Enabling Fast and Slow Lanes for Content Providers Using Software Defined Networking

Hassan Habibi Gharakheili[†], Vijay Sivaraman[†], Tim Moors[†], Arun Vishwanath[*], John Matthews[⋆], Craig Russell[⋆]
[†]University of New South Wales, [*]IBM Research-Australia, [⋆]CSIRO Data61, Sydney, Australia
{h.habibi, vijay, t.moors}@unsw.edu.au, arvishwa@au.ibm.com, {john.matthews, craig.russell}@data61.csiro.au

***Abstract*—Residential broadband consumption is growing rapidly, increasing the gap between ISP costs and revenues. Meanwhile, proliferation of Internet-enabled devices is congesting access networks, degrading end-user experience and affecting content provider monetization. In this paper we propose a new model whereby the content provider explicitly signals fast- and slow-lane requirements to the ISP on a per-flow basis, using open APIs supported through SDN. Our first contribution is to develop an architecture that supports this model, presenting arguments on why this benefits consumers (better user experience), ISPs (two-sided revenue) and content providers (fine-grained control over peering arrangement). Our second contribution is to evaluate our proposal using a real trace of over 10 million flows to show that video flow quality degradation can be nearly eliminated by the use of dynamic fast-lanes, and web-page load times can be hugely improved by the use of slow-lanes for bulk transfers. Our third contribution is to develop a fully functional prototype of our system using open-source SDN components (Openflow switches and POX controller modules) and instrumented video/file-transfer servers to demonstrate the feasibility and performance benefits of our approach. Our proposal is a first step towards the long-term goal of realizing open and agile access network service quality management that is acceptable to users, ISPs and content providers alike.**

***Index Terms*—Software Defined Networking; Access Network; Service Quality; Fast-Lanes; Slow-Lanes**

I. Introduction

Fixed-line Internet Service Providers (ISPs) are increasingly confronting a business problem – residential data consumption continues to grow at 40% per annum [2], increasing the cost of the infrastructure to transport the growing traffic volume. However, revenues are growing at less than 4% per annum, attributable mainly to "flat-rate" pricing [2]. To narrow this widening gap between cost and revenue, ISPs have attempted throttling selected services (such as peer-to-peer), which sparked public outcry (resulting in "net neutrality" legislation), and now routinely impose usage quotas, which can stifle delivery of innovative content and services. It is increasingly being recognised that ensuring sustainable growth of the Internet ecosystem requires a rethink of the business model, that allows ISPs to exploit the *service quality* dimension (in addition to bandwidth and download quota) to differentiate their offerings and tap into new revenue opportunities [3], [4].

Simultaneously, end-user expectations on service quality are evolving as personal and household devices proliferate and traffic types change. Real-time and streaming entertainment content (e.g. Netflix and YouTube) have replaced peer-to-peer as the dominant contributor to Internet traffic [5]. However, maintaining quality of experience (QoE) in online video viewing over best-effort networks remains a challenge. The rapid growth in the number of household devices (computers, phones, tablets, TVs, smart meters, etc.) concurrently accessing the Internet has increased peak-load and congestion on the access link, which is often the bottleneck between the (wired or wireless) residential LAN and the ISP backbone network [6]. The consequent impact on video quality (startup delays and rebuffering events) has been shown to lead to higher user abandonment, lower user engagement, and lower repeat viewership [7].

Content providers (CPs), who monetize their video offerings via ad-based or subscription-based models, are seeing a direct impact on their revenue from reduced user QoE. Though they use sophisticated techniques such as playback buffering, content caching, adaptive coding, and TCP instrumentation to improve video quality, these approaches are inherently limited and often involve trade-offs (e.g. increasing playback buffers can reduce rebuffering but increase startup delay). The frustrations associated with providing good QoE to users over a third-party access network may explain why some CPs (e.g. Google) are building their own fiberhoods, while some other CPs are merging with access network operators (e.g. NBC and Comcast). However, we believe that these proprietary solutions cannot be replicated world-wide (for cost and regulatory reasons), and open solutions are needed that allow any CP to improve the delivery of their services over any ISP access network.

Given the strong motivation for all parties (ISPs, users, and CPs) to want service quality capability in the network, one can rightly ask why it does not already exist. Indeed, user QoS/QoE has been studied extensively over the past two decades, and many researchers (including the authors) have worked to develop numerous technical solutions ranging from ATM-SVC to RSVP and IntServ/DiffServ. However, we believe that the limited success of these prior frameworks is partly because they have not satisfactorily addressed two critical aspects: (a) who exercises control over the service quality? and (b) how is it monetized? These challenges are elaborated next.

Control: Today, the control of network service quality is largely left to the ISP, who carefully hand-crafts policy and device configurations, likely via mechanisms (e.g. mark-

This submission is an extended and improved version of our paper presented at the ACM CoNEXT 2013 conference [1].

ing, policing, resource reservation, and queueing) from the DiffServ frameworks. Users have no visibility into the ISP's doings, and are left powerless and suspicious, wondering if "neutrality" is being violated (e.g. peer-to-peer traffic being de-prioritized). Further, exposing controls to the user also raises challenges around user expertise needed to configure and manage QoS. At the other end, CPs can exert little (if any) control over service quality in ISP networks today. They do not have access to end-to-end quality assurance frameworks (e.g. RSVP/IntServ based) since ISPs deem them either too onerous to operate or too dangerous to expose; at best CPs can indicate relative priority levels for their packets (e.g. via DiffServ code-points), but these assurances are "soft", being qualitative and subject to other traffic in the network. These concerns exacerbate further when the ISP and CP do not peer directly, i.e. connect via a transit provider. Any viable quality enhancement solution therefore has to tackle the issue of how the control is shared amongst the various players involved.

Monetization: An ISP has little incentive to deploy service quality mechanisms unless there is a monetary return. Consumers are very price sensitive, and it is unclear if sufficient consumers will pay enough for the QoS enhancement to allow the ISP to recoup costs. CPs potentially have greater ability to pay; however, current "paid peering" arrangements are based on aggregate metrics such as transfer volume or transfer rate. A CP is unlikely to pay more for "wholesale" improvement in service quality, especially if a non-negligible fraction of their traffic gets delivered at adequate quality anyway. A viable QoS solution should therefore allow the CP to make fine-grained (e.g. per-flow) decisions in an agile way so that service quality can be aligned with their business models. For example, the CP may want to deliver traffic at higher quality only for certain customers or certain content, and these decisions can vary dynamically (e.g. depending on time-of-day or loss/delay performance of the network).

The above two challenges have been poorly addressed in earlier frameworks, dissuading ISPs from deploying service quality mechanisms and causing frustration for CPs and end-users. We believe that the emerging paradigm of **software defined networking** (SDN) provides us a new opportunity to overcome this old impasse. Logical centralization of the control plane under SDN helps in many ways:

1) A central "brain" for the network makes it easier for the ISP to expose (e.g. via APIs) service quality controls needed by an external party, such as the CP. We believe that a software-driven API is a far superior method for information exchange rather than inter-connecting existing protocols (e.g. RSVP) to external parties, since (a) protocols often reveal information (e.g. network topology or network state) that is both private to the ISP and unnecessary for the external entity, whereas APIs can be crafted specifically for the negotiation task at hand, (b) protocols do not easily straddle transit domains, whereas APIs can be invoked by a remote entity that does not peer directly with the ISP, and (c) protocols are typically embedded into network switches and routers, which are not only difficult to upgrade in operation, but also take longer to converge; by contrast, APIs are implemented at the central controller that can respond rapidly to external requests and can be upgraded with relative ease. We believe that the above advantages of APIs make SDN a more suitable paradigm by which the ISP can expose and share QoS control with external entities.
2) The centralized brain in SDN is more amenable for optimal decision making. Since the SDN controller has a global view of resources, it can make informed decisions based on current availability and requests. Indeed, the decision making can also include policy rules and pricing models that could change dynamically (e.g. based on time-of-day or total resource demand and supply), which is difficult to achieve in distributed systems that have limited visibility into global state.
3) Lastly, SDN provides a cross-vendor solution that does not require protocol support from the various forwarding elements. The resource partitioning can be executed by the centralised software across any forwarding element over any access technology that supports a standardized SDN interface such as OpenFlow.

At a high level, our solution encourages the ISP to create fast- and slow-lanes (henceforth referred to as *special lanes*) for specific traffic flows by dedicating bandwidth to them on the last-mile access network link using SDN. The creation of such lanes is driven by open APIs that are exposed to external entities (CPs in our case), who can choose to invoke it to negotiate service quality with the network on a per-flow basis. For the ISP, the API offers a monetization opportunity, while also obtaining explicit visibility into traffic stream characteristics for better resource planning. For CPs, the API provides an enforceable assurance from the access network, and the pay-as-you-go model gives them freedom to align quality requirements with their business models. For users, we equip them with a simple control into the degree to which their access network resources are partitioned, allowing them to match it to their usage patterns. While past experience has taught us that any large-scale deployment of QoS faces significant practical obstacles, we believe our solution approach has the potential to overcome the business, regulatory and administrative impediments, and offers the right set of incentives for ISPs, CPs and users to collaborate for its success.

Our specific contributions are as follows. We use video streaming and file transfers as two motivating examples, and first develop a system architecture and associated APIs that allow the content provider to dynamically request special traffic lanes – video flows avail of "fast-lanes" with dedicated bandwidth over a specified duration, while large file transfers avail of "slow-lanes" that leverage the elasticity of non-time-critical traffic to provide better performance to other (streaming and browsing) traffic over the broadband link. We discuss the incentives for each party to participate in this model, and the SDN mechanisms needed for realizing it. For our second contribution we evaluate the efficacy of our approach via simulations of a real traffic trace comprising over 10 million flows. We show how fast-lanes improve video experience by nearly eliminating stalls, and how slow-lanes can significantly improve web page-load times by leveraging the elasticity of large transfers. For our last contribution, we prototype our system using open-source SDN platforms, commodity switches/access-points, and instrumented video/file-transfer servers, and conduct experiments in a test-bed emulating three residences running real applications to demonstrate how user-experience of video streams and page downloads is benefited from our scheme. We believe our work presents a first step towards a viable and pragmatic approach to delivering service quality in access networks in a way that is beneficial to ISPs, users, and CPs alike.

The rest of the paper is organized as follows: §II describes the use-cases considered in this paper. §III describes our system architecture, trade-offs, and algorithm. In §IV we evaluate our system via simulation with real traffic traces, while §V describes the prototype development and experimentation. Relevant prior work is summarized in §VI, and the paper concludes in §VII.

II. Use-Cases and Opportunities

The set of applications that can benefit from explicit network support for enhanced service quality is large and diverse: real-time and streaming videos can benefit from bandwidth assurance, gaming applications from low latencies, voice applications from low loss, and so on. In this paper we start with two application use-cases: *real-time/streaming video*, chosen due to its growing popularity with users and monetization potential for providers, and (non-real-time) *bulk transfers*, chosen for their large volume and high value to users. The APIs we develop and demonstrate for these use-cases will help illustrate the value of our approach, and can be extended in future work for other application types.

A. Real-Time / Streaming Video

Online video content, driven by providers such as Netflix, YouTube, and Hulu, is already a dominant fraction of Internet traffic today, and expected to rise steeply in coming years. As video distribution over the Internet goes mainstream, user expectations of quality have dramatically increased. Content providers employ many techniques to enhance user quality of experience, such as CDN selection [8], client-side playback buffering [9], server-side bit-rate adaptation [10], and TCP instrumentation [11]. However, large-scale studies [12], [7] have confirmed that video delivery quality is still lacking, with startup delays reducing customer retention and video "freeze" reducing viewing times. Since variability in client-side bandwidth is one of the dominant contributors to quality degradation, an ideal solution is to create network fast-lane to explicitly assure bandwidth to the video stream. Eliminating network unpredictability will (a) reduce playback buffering and startup delays for streaming video, (b) benefit live/interactive video streams that are latency bound and cannot use playback buffering, and (c) minimise the need for sophisticated techniques such as bandwidth estimation and rate adaptation used by real-time and streaming video providers.

There are however important questions to be addressed in realizing the above fast-lane solution: (a) what interaction is needed between the application and the network to trigger the bandwidth reservation? (b) is the bandwidth assured end-to-end or only on a subset of the path? (c) which entity chooses the level of quality for the video stream, and who pays for it? (d) what rate is allocated to the video stream and is it constant? (e) what is the duration of the reservation and how is abandonment dealt with? and (f) how agile is the reservation and can it be done without increasing start-up delays for the user? Our architecture presented in §III will address these non-trivial issues.

B. Bulk Transfer

After video, large file transfers are the next biggest contributors to network traffic. Examples include peer-to-peer file-sharing, video downloads (for offline viewing), software updates, and cloud-based file storage systems [5]. Unlike video, bulk transfers do not need a specific bandwidth, and user happiness generally depends on the transfer being completed within a "reasonable" amount of time. This "elasticity" creates an opportunity for the ISP to provision dynamic slow-lanes to bulk transfers, based on other traffic in the network. This can allow the ISP to reduce network peak load, which is a dominant driver of capital expenditure, improve user experience by reducing completion times for short flows such as web-page loads, and release capacity to admit more lucrative traffic streams (e.g. real-time/streaming video) requiring bandwidth assurances.

Though the idea of slow-lanes to "stretch" bulk data transfers based on their elasticity is conceptually simple, there are challenges around (a) how to identify bulk transfer parameters such as size and elasticity? (b) how to incentivize the user/provider to permit such slow-lanes? and (c) how to dimension the network resource slice for this elastic traffic? These are addressed in §III.

III. System Architecture and Algorithm

Motivated by the above use-cases, we now propose a system architecture for creation of fast and slow lanes on the access link. We first outline the major architectural choices and trade-offs (§III-A), then describe the operational scenario (§III-B), and finally develop the detailed mechanisms for special lanes creation (§III-C).

A. Architectural Choices and Trade-Offs

The aim of creating special lanes is to partition resources dynamically amongst flows in a programmatic way, so that the network is used as efficiently as possible for enhancing application performance or reducing cost. We briefly discuss why open APIs are needed to achieve the creation of special lanes, what part of the network is used for special lanes creation, and who exercises control over the special lanes.

Why Open APIs? Current mechanisms used by ISPs to partition network resources require cripplingly expensive tools for classifying traffic flows (e.g. using DPI), encourage applications to obfuscate or encrypt their communications, and risk causing public backlash and regulation. Therefore, we advocate that the creation of special lanes be driven externally via an explicit API open to all CPs. This allows CPs to choose a resource requirement commensurate with the value of the service, while letting ISPs explicitly obtain service attributes without using DPI.

What is Used for Special Lanes Creation? Assuring application performance ideally requires end-to-end network resource allocation. However, past experience with end-to-end QoS frameworks has taught us that getting the consensus needed to federate across many network domains is very challenging. In this paper we therefore focus on the achievable objective of partitioning resources within a single domain. A natural choice is the last-mile access network as there is evidence [6], [13] that bottlenecks often lie here and not at the interconnects between networks. Our solution can in principle be adapted to any access technology, be it dedicated point-to-point (DSL, PON) or shared (e.g. cable, 3G). In this paper we focus our evaluation on point-to-point wired access technologies, wherein each subscriber has a dedicated bandwidth. The case of shared media (cable or 3G) deserves a separate discussion around the policies needed to be fair to different users who embrace the special lanes scheme to different extents, and is left for future work.

Who Controls the Special Lanes? Though the special lanes APIs can be invoked by any entity, we envisage initial uptake coming from CPs rather than consumers, since: (a) uptake is needed by fewer, since as much of 60% of Internet traffic comes from 5 large content aggregators [13], (b) CPs have much higher technical expertise to upgrade their servers to use the APIs, and (c) client-side charging for API usage can significantly add to billing complexity. For these reasons, we expect CPs to be the early adopters of the fast and slow lanes APIs, and defer consumer-side uptake to future study.

The end-user still needs to be empowered with a means to control the special lanes, e.g. a user might not want her web-browsing or work-related application performance to be overly affected by streaming video that her kids watch. We therefore propose that each household be equipped with a single parameter $\alpha \in [0, 1]$ which is the fraction of its access link capacity that the ISP is permitted to create special lanes. Setting $\alpha = 0$ disables provision of special lanes, and the household continues to receive today's best-effort service. Households that value video quality could choose a higher α setting, while households wanting to protect unpaid traffic (web-browsing or peer-to-peer) can choose a lower α. Higher α can potentially reduce the household Internet bill since it gives the ISP more opportunity to monetize from CPs [14], [15]. Our work will limit itself to studying the impact of α on service quality for various traffic types; determining the best setting for a household will depend on its Internet usage pattern and the relative value it places on the streams, which is beyond the scope of this study.

B. Operational Scenario

We briefly describe the operational scenario, the reference topology, the flow of events, the API specifications, and the roles of the CP and the user.

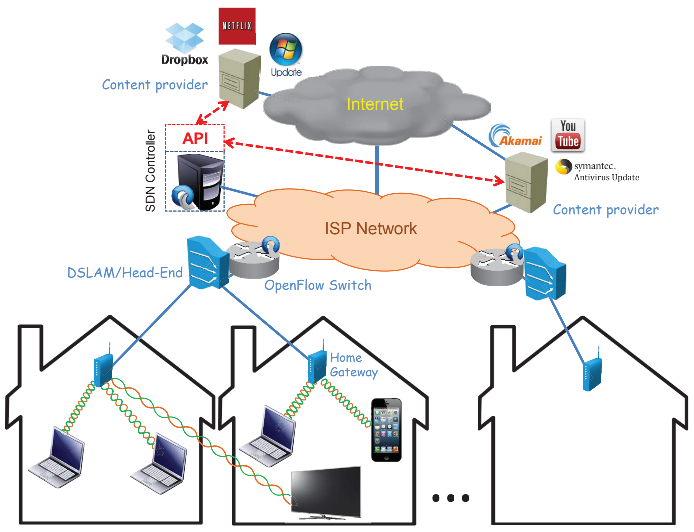


Fig. 1. Network topology of a typical residential broadband access network.

1) Topology and Flow of Events: Fig. 1 shows a typical access network topology. Each residence has a wireless home gateway to which household devices connect. The home gateway offers Internet connectivity via a broadband link (e.g. DSL or PON), connecting to a line termination device at the ISP local exchange, which is in turn back-ended by an Ethernet switch that has SDN capability. The Ethernet switches at each local exchange connect via metro- or wide-area links to the ISP's backhaul network. The ISP network houses an SDN controller that exposes the APIs discussed below, and executes the scheduling of special lanes (described at the end of this section) to reconfigure the network. The ISP network can either peer directly, or via other ISPs, to content providers that source the data that is consumed by users. Our solution works equally well when the data is sourced from CDNs or content caches within or outside the ISP network.

The operational flow of events is as follows. The user's request for content (e.g. YouTube video link click or Dropbox file transfer command) goes to the CP, who can instantly call the API into the ISP network to associate resources for this flow. If the negotiation succeeds, the ISP assures those resources for the flow, and charges/pays the CP for it. In what follows we describe the APIs in more detail and elaborate on the specific actions required by the CP and the user.

2) The APIs: We now develop minimalist specifications of the APIs for the two use-cases considered in this paper; detailed specifications are left for future standardization.

API for Fast-Lanes: This specifies: (a) *Caller id:* The identity of the entity requesting the service. Authentication of some form (such as digital signature of the message) is assumed to be included, but we do not discuss security explicitly in this work. (b) *Call Type:* A type field indicates the service being requested, in this case minimum bandwidth assurance. (c) *Flow tuple:* The 5-tuple comprising the IP source and destination addresses, the transport protocol, and the source and destination port numbers, that identify the flow (consistent with the OpenFlow specification). Note that wildcards can be used to denote flow aggregates. (d) *Bandwidth:* The bandwidth (in Mbps) that is requested by the flow. (e) *Duration:* The duration (in seconds) for which the bandwidth is requested.

This API creates fast-lane and assures minimum bandwidth to a service like video streaming. Note that the flow can avail of extra bandwidth if available, and is not throttled or rate-limited by the network. Further, we have intentionally kept it simple by using a single bandwidth number, rather than multiple (e.g. peak and average) rates. The value to use is left to the CP, who knows best their video stream characteristics (peak rate, mean rates, smoothness, etc.) and the level of quality they want to support for that particular session. The duration of the bandwidth allocation is decided by the caller. To combat abandonment, the CP may choose to reserve for short periods (say a minute) and renew the reservation periodically; however, this runs the risk of re-allocation failures. Alternatively, the caller can choose to reserve for longer periods, and the APIs can be extended to include cancellation of an existing reservation. These implementation decisions are left for future standardization. Lastly, the ISP will charge the caller for providing bandwidth assurance to the stream. The pricing mechanism is outside the scope of the current study, but we refer the reader to our companion study [14] that evaluates the benefits for both ISPs and CPs under various cost/revenue models.

API for Slow-Lanes: This includes: (a) *Caller id:* as before. (b) *Call Type:* in this case bulk transfer. (c) *Flow tuple:* as before. (d) *Size:* The volume of data to be transferred, in MegaBytes. (e) *Deadline:* The duration (in seconds) to which the transfer can be stretched. This API is for large data transfers that are not time critical, namely have a slack deadline. The elasticity can be leveraged by the ISP to stretch the flow, making way for bandwidth-sensitive flows (e.g. video streaming) and latency-sensitive flows (e.g. web-browsing). The incentive for the CP to call the slow-lane API can be monetary, namely the ISP can give a rebate to the CP for relaxing deadlines; in turn, the CP could choose to pass on the discounts to the user who is patient, such as one who is happy to download a movie for later viewing rather than streaming it in real-time (we note that the Apple TV interface does indeed ask the user if they intent to stream or download a movie; soliciting a deadline parameter directly from the user is therefore also conceivable).

The mechanism we propose for implementing slow-lanes is simple: bulk transfers are given low minimum bandwidth guarantees, set periodically by the ISP in proportion to the rate they require in order to meet their deadline. Note that this eliminates the need for the ISP to warehouse the data in transit from the CP to the user, thereby obviating technical complexities (such as proxies and split connections) and associated liabilities (e.g. with pirated data). Further, our approach is work-conserving (i.e. does not waste idle capacity), responsive to changes in demand, and does not require any user-client changes.

3) Changes for Content Provider and User: The changes required at the content servers are well-within the technical expertise of the CPs. They can identify a client's ISP based on the client IP address, and a DNS entry can be created for the controller advertised by that ISP. We note that the CP has full visibility of the flow end-points (addresses and ports), irrespective of whether the home uses NAT or not. For streaming video, the CP has knowledge of the bandwidth requirement based on format and encoding of the content. For bulk transfers, delay bounds can either be explicitly solicited from the user (via an option in the application user interface) or chosen based on previously acquired knowledge about the consumer (e.g. deadlines to ensure delivery before prime time viewing). Lastly, CPs are at liberty to align the API usage with their business models, such as by invoking it only for premium customers or based on network conditions.

Subscribers are provided with a single knob $\alpha \in [0, 1]$ that controls the fraction of their household link capacity that the ISP is permitted to carve special lanes from, adjusted via their account management portal. This parameter can be tuned by the user to achieve the desired trade-off between quality for reserved (video/bulk) flows and unreserved (browsing/peer-to-peer) flows for their household. All user clients (computers, TVs, phones, etc.) running any operating system can thereafter benefit from special lanes without requiring any software or hardware changes. For bulk transfer applications, the user interface may be updated by CPs to explicitly solicit transfer deadlines from users, potentially giving users financial incentive to choose slacker deadlines.

C. The Slow-Lane Scheduling

The time "elasticity" of bulk transfers, inferred from the *deadline* parameter in the slow-lane API call, is used to dynamically adjust the bandwidth made available to such flows. Upon API invocation, the ISP creates a new flow-table entry and dedicated queue for this flow in the switches (though scalability is a potential concern here, we note that a vast majority of flows are "mice" and will not be using the API). Periodically, the minimum bandwidth assured for this queue is recomputed as the ratio of the remaining transfer volume (inferred from the total volume less the volume that has already been sent) to the remaining time (deadline less the start time of the flow). Note that the flow can avail of additional bandwidth (above the specified minimum) if available. Also, the flow bandwidth requirement is reassessed periodically (every 10 seconds in our prototype) – this allows bandwidth to be freed up for allocation to real-time streams in case the bulk transfer has been progressing ahead of schedule, and gives the bulk transfer more bandwidth to catch-up in case it has been falling behind schedule. Lastly, the dynamic adjustment of slow-lane for this flow is largely transparent to the client and server.

IV. Simulation and Trace Analysis

We now evaluate the efficacy of our solution by applying it to real trace data. Obtaining data from residential premises at large scale is difficult; instead we use a 12-hour trace comprising over 10 million flows taken from our University campus network. Though the latter will differ in some ways from residential traces, we believe it still helps us validate our solution with real traffic profiles. We describe the characteristics of the data trace and the network topology, and then quantify the benefits from our scheme of special lanes.

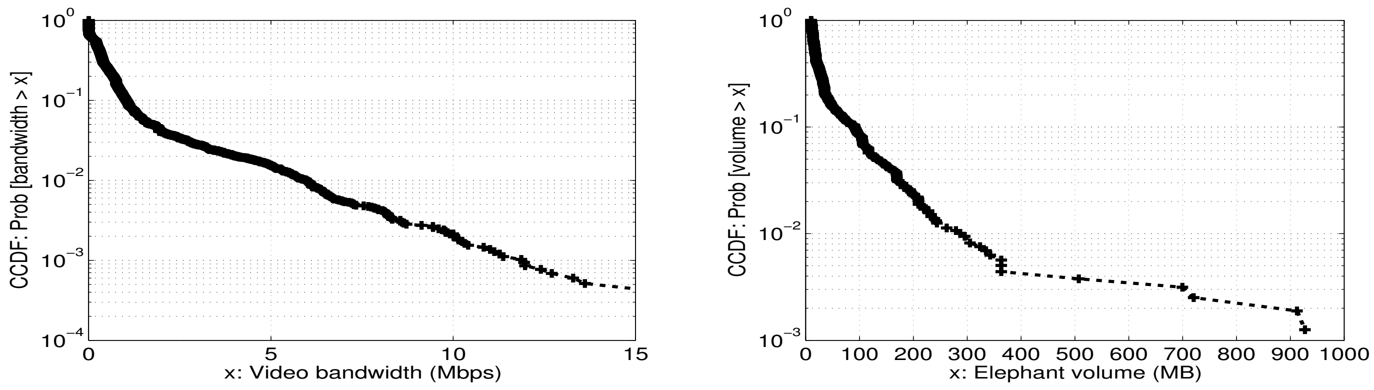


Fig. 2. Campus trace CCDF of (a) video flow bandwidth and (b) elephant flow size.

A. Trace Data and Campus Network

Our trace data was obtained from the campus web cache, containing flow level logs stored in the Extended Log File Format (ELFF). Each row pertains to a flow record, and includes information such as date and time of arrival, duration (in milliseconds), volume of traffic (in bytes) in each direction, the URL, and the content type (video, text, image, etc.). Our flow logs cover a 12 hour period (12pm-12am) on 16th March 2010, comprising 10.78 million flows and 3300 unique clients.

For our evaluation we categorize flows into three types: video, mice, and elephants. Video flows are identified by the *content type* field in the cache log, and were found to be predominantly from YouTube. We categorize the remaining flows as mice or elephants based on their download volume: flows that transfer up to 10 MB we call mice (chosen to be conservatively above the average web-page size of 2.14 MB reported in [16]), and are representative of web-page views for which the user expects an immediate response; flows transferring 10 MB or more we call elephants, and assume that they are "elastic" in that the user can tolerate longer transfer delays. Of the 10.78 million flows, we found that the vast majority (10.76 million or 99.8%) of flows were mice, while there were only $11,674$ video and $1,590$ elephant flows. However, in terms of volume, the three categories were roughly equal, constituting respectively 32%, 32%, and 36% of the traffic download volume. Note that peer-to-peer traffic does not go through the web-cache, and consequently elephant transfers are likely to be under-represented in our trace. Nevertheless, the traffic characteristics of our trace are reasonably consistent with prior observations of Internet traffic.

A time trace of the traffic volume in each category, averaged over 1-minute intervals over the 12-hour period, is shown in Fig. 3. The bottom curve corresponds to mice flows, and we found that very few (0.1%) mice flows download more than 300 KB, consistent with published findings [17].

Video traffic volume (as an increment over the mice traffic volume) is shown by the middle line in Fig. 3. To evaluate the impact of our solution on video quality, we assume that video flows have a roughly constant rate (this allows us to measure quality as the fraction of time that the video stream does not get its required bandwidth). This rate is derived by

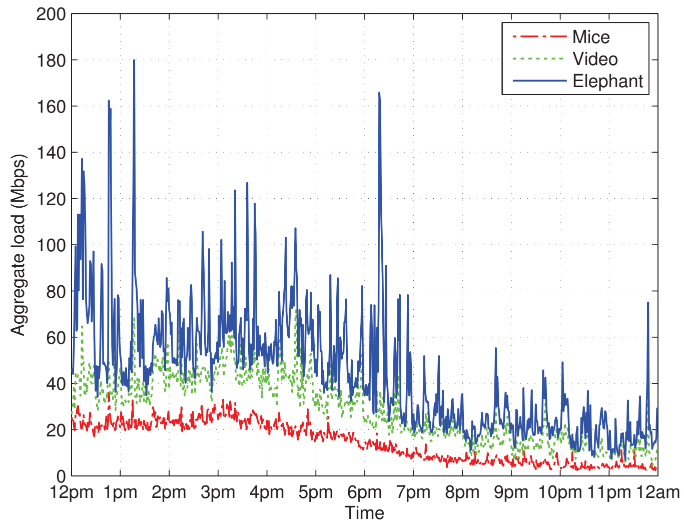


Fig. 3. Aggregate load over a 12 hour period taken from campus web cache.

dividing the video flow traffic volume by its duration. To account for the fact that video streaming uses playback buffers that download content ahead of what the user is watching, we added 40 seconds to the video flow duration, consistent with the playback buffer sizes reported for YouTube [9]. The video flow traffic rate CCDF[1] is depicted in Fig. 2(a), and shows that more than 98% of video flows operate on less than 5 Mbps, and less than 0.2% of flows use more than 10 Mbps. The video flow duration distribution (plot omitted) also decays rapidly – only 10% of video views last longer than 3 minutes, and only 1% are longer than 10 minutes.

The total elephant traffic volume (as an increment over the mice and video traffic) is shown by the top curve in Fig. 3. We observe several large spikes, indicating that bulk transfers can sporadically impose heavy loads on the network. In Fig. 2(b) we plot the CCDF of the file size, and find that it decays rapidly initially (about 8% of flows are larger than 100 MB), but then exhibits a long tail, with the maximum file size being close to 1 GB in our trace. The above traffic trace is simulated over a residential network topology that comprising 10 households, each with a 10 Mbps broadband link, as described next.

[1]Complementary Cumulative Distribution Function

B. Simulation Methodology and Metrics

We wrote a native simulation that takes flow arrivals from the trace as input, and performs slot-by-slot (where a slot is of duration one second) service. Video flows invoke the fast-lane API while elephant flows invoke the slow-lane API. The invocation (and acceptance) of these APIs for each flow is entirely at the CP's (and ISP's) discretion, but to make our study tractable we equip the video CP with a single threshold parameter θ_v, which is the fraction of available bandwidth on the access link below which the fast-lane is invoked for the video flow – a video CP that never wants to use fast-lanes is modeled with $\theta_v = 0$, whereas $\theta_v = 1$ models a video CP that invokes the fast-lane API for every video session irrespective of network load. In general, an intermediate value, say $\theta_v = 0.2$, represents a CP that requests a fast-lane for the streaming video only when the residual capacity on the broadband access link falls below 20%, and takes its chances with best-effort video-streaming otherwise. Similarly, we equip the ISP with parameter θ_b for slow-lane creation for elephant flows: $\theta_b = 0$ prevents slow-lane creation, $\theta_b = 1$ permits creation of slow-lane for every elephant flow, and intermediate values allow the ISP to permit slow-lane creation only when the access link load is higher than a threshold.

User-configured parameter α signifies the fraction of the access link capacity that is available for fast-lane creation. Admitted video flows are allocated their own fast-lane queue, elephant flows invoking the API are assigned their own slow-lane queue, and the remaining flows (including mice flows that do not call any API and video/elephant flows whose API calls are denied) share a best-effort queue. Fast lanes, that can in total take at most fraction α of the broadband link capacity, are assumed to each be served at a constant bit rate. The remaining bandwidth is shared in a weighted-fair manner amongst the best-effort queue and the slow-lane queues. The weights for the slow-lanes are updated dynamically based on their progress, computed as the ratio of the remaining transfer volume to the time remaining to reach the deadline of the respective elephant flow. The best-effort queue has a constant weight of $(1 - \alpha)C$, where C is the access link capacity, i.e. 10 Mbps in our simulation setting. Further, the bandwidth available to the best-effort queue is shared amongst the flows in that queue in a weighted fair manner, whereby the weight for a video stream is its CBR volume over a slot, for an elephant flow is the delay-bandwidth product over a slot (since the elephant flow is expected to be in TCP congestion avoidance phase), and for a mice flow its volume (since it is expected to be in TCP slow-start phase). Our simulation therefore models a weighted-fair-queueing discipline across queues, and TCP-fairness within the best-effort queue, while ensuring that the server is work conserving and does not waste any link capacity if traffic is waiting to be served.

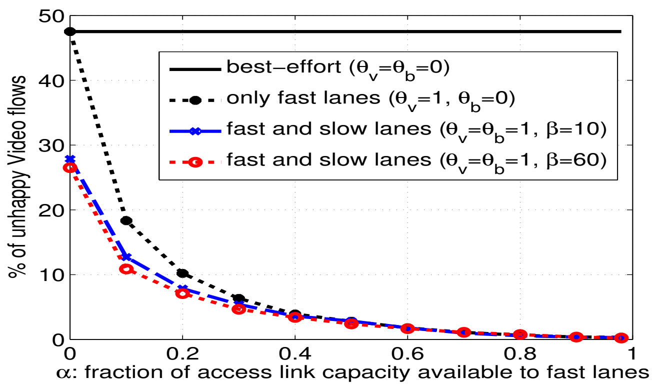
Metrics: A video flow is deemed "unhappy" if it does not receive its required CBR bandwidth for at least 10% of its duration, and a mice flow is deemed "unhappy" if it takes longer than 2 seconds to complete. The performance for an elephant flow is measured in terms of its "elongation", namely the ratio of its finish time when it is put into a slow-lane versus when it is in the best-effort queue. We will be looking into average values as well as distributions of the above metrics.

C. Performance Results

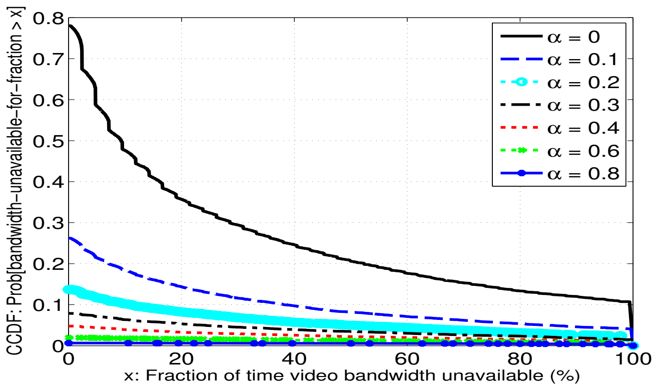
We now quantify how the performance for video, mice, and elephant flows is affected by the following parameters that control fast/slow-lanes: (a) user-chosen parameter α that denotes the fraction of the access link capacity that can be used to carve out fast-lanes; (b) the elasticity parameter β of the delay bound for elephant bulk transfers – this corresponds to the factor by which the transfer time can stretch as a multiple of the time it would require if it had exclusive use of the entire access link bandwidth; thus $\beta = 10$ permits the flow to be squeezed to one-tenth the link capacity on average, while $\beta = 60$ allows it to be squeezed to one-sixtieth (to reduce parameter space we will assume all elephant flows use identical β); (c) parameters θ_v and θ_b that denote the threshold below which residual link capacity has to fall in order for the fast/slow-lane APIs to get invoked (we will restrict ourselves to binary 0/1 values in this paper to disable/enable special lanes). We study the impact of these parameters on performance of video, mice and elephant flows in three scenarios: (a) no special lanes – best-effort service for all flows, i.e. $\theta_v = \theta_b = 0$, (b) only fast-lanes for all video flows, i.e. $\theta_v = 1$, $\theta_b = 0$, and (c) fast-lanes for all video flows and slow-lanes for all elephant flows, i.e. ($\theta_v = \theta_b = 1$), for both small and large β settings.

Impact of fast-lanes: In Fig. 4(a) we plot the percentage of video flows that are unhappy (i.e. obtain less than required bandwidth for at least 10% of their duration) as a function of fraction α of access link capacity that the user allows fast-lane creation from. The top curve shows for reference performance under today's best-effort service with no special lanes, revealing that more than 47% of video flows are unhappy. It can be observed that increasing α and allowing fast-lane creation improves video performance significantly (second curve from top), reducing the number of unhappy video flows to just 10% for $\alpha = 0.2$, corroborating with a real trace that fast-lanes do indeed improve video experience.

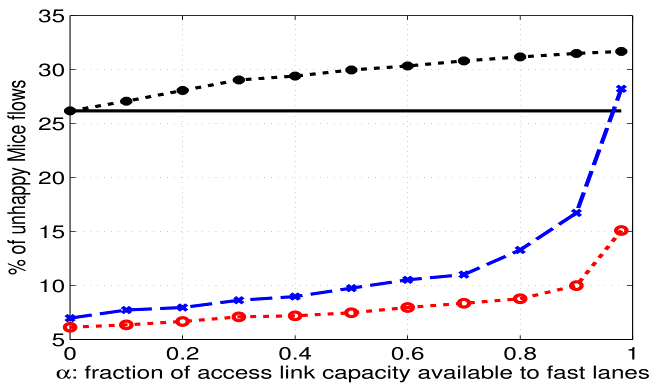
Improving video performance with fast-lanes can degrade performance for other traffic – in Fig. 4(b) we show performance for mice flows. Whereas best-effort service yielded unhappy performance (load-time of more than 2 seconds) for 26% of the mice flows (solid curve), introduction of fast-lanes for video causes the percentage of unhappy mice flows to increase steadily with α (top curve), since the available bandwidth for the best-effort queue shrinks – for example, an $\alpha = 0.2$ increases the percentage of unhappy mice flows to 28% – this can constitute a disincentive for the user to choose a higher α, particularly if they value their web-browsing experience as much or more than video. Similarly, elephant flows also experience lower throughput as α increases: whereas an elephant flow received about 5 Mbps average throughput in the best-effort scenario, this dropped by about 6% when fast-lanes are enabled with $\alpha = 0.2$. This marginal decrease in throughput seems to be a reasonable price to pay for improving video experience via fast-lanes.



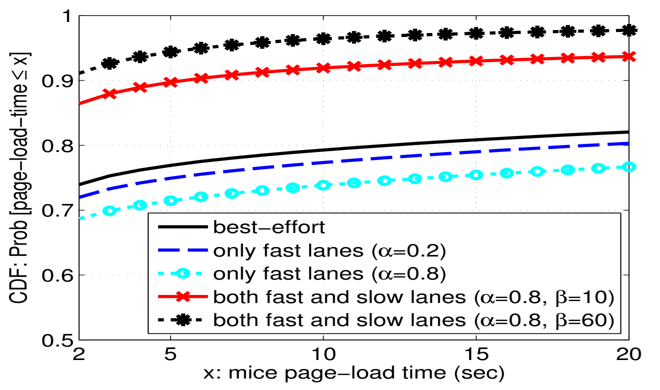
(a) Video unhappiness



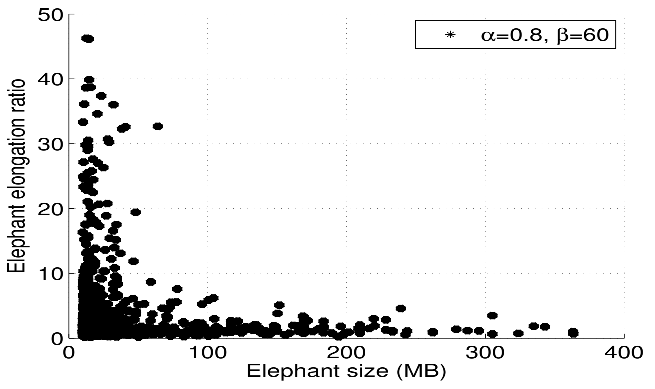
(a) CCDF of video bandwidth unavailability



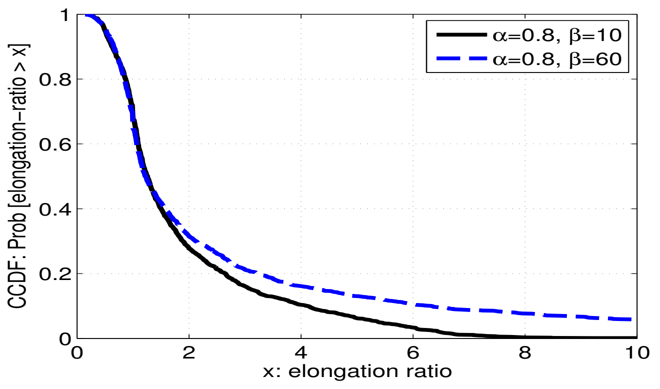
(b) Mice unhappiness



(b) CDF of mice page-load time



(c) Elephant elongation ratio



(c) CCDF of elephant elongation ratio

Fig. 4. Performance of video, mice and elephant flows.

Fig. 5. A detailed look on performance of video, mice and elephant flows

Impact of slow-lanes: The results discussed above showed that the negative impact of fast-lanes on mice flows can cause users to set their fraction α of access capacity that can be used for fast-lane creation to be low, so as to protect their web-browsing experience. This reduces the ISP's ability to monetize on fast-lanes, which can be disastrous. Slow-lanes have the ability to combat this problem, whereby large downloads (elephants) are peeled off into separate queues and elongated (as per their specified stretch factor β) to allow better service for other traffic flows. Indeed, Fig. 4(a) shows that when elephant flows are stretched ($\beta = 10, 60$ in bottom two curves) using slow-lanes, the number of unhappy video flows reduces significantly, though the benefits diminish with α, since a high α allows video flows to have their own fast-lanes anyway.

The most dramatic impact of slow-lanes is on mice flows. In Fig. 4(b), the bottom two curves (corresponding to stretch factors $\beta = 10$ and $\beta = 60$) represent the percentage of unhappy mice flows (that have load-time longer than 2 seconds) – it is seen that at $\alpha = 0.2$, introduction of slow-lanes reduce the number of unhappy mice flows from 28% to below 8%. Though the mice performance still degrades with α, the use of slow-lanes for elephant flows permit the ISP to serve mice flows far better than before – indeed, even if the user chooses a fairly high α of say 0.8, only 8% of their mice flows take longer than 2 seconds to complete, provided elephant transfers are given slow-lanes with elasticity $\beta = 60$ (bottom curve in Fig. 4(b)).

The impact of slow-lanes on elephants is shown in Fig. 4(c). Each point in the scatter plot corresponds to a flow, and shows how much it got elongated in time (in multiples of the baseline time obtained from best-effort service with no special lanes) as a function of the file size (this plot is for chosen stretch factor $\beta = 60$). For small file sizes (10-50 MB), the file transfer time can be elongated ten-fold or more – this should not be surprising, since the slow-lane is meant to take advantage of elephant elasticity to better accommodate the transient needs of other traffic. What is interesting to note in this scatter plot is that as the elephant size gets larger, the elongation drops (elephants larger than 200 MB rarely get elongated more than two- or three-fold) – a little thought will reveal that this should indeed be expected, since large elephants in slow-lanes will give way to transient spikes in other traffic, but will catch-up during lulls in other traffic (since the scheduling is work-conserving), so their long-term average rate will be no worse than in a best-effort queue.

A detailed look at performance: In Fig. 5 we show in more detail the impact of special lanes on performance quality for video, mice, and elephant flows. Fig. 5(a) plots the CCDF of the fraction of time for which a video flow does not receive its required bandwidth, for various values of α. In the absence of fast-lanes ($\alpha = 0$, top curve), more than 78% of video flows experience some level of degradation, with around 21% of flows not receiving their required bandwidth more than half the time. By contrast, allowing video fast-lanes using even just $\alpha = 0.1$ fraction of the link capacity (second curve from the top) reduces the number of flows experiencing any degradation to 26%, and this can be reduced to below 10% by setting $\alpha = 0.3$.

Fig. 5(b) shows the CDF of mice flow completion times. Best-effort service (solid line) with no special lanes allows 74% of mice flows to finish within 2 seconds and 80% within 10 seconds. Creation of fast-lanes worsens latency for mice flows (bottom two curves), with the number of mice finishing within 10 seconds falling to 77% for $\alpha = 0.2$ and 74% for $\alpha = 0.8$. However, when fast-lanes (for video) and slow-lanes (for elephants) are both invoked using their respective APIs (top two curves), well over 95% of mice flows complete within 10 seconds (for $\alpha = 0.8$ and $\beta = 60$), corroborating that high α values are compatible with good mice performance.

Lastly, Fig. 5(c) shows the CCDF of the elongation experienced by elephant flows using slow-lanes. It is observed that with $\beta = 10$, only 28% of elephants are elongated two-fold or more, and 0.1% ten-fold or more. When elasticity is increased to $\beta = 60$ (dashed line), about 30% of elephants elongate two-fold or more while 6% elongate ten-fold or more. We believe this is an acceptable price to pay for improved performance of video and mice flows.

Summary: The key observations to emerge from our evaluations are: (a) Fast-lanes significantly improve bandwidth performance for video traffic, though at the expense of increasing latency for mice flows; (b) Slow-lanes leverage the elasticity of bulk transfers to improve performance, particularly for mice flows; (c) Combined use of fast- and slow-lanes allow the user to obtain good performance for both streaming and browsing traffic, while allowing the ISP to monetize them from content providers (economic models to support this are discussed separately in [14], [18]).

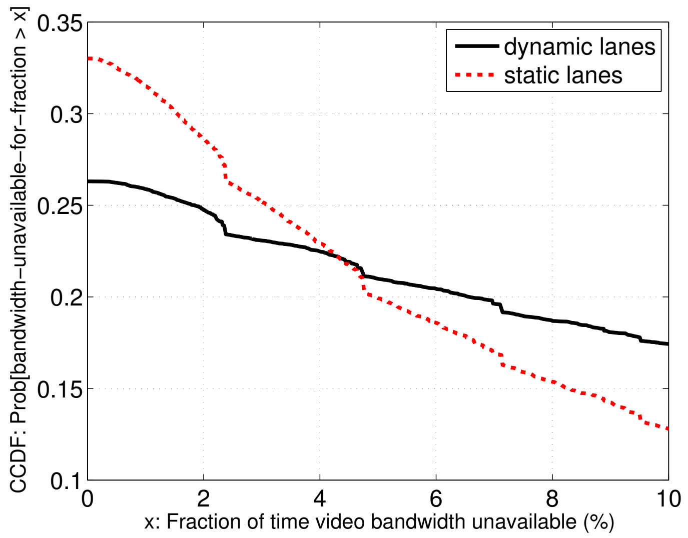


Fig. 6. CCDF of video bandwidth unavailability ($\alpha = 0.1$, $\beta = 10$)

D. Comparison to Static Lanes

We now compare the performance of our dynamically signalled fast/slow-lane scheme to a statically provisioned approach, whereby traffic flows are classified into static queues of different priority levels. We specifically choose the Metro Ethernet Forum (MEF) 23.1 [19] framework that advocates the use of three Class-of-Service (CoS) priority labels namely high (H), medium (M) and low (L); though our comparison can equally be applied to other frameworks like DiffServ that use a larger number of static queues. A set of performance objectives (e.g. bandwidth., latency, jitter) is associated to each service label. To make our comparison study fair, we provision three queues as per the MEF specifications, and serve video, mice and elephant flows by queue H, M and L respectively, with the following minimum rate configurations: we allocate α fraction of total link capacity to high priority queue H (to match the user-selected fraction of access link capacity that is allocated to video flows in our dynamic fast-lane scheme); we allocate fraction $1/\beta$ of the link capacity for the low-priority queue L (to match the bandiwdth given to an elephant flow at its commencement in our dynamic slow-lane scheme); and the remaining capacity is allocated to the M queue serving mice flows.

We simulate the static queue provisioning method above, and compare the performance of video, mice, and elephant flows against our dynamic allocation scheme. Fig. 6 depicts the CCDF of the fraction of time for which a video flow does not receive its required bandwidth under both schemes, for $\alpha = 0.1$ and $\beta = 10$. It is seen that with static allocation around 33% of video flows experience some level of degradation (dotted curve), while under our dynamic fast-lane scheme this fraction reduces to 26%. This can be explained

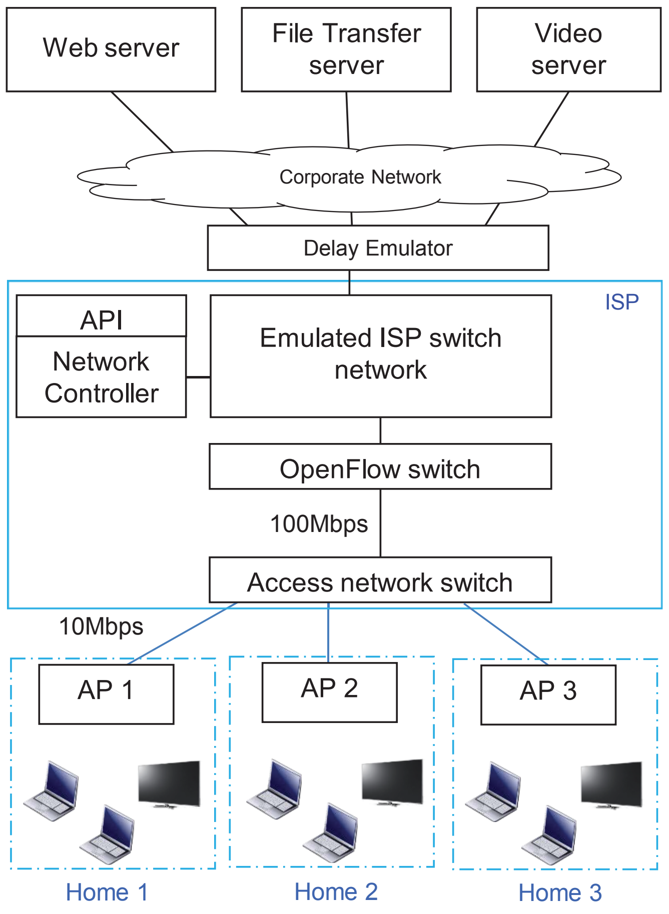


Fig. 7. Network Architecture

as follows: when video bandwidth demand exceeds the bandwidth available to video flows (fraction $\alpha = 0.1$ of access link capacity in this case), under the static scheme all video flows will suffer since they share the common (priorty H) queue, whereas under our dynamic scheme the video flows that are accepted will receive their requested bandwidth, while video flows whose request is rejected will be served in the best-effort queue. In other words, during times of over-subscription, the latter preserves quality for video flows that are accepted for fast-lane service, while the former violates it for all video flows. This demonstrates the value of dynamic allocations that can be honored at all times, unlike a static allocation that relies on over-provisioning.

V. Prototype Implementation and Experimentation

We prototyped our scheme in a small testbed, depicted in Fig. 7, hosted in a 18m×12m two-level shed, to emulate a small part (3 homes, each with multiple clients) of a residential ISP network. The objectives of this experimental setup are to demonstrate the feasibility of our scheme with real equipment and traffic, and to evaluate the benefits of special lanes for real video and bulk-transfer streams.

A. Hardware and Software Configuration

Network Topology: The clients are connected wirelessly to their home AP, each of which has uplink broadband capacity of 10 Mbps emulating a DSL/cable/PON service. The APs connect back to an access switch (emulating a DSLAM, cable head-end, or OLT), which is back-ended with an OpenFlow capable Ethernet switch. This connects through a network of switches (emulating the ISP backbone network) to the controller (that implements the API) and to a delay emulator that introduces 5 ms of delay before forwarding traffic on to the servers through the corporate network (the delay emulator and corporate network together emulate the Internet).

Openflow switch: Our switch was a 64-bit Linux PC with 6 Ethernet ports, running the OpenFlow 1.0.0 Stanford reference software implementation. It supported 200 Mbps throughput without dropping packets, which is sufficient for our experiments. The switch has a default best-effort FIFO queue for each home, and a separate queue was created for each flow that made a successful API call to the controller. Linux Hierarchical Token Buckets (HTBs) assure minimum bandwidth to those queues in proportion to their weights.

Network controller: We used the POX OpenFlow controller and developed Python modules that used the messenger class to execute the API calls using JSON from our video and bulk-transfer servers. Successful API calls result in the installation of a flow table entry at the OpenFlow switch to direct the traffic along the desired path. We also implemented the mechanism at the controller, which makes call admission decisions, and polls the switch every 10 seconds to check the volume sent for each bulk-transfer flow, computes the minimum bandwidth required to complete the transfer within the agreed time, and configures this for the HTB queue at the switch. This periodic reconfiguration of bandwidth for bulk-transfer flows involved very low transmission overhead (of the order of a few bytes per second per flow).

Video server: A Python scripted video on demand server was developed using Flup. For each user video request, the server calls the fast-lane API via a JSON message to the network controller. An example is: {hello: jukebox, type: minbw, nwsrc: `10.10.7.31/32`, nwdst: `10.10.5.18/32`, proto: 6, sprt: 8080, dprt: 22400, bw: 7600}. In this case the server requests a fast-lane with minimum bandwidth of 7.6 Mbps for TCP on the path from `10.10.7.31:8080` (server) to `10.10.5.18:22400` (client). The server executes a new VLC (v2.0.4) instance for each video stream, and periodically renews the bandwidth reservation until user video playback ends with TCP disconnection.

Bulk transfer server: When the bulk transfer server receives a request from a client, it calls the slow-lane API at the network controller via a JSON message. An example is: {hello: jukebox, type: bulk, nwsrc: `10.10.7.31/32`, nwdst: `10.10.5.18/32`, proto: 6, sprt: 24380, dprt: 20, len: 1800000, deadline: 3600}. In this case the server requests a bulk transfer of 1.8GB by TCP on the path from `10.10.7.31:24380` to `10.10.5.18:20`. The deadline parameter indicates that the transfer can take up to 1 hour. If the controller accepts the request, the flow is given a dedicated queue, whose weight is adjusted periodically as described earlier.

Wireless APs: We used standard TP-LINK WR1043ND APs, which were run in layer-2 mode (i.e. routing, DHCP,

TABLE I
VIDEO, BROWSING, AND FTP PERFORMANCE WITH VARYING α.

App	$\alpha = 0$		$\alpha = 0.8$		$\alpha = 1$	
	mean	std	mean	std	mean	std
C1 MOS	2.87	0.44	3.10	0.31	3.25	0.01
C2 MOS	3.25	0.00	3.25	0.01	3.25	0.01
Page load (s)	2.84	0.86	3.10	1.61	4.85	3.55
FTP stretch	1.60	0.20	1.97	0.77	2.45	1.07

and NAT disabled) with dd-wrt v24.

User clients: Each home has three clients, implemented using standard computers. Client C1 represents a large-screen device (e.g. PC or TV) and client C2 a small-screen device (e.g. tablet/phone) on which users watch videos, while client C3 represents a PC or media gateway that does both web-browsing and bulk transfers. Browsing took place within Internet Explorer (IE) v10, and a web-page of 1.1 MB containing text and images is accessed. All videos were played by the VLC IE plugin.

User Traffic: Clients run PowerShell scripts to automatically generate traffic representative of the average home. Clients C1 and C2 are either idle or streaming video, and a Markov process controls the transitions, as in [20], with 40% of time spent idle and 60% watching video. Client C1 streams a high bandwidth video in MPEG-1/2 format, allocated a peak bandwidth of 7.5 Mbps, and having mean rate of 5.9 Mbps averaged over 3-second interval samples. Client C2 streams a lower bandwidth video in MPEG-4v format, allocated a peak bandwidth of 2.1 Mbps and having a mean rate of 1.3 Mbps. Client C3 can be in idle, browsing, or bulk-transfer states. For browsing it opens IE and loads a 1.1 MB web-page from our web-server. The user is assumed to read the web-page for 10 seconds, reloads the web-page, and the process repeats. We disabled IE's cache so that it downloaded the full web page on every access, which lets us compare the download times for the page across various runs. For bulk-transfers the file sizes were chosen from a Pareto distribution with shape parameter 4.5, and scale parameter such that files are between 100 and 500 MB with high probability. The idle periods are log-normal with mean 10 minutes and standard deviation 2 minutes.

Metrics: The video streaming quality is measured in terms of Mean Opinion Scores (MOS). To automatically evaluate MOS, we rely on the technique of [21] that combines initial buffering time, mean rebuffering duration, and rebuffering frequency to estimate the MOS (with a configured playback buffer of 3 seconds). Our VLC IE plugin was instrumented with Javascript to measure these parameters and compute the MOS. Our client script also measured the times taken for each bulk transfer and web-page download.

B. Experimental Results

We conducted tens of experiments varying the user selected parameter α that controls the extent to avail special lanes, and the elasticity β for bulk transfer flows. The impact of these parameters on video quality, file transfer times, and browsing delays is discussed next.

In Table I, we show how the quality for the various applications depends on the fraction α of household link capacity that is made available by the user. The low-rate video (2.1 Mbps peak) on client C2 always gets a near-perfect MOS of 3.25. This is unsurprising, since a fair share of the link capacity suffices for this video to perform well in our experiments, and fast-lane reservations are not necessary. The high-rate video stream (7.5 Mbps peak) on client C1 however sees marked variation in quality: disabling special lanes with $\alpha = 0$ makes the video unwatchable most of the time, with low average MOS of 2.87 (standard deviation 0.44), while complete fast-lane provisioning with $\alpha = 1$ always successfully allocates bandwidth to this stream, yielding a perfect MOS of 3.25. With $\alpha = 0.8$, the average MOS degrades to 3.10 (standard deviation 0.31) since allocations fail when the other video stream is also active.

The table also shows that α has the converse effect on web-page load time: when $\alpha = 0$, the web-page loads in 2.84s on average (standard deviation 0.86s), while increasing α to 0.8 and 1 steadily increases the average time taken for page loads; furthermore, the standard deviation also increases, indicating that download times become more erratic as α increases. This is not surprising, since web-page downloads (and mice flows in general) will not allocate resources via the API call, and their performance suffers when bandwidth is allocated to other reserved flows. This trade-off between video quality and web-page load-time illustrates that users should adjust their household α value (via trial-and-error or other means beyond the scope of this paper) in line with their traffic mix and the relative value they place on each traffic type.

The performance of a bulk transfer flow is measured in terms of its "stretch", i.e. the factor by which its transfer delay gets elongated compared to the baseline case where it has exclusive access to the entire access link capacity. Table I shows that with no special lanes, bulk transfer flows get stretched by a factor of 1.6, and the stretch increases to 1.97 at $\alpha = 0.8$ and 2.45 at $\alpha = 1$. This is both expected and desired, since increasing α allows the video streams to get higher quality, which comes at the cost of stretching the elastic bulk-transfers.

VI. RELATED WORK

The body of literature on QoS/QoE is vast, and *bandwidth-on-demand* capabilities have been envisaged since the days of ATM, IntServ and RSVP. These mechanisms equip the ISP with tools to manage quality in their own network, but little has been done by way of exposing controls to end-users and content providers. Early attempts at exposing QoS to external entities include the concept of bandwidth broker for ATM networks [22], and protocols for QoS negotiation (e.g. XNRP [23]). Tools for exposing network bandwidth availability are starting to emerge, though predominantly for data center users, such as Juniper's Bandwidth Calendaring Application [24] implemented over an OpenFlow-based network. Bandwidth-on-demand for bulk data transfers between data centers has also been explored in the Globally Reconfigurable Intelligent Photonic Network [25] and NetStitcher [26], with the latter exploiting the elasticity in bulk data transfer to schedule it during diurnal lulls in network demand. Elasticity has also been leveraged by [27] to improve ISP access network performance, with time-dependent pricing explored in [28].

ISPs can also provide class-based QoS support via frameworks like MEF Carrier Ethernet interfaces [19], which supports multiple classes of service over Ethernet Virtual Connections. However, these data-plane interfaces (which mark packets) are only accessible to entities directly connected to the ISP (or a set of contiguous Operator Metro Ethernet Networks, in the case of Operator Virtual Connections) and so are unsuitable when the ISP/Operator-MEN and CP do not peer directly, and would require the CP to be able to identify users by their MAC address if it is to differentiate service for/between its users. Carrier Ethernet also differentiates traffic on the basis of SLAs, which are unlikely to be as agile as flow based mechanisms, e.g. operating on the time scale of days rather than sessions, preventing a CP from differentiating flows of the same form, e.g. movie vs advertising video. A Carrier might use the IEEE 802.1Q Stream Reservation Protocol [29] to implement traffic control within its own network, though it remains unclear how this mechanism would be exposed to the CP.

Greenstein describes [30] how recent US regulatory changes have allowed ISPs to provide fast lane services, but that an impediment to such services is user demand for options to control such prioritisation. Our approach gives users direct control of the fraction of the access link capacity that can be used to carve out fast-lanes through the parameter α. Google's Bandwidth Enforcer [31] is another system that uses SDN techniques to manage bandwidth allocations across a network. The weight that it gives to best-effort traffic performs a role that is similar to our elasticity parameter β. While it identifies such bulk transfer parameters and dimensions network resources in a globally max-min fair manner, it is designed for use in a private WAN, and so does not address user (receiver) incentives for allowing slow lanes, and so has no equivalent of our user-controlled parameter α.

The works closest to ours are those that virtualize the access [32] and home [33], [34] networks. NANDO [32] allows multiple ISPs to share infrastructure, and consumers can choose the ISP on a per-service basis. This model is very attractive for public access infrastructure (e.g. in Australia or Singapore), but it remains to be seen if private ISPs will be willing to share infrastructure with each other. In [33], the home network is sliced by the ISP amongst multiple content providers. With this approach the ISP cedes long-term control of the slice to the CP (it is however unclear what policies dictate the bandwidth sharing amongst the slices), which is different from our architecture in which the ISP only "leases" well-specified resources to the CP on a short-term per-flow basis. Both models have merits and are worth exploring, though we believe our approach is likely to be more palatable to ISPs as they can retain more control over their network. Lastly, [34] gives users control of how their home network is sliced, but requires a higher level of user sophistication that we have tried to bypass.

VII. Conclusions

In this paper we have proposed an architecture for fast- and slow-lanes in the access network that can be invoked by an external entity via open APIs. Our architecture provides the motivation and means for all parties to engage: content providers can selectively choose to avail fast or slow lanes for flows in line with their business models; ISPs can monetize their access infrastructure resources on a per-flow basis rather than relying on bulk-billed peering arrangements; and users can readily adjust the degree of (or opt out of) special lanes provisioning to suit their usage pattern. We developed a mechanism that achieves efficient creation of special lanes via SDN-based centralized control. We simulated our algorithm on real traffic traces comprising over 10 million flows to show that fast lanes can almost eliminate video quality degradations and slow lanes enhance web page-load time significantly for a modest increase in bulk transfer delays. Finally, we prototyped our scheme on a small testbed comprising OpenFlow-compliant switches, off-the-shelf access points, and unmodified clients, to show how the user can control the trade-off between video experience, bulk transfer rates, and web-page load-times.

Our work is a first step towards showing how the agility and centralization afforded by SDN technology presents a unique opportunity to overcome the long-standing impasse on service quality in access networks. Needless to say, many challenges are yet to be overcome to make this a reality, such as enriching the API to include other application use-cases (e.g. low-latency gaming or virtual reality applications), extending the API end-to-end across network domains via federation, and ultimately developing appropriate pricing models that can derive economic benefits for ISPs, CPs, and end-users.

VIII. Acknowledgements

The authors thank Google for supporting this project via a Research Award, and specifically Josh Bailey who has been a close collaborator throughout this work.

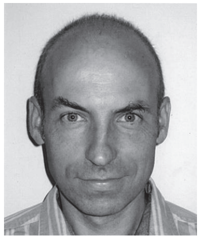
References

[1] V. Sivaraman, T. Moors, H. Habibi Gharakheili, D. Ong, J. Matthews, and C. Russell. Virtualizing the Access Network via Open APIs. In *Proc. ACM ConNEXT*, CoNEXT '13, December 2013.
[2] Cisco Internet Business Solutions Group. Moving Toward Usage-Based Pricing. http://goo.gl/QMEQs, 2012.
[3] The European Telecom. Network Operators' Association. ITRs Proposal to Address New Internet Ecosystem. http://goo.gl/VutcF, 2012.
[4] M. Nicosia, R. Klemann, K. Griffin, S. Taylor, B. Demuth, J. Defour, R. Medcalf, T. Renger, and P. Datta. Rethinking flat rate pricing for broadband services. White Paper, Cisco Internet Business Solutions Group, July 2012.
[5] Sandvine. Global Internet Phenomena Report. http://goo.gl/l7bU2, 2012.
[6] 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.
[7] S. Krishnan and R. Sitaraman. Video Stream Quality Impacts Viewer Behavior: Inferring Causality Using Quasi-Experimental Designs. In *Proc. ACM IMC*, November 2012.
[8] X. Liu, F. Dobrian, H. Milner, J. Jiang, V. Sekar, I. Stoica, and H. Zhang. A Case for a Coordinated Internet Video Control Plane. In *Proc. ACM SIGCOMM*, August 2012.
[9] A. Rao, Y. Lim, C. Barakat, A. Legout, D. Towsley, and W. Dabbous. Network characteristics of video streaming traffic. In *Proc. ACM CoNEXT*, December 2011.

[10] S. Akhshabi, A. Begen, and C. Dovrolis. An Experimental evaluation of rate-adaptation algorithms in adaptive streaming over HTTP. In *Proc. ACM MMSys*, February 2011.
[11] M. Ghobadi, Y. Cheng, A. Jain, and M. Mathis. Trickle: Rate Limiting YouTube Video Streaming. In *Proc. USENIX ATC*, June 2012.
[12] 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.
[13] Internet Society. Bandwidth Management: Internet Society Technology Roundtable Series. http://goo.gl/ZMOTx, Nov 2012.
[14] 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)*, April 2015.
[15] H. Habibi Gharakheili, A. Vishwanath, and V. Sivaraman. An Economic Model for a New Broadband Ecosystem Based on Fast and Slow Lanes. *IEEE Network*, 30(2):26–31, 2016.
[16] HTTP Archive. http://www.httparchive.org/.
[17] S. Ramachandran. Web metrics: Size and number of resources. http://goo.gl/q4O4X, 2010.
[18] 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.
[19] MEF Forum. Implementation Agreement MEF 23.1: Carrier Ethernet Class of Service, Phase 2, January 2012.
[20] X. Cheng, C. Dale, and J. Liu. Statistics and Social Network of YouTube Videos. In *Proc. IEEE/ACM IWQoS*, June 2008.
[21] 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.
[22] K. Nahrstedt and J. M. Smith. The QoS Broker. *IEEE Multimedia*, 2:53–67, 1995.
[23] K. Rothermel, G. Dermler, and W. Fiederer. QoS negotiation and resource reservation for distributed multimedia applications. In *Proc. IEEE International Conference on Multimedia Computing and Systems*, June 1997.
[24] H. Sugiyama. Programmable Network Systems Through the Junos SDK and Junos Space SDK. In *World Telecommunications Congress*, 2012.
[25] A. Mahimkar, A. Chiu, R. Doverspike, M. Feuer, P. Magill, E. Mavrogiorgis, J. Pastor, S. Woodward, and J. Yates. Bandwidth on Demand for Inter-Data Center Communication. In *Proc. ACM HotNets Workshop*, November 2011.
[26] N. Laoutaris, M. Sirivianos, X. Yang, and P. Rodriguez. Inter-Datacenter Bulk Transfers with NetStitcher. In *Proc. ACM SIGCOMM*, August 2011.
[27] P. Danphitsanuphan. Dynamic Bandwidth Shaping Algorithm for Internet Traffic Sharing Environments. In *Proc. World Congress on Engineering*, July 2011.
[28] C. Joe-Wong, S. Ha, and M. Chiang. Time-dependent broadband pricing: feasibility and benefits. In *Proc. IEEE ICDCS*, June 2011.
[29] IEEE. IEEE Standard for Local and metropolitan area networks – Bridges and Bridged Networks. *IEEE Std 802.1Q-2014*, 2014.
[30] S. Greenstein. The fault lines along fast lanes. *IEEE Micro*, 34(2):64–64, Mar 2014.
[31] A. Kumar et al. BwE: Flexible, Hierarchical Bandwidth Allocation for WAN Distributed Computing. In *Proc. ACM SIGCOMM*, August 2015.
[32] J. Matias, E. Jacob, N. Katti, and J. Astorga. Towards Neutrality in Access Networks: A NANDO Deployment With OpenFlow. In *Proc. Int'l Conf. on Access Networks*, June 2011.
[33] Y. Yiakoumis, K. Yap, S. Katti, G. Parulkar, and N. McKeown. Slicing Home Networks. In *Proc. SIGCOMM HomeNets Workshop*, August 2011.
[34] Y. Yiakoumis, S. Katti, T. Huang, N. McKeown, K. Yap, and R. Johari. Putting Home Users in Charge of their Network. In *Proc. ACM UbiComp*, September 2012.

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.



Tim Moors is a Senior Lecturer in the School of Electrical Engineering and Telecommunications at the University of New South Wales, in Sydney, Australia. He researches transport protocols for wireless and optical networks, wireless LAN MAC protocols that support bursty voice streams, communication system modularity, and fundamental principles of networking. Previously, he was with the Center for Advanced Technology in Telecommunications at Polytechnic University in New York, and prior to that, with the Communications Division of the Australian Defence Science and Technology Organisation. He received his PhD and BEng(Hons) degrees from universities in Western Australia (Curtin and UWA).

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.

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.