in their network differently based on type (e.g. video) or source (e.g. Netflix) – we refer the reader to our recent comprehensive survey on this topic [8]. The implicit assumption in this whole debate has been that fast-lane decisions are made unilaterally by the ISP (possibly in conjunction with CSPs), and are entirely out of the control of consumers. We challenge this assumption, and consider an alternative approach in which consumers are empowered to make choices on whether and which of their traffic streams get priority treatment in the network – consequently, one subscriber could choose that their video streams get priority treatment over their peer-to-peer traffic, while another household could choose to maintain neutrality of access for all streams over their broadband link. This paper is an attempt to show that such a system, in which control over fast-lanes is shared with consumers, is both technologically viable (leveraging the software defined networking paradigm) and potentially beneficial for ISPs, CSPs, and consumers alike.

There are surprisingly few proposals that try to bring both CSPs and consumers into the fast-lane negotiation. In Oct 2014 AT&T proposed fast-lanes that are controlled by end-users [9]; the proposal unfortunately reveals little technical or business detail, and it remains unclear what interfaces will be exposed to users and how these will be priced. Our proposal in [10], supported by some economic modeling in [11], develops APIs by which the CSP can dynamically request fast-lane creation from the ISP at run-time; this gives per-flow control to the CSP without having to enter into bulk-billed peering arrangements with the ISP. While our prior work does not provide much control (other than an opt-in/out button) to the end-user to control the fast-lanes, in this paper, we seek to fill this important gap by developing and evaluating an architecture that allows both the end-user and the CSP to create, dimension, and use broadband fast-lanes.

The challenges in developing a two-sided fast-lane architecture are manifold: (a) End-users and CSPs will often have different motives for traffic prioritization, leading to conflicts

This submission is an extended and improved version of our paper presented at the IEEE/ACM IWQoS 2015 conference [1].

whose resolution needs to be customized per-user based on their desires; (b) Users typically have much lower technical sophistication than CSPs, so the interfaces for control have to be quite different at the two ends; (c) The economic capacity of the two ends is again quite different, with the CSP expected to bear the cost of the fast-lane, but the end-user still being given some means of control over it. Any solution has to take the above sensitivities into account, and yet be attractive to all parties from an economic and performance point-of-view.

In this paper we develop a new architecture that addresses the above challenges. We begin by devising appropriate APIs that are suitable for the two ends of the fast-lanes, and argue that they are realizable using emerging software defined networking (SDN) technology. We then address the economic aspect of two-sided fast-lanes by devising a model that captures the trade-off between end-user and CSP happiness, and providing the ISP with means to control this trade-off. We evaluate our model using simulation with trace data of over 10 million flows taken from an enterprise network.

The rest of this paper is organized as follows: §II summarizes relevant prior work; §III describes our two-sided fast-lane system architecture and APIs. In §IV we develop a model that captures the economic gains of fast-lanes, and §V evaluates it using real trace data. Our prototype implementation is described in §VI along with experimental evaluation in a campus network, and the paper is concluded in §VII.

II. Related Work

Recent SDN-based approaches have proposed various frameworks to control service quality: APIs have been developed in [12], [13] to allow applications to dynamically interact with the network and set QoS configurations. The work in [10] develops APIs for a content provider to dynamically negotiate QoS with the ISP. However, none of these APIs specifically target home networks and deal with consumer interfaces.

In the context of home networks, [14] advocates wholesale slicing of the home broadband access network by the ISP into independent entities for sharing by multiple content providers such as video services and smart grid utilities. HomeVisor [15] offers an SDN-based management tool for a home network to enable remote administration and troubleshooting via high-level network policies, while [16] presents interfaces and apps (similar in spirit to ours) to allow the user to interact with the underlying network to control service quality for different applications. Improving home user experience using dynamic traffic prioritization is studied in [17], which actively identifies traffic flows of interest (by monitoring the application window) and signals the home router to serve the flows with a higher priority, and [18] develops a client hosted application for QoE control. While all the above works are relevant, we distinguish our work in this paper by considering two-sided control in which both the end-user and the CSP simultaneously exert influence over traffic prioritization, and develop an economic model to support it.

Several different pricing models by ISPs, termed smart data pricing (SDP), have been proposed in the literature, ranging from models for pricing only the end-users [19], [20] to two-sided pricing [21], [22], i.e. pricing both end-users and CSPs.

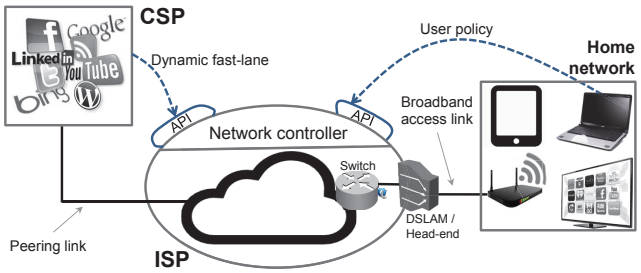


Fig. 1. A typical broadband access network topology.

These models consider usage-, time-of-day- and congestion-based pricing to affect user activity (for e.g. deterring usage by charging users more during peak hours than off-peak hours), or propose (semi-)static payment arrangements between ISP and CSP to increase their utility. By contrast, our model does not aim to charge the end-user or affect change in user behaviour, and prices dynamic fast-lanes (at a per-flow level) initiated by the CSP, as opposed to today's (semi-)static payment models between the ISP and CSP.

III. Two-Sided Fast-Lane System Architecture

Consider a representative broadband access network topology shown in Fig. 1. As is prevalent today, each household consists of a variety of devices (e.g. laptops, smart phones, tablets, smart TVs, etc.) connecting to the wireless home gateway, which offers broadband Internet connectivity via the DSLAM at the ISP's local exchange. The ISP peers directly with a number of CSPs (such as YouTube, Hulu, and Netflix) or indirectly via CDNs (such as Akamai) and other ISPs. In our proposed architecture, the DSLAM is connected to an SDN Ethernet switch (e.g. OpenFlow switch) which in turn connects to the ISP's backhaul network providing access to the global Internet. The SDN switch is controlled by an SDN controller which is housed within the ISP's network and exposes the APIs to be called – by both the end-user and the CSPs – for the creation of fast-lanes.

A. End-user facing APIs

Consider a family of four living in a household – the father uses his laptop at home for various work-related activities such as video-conferencing and Skyping, the mother uses a smart TV to watch shows or movies (e.g. Internet-TV), the son uses his laptop for gaming and watching videos on YouTube, and the daughter uses her tablet to spend time on Facebook and browse the Internet. To ensure that the users in the household get the required QoS, we permit the subscriber (e.g. the father) to configure a *minimum* bandwidth (on the broadband access link from the ISP to the household) that he deems is necessary for each of the devices in the household. We note that the link bandwidth is multiplexed in a work-conserving manner. An example of such a configuration could be: 40% of the broadband capacity is assured to the father's laptop, 30% to the smart TV, 15% to the son's laptop, 10% to the daughter's tablet, and 5% for the remaining devices in the house. The key tenets of this approach are as follows:

- *Device-level control:* We have intentionally chosen to configure bandwidth partitions at a device-level, rather than at a content-level (e.g. video, documents, HTML/images), or service-level (e.g. YouTube, Netflix, Skype, etc.) or flow-level (e.g. specific Skype call or video session). Flow-level control is too onerous for the user, requiring them to interact with the user-interface to configure fast-lane access rights for every session. Service-level control may seem easier to conceive, for example a subscriber could say that Netflix traffic is to be prioritized over Bit-Torrent. However, we feel that this approach does not capture the fact that the importance of a service often depends on the user within the household accessing it - for example YouTube/Netflix may be more important if the father or mother is accessing it, but less so if the son/daughter is accessing it; moreover, it runs the risk that subscribers will strongly favor established content providers (YouTube, Netflix) over smaller lesser-known ones. We also believe that content-level control has similar issues as service-level control. Specifically, it does not capture the fact that the importance of a content-type (e.g. video) often depends on the person within the house accessing it for example Youtube/Netflix may be more important if the family is watching TV together, but less so if the child is watching it on their personal device. Moreover, identification of various content types is more expensive and requires deep packet inspection or analysis engine. We therefore believe that device-level bandwidth control is more in line with the subscriber's view on how bandwidth should be shared within the household. Of course device-level control can be combined with service-level control (e.g. give some bandwidth to Skype on the father's laptop), but this requires more configuration on the subscriber's part (cross-product of devices and services), and does not add much value.
  Lastly, since many users have multiple devices, a per-user (or per-set-of-devices) allocation of bandwidth may seem desirable. We note that a per-user bandwidth allocation is technically achievable by mapping a group of devices (instead of one device) to a queue. However, the broader question is whether this is what users want. If they are active on multiple devices at the same time, they may want to distinguish the bandwidth requirement of the separate devices if this is the case, the user-interface becomes considerably more complex, as does the queueing mechanism that has to perform some form of hierarchical bandwidth scheduling. In order to avoid these complexities, in this paper we tackle the simpler per-device allocations, and leave per-user allocations for further study.
- *Single parameter:* We have intentionally chosen the APIs to have only a single control knob (i.e. the minimum bandwidth) because a vast-majority of end-users lack the sophistication to configure a multiplicity of parameters. A single, but intuitive, parameter reduces the barrier for end-users to adopt fast-lanes for improved QoS, and gives them control over it, which has hitherto remained elusive.
- *Proactive approach:* The crux of the QoS problem in a residential setting is bandwidth sharing amongst several household devices. To combat this problem, we advocate a set and forget device centric QoS policy, but leave the door open for end-users to seek additional bandwidth (i.e. create fast-lanes) as and when necessary for the duration of the traffic stream.
- *Scalability:* In our architecture depicted in Fig. 1, given the SDN access switch has p ports, each port serves a household of average N devices. We propose that a separate queue for each device is maintained on the pertinent port of the switch for static fast-lanes and thus a rule is associated with that. Therefore, we will have $p.N$ number of queues and rules inside an access switch for static fast-lane provisioning. For example, given a switch of 48 ports with average 10 devices per household, the switch needs to accommodate about 500 static queues and rules which can be easily achieved with todays SDN-enabled switches in the market.

B. Content Service Provider facing APIs

In contrast to the APIs exposed to the end-user, the APIs exposed to the CSP allow the latter to reserve access-link bandwidth at a per-flow level. There are several reasons why we believe such fine-grained control is the most appropriate for CSPs:

- *Economics:* Instead of paying in bulk for all the traffic they are sending via the ISP, the CSPs can exercise discretion in selecting the subset of flows for which they call the bandwidth reservation API into the ISP. For example, they may choose to reserve bandwidth only when there is congestion, or only for certain premium customer traffic. The important point here is that the per-flow API allows the CSP to make dynamic decisions on fast-lane usage, allowing them to align it with their own business models.
- *Control*: Unlike end-users, CSPs have the technical expertise to conduct per-flow negotiations on fast-lane access and the associated pricing, and are indeed expected to have automated their algorithms for doing so. This gives them the flexibility to account for various factors (time-of-day, user net-value, etc.) in making dynamic fast-lane decisions to maximize their returns.
- *Reactive approach:* The CSPs are not obliged to call the API every time a flow request is received from the end-user. Instead, it is left to the discretion of the CSP; the API can be called in a reactive manner (i.e. dynamically) such as when the QoS/QoE of the traffic flow is unsatisfactory.

The API itself for per-flow bandwidth reservation is relatively simple, and specifies the following attributes (much like in [10]): *CSP id*, the identity of the CSP making the request; *Flow tuple*, denotes the IP address and port number of the source and destination, and the transport protocol; *Bandwidth*, the minimum bandwidth that the flow, such as a YouTube video, requires; and *Duration*, the duration for which the bandwidth is requested.

C. Challenges with two-sided control

When an ISP receives a request for creating fast-lanes from the CSPs and/or end-users, the ISP has to decide whether or not to instantiate the fast-lane. On the one hand, satisfying all fast-lane requests from CSPs will generate greater revenue for the ISP, because the CSP pays the ISP for the creation of fast-lanes. On the other hand, creating a dynamic fast-lane for the CSP may violate the minimum bandwidth fast-lanes set by the end-users for their specific devices, causing annoyance to the user and potentially leading to consumer churn. The ISP therefore has to balance the revenue benefits from the CSP against the risk of subscriber dissatisfaction whenever the fast-lane configurations from the two ends conflict.

Consider the following possible scenario: seeing that the video quality of a YouTube stream on the daughter's iPad is not adequate, YouTube calls the API into the ISP network to create a fast-lane for this stream on this subscriber's broadband link. This presents an opportunity to the ISP to charge the CSP for the dynamic fast-lane. However, suppose the bandwidth requested by the YouTube stream is not currently available because the father is doing a Skype session. The ISP then has to decide whether to let YouTube access the fast-lane, in violation of the father's policy that his laptop gets a higher bandwidth share than the daughter's iPad, thereby causing subscriber frustration, or instead to just deny YouTube the requested bandwidth, thereby foregoing the revenue opportunity. Making the appropriate decision requires a cost-benefit analysis by the ISP, for which we develop an economic model in the next section.

We would like to point out that the challenges associated with two-sided control of fast-lanes is not just about resolving the policy conflicts. Indeed, there are existing frameworks (e.g. PANE [13]) that explore various techniques for conflict resolution. Our objective is to evaluate the underlying economic and performance incentives that influence how the conflicts get resolved in this fast-lane architecture with two-sided control.

IV. Dynamic Negotiation and Economic Model

We now present the dynamics of fast-lane creation, and develop an economic model to aid the ISP in making admission decisions that balances the user's needs with the CSP's.

A. Dynamic negotiation framework

Broadband fast-lanes are created via two sets of API calls: (a) relatively static policies configured by the end-user that establish per-device fast-lanes, and (b) dynamic API calls coming from the CSP for establishment of per-flow fast-lanes. We assume that the user-facing APIs do not generate revenue, and are given free-of-charge to the end-user. API calls from the CSP are however revenue-generating, with the per-flow fast-lane being associated with a micro-payment dependent on the size and duration of the flow (detailed model to follow). Further, the CSP's request for fast-lane may conflict with the user-set preferences, such as when the bandwidth requested for a video streaming flow exceeds the user-set bandwidth portion for the specific client device. The ISP is still permitted to accept the CSP call, thereby generating revenue; however this leads to violation of the user-set preferences, which can lead to user annoyance – in what follows we will assign a monetary cost to this annoyance by mapping it to a churn probability and consequent loss of revenue for the ISP.

The decision to invoke a dynamic fast-lane via the API call is entirely up to the CSP. The CSP could choose to invoke it for every video stream, or more realistically, when network conditions and/or user importance make bandwidth reservation beneficial. The CSP may even involve the user in this decision, say by embedding a "boost" button in the application that the user can press to trigger fast-lane creation to enhance QoS for this stream (such boosting capability may entail extra payment from the user to the CSP, which could partly or wholly support the cost of the fast-lane API invocation). The ISP charges the CSP each time a call from the latter is admitted. The ISP is at liberty to accept or reject the CSPs fast-lane request, and we make the reasonable assumption that if accepted, the allocation commitment is maintained over the duration of the flow (indicated in the API call from the CSP) and not modified mid-stream.

The ISP's dilemma on whether or not to accept the CSP's dynamic fast-lane request is illustrated with a simple example: Suppose a dynamic fast-lane of 2 Mbps is requested for a YouTube HD video stream to be delivered to the daughter's tablet, while the father has configured a static fast-lane of 1 Mbps for that device and 2 Mbps for his own laptop. If the father is doing a Skype video conference on his laptop at the same time, accepting a fast-lane creation call for the daughter's video stream at 2 Mbps will reduce the father's Skype quality and cause him annoyance, since this does not respect his desire to have higher bandwidth fraction for his laptop compared to his daughter's tablet.

To quantify this user annoyance, we track "violation" metric v for the household, measured as the sum across all *active* devices of the shortfall between the minimum bandwidth configured by the user for the device and the bandwidth assigned by the ISP at run-time for this device as a consequence of allocating fast-lane bandwidth to other devices, this number being normalized by the total capacity of the broadband link. For example, in the situation explained above, the ISP creates a fast-lane of 2 Mbps for the daughter's video stream and correspondingly squeezes the minimum bandwidth for the father's laptop by 1 Mbps – for a 10 Mbps broadband link, this reduction of 1 Mbps corresponds to a 10% violation, that can be computed at run-time by measuring bandwidth usage on a per-device basis. We keep track of this violation measure over time via exponential averaging – it rises in any time slot in which an *active* device gets below its configured minimum rate, and falls in any time slot in which *all active* devices are getting at least their minimum configured rates. Based on this measure, we propose a simple algorithm that the ISP can use to make call admission decisions: for a specific household, the ISP uses a target threshold (v_{th}) to cap the violations, and a call from the CSP is admitted if and only if the current violation measure v is below the threshold v_{th}. The decision to create a fast-lane is based purely on the (historical) computed value of v, and is independent of whether any device is active or not. It is easy to see that an ISP that never wants to violate

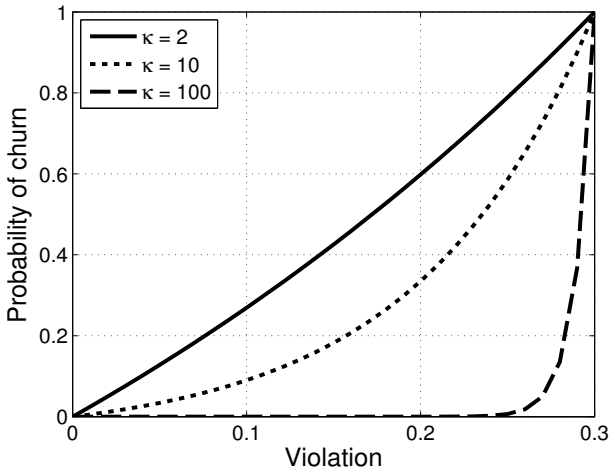


Fig. 2. Churn probability.

the user preference can choose $v_{th} = 0$, whereas an ISP that wants to accept every API call from the CSP irrespective of user preferences chooses $v_{th} = 1$. In general, an ISP could choose an intermediate value, say $v_{th} = 0.2$, that accepts CSP-side fast-lane requests that maintain user-side violations at this acceptable level.

We now attempt to convert the user-preference violation metric above into a measure of damage incurred by the ISP. Prior observations in [23], [24] show that QoE-decay is tightly bound to user-engagement (e.g. Figures 2b, 11a, 12, 13 in [23] and Figures 3, 4 in [24]) and violation of service-level agreement is a key contributor to subscriber churn [25], [26], though the relationship is not easy to capture mathematically in a succinct way. We note that each ISP will develop its own proprietary model of subscriber churn, based on its experience with its customer base. Currently, there is no such model publicly available, and therefore we resort to a simplified mathematical expression in which the user's probability of churn (i.e. of changing ISP) at the end of the billing period is an exponentially increasing function of the violation measure (consistent with reported studies in [25], [26]), given by:

$$P_{churn} = \frac{e^{\kappa v} - 1}{e^{\kappa v_0} - 1} \tag{1}$$

Here P_{churn} denotes the user's churn probability, κ in the exponent corresponds to the user's level of flexibility (discussed below), v_0 denotes the maximum tolerable violation at which the user will undoubtedly leave, and $v \in [0, v_0]$ is the measure of actual violation (computed by the ISP using an exponential moving average). The expression is chosen so that the two end-points $v = 0$ and $v = v_0$ correspond to $P = 0$ and $P = 1$ respectively. Fig. 2 depicts the curve for churn probability with three value of $\kappa = 2, 10, 100$ corresponding to increasing levels of user flexibility: at a given violation, churn will less likely occur with a larger κ. The user-flexibility parameter κ can either be explicitly solicited from the user, or learnt by the ISP based on user behaviour. Further, the ISP can give users financial incentives to choose a larger κ, since this allows the ISP to make more revenue from CSPs by accepting their fast-lane API calls; however, discussion of such financial incentives is out of the scope of the current paper.

B. Economic Model

The fast-lane service offering is free for users, but paid for by the CSP. The pricing structure we employ for dynamic fast-lanes is one in which the cost of the resource changes as a continuous function of its availability. A convenient and commonly used such function is the exponential [11], wherein the unit price of bandwidth is a function of spare capacity available on the broadband access link. The bandwidth cost is therefore set high when the spare capacity (link rate minus load) is low, and we assume it falls exponentially as the spare capacity increases, expressed by:

$$C = \lambda e^{-\delta x}, \tag{2}$$

where C is the spot cost of bandwidth (i.e. for 1 Mbps over a 1-second interval), x is the variable denoting fraction of available link capacity (computed by the ISP using an exponential moving average), λ is a constant corresponding to the peak spot-price (we use $\lambda = 1, 1.5$ cents-per-Mbps-per-sec in our simulations), and δ is a constant corresponding to the rate at which the spot price of bandwidth falls with available capacity x. Our simulations will employ bandwidth pricing with $\delta = 2$.

Shifting focus to the user-side, the violation of their per-device fast-lane policies by virtue of dynamic fast-lane creation for CSPs will cause annoyance to the subscriber; to capture the economic cost of this, we associate such annoyance with churn, i.e. the user's likelihood of changing ISPs, leading to loss of revenue for the ISP. The ISP's (monthly) change in revenue from fast-lanes will therefore equal the revenue generated from admission of CSP calls, minus the revenue lost from user churn, denoted mathematically as:

$$\sum_k (C.f_k^{rate}.f_k^{duration}) - S.P_{churn}, \tag{3}$$

where f_k^{rate} and $f_k^{duration}$ are the rate (in Mbps) and length (in seconds) respectively for the k-th fast-lane admitted by the ISP. These are multiplied with the spot price C (in dollars-per-Mbps-per-sec) of unit bandwidth (following congestion-based pricing given in Eq. (2)), and summed over all calls k admitted over the month; S is the subscription fee (in dollars-per-household per month), and is multiplied by churn probability P_{churn} to derive the loss in revenue from subscribers. Our simulations will use $S = \$60$ for a broadband service of 10 Mbps, consistent with the typical price for a 10 Mbps broadband link in most developed countries.

The objective for the ISP is to operate the fast-lanes in a way that maximizes profit in Eq. (3), by tuning the violation threshold parameter v_{th}: a larger v_{th} allows the ISP to admit more CSP calls (generating revenue), but amplifies user frustration leading to elevated churn probability (with consequent revenue loss): this trade-off, and the various parameters that affect it, are studied via simulation of real trace data next.

V. Simulation Evaluation and Results

We now evaluate the efficacy of our proposal by applying it to a 12-hour real trace comprising over 10 million flows taken from an enterprise campus network. We focus on how two critical parameters – violation threshold v_{th} chosen by the ISP, and user-churn probability exponent κ – influence revenues for the ISP and performance benefits for all the parties involved.

A. Simulation Trace Data

Our flow-level trace data was taken from a campus web cache, spanning a 12 hour period (12pm-12am). Each entry consists of flow attributes such as arrival date and time, duration (in milliseconds), volume of traffic (in bytes) in each direction, the URL, and the content type (video, text, image). The log contains 10.78 million flow records corresponding to 3300 unique end-user clients. Of these flows, $11,674$ were video flows (predominantly from YouTube, identified by the content type field), $9,799$ were elephant flows (defined as transfers of size greater than 1 MB), and the remaining 10.76 million flows were mice (defined as transfers of size 1 MB or less, representative of web pages). Though mice flows dominate by number (consistent with other measurements of residential broadband Internet traffic [27]), the three flow types contribute roughly equally by volume (32%, 32% and 36% respectively) to the total traffic downloaded. We found that: 98% of video flows required less than 5 Mbps, and only 0.2% of the flows required more than 10 Mbps; in terms of duration, 90% of the video flows last under 3 minutes, and only 1% of the flows last for longer than 10 minutes. For completeness, we note that the file transfer size of elephant flows exhibits a heavy tail, with 99% of the flows transferring under 100 MB, and the maximum file size was about 1 GB; further, 93% of the mice flows complete their transfers within 1 second, and about 0.3% of the flows transferred more than 300 KB, consistent with published findings such as [28].

B. Simulation Methodology

We developed a native simulation that reads the flows information (arrival time, duration, type, rate/volume) and injects them into the slotted simulation. Flows are serviced slot-by-slot (a slot is of duration 1 second) over a broadband access link of capacity 100 Mbps. For simplicity, we assume this access link emulates a "mega-household" representing a collection of households, each having an average DSL connection of 10 Mbps. The mega-household is assumed to house four premium mega-devices namely family TV, father's laptop, mother's laptop and daughter's tablet and one ordinary mega-device (representing all IoT devices that do not generate high volume of traffic). Each mega-device is serviced at a statically configured minimum rate (assumed to be configured by the user using the user-side API); for our experiments the family TV, father's laptop, mother's laptop, daughter's tablet, and ensemble of IoT devices are respectively set to receive at least 40%, 25%, 25%, 5% and 5% of link capacity. In each simulation run, flows are mapped into a randomly chosen mega-device proportionate to the weights mentioned above.

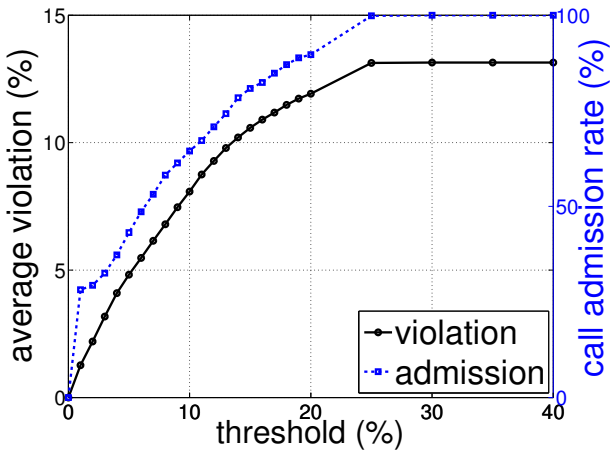


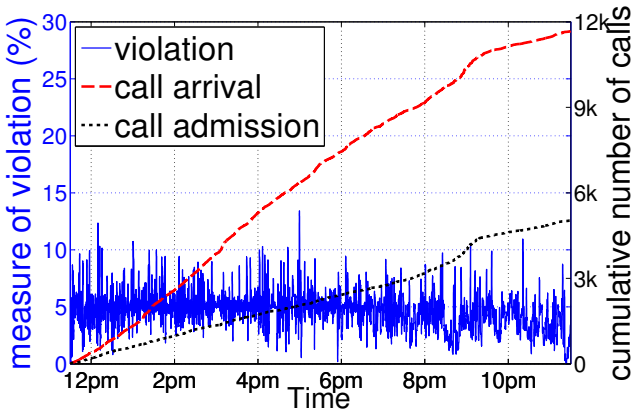
Fig. 3. Violation of user demands.

The video flows that are accommodated by the API – assumed to be constant bit rate – are allocated their own reserved queue, while the other flows (mice, elephants, and video flows not accepted by the API) share a best-effort device-specific queue. Within the best-effort queue, the mice flows (that transfer less than 1 MB) are assumed to obtain their required bandwidth first (since they are typically in the TCP slow-start phase), and the remaining bandwidth is divided fairly amongst the video and elephant flows, which are expected to be in the TCP congestion avoidance phase. The scheduling is work-conserving, so any bandwidth unused by any queues are given to the remaining best-effort queues.

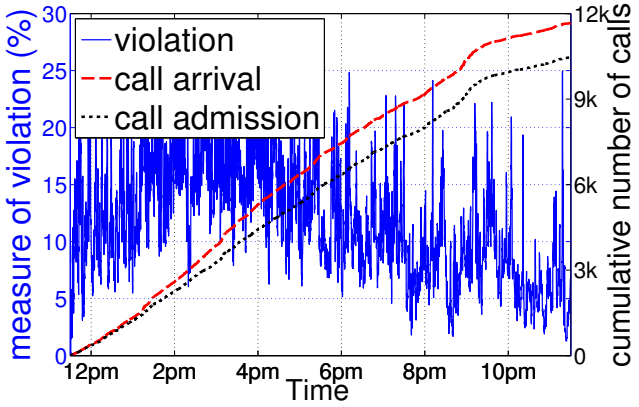
C. Performance Results

1) Impact of Violation Threshold (v_{th}): We first discuss the impact of the ISP-knob v_{th} on the overall experience of both the user and the CSP. In Fig. 3 we show by a solid black line the average violation (left-side y-axis) as a function of the chosen violation threshold v_{th}. As expected, when $v_{th} = 0$, no API call from the CSP is accepted, (the ISP therefore makes no money from the CSP); correspondingly, the user's policy is never violated and each mega-device receives its configured minimum rate at all times. As the ISP increases v_{th}, the average violation increases roughly linearly as well, saturating at about 13.14%. That is because the average video load in our trace data is about 13.76 Mbps. Thus, even if all video flows are granted fast-lanes, the bandwidth deficit would be fraction 13.76% of the link capacity, which provides an upper bound for average violation. The call admission rate for dynamic fast-lanes (dash-dotted blue curve, right-side y-axis) increases with threshold v_{th}, meaning that the CSP can exercise more control over fast-lane creation (and pay for it). At $v_{th} = 35\%$, all video flows are reserved. Increasing the violation threshold to $v_{th} = 25\%$ leads to saturation, since at this point 99.85% of CSP requests for fast-lane creation have been admitted; for this reason we truncate the plot at $v_{th} = 40\%$.

Fig. 4 shows the temporal dynamics (i.e. behaviour over the 12-hour span of the data) of violation and call admission rate with two sample threshold values: (a) $v_{th} = 5\%$, and (b)



(a) $v_{th} = 5\%$



(b) $v_{th} = 20\%$

Fig. 4. Temporal dynamics of violation and call arrival/admission.

$v_{th} = 20\%$. The observed violation rate (solid blue line, left-side y-axis) oscillates around the chosen threshold value, as expected. It is also seen that the gap between the call arrivals (dashed-red line) and call acceptance (dotted-back line) is much narrower when $v_{th} = 20\%$ (Fig. 4(b)) rather than when $v_{th} = 5\%$ (Fig. 4(a)), since a higher threshold allows the ISP to accept more CSP calls by violating the user-defined policy more frequently.

2) Impact of User Flexibility (κ): We now evaluate how the user's flexibility, captured by the parameter κ that translates their policy violation into a churn probability, affects the ISP's economics. For this study the pricing parameter δ is fixed to 2. In Figures. 5 and 6 we show the ISP profit in units of dollars, normalized per-user-per-month. We consider three types of users: (a) inflexible user corresponding to $\kappa = 2$ for whom the probability of churn rises steeply with minor increase in average violations, (b) moderate user corresponding to $\kappa = 10$ who can tolerate violation to some extent, and (c) flexible user corresponding to $\kappa = 100$ who is very permissive in letting the ISP carve dynamic fast-lanes for CSPs.

For the inflexible user corresponding to $\kappa = 2$, Fig. 5(a) shows that the ISP profit largely falls as the violation threshold is increased (bandwidth is priced at a peak rate of $\lambda = 1$ cent-per-Mbps-per-sec for this plot). This is because the risk of losing the customer due to their annoyance at violation of their policy outweighs the revenue obtained from the CSP. Fig. 6(a) shows the situation is roughly the same when the bandwidth peak price is increased to $\lambda = 1.5$, though the numerical profit is less negative. An "inflexible" user therefore poses a high economic risk for the ISP; to retain such users, the ISP has to either reject the majority of CSP API calls pertaining to this subscriber, or offer the customer some incentive (such as a rebate) to increase their flexibility parameter κ.

Increasing the user's flexibility to $\kappa = 10$ (we label such a user as being "moderately-flexible") results in an ISP profit curve shown in Fig. 5(b). In this case the ISP is able to gain an extra maximum profit of \$2.2 per-month per-user by adjusting the violation threshold to $v_{th} = 2\%$, when the bandwidth peak-price is set at $\lambda = 1$ cent-per-Mbps-per-sec. Increasing the violation threshold any higher is however detrimental, since the user annoyance over-rides the gains from the CSP. When the peak-price of bandwidth is increased to $\lambda = 1.5$ cents, Fig. 6(b) shows that the ISP can maximise profit by increasing violations for the user to about 10%, since the dynamic fast-lanes are more lucrative, thereby nearly doubling the profits to \$4.3 per-user per-month, which could even be used to subsidize the user's \$60 monthly bill.

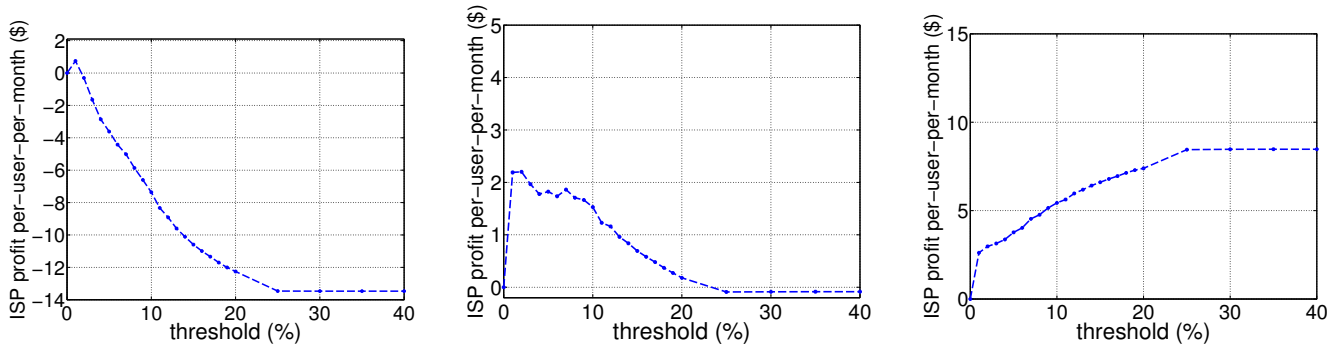
Lastly, we consider an extremely "flexible" user with $\kappa = 100$, for whom the ISP profit is shown in Fig. 5(c) (for $\lambda = 1$) and Fig. 6(c) (for $\lambda = 1.5$). As expected, we see in this case that the ISP profit rises monotonically with threshold, since the low chance of user churn encourages the ISP to accept all CSP requests for fast-lanes and charge for them. The ISP's substantial profits in this case (\$8.45 and \$12.67 per-subscriber per-month respectively for $\lambda = 1$ and 1.5 cents-per-Mbps-per-sec) can be passed on as a rebate back to the subscriber, though rebate mechanisms are beyond the scope of study of the current work.

VI. Prototype Implementation

We have implemented a fully functional prototype of our system that uses proposed the APIs to provide two-sided control of fast-lanes. Our system includes the access switch (OVS) enhancements and controller (FloodLight) modules for the ISP network, and the service orchestrator (Ruby on Rails) and web-GUI (Javascript/HTML) operated by the ISP. Our ISP controller operates in the campus data-center, while the orchestrator and GUI run in the Amazon cloud. Our implementation is currently deployed in an SDN-enabled campus network (emulating an ISP network) spanning over 3000 WiFi access points.

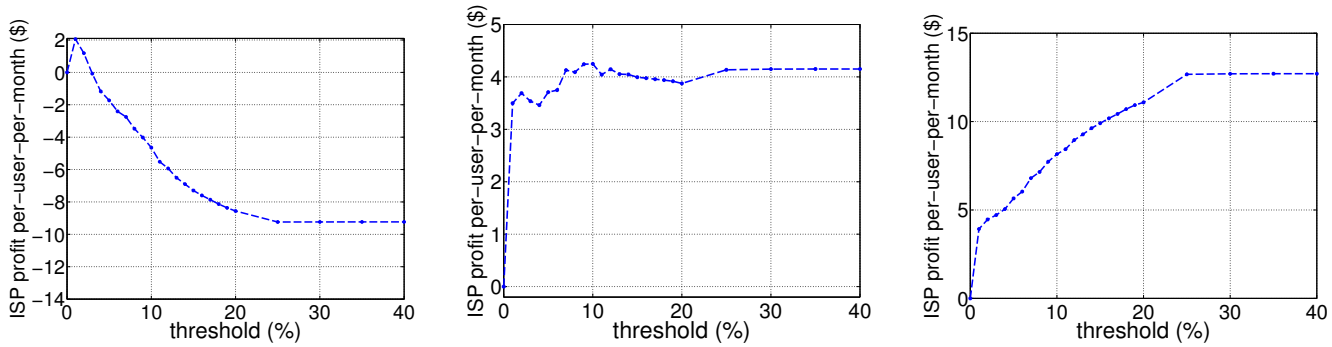
Our implemented design is depicted in Fig. 7. We assume that the ISP's access switches are SDN-enabled, and further assume that the ISP has visibility of the subscriber's household devices. This starting point is chosen for convenience since: (a) existing SDN controllers have better support for Layer-2 protocols, (b) MAC addresses are static unlike IP addresses that are usually dynamic, and (c) there is a trend towards ISPs providing managed home gateways, either by giving the subscriber a physical home gateway or a virtual instance in the cloud (e.g. vCPE).

ISP Access switch: Our access switch runs `Open vSwitch 1.9.0` (OVS), and as shown in Fig. 7, exposes



(a) Inflexible user: $\kappa = 2$ (b) Moderately-flexible user: $\kappa = 10$ (c) Flexible user: $\kappa = 100$

Fig. 5. ISP profit for $\lambda = 1$.



(a) Inflexible user: $\kappa = 2$ (b) Moderately-flexible user: $\kappa = 10$ (c) Flexible user: $\kappa = 100$

Fig. 6. ISP profit for $\lambda = 1.5$.

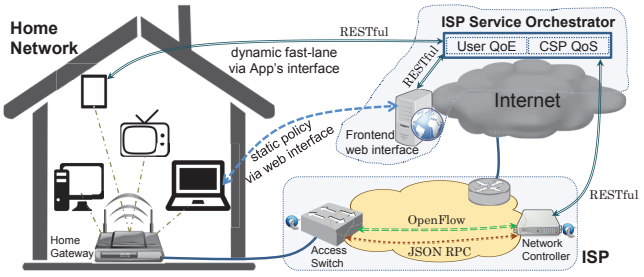


Fig. 7. Overview of prototype design.

both standard OpenFlow APIs as well as JSON RPCs for queue management (explained below). Each home is associated with a physical port on this switch, and for each home we create an instance of a virtual bridge within OVS that maintains the flow-rules and queues for that household. We found that dynamic QoS management APIs were lacking[1] in OVS, so we wrote our own module in C++ called ***qJump*** that bypasses OVS and directly manages queues in the Linux kernel using `tc` (traffic controller). The ***qJump*** module exposes JSON RPC APIs to the SDN controller for queue creation and modification.

For example the API `{"cmd":"setrate", "type":"request","tid":<tid>,"queue":<qid>, "rate":X}` allows the controller to set a minimum rate of X Kbps for queue `qid`. Upon success the ***qJump*** module responds with `{"cmd":"setrate","type":"response", "tid": <tid>, "rc":"ok"}`, or an error code if unsuccessful. Note that the transaction-id `tid` allows the response to be correlated with the request.

ISP Network controller: We used the Floodlight (v0.9) OpenFlow controller for operating the ISP network, and developed Java modules to implement the APIs presented in §III (these APIs are exposed via a RESTful interface to the service orchestrator, as shown in Fig. 7). Successful API calls result in appropriate actions (e.g. flow table rules and queue settings) at the respective OVS bridge serving this subscriber. We added new modules to FloodLight to implement the API functionalities described in §III:

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

2) ***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

[1]Very recently, similar functionality has been added via the Floodlight QueuePusher module [29] and the code has been made available in GitHub [30]

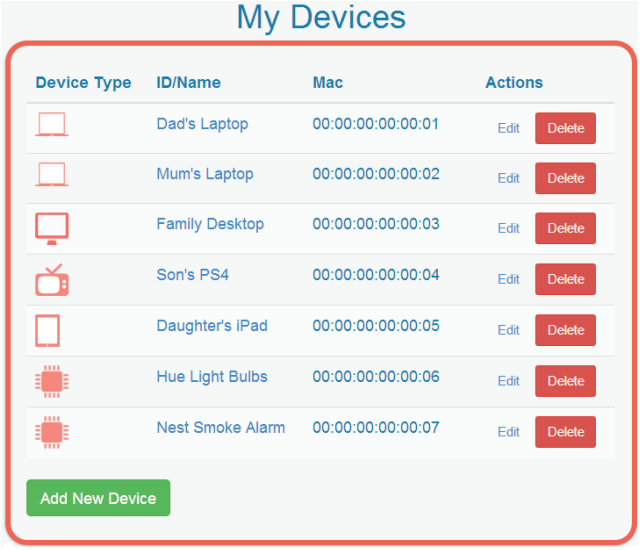


Fig. 8. Home network devices.

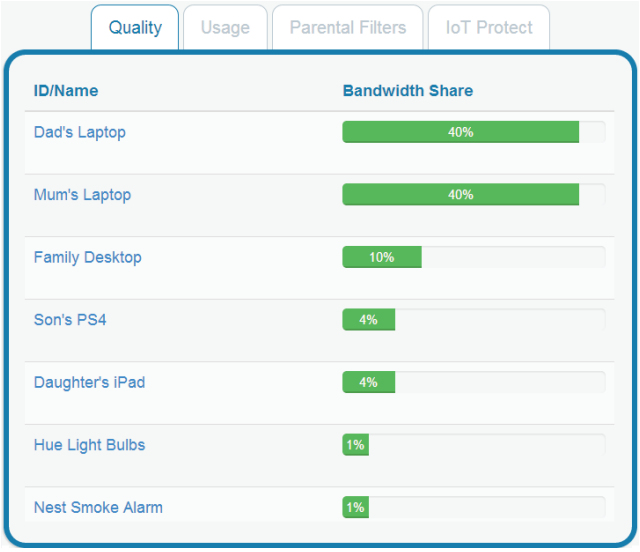


Fig. 9. QoE control.

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 to manage services for the subscriber. It interacts on one side with the ISP 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. 7. 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 (described next) by retrieving the appropriate state information corresponding to the subscriber's command, and calling the appropriate sequence of ISP 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 Figures 8 and 9. Upon signing in, the user sees their household devices listed in the left panel, while the right panel shows a "Quality" tab. Fig. 8 shows 7 devices for this user (the subject of the experiments described in §V), comprising laptops, desktop, iPad, TV, and IoT devices. Each service tab is described next.

Fig. 9 depicts the ***Quality*** control provided to the user with 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 ISP's API to set the bandwidth for the appropriate queue, first creating the queue if needed.

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 [16], that give a "boost" button to the user to react to poor QoE by dynamically dilating bandwidth. Pressing this button allows the user to signal the CSP, who can then in turn call the ISP API to create a dynamic fast-lane for the specific audio/video session. In our experiments, for Skype+ we reserve 2 Mbps for HD quality, and for YouTube+ we hardcode a static mapping of video resolution to bitrate. The impact of fast-lane configuration on user experience is evaluated in the experiments described next.

A. Campus Experimental Results

We have deployed our system in a campus network emulating an ISP access network. A guest SSID was created and runs in parallel with the regular enterprise WiFi network, giving us coverage of over 3000 wireless access points across campus to which any user device can connect using existing login credentials. Additionally, several wired ports from a lab were also added to this network. All wired and wireless traffic is delivered at Layer-2 to a set of 8 Gigabit Ethernet ports on our Dell PowerEdge R620 SDN switch (emulating the ISP access switch) running OVS (with our additions). We run our own DHCP server with a /25 public-IP address block, and default all outgoing traffic into the campus backbone network. Our controller (FloodLight augmented with our modules) runs on a VM in the campus data center. For the experiments described in this section, we throttled the access link to 5 Mbps downstream and 1 Mbps upstream, so as to represent a typical residential broadband link capacity (these bandwidth can be increased/descreased on-demand via our portal). The user portal (and the underlying service orchestrator) operated by the ISP run in the Amazon cloud, and communicate with the network controller via the APIs described earlier.

We connected several user devices across the campus, including PCs, laptops, a Google TV, and a handful of IoT devices, to emulate a large household. For our experiment illustrating QoE control, we created a scenario with concurrent access: the father is video conferencing via Skype, the mother is watching a 1080p HD video on YouTube, the son is playing Diablo-III online, the daughter is web-browsing (alternating between Facebook and Google), and

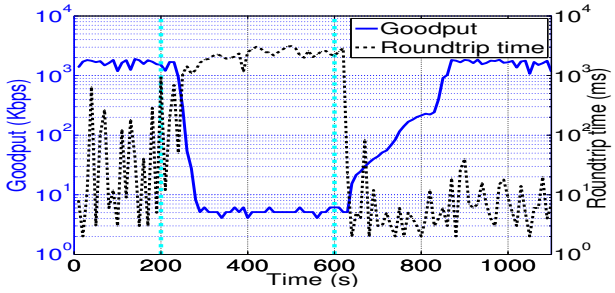


Fig. 10. Skype video call.

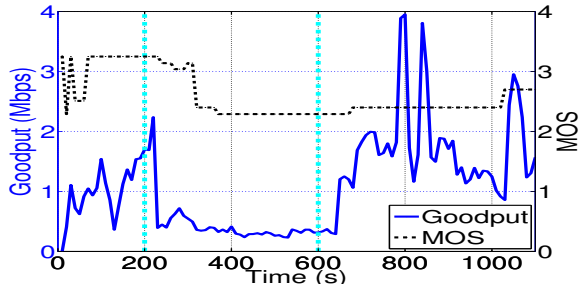


Fig. 11. YouTube streaming.

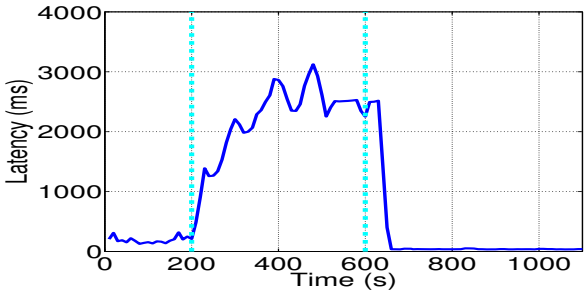


Fig. 12. Online gaming (Diablo III).

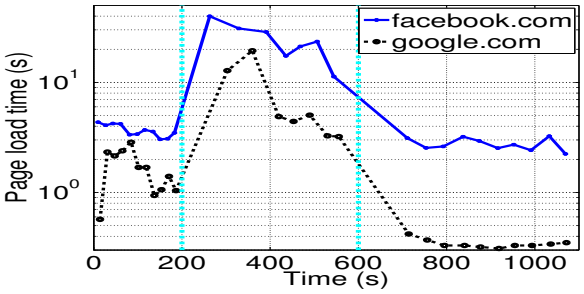


Fig. 13. Web browsing (Facebook and Google).

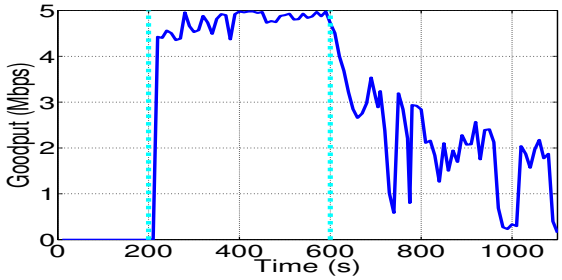


Fig. 14. Large download using IDM.

the family PC is downloading a large file (Ubuntu ISO of size 964 MB) using Internet Download Manager (IDM). The experiment runs for 1100 seconds, and the performance seen by the various household devices (depicted in Figures 10-14) is demarcated into three regions: in $[0, 200]$s, all household members except the IDM download are active, and share the broadband capacity in a best-effort manner, namely that any unused bandwidth by one device is used by other devices and does not go waste. At time 200s, IDM on the family PC starts a download, impacting QoE for all users. At 600s, the father invokes bandwidth sharing via the portal (Fig. 9), and QoE gradually recovers.

Fig. 10 shows the father's Skype video call goodput (left axis) and Round Trip Time (RTT, right axis) on log-scale at 10s intervals. Up until 200s, he gets an average goodput of 1.6 Mbps at 720p resolution and 29.85 fps. When IDM kicks in at 200s, it grabs nearly 95% (4.747 Mbps) of the broadband bandwidth (see Fig. 14). This decimates Skype, reducing its bandwidth to below 10Kbps at 180p resolution and 15fps, with RTT rising over 2s, resulting in very poor experience. At 600s the father uses our portal to configure bandwidth shares: 40% each for his and the mother's device, 4% for each of his kids' devices, and 10% for the family PC. This triggers the service orchestrator to make API calls into the network, separating the user's traffic into multiple per-device queues with minimum bandwidth guarantees. The 2 Mbps now available to the father's laptop allows Skype to slowly ramp up quality, recovering to a goodput of 1 Mbps, at 720p resolution and 30fps with RTT below 10ms. We note that Skype's recovery is somewhat slow, taking about 260s – this is because in this experiment the QoE was configured reactively by the user; once configured, Skype's performance the next time around is not impacted at all by other traffic in the house.

Fig. 11 shows the mother's YouTube experience in terms of bit-rate and Mean Opinion Score (MOS, computed by a Javascript plugin [31] that combines initial buffering time with rebuffering frequency and duration). Over the first 200s, the bit-rate is erratic (averaging 996Kbps) and there were two stalls, due to best-effort sharing with other devices. When IDM starts, YouTube's bandwidth drops dramatically to 478Kbps, it's playback buffers empty out, and several rebuffering events cause the MOS to drop from 3.25 to 2.29. Once the bandwidth partitioning is configured via the web portal, average goodput rises to 1760Kbps and there are no further rebuffering events.

Fig. 12 shows the son's Diablo-III gaming latency experience (reported by the game interface). Initial latencies average 192.2ms, peaking at 300ms, resulting in a perceptible slowdown of game reaction time. Once IDM starts, average latency degrades to 2.3s, making the game unplayable. About 50 seconds after partitioning is configured via the portal, the latency falls below 50ms, and the game experience is wonderful. Similar observations are made for the daughter's page-load times (obtained from a Chrome add-on) for Facebook (1.6MB) and Google (150KB) home-page, as shown on log-scale in Fig. 13. Initial load-times average 3.68s and 1.65s (standard deviation 0.47s and 0.72s) respectively. Once IDM starts, load-times balloon to over 25s and 7s (standard deviation 9.42s and 6.17s), but when bandwidth partitioning is enabled, load times fall to 2.76s and 0.34s (standard deviation 0.35s and 0.03s) respectively, giving the user a much better browsing experience. This experiment is but one of many where we

have seen perceptible improvement in user-experience from our scheme, and illustrates the ease with which the subscriber can control Internet bandwidth sharing in their home. The scenarios we could run on our testbed were limited by the number of devices/users we could connect to our SDN WiFi network (due to ethics and IT constraints). The point of our experiments was to validate the system to show that it is real and practical.

VII. Conclusions

As access networks continue to be the bottleneck between CSPs and end-users, ISPs are best poised to manage the scarce bandwidth on the access link. In this paper, we advocated fast-lanes with two-sided software control for bandwidth resource management, and argued how it benefits all the three entities involved, namely the end-user, CSP and ISP. We developed an architecture, powered by SDN technology, that permits an ISP to create and allocate bandwidth to fast-lanes, provides control of fast-lanes to the end-user on a per-device basis, and allows fast-lanes to be initiated dynamically by a CSP on a per-flow basis. Using simple but representative models for fast-lane economics by ISPs, associated revenue-generation for CSPs, and churn-rates for subscribers, we have shown that our approach can open doors for ISPs to monetize on fast-lanes, assure quality of flows for CSPs, and adhere to desired end-user quality of service preferences. Using simulations from real traffic traces comprising over 10 million flows, we showed that dynamic-fast-lanes is an attractive revenue stream for ISPs while limiting end-user annoyance to controllable levels. We also prototyped our system on a campus scale SDN-enabled testbed and demonstrated its efficacy via improved service quality for end-users. We believe that our solution is a candidate worthy of consideration in the continuing debate surrounding Internet fast-lanes.

References

[1] H. Habibi Gharakheili, V. Sivaraman, A. Vishwanath, L. Exton, J. Matthews, and C. Russell. Broadband Fast-Lanes with Two-Sided Control: Design, Evaluation, and Economics. In *Proc. IEEE/ACM IWQoS*, June 2015.

[2] The Guardian. The FCC is about to axe-murder net neutrality. Don't get mad get even. http://www.theguardian.com/commentisfree/2014/apr/24/fcc-net-neutrality-tom-wheeler-stop-rules, Apr 2014.

[3] Wall Street Journal. FCC to Propose New 'Net Neutrality' Rules. http://online.wsj.com/news/articles/SB10001424052702304518704579519963416350296, Apr 2014.

[4] FinancialTimes. Netflix wants to put Comcast genie back in fast lane bottle. http://www.ft.com/cms/s/0/0bc54d54-639e-11e4-8216-00144feabdc0.html#axzz3KreEKl59, Nov 2014.

[5] S. Sundaresan, W. de Donato, N. Feamster, R. Teixeira, S. Crawford, and A. Pescapè. Broadband Internet Performance: A View from the Gateway. In *Proc. ACM SIGCOMM*, August 2011.

[6] Internet Society. Measuring Internet congestion:A preliminary report. https://goo.gl/aUyWyX, Nov 2012.

[7] MIT Information Policy Project. Bandwidth Management: Internet Society Technology Roundtable Series. http://goo.gl/wodfTE, Aug 2014.

[8] H. Habibi Gharakheili, A. Vishwanath, and V. Sivaraman. Perspectives on Net Neutrality and Internet Fast-Lanes. *ACM Computer Communications Review*, 46(1):64–69, 2016.

[9] CNN Money. AT&T wants you to design your own Internet fast lane. http://money.cnn.com/2014/10/13/technology/net-neutrality-compromise/, Oct 2014.

[10] 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*, Dec 2013.

[11] H. Habibi Gharakheili, A. Vishwanath, and V. Sivaraman. Pricing user-sanctioned dynamic fast-lanes driven by content providers. In *Proc. IEEE INFOCOM Workshop on Smart Data Pricing (SDP)*, Apr 2015.

[12] 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.

[13] A. Ferguson, A. Guha, C. Liang, R. Fonseca, and S. Krishnamurthi. Participatory Networking: An API for Application Control of SDNs. In *Proc. ACM SIGCOMM*, Hong Kong, Aug 2013.

[14] Y. Yiakoumis, K.-K. Yap, S. Katti, G. Parulkar, and N. McKeown. Slicing home networks. In *Proc. ACM SIGCOMM HomeNets*, Toronto, Ontario, Canada, August 2011.

[15] T. Fratczak, M. Broadbent, P. Georgopoulos, and N. Race. Homevisor: Adapting home network environments. In *PRoc. EWSDN*, Oct 2013.

[16] 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, September 2012.

[17] J. Martin and N. Feamster. User-driven dynamic traffic prioritization for home networks. In *Proc. ACM SIGCOMM W-MUST*, aug 2012.

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

[19] S. Sen, C. Joe-Wong, S. Ha, and M. Chiang. A Survey of Smart Data Pricing: Past Proposals, Current Plans, and Future Trends. *ACM Comp. Surveys*, 46(2), 2013.

[20] S. Sen, C. Joe-Wong, S. Ha, and M. Chiang. Incentivizing Time-Shifting of Data: A Survey of Time-Dependent Pricing for Internet Access. *IEEE Communications Magazine*, 50(11):91–99, 2012.

[21] E. Altman, A. Legout, and Y. Xu. Network Non-neutrality Debate: An Economic Analysis. In *Proc. IFIP Networking*, Spain, May 2011.

[22] N. Economides and J. Tag. Network neutrality on the Internet: A two-sided market analysis. *Inf. Econ. and Policy*, 24:91–104, 2012.

[23] 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*, August 2013.

[24] F. Dobrian, V. Sekar, A. Awan, I. Stoica, D. Joseph, A. Ganjam, J. Zhan, and H. Zhang. Understanding the Impact of Video Quality on User Engagement. In *Proc. ACM SIGCOMM*, August 2011.

[25] G. Klepac, R. Kopal, and L. Mri. *Developing Churn Models Using Data Mining Techniques and Social Network Analysis*. Information Science Refrence, 2014.

[26] H. Li, D. Wu, G-X. Li, Y-H. Ke, Liu W-J., Y-H. Zheng, and X-L. Lin. Enhancing Telco Service Quality with Big Data Enabled Churn Analysis: Infrastructure, Model, and Deployment. *Computer Science Technology*, 30(6):1201–1214, 2015.

[27] G. Maier, A. Feldmann, V. Paxson, and M. Allman. On dominant characteristics of residential broadband internet traffic. In *Proc. of ACM IMC*, Nov 2009.

[28] S. Ramachandran. Web metrics: Size and number of resources. https://developers.google.com/speed/articles/web-metrics, 2010.

[29] D. Palma, J. Gonalves, B. Sousa, L. Cordeiro, P. Simoes, S. Sharma, and D. Staessens. The QueuePusher: Enabling Queue Management in OpenFlow. In *Proc. EWSDN*, Sept 2014.

[30] OneSource. Floodlight QueuePusher. https://github.com/OneSourceConsult/floodlight-queuepusher.

[31] R. Mok, E. Chan, and R. Chang. Measuring the quality of experience of http video streaming. In *Proc. IFIP/IEEE Int'l Symp. on Integrated Network Management*, May 2011.

Hassan Habibi Gharakheili received his B.Sc. and M.Sc. degrees of Electrical Engineering from the Sharif University of Technology in Tehran, Iran in 2001 and 2004 respectively, and his Ph.D. of Electrical Engineering and Telecommunications from the University of New South Wales in Sydney, Australia in 2015. He is currently a postdoctoral researcher in the School of Electrical Engineering and Telecommunications at the University of New South Wales. His research interests include network architectures, software-defined networking and broadband networks.

Vijay Sivaraman (M '94) received his B. Tech. degree from IIT in Delhi, India, in 1994, his M.S. from North Carolina State University in 1996, and his Ph.D. from the University of California at Los Angeles in 2000, all in Computer Science. He has worked at Bell-Labs and a silicon valley startup. He is now an Associate Professor at the University of New South Wales in Sydney, Australia. His research interests include software-defined networking, and sensor networks for environmental and health applications.

Arun Vishwanath (SM '15, M '11) is a Research Scientist at IBM Research - Australia. He received the Ph.D. degree in Electrical Engineering from the University of New South Wales in Sydney, Australia, in 2011. He was a visiting Ph.D. scholar in the Department of Computer Science at North Carolina State University, USA in 2008. His research interests include software defined networking and energy-efficient networking. Arun is a Senior Member of IEEE.

Luke Exton received his Bachelor's degree of Electrical Engineering and Telecommunications from the University of New South Wales in Sydney, Australia in 2015. His research interests include automated network technologies and software defined networking.

John Matthews received his B.Sc. (with honours) in 1988 from the Electronic Engineering Department of Southampton University, UK. He is a Software Engineer at CSIRO Astronomy and Space Science where he is presently active with the design of the Central Signal Processor for the international Square Kilometre Array radio telescope. His research interests include high speed and low latency networking using FPGAs and network automation.

Craig Russell received his Ph.D. in Applied Mathematics from Macquarie University, Sydney in 1997. He is currently a Principal Research Engineer in the Cyber-Physical Systems Program of CSIRO Data61. He has design, implementation and operational experience in a wide range of advanced telecommunications equipment and protocols. His professional interest is the application of advanced Ethernet and IP-based technologies to Australian industries.