# FlowQoS: Per-Flow Quality of Service for Broadband Access Networks

M. Said Seddiki[*†‡], Muhammad Shahbaz[*], Sean Donovan[*], Sarthak Grover[*], Miseon Park[*], Nick Feamster[*], Ye-Qiong Song[†]
[*] Georgia Tech, [†] Loria, Lorraine University, [‡] Sup'Com, University of Carthage

## Abstract

In broadband access networks, one application may compete for the bandwidth of other applications, thus degrading overall performance. One solution to this problem is to allocate bandwidth to competing flows based on the application type at the gateway of the home network. Unfortunately, application-based quality of service (QoS) on a home network gateway faces significant constraints, as commodity home routers are not typically powerful enough to perform application classification, and many home users are not savvy enough to configure QoS parameters. This paper describes FlowQoS, an SDN-based approach for application-based bandwidth allocation where users can allocate upstream and downstream bandwidths for different applications at a high level, offloading application identification to an SDN controller that dynamically installs traffic shaping rules for application flows. FlowQoS has two modules: a flow classifier and an SDN-based rate limiter. We design a custom DNS-based classifier to identify different applications that run over common web ports; a second classifier performs lightweight packet inspection to classify non-HTTP traffic flows. We implement FlowQoS on OpenWrt and demonstrate that it can improve the performance of both adaptive video streaming and VoIP in the presence of active competing traffic.

## 1 Introduction

Broadband Internet access is proliferating in settings where upstream and downstream throughput may be limited, ranging from home networks to small offices to developing countries. In these settings, the traffic of bandwidth-intensive applications (*e.g.*, video streaming) and interactive applications (*e.g.*, VoIP, gaming) compete for relatively scarce bandwidth resources. A common approach for dealing with limited throughput is to configure the network routers to prioritize some applications' traffic flows over others, effectively enforcing Quality of Service (QoS). Many QoS mechanisms have been proposed, standardized and implemented [2, 11, 18, 20, 22, 25], but these mechanisms have not seen deployment in broadband access networks, for several reasons. First, many home routers have limited memory and processing resources and may not be able to perform application classification "on-the-fly" [3]; second, most of these devices cannot be easily configured to perform QoS functions because the mechanisms for doing so are complicated and obtuse, not based on specific application, devices, or users.

One approach to deploying QoS in broadband access networks is to delegate the functions that perform QoS both application identification and router-level configuration to separate control logic, which permits a user to configure QoS policies at higher levels of abstraction (*e.g.*, per application) and installs the results of the QoS configuration into the home router in the form of flow-table entries. The emergence of *software-defined networking (SDN)* and, specifically, the OpenFlow specification [19] makes such a refactoring possible. Such control software could run directly on the router itself as a separate program (in the case where the router is powerful enough), or on a separate device, either inside the home or even from a remote location.

We present FlowQoS, a system for performing per-flow application-based QoS by delegating application identification and QoS configuration to an SDN controller. In FlowQoS, the user of the broadband access network simply specifies the high-level applications that should have higher priority (*e.g.*, video streaming, VoIP), and the FlowQoS controller performs the appropriate application identification and QoS configuration for both upstream and downstream traffic to implement the user's preferences. FlowQoS identifies application types for each flow in real time and installs rules in the data plane that forward individual flows according to user-specified priorities for those applications. To do so, FlowQoS creates links in a virtual topology in the home router, configures each of these links with a particular rate, and assigns flows to these links on-the-fly, as flows are mapped to corresponding applications (and priorities).

FlowQoS makes it easier for a typical user to configure priorities and facilitates more sophisticated per-flow application-based QoS, but doing so imposes its own set of challenges. In particular, if traffic classification must occur at a remote location (in cases where the processing power on the home gateway is not sufficient to perform classification on-the-fly), latency for performing traffic classification might be higher. We design a fast, lightweight application classification algorithm that is based on the DNS lookups that each application issues. This classification algorithm performs early identification of the application type for each flow—often before the first packet of the flow is even sent. Second, FlowQoS performs "lazy" enforcement of QoS parameters, forwarding the first packets of each flow uninhibited before ultimately installing the appropriate QoS rules; this approach (which has also been applied before in IP backbone networks [22]) lets application traffic realize lower latencies than they would otherwise, at the expense of a lack of QoS for the first few packets of each flow. Our evaluation shows that this tradeoff still allows preferred flows to achieve good performance in the face of competing traffic.

This paper presents several contributions. First, we present details about the design of the different components, the implementation, and the evaluation of FlowQoS, which allows users to specify priorities for different application flows. Our design ensures that both application identification and QoS configuration can be offloaded to a remote third-party controller without adversely affecting application performance. Second, we introduce a new mechanism for performing lightweight application classification based on DNS lookups; this mechanism can perform early application classification for flows and also works for encrypted traffic (*e.g.*, HTTPS). In previous work [30] we have presented the high level architecture of FlowQoS. This paper presents details description and evaluation of our approach.

The rest of the paper proceeds as follows. Section 2 presents related work in QoS, as well as SDN-based solutions for home and broadband access networks. Section 3 describes the design of FlowQoS, as well as its implementation using OpenWrt and Open vSwitch. Section 4 evaluates FlowQoS for video streaming and VoIP applications in the context of competing flows. Section 5 discusses future work and open research avenues, and Section 6 concludes.

## 2 Related Work

Although there is significant previous work for QoS in IP networks [2, 18, 22], traffic shapers [20, 25], and methods for performing application identification to quickly classify IP traffic flows into the appropriate traffic class [13, 29], we focus in particular on making per-flow, application-based QoS easy to deploy and configure in home networks, where networking devices have less processing power than typical networking devices and the users are not skilled network operators. SDN's flexibility has engendered a resurgence of work on QoS, particularly as SDN test beds begin to provide more support for QoS functions [31].

Kim *et al.* presented a solution most closely related to FlowQoS, albeit with a different approach [15]. The approach sets rate limiters at the edge switches and priority queues for flow at each path hop and uses a QoS control framework for automated fine-grained management of OpenFlow networks with multiple switches. FlowQoS provides similar automated traffic shaping but does so at a *single* gateway (*e.g.*, in a home network), while offloading application identification to the controller. Ishimori *et al.* developed QoSFlow [12], a system for providing QoS in OpenFlow networks. The traffic shaping is similar to FlowQoS's shaping mechanism, but the system does not focus on providing usable QoS for broadband access networks. The QoSFlow prototype is still under development, and the system has not been evaluated. Ko *et al.* proposed a two-tier flow-based QoS management framework [16] that divides the flow table into three tables: one for the flow state, the second for the forwarding rules, and the third for the QoS rules where multiple micro-flows can share a single QoS profile. This system requires multicore processors and was not designed for home networks. Ferguson *et al.* developed PANE, a system that allows a user to reserve guaranteed minimum bandwidth between two hosts [8]; PANE addresses on a much broader set of network configuration problems (*e.g.*, access control, path configuration) and does not focus specifically on QoS in broadband access networks or application identification. Williams *et al.* [36] developed an automated IP traffic classification algorithm based on statistical flow properties and evaluated its performance on home gateways; this approach limited the throughout of commodity home routers to 28 Mbps. In contrast, FlowQoS faces no such limitations. Risso *et al.* [28] developed an OpenFlow-based mechanism for customizing data-plane processing in home routers, but the architecture is focused on more general data-plane modifications, not QoS.

Several other approaches explore QoS in home networks. Yiakoumis *et al.* proposed letting users notify the ISP about their bandwidth needs for a given application; in this case, provisioning occurs in the ISP's last mile, not in the home [37]. Georgopoulos *et al.* proposed an OpenFlow-assisted framework that improves users' quality of experience (QoE) in home networks for multimedia flows, subject to fairness constraints [10]. The system allocates resources to each device but does not perform per-application or per-flow QoS. Mortier *et al.* developed Homework, a home networking platform that provides per-flow measurement and management capabilities for home networks [21]. Homework allows users to monitor and control per-device and per-protocol usage, but it does not provide QoS support or perform any application classification.

Carbone *et al.* developed a port of Dummynet for OpenWrt [4]; the emulator allows configuration of various links for traffic shaping, and could be used as FlowQoS's data-plane traffic shaper. The system does not perform automated application identification or an interface for defining QoS based on application, but FlowQoS's mechanisms for establishing mappings between applications and traffic classes could be used as an interface to Carbone's lower-level Dummynet-based mechanisms.

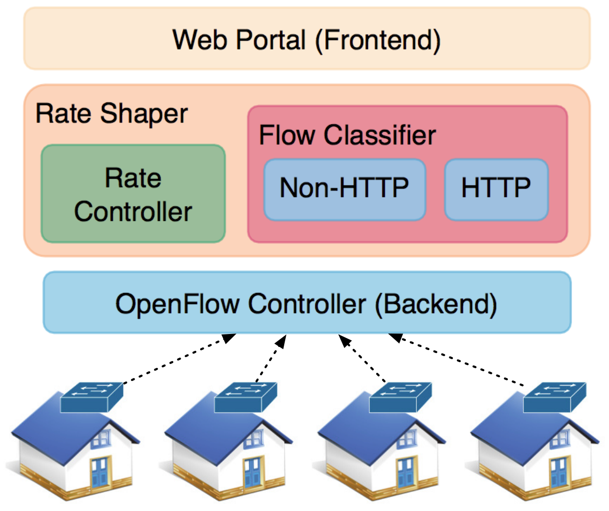


**Figure 1:** *FlowQoS architecture. Although this diagram shows the control architecture sitting outside the home (which is how our prototype is implemented), each router could run its own controller (and front-end portal).*

## 3 FlowQoS

In this section, we describe the design and implementation of FlowQoS. We present an overview of the design and then describe the system components in detail.

### 3.1 Overview

Figure 1 shows the high-level architecture of FlowQoS. Users configure priorities for specific high-level applications from the edge network (*e.g.*, the home network) using a Web portal. Users specify the percentage of bandwidth they want to allocate to each application. We consider the configuration a separate problem. The output from the portal is a configuration file that the rate shaper uses to ultimately implement the respective controls for shaping application traffic. The two main components of the rate shaper are an SDN-based rate controller and a flow classifier. The flow classifier has two modules: one that can perform early application identification of HTTP and HTTPS traffic using DNS information, and a second that performs application identification for other flows.

When the first packet of a flow arrives at the switch, a copy of this packet is forwarded to the controller. The switch continues to perform default forwarding of the traffic flows until application identification has been performed, essentially "lazily" implementing QoS only once application classification is complete. The controller determines which classifier should classify the application type for the flow. Each application type is associated with a different queue, each of which is shaped according to the traffic shaping policy for that application class. Section 3.3 describes this mechanism in more detail.

### 3.2 Flow Classifier

The flow classifier maintains a lookup table where the key is a flow's five-tuple (*e.g.*, source IP address, destination IP address, protocol, source port, and destination port). When a sender initiates a new flow, the switch sends a copy of the first packet of the flow to the controller. The flow classifier then checks whether the flow's tuple correspond to an entry in this lookup table. The lookup then returns the type of application, such as video, VoIP, P2P, gaming, or web.

FlowQoS uses two different modules to classify traffic. FlowQoS classifies application traffic on ports 80 and 443 (*i.e.*, HTTP and HTTPS, respectively) using the DNS classifier, which performs more fine-grained classification; many classifiers would otherwise classify many types of application traffic on these ports simply as "web traffic". We classify most other traffic using `libprotoident` [1]. When a new flow arrives at the switch, the five-tuple in the first packet can identify the flow, which is sufficient for the DNS-based classifier. Invoking `libprotoident` requires knowing an additional four fields: the first four bytes sent, first four bytes received, first payload size sent, and first payload size received. The requirement for parsing these additional fields might require the controller to see additional few packets before processing the flow, but because QoS enforcement is lazy (*i.e.*, before a flow is classified, its are forwarded on the default queue), the requirement to see additional packets is not prohibitive. We chose to use `libprotoident` for application identification because it has higher accuracy compared to other DPI-based solutions; previous work shows that `libprotoident` was able to properly classify 94% of 1,262,022 flows captured over 66 days [5].

FlowQoS's DNS-based classifier maintains a table that it builds using the DNS responses that the switch forwards. The table includes the A or CNAME record, the corresponding IP address (in the case of an A record response), and the time-to-live for the record. To classify flows based on this information, the classifier checks the A or CNAME record against a list of regular expressions, each of which corresponds to some application. Because the sender initiates a DNS request before the corresponding TCP connection, the classifier can associate a flow with an application before the sender even sends the first packet of the flow. In the case of HTTP or HTTPS flows, the controller can proactively install a rule based on the source IP of the sender and the destination IP and port where the sender will initiate a connection. FlowQoS's per-flow classifier can also prioritize distinct flows within the same Web page. For instance, most video streaming providers use well-known content distribution networks (CDNs) that are distinct from the site's primary domain. In the case of Youtube, for example playing a video loads HTML content, advertisements, and the video stream itself, each from a different domain.

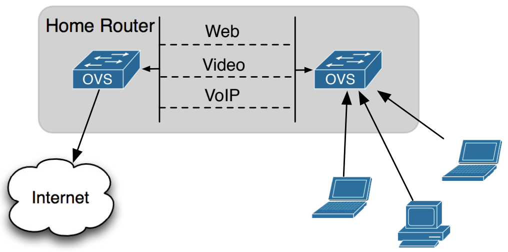


**Figure 2:** *The virtual switch topology that performs traffic shaping inside the home router.*

## 3.3 SDN-Based Rate Controller

Based on the flow classifier's determination of the associations between flows and applications, FlowQoS's SDN-based rate controller assigns each flow to the appropriate rate. In existing systems, assigning a priority to each traffic flow according to user configuration is complicated by the limitations of today's home routers: Existing home routers do not support per-flow rate control; existing mechanisms such as `tc` still require configuring virtual interfaces or tagging via `iptables`. Even the Open vSwitch (OVS) implementation for OpenWrt, does not yet support the parts of the OpenFlow 1.3 specification for per-flow QoS.

To overcome these limitations, FlowQoS enables per-flow QoS by instantiating a two-switch virtual topology on the home router, as shown in Figure 2. Each virtual link between the two switches corresponds to a different application group (*e.g.*, video, web, gaming). To implement rate limiting, each link has a traffic shaper (implemented with Linux's `tc` utility) that corresponds to the user-specified rate. When a new flow arrives at the switch, it is sent to the appropriate flow classifier. Based on the results from classification, the controller installs OpenFlow rules into the Open vSwitch components such that the new flow is forwarded on the virtual links with the appropriate shaping parameters. Open vSwitch's QoS rate-Limiting feature uses Linux `tc` capability for rate shaping. [1]

To perform rate limiting once the classifier has identified the application type for a flow, the switch refers to its existing rules to determine which inter-switch connection corresponds to that traffic class. Only flows that need to be rate-limited will be categorized in a different manner. The rate controller also handles installation of other rules, such as the rules associated with DNS responses. A copy of the initial packets for each new flow are forwarded to the controller while default forwarding takes place; default forwarding continues until application classification potentially overrides the default settings. Once a rule that forwards the packets to the appropriate inter-switch link is created, the controller will not receive any more frames from that particular flow.

FlowQoS can place flows belonging to different applications into queues and priorities that the user configures. In the case of video traffic, FlowQoS places each flow in a different queue and shares the available bandwidth for video among competing applications; in the case of competing video streams, this strategy results in fewer bitrate oscillations and better overall fairness. To ensure that the access link is fully utilized, FlowQoS monitors the links for active traffic flows and reconfigures traffic shapers for the active queues to ensure complete utilization of available capacity.
Our prototype provides static and dedicated bandwidth allocation to every application. Due to the limitation of the current implementation of OVS we were not able to perform dynamic allocation based on one scheduler. We are currently working on how we can offer full bandwidth utilization to the active flows.

## 4 Evaluation

We now present the evaluation of FlowQoS. We describe the experimental setup and results of FlowQoS in terms of both application performance and CPU and memory requirements.

### 4.1 Experiment Setup

We used an OpenWrt-based router for our prototype implementation of FlowQoS. We integrated Open vSwitch (OVS) with OpenWrt to enable the control of an OpenWrt switch using OpenFlow. We used a Raspberry Pi [33] for the controller hardware, and implemented the control application on top of Pox [26], a popular open-source OpenFlow controller. We have released the source code for FlowQoS [9].

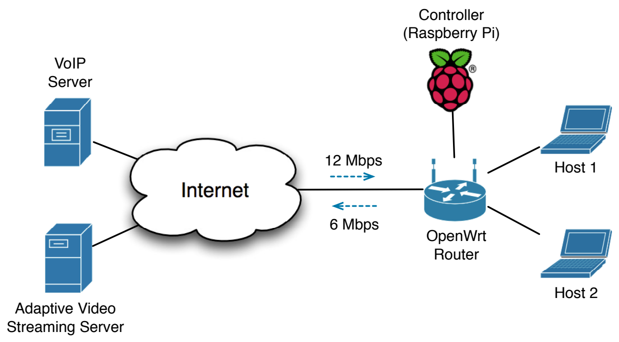


**Figure 3:** *Experiment setup.*

Figure 3 shows the experiment setup. We configured an Internet connection to be 12 Mbps downstream and 6 Mbps upstream. We allocated 7 Mbps for video, 3 Mbps for VoIP, and 2 Mbps for web applications and other traffic. Before application classification takes place, the switch forwards traffic to according to the web application traffic, which is the default traffic class.

[1]Configuring `tc` on virtual inter-switch links is conceptually similar to what OF-config [23] might enable if Open vSwitch supported the flow-based QoS functions outlined in OpenFlow 1.3. Our dual-switch topology is a workaround for the limitations of the Open vSwitch implementation.

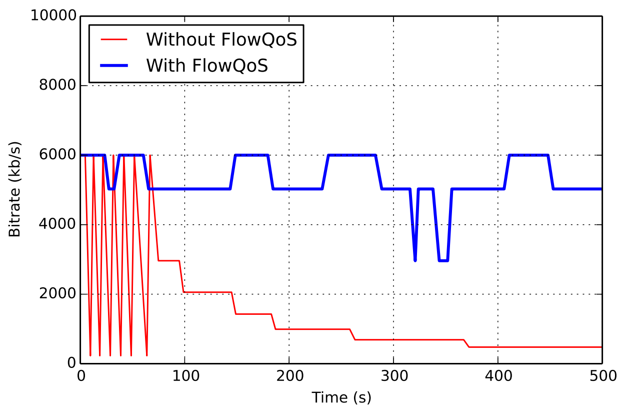


**Figure 4:** *Comparison of video bitrate oscillations for adaptive video streaming, with and without FlowQoS. FlowQoS allows the video streaming player to more quickly converge to a higher bitrate.*

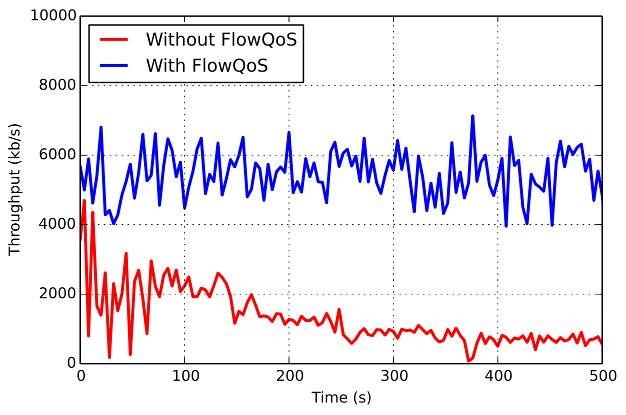


**Figure 5:** *Performance of HTTP-based adaptive video streaming with and without FlowQoS.*

During the experiments, host 1 is either watching a video or making a VoIP call, depending on the experiment; host 2 generates background traffic by downloading a large file. FlowQoS segregates the traffic flows accordingly, sending the respective traffic flows along the corresponding paths between the two OVS switches to enforce the appropriate traffic control, using flow table rules to forward traffic through the appropriate rate shapers.

## 4.2 Results

We show how FlowQoS improves the performance of both adaptive video streaming and VoIP in the face of competing traffic. We also evaluate the CPU and memory requirements of FlowQoS compared to conventional traffic shapers.

### 4.2.1 Improvements to application performance

**Adaptive video streaming.** We first evaluated FlowQoS's ability to apply per-flow application-based QoS to improve the performance of adaptive video streaming, which relies on TCP to determine the available bandwidth and video streaming rate. We performed these experiments using the DASH dataset [17], using the Big Buck Bunny benchmark video with a four-second segment length at ten different bitrates with Dash-JS player.

Figure 4 illustrates the performance in terms of bitrates of the benchmark video. The results show that FlowQoS allows the system to quickly converge to a higher bitrate than it otherwise would without FlowQoS enabled. This prevents bitrate oscillations and ensures the stability of the adaptive video player in terms of requested bitrates and video quality. Thus, FlowQoS improves the quality of the adaptive streaming video by both reducing bitrate oscillation and achieving a higher overall bitrate. Figure 5 shows the performance in terms of video throughput for the DASH dataset; we computed moving averages using a four-second window. Even if the user downloads many files, as is the case with typical browsing, using FlowQoS will improve the performance of the adaptive video streaming, since the control application will forward web traffic over a separate virtual link than the virtual link dedicated to video traffic.

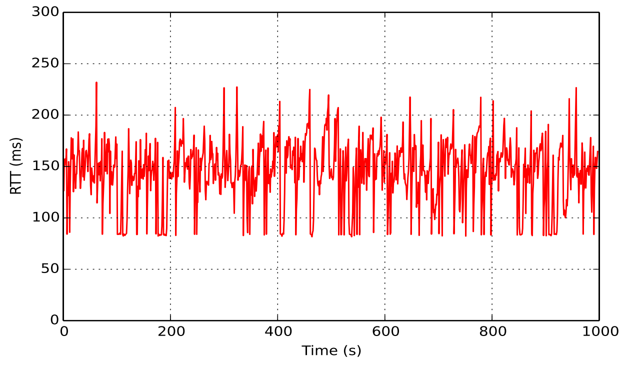
Figure 5 shows the evolution of instantaneous throughput for an adaptive streaming video application with and without FlowQoS. The results show that FlowQoS prevents the client from switching to experiencing lower instantaneous throughput values (and thus prevents the client from switching to lower bitrate).

**VoIP.** We also evaluated FlowQoS in the context of VoIP application traffic. VoIP is sensitive to delay and variation in packet arrival times, so lower jitter is essential for good performance. We monitor the packet delay and jitter of the VoIP application using `ping` and `iperf` to monitor the RTT and the packet arrival times throughout the experiment. Figure 6 shows the round trip latency and jitter of the resulting application traffic when host 2 generates background traffic and host 1 makes a VoIP call over the Internet connection.

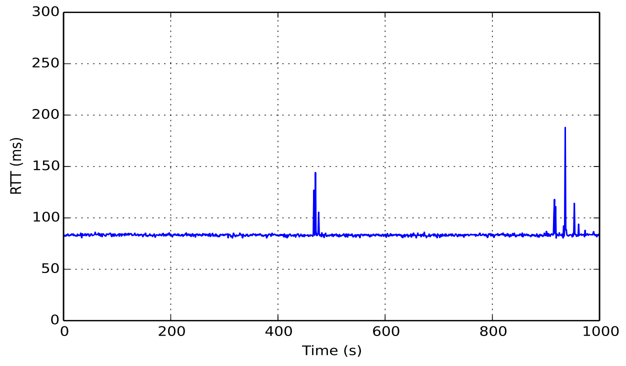
Figure 6 shows the delay and jitter of the VoIP application in the face of competing cross traffic for 1,000 seconds. The evaluation shows that FlowQoS can provide delay guarantees and reduce jitter for these types of applications. We show jitter separately because VoIP calls are quite sensitive to high jitter, which is exhibited in the case without FlowQoS. FlowQoS maintains the strict delay and jitter requirements for VoIP when there is active competing background traffic.

### 4.2.2 Classification delays

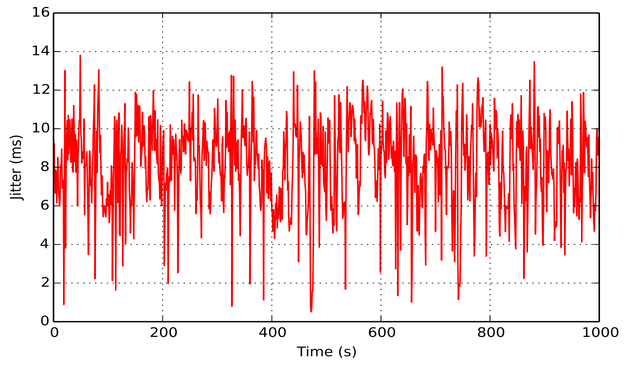
We evaluated the average classification delays of both scenarios of deployment of FlowQoS. Figure 7 illustrates how long it takes the FlowQoS classifier to identify traffic when it is running on a local controller (Raspberry PI) and on a distant controller (Server). The Raspberry Pi has a 700 MHz ARM CPU and 512 MB of RAM. The server has a 2.4GHz Intel Core 2 CPU and 8 GB of RAM. This delay is equal to



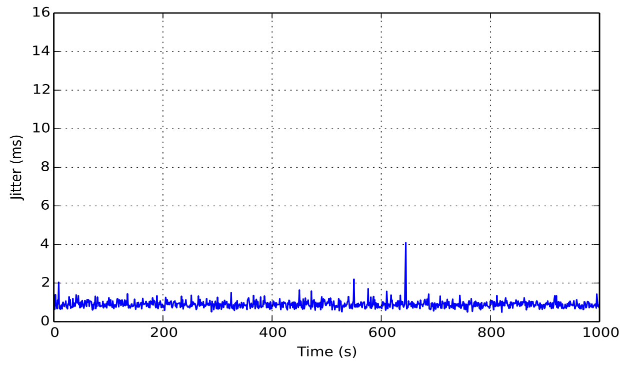
(a) Round-trip latency without FlowQoS.



(b) Round-trip latency with FlowQoS.



(c) Jitter without FlowQoS.



(d) Jitter with FlowQoS.

**Figure 6:** *The performance of VoIP with and without FlowQoS.*

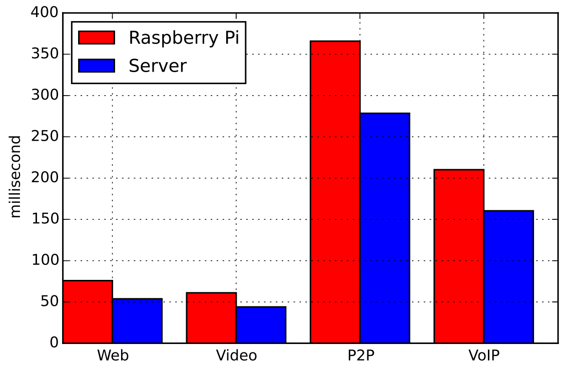


**Figure 7:** *Comparison of classification delays.*

the time to create an entry for an identified flow when the controller deployed locally next to the home gateway. When the controller is deployed by the ISP at the last mile hop, an additional delay should be added that is equal to two times the last-mile hop latency. The additional delay is varying from about 20 ms to nearly 80 ms according to the measurements conducted in previous work [32]. During our experiments, only 2.34% of over 20,000 flows were unclassified (and thus forwarded over the default virtual link).

### 4.2.3 CPU and memory requirements

To quantify the overhead of running a two-switch virtual network over either a single Open vSwitch instance or a simple traffic shaping script, we compared the average CPU and memory of an unmodified OpenWrt and the dual OVS topology connected by a single link. Every 30 seconds, the user requests the video from another server with a different IP address. We compare the results in the presence of one user streaming YouTube videos. Table 1 illustrates that the dual OVS topology adds only minimal CPU and memory overhead compared to the baseline installation of OpenWrt.

| | CPU usage | Memory usage |
|---|---|---|
| Baseline (unmodified OpenWrt) | 11.34% | 43.03MB (33.62%) |
| Dual OVS | 16.78% | 52.77MB (41.23%) |

**Table 1:** *CPU and memory consumption for an unmodified OpenWrt and the dual OVS topology.*

We run again the same scenario to compare FlowQoS and a QoS traffic shaping script . The script performs per-flow traffic shaping. It aims to control the rate at which packets are sent; this script performs per-flow traffic shaping by creating a dedicated token bucket filter for both incoming and outgoing traffic flows. This queue smoothes bursts in traffic flows. We rate-limit the download traffic of each flow to 7 Mbps for download and 3 Mbps for upload and monitor the memory and CPU usage and compute the average over ten minutes.

| | CPU usage | Memory usage |
|---|---|---|
| Traffic shaping script | 25.39% | 62.98MB (49.21%) |
| FlowQoS | 23.63% | 57.19MB (44.68%) |

**Table 2:** *CPU and memory consumption for FlowQoS and a comparable traffic shaping script.*

Table 2 shows that FlowQoS uses considerably less CPU and memory than a traffic-shaping script that assigns new flows to unique queuing disciplines when new traffic flows arrive. Because FlowQoS sets up QoS queues once per traffic class and assigns flows to queues based on application identification, CPU usage is also lower than OpenWrt-based scripts that assign each new flow to a new queuing discipline, as setting up these queues per-flow can consume significant CPU.

## 5 Discussion

One significant unresolved question in the design of FlowQoS concerns whether the controller (and corresponding application-identification mechanisms) should be co-located with the router inside the home or "outsourced" to an offsite location. We point out that FlowQoS can function with either design, and the location of FlowQoS's control mechanisms depends on the computational power of the home gateway, the user's tolerance for higher latency for application classification, and who ultimately controls the quality of service parameters. Our prototype implementation of FlowQoS currently resides on a device that is separate from the router, but this need not be the case long-term, as the computational power of home routers evolves. A recent study illustrated that home gateway devices typically cannot perform detailed traffic analysis at rates of more than 5 Mbps [27]. Some home gateways may eventually be able to perform more sophisticated operations [34, 35], which may ultimately make it possible to co-locate FlowQoS with the home gateway.

FlowQoS presents several possible avenues for future work. First, FlowQoS could support more flow classification engines, such as a classifier that handles non-TCP or UDP messages; or a series of parallel classifiers, with weights given to each classifier depending on input depending on the accuracy of each classifier.

Time-based rate limiting may be useful, as previous work has identified the need to impose bandwidth restrictions in home networks after certain hours [7]. Such a mechanism could be used in the home to limit usage of, for instance, social media after 9 p.m. As another example, time-based rate limiting mechanisms could be used to rate-limit downloads of bandwidth intensive operations such as operating system updates or backups until off-peak hours. In the same vein, FlowQoS could support different QoS rules for different users or devices. This mechanism could be combined with time-based rate limiting: for example, a parent may limit video during the evening hours for their children and their devices, while still allowing high-bandwidth streams to the television.

FlowQoS could provide per-device and per-application usage statistics to help users better track their usage for different devices and applications. On a home Internet connection with a monthly usage cap, a user may want to know how different applications or devices are consuming data [6]. Adding hard and soft usage caps for specific users or devices within a home network, such as those that have been proposed in previous work [6, 14] is another possible extension. Such a mechanism would be similar to many mobile phone plans where there is a certain quota of full-speed connectivity, after which point an application or device's data rate is throttled for the remainder of the billing cycle.

As technologies continue to mature, systems like FlowQoS will likely become easier to deploy. For example, OpenFlow 1.3 supports per-flow QoS, which makes the dual OVS topology unnecessary: instead of implementing rate limiters with `tc` on virtual links between two Open vSwitch switches, the controller could instead simply install a flow-table entry with the appropriate QoS parameters. Furthermore, various ISP consortia are exploring the possibility of installing routers in homes that are better provisioned [24]; enhancements to the capabilities of the home router may make it possible to shift some functions from the controller to the router itself (or even run the controller directly on the router), thus improving performance.

## 6 Conclusion

FlowQoS enables QoS in broadband access networks by offloading expensive traffic classification algorithms to an SDN controller, which installs the appropriate flow table entries in the home gateway. Offloading QoS function facilitates per-flow, application-based QoS on commodity home routers and simplifies QoS configuration for ordinary home users. We have implemented a prototype of FlowQoS on OpenWrt and are currently extending the system to support additional features and applications.

## References

[1] S. Alcock and R. Nelson. Libprotoident: Traffic classification using lightweight packet inspection (technical report). University of Waikato, 2012.

[2] C. Aurrecoechea, A. T. Campbell, and L. Hauw. A survey of qos architectures. *Multimedia systems*, 6(3):138–151, 1998.

[3] J. But, G. Armitage, and L. Stewart. Outsourcing automated qos control of home routers for a better online game experience. *Communications Magazine, IEEE*, 46(12):64–70, December 2008.

[4] M. Carbone and L. Rizzo. Dummynet revisited. *SIGCOMM Comput. Commun. Rev.*, 40(2):12–20, Apr. 2010.

[5] V. Carela-Espaol, T. Bujlow, and P. Barlet-Ros. Is our ground-truth for traffic classification reliable? In M. Faloutsos and A. Kuzmanovic, editors, *Passive and Active Measurement*, volume 8362 of *Lecture Notes in Computer Science*, pages 98–108. Springer International Publishing, 2014.

[6] M. Chetty, R. Banks, A. Brush, J. Donner, and R. Grinter. You're capped: understanding the effects of bandwidth caps on broadband use in the home. In *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems*, pages 3021–3030. ACM, 2012.

[7] M. Chetty, R. Banks, R. Harper, T. Regan, A. Sellen, C. Gkantsidis, T. Karagiannis, and P. Key. Who's hogging the bandwidth: the consequences of revealing the invisible in the home. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 659–668. ACM, 2010.

[8] A. D. Ferguson, A. Guha, C. Liang, R. Fonseca, and S. Krishnamurthi. Participatory networking: An API for application control of SDNs. In *ACM SIGCOMM*, pages 327–338. ACM, 2013.

[9] FlowQoS. `http://flowqos.noise.gatech.edu/`.

[10] P. Georgopoulos, Y. Elkhatib, M. Broadbent, M. Mu, and N. Race. Towards network-wide QoE fairness using openflow-assisted adaptive video streaming. In *Proceedings of the 2013 ACM SIGCOMM workshop on Future human-centric multimedia networking*, FhMN '13, 2013.

[11] J. Gozdecki, A. Jajszczyk, and R. Stankiewicz. Quality of service terminology in ip networks. *Communications Magazine, IEEE*, 41(3):153–159, 2003.

[12] A. Ishimori, F. Faria, I. Carvalho, E. Cerqueira, and A. Abelem. Automatic QoS Management on OpenFlow Software-Defined Networks. June 2012.

[13] T. Karagiannis, K. Papagiannaki, and M. Faloutsos. Blinc: multilevel traffic classification in the dark. In *ACM SIGCOMM Computer Communication Review*, volume 35, pages 229–240. ACM, 2005.

[14] H. Kim, S. Sundaresan, M. Chetty, N. Feamster, and W. K. Edwards. Communicating with caps: Managing usage caps in home networks. In *ACM SIGCOMM Computer Communication Review*, volume 41, pages 470–471. ACM, 2011.

[15] 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 *Proceedings of the 2010 internet network management conference on Research on enterprise networking*, 2010.

[16] N.-S. Ko, H. Heo, J.-D. Park, and P. Hong-Shik. OpenQFlow: Scalable OpenFlow with Flow-Based QoS. *IEICE Transactions*, 96-B(2), 2013.

[17] S. Lederer, C. Müller, and C. Timmerer. Dynamic adaptive streaming over http dataset. In *Proceedings of the 3rd Multimedia Systems Conference*, MMSys '12, pages 89–94. ACM, 2012.

[18] D. McDysan. *QoS and traffic management in IP and ATM networks*. McGraw-Hill, Inc., 1999.

[19] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. Openflow: enabling innovation in campus networks. volume 38, pages 69–74. ACM, 2008.

[20] Meraki Traffic Shaper. `http://goo.gl/IL24ek`.

[21] R. Mortier, B. Bedwell, K. Glover, T. Lodge, T. Rodden, C. Rotsos, A. W. Moore, A. Koliousis, and J. Sventek. Supporting novel home network management interfaces with OpenFlow and NOX. In *ACM SIGCOMM*, 2011.

[22] P. Newman, W. Edwards, R. Hinden, E. Hoffman, F. C. Liaw, T. Lyon, and G. Minshall. Ipsilon flow management protocol specification for IPv4 version 1.0. 1996.

[23] OF-CONFIG 1.2: OpenFlow Management and Configuration Protocol. `http://goo.gl/1JnFWN`, 2014.

[24] Open Home Gateway Forum. `http://www.ohgf.org/`.

[25] PacketShaper. `http://www.bluecoat.com/products/packetshaper`.

[26] POX. `http://www.noxrepo.org/pox/about-pox/`.

[27] A. Reggani, F. Schneider, and R. Teixeira. Tracking application network performance in home gateways. In *Wireless Communications and Mobile Computing Conference (IWCMC), 2013 9th International*, pages 1150–1155, July 2013.

[28] F. Risso and I. Cerrato. Customizing data-plane processing in edge routers. In *2012 European Workshop on Software Defined Networking (EWSDN)*, pages 114–120, Oct 2012.

[29] M. Roughan, S. Sen, O. Spatscheck, and N. Duffield. Class-of-service mapping for QoS: a statistical signature-based approach to IP traffic classification. In *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pages 135–148. ACM, 2004.

[30] M. S. Seddiki, M. Shahbaz, S. Donovan, S. Grover, M. Park, N. Feamster, and Y.-Q. Song. Flowqos: Qos for the rest of us. In *Proceedings of the Third ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, HotSDN '14, 2014.

[31] B. Sonkoly, A. Gulyas, F. Nemeth, J. Czentye, K. Kurucz, B. Novak, and G. Vaszkun. On QoS Support to Ofelia and OpenFlow. In *2012 European Workshop on Software Defined Networking (EWSDN)*, 2012.

[32] S. Sundaresan, W. de Donato, N. Feamster, R. Teixeira, S. Crawford, and A. Pescapè. Broadband internet performance: A view from the gateway. In *Proceedings of the ACM SIGCOMM 2011 Conference*, SIGCOMM '11, pages 134–145, New York, NY, USA, 2011. ACM.

[33] E. Upton and G. Halfacree. *Raspberry Pi User Guide*. John Wiley & Sons, 2012.

[34] V. Valancius, N. Laoutaris, L. Massoulié, C. Diot, and P. Rodriguez. Greening the internet with nano data centers. In *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies*, CoNEXT '09, pages 37–48, New York, NY, USA, 2009. ACM.

[35] J. Whiteaker, F. Schneider, R. Teixeira, C. Diot, A. Soule, F. Picconi, and M. May. Expanding home services with advanced gateways. volume 42, pages 37–43, Sept. 2012.

[36] N. Williams and S. Zander. Real time traffic classification and prioritisation on a home router using diffuse, 2011.

[37] Y. Yiakoumis, S. Katti, T.-Y. Huang, N. McKeown, K.-K. Yap, and R. Johari. Putting home users in charge of their network. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, pages 1114–1119, 2012.