Dynamic Bandwidth Allocation with Minimum Rate Guarantees Using OpenFlow

Joel van Egmond Maryam Elahi Mea Wang

Department of Computer Science
University of Calgary

Abstract

Quality of Service (QoS) control typically refers to network mechanisms used to guarantee a certain service from the network [7]. Many service providers use hard QoS controls to guarantee bandwidth to their clients. In current solutions excess idle bandwidth is wasted, but with the emergence of Software-Defined Networking (SDN) and the OpenFlow protocol, it is possible to program a network to respond to this problem cheaply and efficiently. This paper describes the design of an SDN/OpenFlow solution that ensures minimum bandwidth guarantees for clients while fairly distributing the excess bandwidth among current users to maximize link capacity utilization.

Key words: OpenFlow, Ryu, software defined networking, traffic shaping, quality of service.

1 Introduction

Quality of Service (QoS) control typically refers to network mechanisms used to guarantee a certain service from the network [7]. For example, minimum bandwidth is a service frequently guaranteed by QoS. Internet service providers (ISPs) are common users of QoS technology. Traditional ISPs such as Bell, Telus, and Shaw aim to maximize profits and avoid having excess unused bandwidth by overselling their channel to more clients than it can support, with the assumption that not all clients will be using the channel simultaneously. In contrast, many other large-scale service providers, such as the research and education focused Cybera, ORION, and MRnet do not oversell their channel, opting to use QoS controls to guarantee each client that their allocated bandwidth will always be available even if every other client is currently using their own maximum allocated bandwidth. Consequently, these networks often have significant amounts of excess idle bandwidth, since it is very unlikely that every client will be actively using the network at all times. Currently this bandwidth is wasted while the operation and maintenance costs of the full channel must still be paid. This leads to the problem of how to make use of all of this excess bandwidth, while still maintaining the minimum bandwidth guarantees demanded by the clients. One potential technology for solving this problem is Software-Defined Networking.

Software-Defined Networking (SDN) is an emerging network paradigm that decouples network control and forwarding functions, enabling network control to be directly programmable [4]. This is done by implementing the network control function in a centralized controller node, which communicates to all the network devices in the network through a standardized protocol. OpenFlow is one such protocol; an open standard protocol that acts as an abstracted switch interface to allow the SDN controller to control the routers/switches in its domain [8].

The purpose of this research is to design and implement an SDN solution to the excess bandwidth problem, in order to maximize the utilization of a network while providing minimum bandwidth guarantees to its clients. This is done by directing network traffic through an OpenFlow enabled switch. The switch is connected to a SDN controller device, which uses the switch's features to guarantee minimum bandwidths while dynamically allocating the excess bandwidth to clients according to a dynamic allocation algorithm, of which three have been developed. The algorithms in question are inspired by common CPU scheduling algorithms, but have been custom designed to approximate the functionality of their inspirations within the limitations of the OpenFlow switch environment.

This research contributes to the body of QoS research by presenting an original SDN/OpenFlow traffic shaping design, which provides both minimum bandwidth as a guaranteed service, as well as the ability to utilize and fairly distribute the excess bandwidth in the link. Once implemented the solution will be evaluated first on its ability to guarantee minimum rates, and then on each algorithm's link capacity utilization and fairness of allocation.

1.1 Key Definitions

When discussing an OpenFlow switch in the context of this research there are several important terms: flows, meters, meter bands, Differentiated Services Code Point (DSCP) remark, and queues.

In OpenFlow, a flow describes a set of packets transferred from one network endpoint to another endpoint. The endpoints may be defined as IP addresses, TCP/UDP ports, VLAN identifiers, or physical device ports, among others. A flow additionally defines the of rules describing the forwarding actions that the device should take for all packets belonging to that flow [5]. Flows may be manually installed on a switch, or installed by the SDN controller.

Meters are flow monitoring tools which allow the rate monitoring of traffic going into the switch (ingress traffic). Meters are attached to specific flows, and can perform operations based on the rate of traffic entering those flow, specifically dropping or DSCP remarking packets.

Meter bands are a feature of meters that allow them to perform operations on traffic in a flow. A meter band defines a rate threshold and an action, such that when the rate of traffic passing through the meter passes that threshold, the action is applied to all traffic over the threshold.

DSCP remark is a possible action that can be chosen for a meter band. The DSCP remark action is provided a 6-bit value, and will write that value to the Type of Service (TOS)/DSCP field in the IP header of all packets that trigger the meter band. This is important because flows are able to match on the DSCP field, allowing for traffic differentiation.

Finally, queues are the complements to meters for traffic leaving the switch (egress traffic), in that they provide a mechanism for controlling the rate of egress traffic instead of ingress traffic. They are also attached to specific flows, and all traffic passing through the flow is placed into the attached queue. The packets in the queue are then processed at specified minimum and/or maximum rates. On most switches queues are given a priority and are serviced strictly according to their priority, but some switches have support for other queueing algorithms such as round-robin or weighted round-robin.

2 Previous Work

The body of existing research relevant to this project can be classified in four major sections: hard guarantees, soft guarantees, flow monitoring, and scheduling algorithms. The first two sections refer to the two main paradigms that have emerged from the various methods of using OpenFlow enabled switches to obtain minimum bandwidth guarantees. A hard bandwidth guarantee uses a static reservation of network resources for the specified traffic, whereas a soft bandwidth guarantee uses a differentiated service approach with no reservation, in which some traffic is given priority over other traffic using only currently available network resources [6]. The third sections refers to methods of monitoring network traffic passing through OpenFlow flows. The final section refers to various network packet scheduling algorithms and discussions on their efficiency and fairness.

2.1 Hard Guarantee

Designing methods of achieving hard bandwidth guarantees for SDN enabled networks was the focus of research by Tomovic et al. [9] and Dwarakanathan et al. [3].

Tomovic et al. [9] argue that providing QoS guarantees to applications is an important objective of modern networks. With the rising popularity of real-time applications, especially in multimedia, the need arises for a network service model with performance guarantees. Current models (such as IntServ and DiffServ) lack adequate control, so QoS is currently often implemented by assigning separate physical networks with their own hardware to each class of traffic, incurring high operational expenditures and infrastructure costs. The paper aims to address this issue by designing a Software-Defined Networking (SDN) control environment to provide bandwidth guarantees between two points. The solution works by shifting network intelligence into a PoX SDN controller, which accepts/denies bandwidth requests, monitors network resources, and calculates traffic routes [9]. The solution is evaluated through a pair of experiments, which aim to demonstrate that the system can actually provide the minimum bandwidth guarantees between two points in the work. IntServ has been chosen as a baseline, and the results of each experiment are compared this baseline to measure improvement. This paper contributes to the body of QoS research by demonstrating the design of an SDN/OpenFlow control environment that provides bandwidth guarantees for priority flows in an automated manner, alleviating the need for distinct physical networks for multiple classes of service.

Dwarakanathan et al. [3] argue that current QoS solutions are not sufficiently scalable. To address this they implemented a similar SDN environment to that of Tomovic et al. [9], using a Floodlight SDN controller as the point of network intelligence instead of a PoX controller. Dwarakanathan's [3] solution aims to be more scalable than that of Tomovic et al. [9] by using fewer queues in each switch. Instead of creating one queue per guaranteed flow as in Tomovic's [9] system, it instead uses only one queue shared between all guaranteed flows, whose minimum and maximum are dynamically adjusted as bandwidth requests are granted and previously granted flows expire. The solution is evaluated through experiments ran in the Mininet simulator, in which theoretical minimum rates are calculated and compared against the experimental results. This paper contributes to the body of QoS research by demonstrating that it is possible to create a scalable QoS solution with SDN.

In all of these systems, the maximum rate is equal to the guaranteed rate, preventing the flow from ever exceeding its guaranteed rate. This means that any unreserved bandwidth goes unused, resulting in sub-optimal link capacity utilization. They also both rely on all nodes in the network between the start and end points allocating resources, which introduces scalability difficulties.

2.2 Soft Guarantee

Designing methods of achieving soft bandwidth guarantees for SDN enabled networks was the focus of research by Krishna et al. [7] and Wallner and Cannistra [11].

Krishna et al. [7] argue that QoS is currently underutilized in provider networks, because it is too complicated to install and maintain. A response to this issue would be of great interest to large-scale internet service providers, who currently massively overprovision their networks at great cost. This paper aims to demonstrate how to use Software-Defined Networking (SDN) and the OpenFlow protocols QoS features to cheaply and efficiently implement minimum bandwidth guarantees in a network. The solution described in the paper uses a Ryu SDN controller communicating with an OpenFlow enabled switch, through which all QoS traffic will pass [7]. The minimum bandwidth guarantee is applied by using a combination of the switchs ingress meter table, and and egress queues. The ingress meters are used to aggregate traffic and redirect packets over the guaranteed minimum rate, and the egress queues and used to ensure priority traffic is transmitted first. The success of their solution is measured by the results of their three experiments. The first experiment tests the creation of new QoS flows in the system, the second tests traffic aggregation and prioritization, and the the third tests the responsiveness of the system. The experiments all have calculated theoretical expected results, and the effectiveness of the system is measured by how close the actual results come the expected results. The key metrics analyzed are bandwidth use for each flow, bandwidth use of the full channel, and time to adjust to changes in bandwidth usage. The paper contributes to the body of QoS knowledge by demonstrating how to implement minimum bandwidth guarantees using OpenFlow, while still allowing excess bandwidth to be shared without causing packet loss.

Wallner and Cannistra [11] argue that different network services (such as Voice Over IP (VOIP), stored video streaming, and live video streaming) have different requirements in terms of latency and bandwidth, and that these services can be divided into classes based on these requirements. Their paper addresses this by presenting a proof of concept model for class-based QoS control using SDN. A floodlight SDN controller is used to control many switches in a network, and in each switch incoming packets are classified based on their Type of Service (ToS) bits in the IP header. From this point different classes are given different priorities as they leave the switch. In this system minimum rates are only guaranteed by class and not by individual client, and the excess bandwidth is only distributed on a first-come first-serve basis. The effectiveness of this solution is unclear, as no empirical methods of evaluation are mentioned in the paper. The paper contributes to the body of QoS research by demonstrating that it is possible to use SDN to implement class-based QoS.

In Wallner and Cannistra's [11] system minimum rates are only guaranteed by class and not by individual client, and in both systems the excess bandwidth is only distributed on a first-come first-serve basis, allowing clients with high traffic to unfairly dominate the channel.

2.3 Flow Monitoring

Designing methods of performing real-time monitoring of flow statistics for SDN enabled networks was the focus of a paper by van Adrichem et al. [10].

Van Adrichem et al. [10] argue that a key requirement to implement effective QoS in an SDN network is accurate traffic monitoring, on a finely-grained (per flow) basis. They state that current flow-based measurement techniques are too resource hungry in terms of CPU and bandwidth usage. To address this issue they propose *OpenNetMon*, a POX OpenFlow controller module enabling accurate monitoring of per-flow throughput, packet loss and delay metrics [10]. *OpenNetMon* improves upon current measurement techniques by optimizing the choice of which switches are polled, and the frequency which they are polled, in order to minimize network resource usage while maintaining high accuracy [10]. *OpenNetMon*'s effectiveness was evaluated by running an *OpenNetMon* module as well as *tcpstat* on a node, and comparing their monitoring statistics as well as CPU and bandwidth usage. This paper contributes to the body of SDN research by demonstrating that is is possible to efficiently perform per-flow monitoring.

2.4 Scheduling Algorithms

Design, and evaluation of network scheduling algorithms for efficient and priority-based traffic was the focus of a paper by Balogh et al. [2].

Balogh et al. [2] argue that in a networking ecosystem that is increasingly making use of class-based priority QoS, the way how waiting packets in network nodes are selected for output is a significant factor for providing that QoS. In their paper they investigate three priority queuing based scheduling algorithms. The algorithms are Priority Queuing (PQ), Weighted Round Robin with Priority Queueing (WRRPQ), and Low Latency Queuing (LLQ) [2]. These algorithms all support strict priority queueing, in which time critical priority data can be transmit first. PQ is a simple strict priority algorithm, in which priority packets are sent first, and all other packets are sent in first-come first-serve [2]. WRRPQ is a combination of PQ and Weighted Round Robin, in which priority packets are sent first, and other packets are sent according to Weighted Round Robin [2]. LLQ is an algorithm which considers the time required to send each packet to minimize latency [2]. The algorithms were evaluated by testing each one in a simulated network using different kinds of traffic. The results found that effectiveness of each algorithm depended on the kind of traffic (LLQ better for VOIP, WRRPQ better for fixed size packets). This paper contributes to the body of QoS research by showing that there are different optimal priority queuing algorithms for class-based QoS depending on the type of traffic.

All these models implement bandwidth guarantees, but with shortcomings in the areas of link capacity utilization (in hard guarantees) and fairness of bandwidth distribution (in soft guarantees). This project addresses these shortcomings by allowing excess link capacity to be used, and dynamically fairly allocating the excess bandwidth.

3 Methodology

This project implements a hybrid soft/hard QoS system, in a two-part solution. The first part of the solution is to achieve a minimum bandwidth guarantee for each client while retaining the use of the excess bandwidth on a first-come first-serve basis. The second part of the solution addresses the dynamic fair allocation of the excess bandwidth beyond only first-come first-serve. Both parts of the solution are implemented using an OpenFlow enabled Pica8 switch, with a Ryu SDN controller managing the details of the switch's flows, meters, and queues.

3.1 Technical Limitations

The unfortunate reality of implementing on a real switch is that OpenFlow is only a standard, and switch manufacturers may not correctly or completely implement the standard. This becomes increasingly apparent when taking advantage of OpenFlow's more recent or esoteric features. While most OpenFlow features work as expected on the Pica8, there are a few limitations that impacted the solution design or required workarounds. The specific problematic limitations are as follows:

- The OpenFlow standard supports an arbitrary number of queues, whereas the Pica8 switch only supports 8 queues. This is a realistic hardware limitation, but it imposes scalability restrictions.
- The OpenFlow standard supports an arbitrary number of meter bands, whereas the Pica8 switch only supports one meter band. This is also a realistic hardware limitation, but it significantly limits the potential utility of meters.
- The OpenFlow standard states that the DSCP remark meter action should occur before the flow action of its attached flow. On the Pica8 the DSCP remark action and the flow action happen in parallel. In most cases this results in the same behaviour, except when the flow action is to direct the packet to another set of flows. According to OpenFlow the packet should be remarked and then sent to the next set of flows, allowing the next set of flows to match on the remarked field. In reality the remark is not applied until the packet leaves the switch, and the next set of flows will not be able to match on the remarked field. This is a significant departure from the intended functionality and requires a substantial workaround.

The details of the design changes resulting from these limitations are mentioned in the solution descriptions.

3.2 Minimum Bandwidth Guarantee

The minimum bandwidth guarantee was initially planned to be achieved through a system in which each client would have a corresponding meter and queue assigned to them, with one best effort queue to handle excess traffic (Figure 1). The meter would use one meter band set to the client's minimum allocated bandwidth. All traffic under the client's minimum rate would be put into its corresponding flow, while all excess traffic would be DSCP remarked by the meter and directed to the best effort queue. Each client's queue's minimum rate would be set to the client's allocated rate, guaranteeing that the client would never go below that rate. Any excess bandwidth would be used for traffic in the excess traffic queue, allowing for the use of the excess bandwidth. While promising, this system was abandoned because it was not scalable, as for n clients it would require $n + 1$ queues, and the Pica8 switch can only support 8 queues.

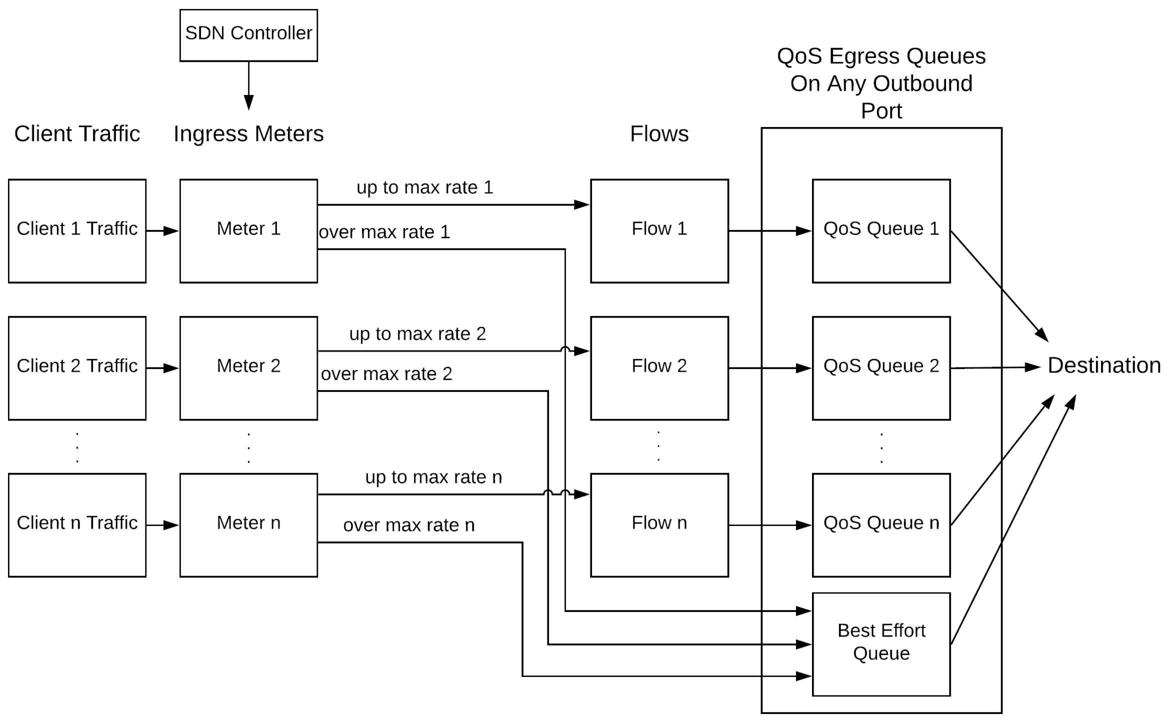

Figure 1: *Initial Pica8 setup for minimum bandwidth guarantee.*

To address the scalability issue, the minimum bandwidth guarantee design was reworked to use a similar method to Krishnas [7] system (Figure 2). It was implemented using an OpenFlow enabled Pica8 switch, using egress queues, ingress meters, and a Ryu SDN controller remotely controlling the switch. The egress queues are set in a strict priority system, in which the higher priority queue always goes first. The highest priority queue is used for guaranteed rate traffic, and the lower priority queues are used for traffic over clients guaranteed minimum rates. The ingress meters are used for traffic aggregation and setting the minimum rate for each client. Each individual client has a meter assigned to them, which measures incoming/outgoing traffic. The meter directs all traffic under the guaranteed bandwidth (meter maximum rate) to the highest priority queue, ensuring that it will be transmitted first. All traffic over the guaranteed rate is DSCP remarked and directed to a second set of flows, which redirected to the lower priority queues by matching on the packets' remarked DSCP field. This redirection guarantees the minimum rate for each client so long as the system obeys the equation:

$$\sum M_{max} \quad \leq \quad R$$

where M_{max} is a meter maximum rate, and R is the link transmission rate. This is to say the minimum rates will always be guaranteed so long as the sum of the minimum rates is not greater than the maximum link rate. This solution requires a minimum of only two queues for n clients, addressing the initial design's scalability issue.

Unfortunately the second design was also subject to a technical limitation, as using DSCP remark to redirect traffic to lower priority queues did not perform as expected, due to the Pica8's aforementioned incorrect implementation of DSCP remark. Since the DSCP remark would still apply after the packet left the switch, the issue was resolved by routing remarked traffic to a second Pica8 switch, and installing the egress queues and second set of flows on the second switch instead of the first. This workaround treats the two switches combined as one entity, with the first switch functioning as the ingress point performing the metering and DSCP remark, and the second switch functioning as the egress point redirecting remarked packets to their appropriate queues.

3.3 Dynamic Bandwidth Allocation

Having implemented minimum bandwidth guarantees, the second part of the solution is the ability to distribute the excess bandwidth. This is done by implementing various distribution algorithms in the SDN controller. The controller reads real-time bandwidth usage for each client from the flows and adjusts how the traffic is redirected dynamically according to what algorithm is in use. This

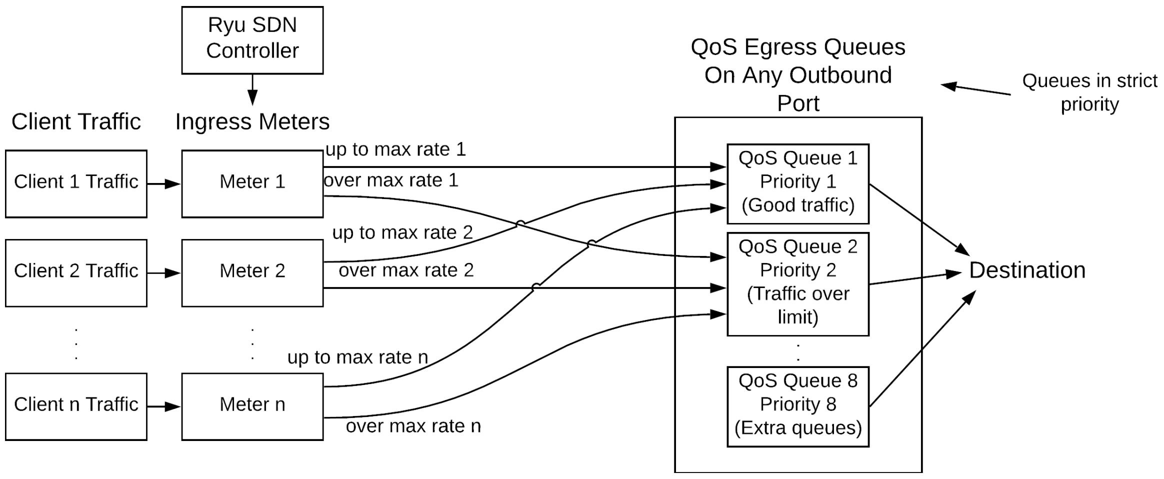

Figure 2: *Modified Pica8 setup for minimum bandwidth guarantee.*

system builds onto the guaranteed bandwidth solution by adding another meter band for each meter, and one more queue to the switch (Figure 3). The first meter band's trigger rate is set to the the client's guaranteed minimum bandwidth, sending all traffic under this rate to the high priority queue. The second meter band's trigger rate is set higher than the minimum bandwidth, but is not fixed. Traffic above the first band's rate but below the second band's rate is sent to the medium priority queue. Traffic above the second band's rate is sent to the low priority queue. This creates a three tier priority differentiation of traffic, the first tier for guaranteed traffic, the second tier for dynamically allocated excess traffic, and the third tier for extreme excess traffic. What the second meter band's trigger rate is set to depends on the dynamic allocation algorithm in use, but all algorithms will have some common elements. In each algorithm the SDN controller repeatedly polls the switch flows for their bandwidth usage, allowing the controller to determine the current demanded rate for each client. With this information the controller calculates the amount of excess bandwidth available, and divides it among the currently active clients depending on their current demand and the chosen dynamic bandwidth algorithm. The division is done by setting the second meter bands' trigger rates such that for all clients the sum of the bandwidth captured in the second tier of traffic is equal to the current excess bandwidth available in the link. The controller frequently repeats this process to continuously adapt the system to changing traffic demands, and in this fashion dynamically allocates the excess bandwidth. Because this is a reactive system, the third tier of traffic (extreme excess) exists to capture traffic that the controller has not yet compensated for, ensuring that no packets are dropped and the demand polling remains accurate.

Unfortunately this solution was also subject to hardware limitations, specifically regarding the number of available meter bands. The Pica8 switch supports only one meter band per meter, meaning that traffic can only be split in two ways at each meter; unmarked and marked traffic. To compensate for this the system was redesigned to use a second set of meters in place of a second set of meter bands (Figure 4). In this modified setup the trigger rate of the second set of meters is modified by the controller instead of the second set of meter bands, but otherwise the setup behaves similarly. On the first switch, the first set of meters (tier 1 meters) split traffic into guaranteed bandwidth (tier 1 traffic), and the combination of dynamically allocated excess traffic and extreme excess traffic (tier 2 and tier 3 traffic). The tier 1 traffic is sent to the high priority queue on the third switch, while the tier 2 and tier 3 traffic is sent to second set of meters (tier 2 meters) on the second switch. The tier 2 meters then split the combined tier 2 and tier 3 traffic into separate tier 2 and tier 3 streams. The tier 2 stream is sent to the medium priority queue on the third switch, and the tier 3 stream is sent to the low priority queue of the third switch. This setup ultimately produces the same results as the original would have, at the cost of requiring a three switches instead of just two.

The design required for bandwidth guarantees puts some restrictions on algorithm design. Only the ingress meter settings can be changed dynamically, and the egress queues must have a strict priority system, meaning the higher priority will always transmit first. This means that it is non-trivial to implement any dynamic allocation algorithm outside first-come first-serve, even an algorithm as simple as round robin will require some

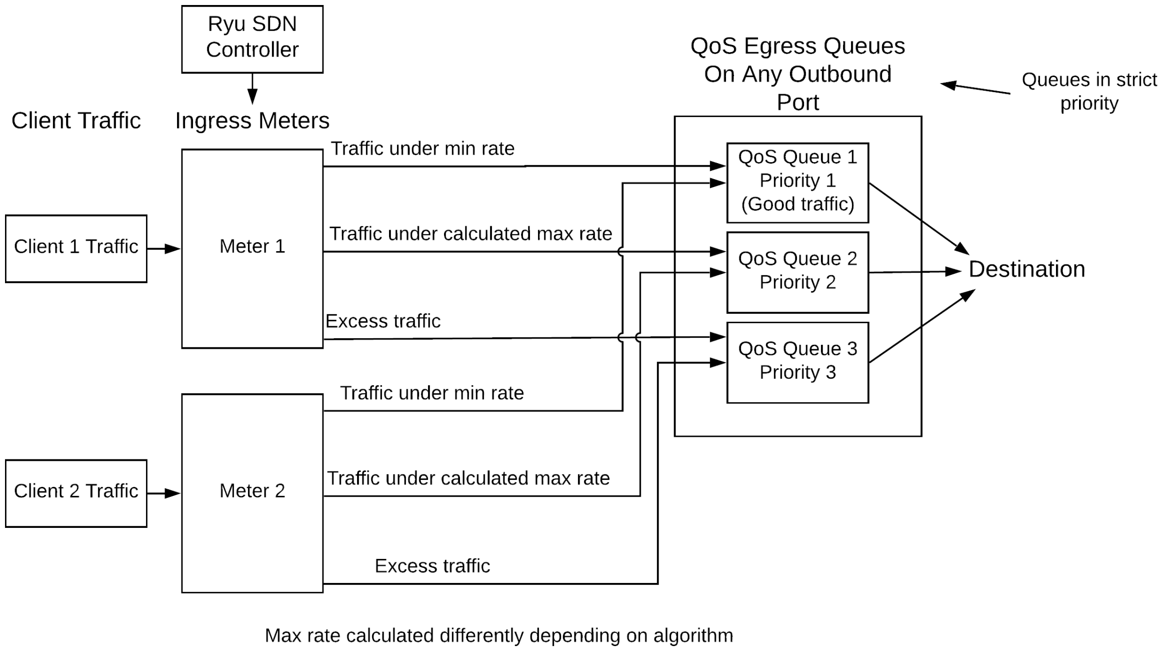
Ryu SDN Controller
Client Traffic
Ingress Meters
QoS Egress Queues On Any Outbound Port
Queues in strict priority
Client 1 Traffic
Meter 1
Traffic under min rate
Traffic under calculated max rate
Excess traffic
QoS Queue 1 Priority 1 (Good traffic)
QoS Queue 2 Priority 2
QoS Queue 3 Priority 3
Destination
Client 2 Traffic
Meter 2
Traffic under min rate
Traffic under calculated max rate
Excess traffic
Max rate calculated differently depending on algorithm

Figure 3: *Pica8 setup with support for dynamic bandwidth allocation algorithms.*

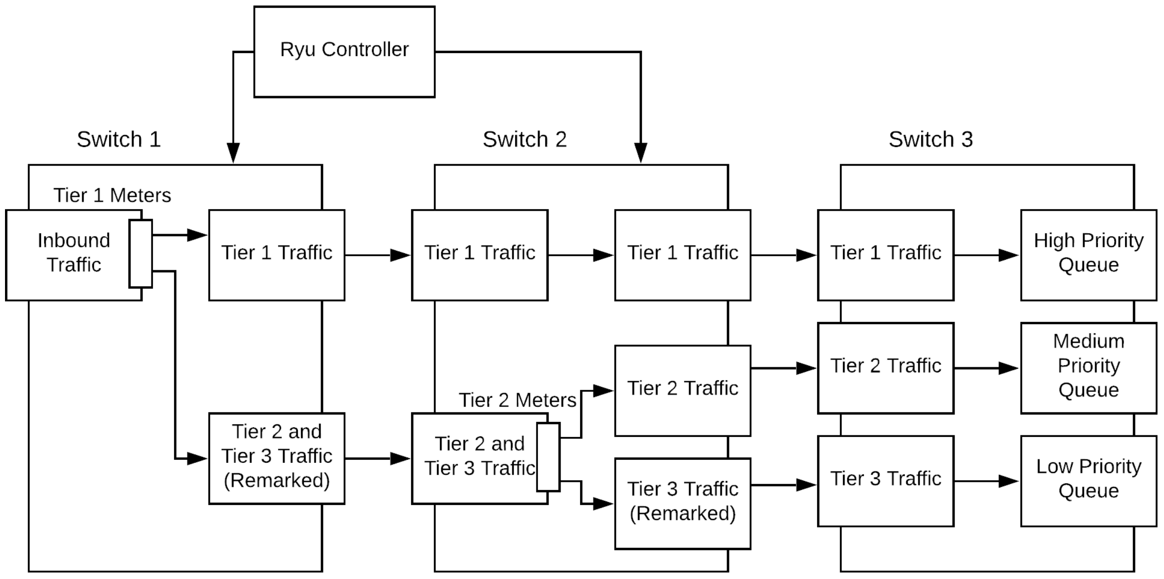

Figure 4: *Modified Pica8 setup for minimum bandwidth guarantee to compensate for insufficient meter bands.*

creativity to implement while maintaining the bandwidth guarantees. Accordingly, the three implemented allocation solutions are relatively simple: egalitarian fairness, proportional fairness, and hybrid proportional fairness.

In egalitarian fairness (Table 1), an equal share of the excess bandwidth is given to all members upon demand. This aims to treat all members as equals regardless of the minimum service they use. This solution is implemented by dividing the excess bandwidth equally between all currently active clients up to a maximum of their demand. This solution is inspired by the round robin scheduling algorithm.

Client	Minimum	Demand	Allocation
Client 1	100	300	200
Client 2	200	400	300
Client 3	300	500	400

Table 1: *Example of egalitarian fairness (Total bandwidth: 900 Mbps)*

In proportional fairness (Table 2), the share of the excess bandwidth given to each member upon demand is in proportion to its allocated minimum bandwidth guarantee. This aims to give members attributed a higher minimum service a better share of the excess bandwidth. This solution is implemented by dividing the excess bandwidth between all currently active clients up to a maximum of their demand, weighting the allocation by their minimum bandwidth. This solution is inspired by weighted round robin scheduling algorithm.

Client	Minimum	Demand	Allocation
Client 1	100	300	150
Client 2	200	400	300
Client 3	300	500	450

Table 2: *Example of proportional fairness (Total bandwidth: 900 Mbps)*

In hybrid proportional fairness (Table 3), each member has priority to use the excess bandwidth up to a fraction of its allocated minimum bandwidth, and the remainder is equally shared. This aims to give preference to higher usage members as in proportional fairness, but with a limitation that prevents them from dominating the channel. This solution is implemented by dividing the excess bandwidth between all currently active clients up to a maximum of their demand, starting by allocating up to $1/X$th of their minimum bandwidth for each client, and then equally after that.

In all cases all excess bandwidth is used and fairly distributed according to the algorithm's definition of fairness.

Client	Minimum	Demand	Allocation
Client 1	100	300	190
Client 2	200	400	300
Client 3	300	500	410

Table 3: *Example of hybrid proportional fairness (Total bandwidth: 900 Mbps, 1/10th of minimum prioritized)*

4 Results

All results were gathered by running a series of test cases on actual hardware setups, in which multiple hosts sent traffic to one other host. Each test gathered data about how much bandwidth was received in each of the priority queues, and how much effective bandwidth each host received (accounting for retransmissions). Success of the tests was evaluated in two ways: First by link capacity utilization, which compares the sum bandwidth usage for all hosts to the full link capacity. Second by fairness of distribution, where the bandwidth usage for each individual host is compared to the theoretical fair distribution as determined by each dynamic allocation algorithm.

4.1 Test Setups

Three different test setups were used to evaluate this construction. A no guarantee setup, a minimum guarantee setup, and a dynamic allocation setup. The no guarantee setup (Figure 5) is a single switch with three hosts and no controller, meant to demonstrate the behaviour of a switch with no minimum rate guarantees or dynamic allocation. The minimum guarantee setup (Figure 6) implements the minimum rate guarantee design (Figure 2). It is two switches, three hosts, and no controller, meant to demonstrate the behaviour of a switch with minimum rate guarantees but without any controller-based dynamic allocation. The dynamic allocation setup (Figure 7) implements the dynamic allocation design (Figure 4). It is two switches, three hosts, and a controller connected to each host, meant to demonstrate the behaviour of a switch with dynamic allocation enabled. The same dynamic allocation setup is used to test all three dynamic allocations. Notice that this setup only uses two switches, even though it was previously explained that three switches are required for the dynamic allocation algorithms to work. For this project only two switches were available, so to simulate a third switch, traffic was routed in a loop out of one port in the second switch and back into another port in the same switch. With careful flow management and manual ARP routing this allowed the second switch to both output and match on remarked packets, mimicking a third switch.

4.2 Tests

Five main classes of test were used for evaluation, each class testing two different scenarios, once with UDP and

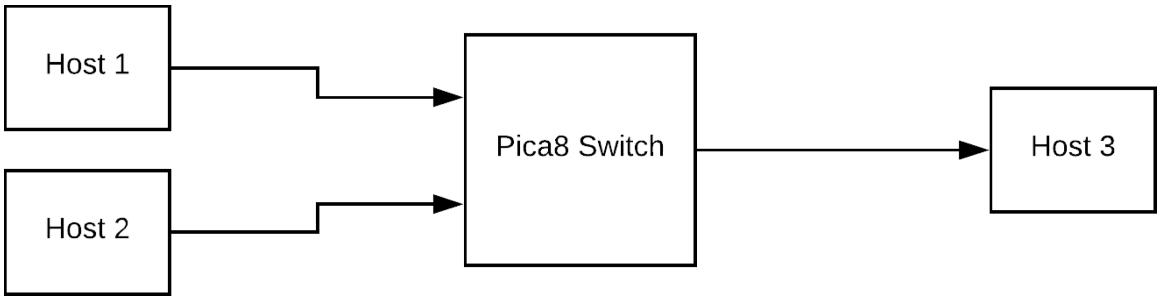

Figure 5: *No guarantee test setup.*

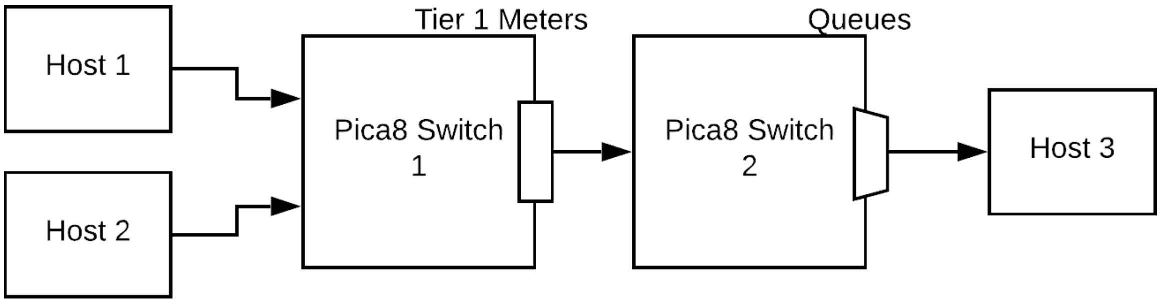

Figure 6: *Minimum guarantee test setup. Implements the minimum rate guarantee design (Figure 2).*

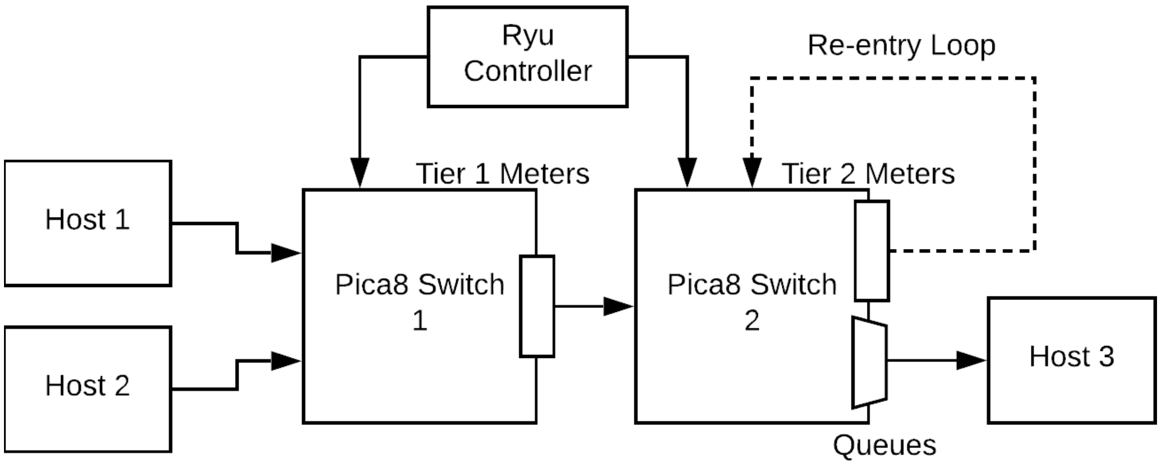

Figure 7: *Dynamic allocation test setup. Implements the dynamic allocation design (Figure 4).*

once with TCP. The first two classes of test correspond to the no guarantee and minimum guarantee test setups. The other three classes of test correspond to the three dynamic allocation algorithms; dynamic egalitarian fairness, dynamic proportional fairness, and dynamic hybrid proportional fairness, all of which used the dynamic allocation test setup. For each of the test classes the same two scenarios were tested with both UDP and TCP, one with non-competing hosts and another with competing hosts. The non-competing hosts scenario tested two hosts sending to a third host through the setup, but their total bandwidth usage was below the link capacity, with the goal of demonstrating the setup/algorithm will behave as expected when not all bandwidth is in use. The competing hosts scenario tested two hosts sending to a third host through the setup, but their total bandwidth usage was above the link capacity, with the goal of demonstrating the setup/algorithm will allocate bandwidth (or not allocate bandwidth in the case of no guarantee and minimum guarantee) as expected when some bandwidth must be given low priority.

All test scenarios have a maximum link capacity of 600 Mbps, and use a minimum rate of 100 Mbps for Host 1, and 200 Mbps for Host 2. In the non-competing scenario, Host 1 has a demand of 200 Mbps, and Host 2 has a demand of 300 Mbps. In the competing scenario, both Host 1 and 2 have demands of 400 Mbps.

An additional test was run for each of the dynamic allocation algorithms, in which the hosts change their demand over the course of the test to demonstrate the dynamic adaptivity of the algorithms. In this test one host begins by sending the receiver packets at 400 Mbps, with no competition. After 15 seconds, the second host begins sending the receiver packets at 500 Mbps, creating competition between the two hosts. Another 15 seconds later, and the first host ceases to send packets, allowing the second host to transmit at 500 Mbps for the remaining 15 seconds of the test.

In each test the receiving host (Host 3) is ran iperf[1] in server mode, and the other hosts (Host 1 and 2) connected using iperf in client mode. The sending hosts then send data at the specified bandwidth through iperf until the test is complete.

4.3 No Guarantee

Both non-competing (Table 4) and competing (Table 5) scenarios were run on the no guarantee setup for both UDP and TCP. The results recorded were the demand for each host (Demand), and the actual bandwidth achieved according to iperf (Actual).

The non-competing (Table 4) results show that actual bandwidth was within 5% of the demand bandwidth for each host for both UDP and TCP. These results indicate that with sum bandwidth demands under link capacity, host demand can be met in full for both protocols.

Protocol	Host	Demand	Actual
UDP	Host 1	200	210
	Host 2	300	315
TCP	Host 1	200	210
	Host 2	300	315

Table 4: *Bandwidth usage for non-competing hosts in no guarantee setup.*
Host 1 minimum rate guarantee: 100 Mbps
Host 2 minimum rate guarantee: 200 Mbps
Total available bandwidth: 600 Mbps

Protocol	Host	Demand	Actual
UDP	Host 1	400	301
	Host 2	400	301
TCP	Host 1	400	315
	Host 2	400	277

Table 5: *Bandwidth usage for competing hosts in no guarantee setup.*
Host 1 minimum rate guarantee: 100 Mbps
Host 2 minimum rate guarantee: 200 Mbps
Total available bandwidth: 600 Mbps

The competing (Table 5) results show that the sum of the actual bandwidths came within 1% of the link capacity, showing ideal link capacity utilization for both UDP and TCP traffic. The distribution of excess bandwidth was based only on demand with no regard for the minimum rates, as each host demanded the same bandwidth and while the full bandwidth was split evenly, the excess 300 Mbps above the min rates was not distributed evenly, with Host 1 receiving 66% and Host 2 receiving 33%. These results indicate that with no guarantees link capacity utilization is good, but there is no form of fairness, the host who demands more bandwidth above their minimum rate receiving more without consideration for other hosts.

4.4 Minimum Rate Guarantee

Both non-competing (Table 6) and competing (Table 7) scenarios were run on the minimum guarantee setup for both UDP and TCP. The results recorded were the demand for each host (Demand), the actual bandwidth achieved according to iperf (Actual), and the bandwidth going to the high priority (High Prio) and remarked (Remarked) egress queues according to the controller.

The non-competing (Table 6) results show that actual bandwidth was within 5% of the demand bandwidth for each host for both UDP and TCP. Additionally, high priority bandwidth was within 5% of the minimum rate for each host. These results indicate that with sum bandwidth demands under link capacity, host demand can be met in full for both protocols, and that minimum rates are given priority.

Protocol	Host	Demand	Actual	High Prio	Remarked
UDP	Host 1	200	210	96	111
	Host 2	300	315	192	119
TCP	Host 1	200	210	96	115
	Host 2	300	315	192	125

Table 6: *Bandwidth usage for non-competing hosts in minimum guarantee setup.*
Host 1 minimum rate guarantee: 100 Mbps
Host 2 minimum rate guarantee: 200 Mbps
Total available bandwidth: 600 Mbps

Protocol	Host	Demand	Actual	High Prio	Remarked
UDP	Host 1	400	322	96	319
	Host 2	400	322	192	223
TCP	Host 1	400	233	96	103
	Host 2	400	359	191	145

Table 7: *Bandwidth usage for competing hosts in minimum guarantee setup.*
Host 1 minimum rate guarantee: 100 Mbps
Host 2 minimum rate guarantee: 200 Mbps
Total available bandwidth: 600 Mbps

The competing (Table 7) results show that the sum of the actual bandwidths were within 1% of the link capacity for TCP, and above the link capacity for UDP (likely due to iperf bandwidth burst during polling), showing ideal link capacity utilization for both UDP and TCP traffic. The distribution of excess bandwidth was similar to no guarantee; based on demand with no regard for the minimum rates, as each host demanded the same bandwidth and while the full bandwidth was split relatively evenly, the excess 300 Mbps above the minimum rates was not distributed evenly, with Host 1 receiving 66% and Host 2 receiving 33%. These results indicate that with minimum guarantees link capacity utilization is good, but there is no form of fairness, the host who demands more bandwidth above their minimum rate receiving more without consideration for other hosts.

4.5 Egalitarian Fairness

Both non-competing (Table 8) and competing (Table 9) scenarios were run on the minimum guarantee setup for both UDP and TCP. The results recorded were the demand for each host (Demand), the theoretical expected throughput according to the egalitarian allocation algorithm (Expected), the actual bandwidth achieved according to iperf (Actual), and the bandwidth going to the high priority (High), medium priority (Mid), and low priority (Low) egress queues according to the controller. Additionally, the dynamic adaptivity test (Figures 8, 9, 10) was run for this setup.

The non-competing (Table 8) results show that actual bandwidth was within 5% of the expected bandwidth, and the high priority bandwidth was within 5% of the minimum rate for each host for UDP traffic. This indicates that for UDP traffic under egalitarian fairness with sum bandwidth demands under link capacity, theoretical expected fairness can be met in full, and minimum rates are given priority. Inversely, TCP performance was overall very poor. Host 2 saw an acceptable 5% deviation from actual to expected with minimum rate maintained, but Host 1 fell a full 62% below the expected bandwidth, and its high priority traffic fell 37% below its minimum rate. This disparity is likely caused by retransmissions due to TCP sequences being separated into different queues, elaborated upon further in the overall results analysis.

Protocol	Host	Demand	Expected	Actual	High	Mid	Low
UDP	Host 1	200	200	199	98	109	4
	Host 2	300	300	307	199	117	3
TCP	Host 1	200	200	75	63	5	13
	Host 2	300	300	315	199	127	1

Table 8: *Bandwidth usage for non-competing hosts in egalitarian fairness setup.*
Host 1 minimum rate guarantee: 100 Mbps
Host 2 minimum rate guarantee: 200 Mbps
Total available bandwidth: 600 Mbps

Protocol	Host	Demand	Expected	Actual	High	Mid	Low
UDP	Host 1	400	250	242	98	178	175
	Host 2	400	350	370	197	175	38
TCP	Host 1	400	250	85	73	6	14
	Host 2	400	350	419	197	234	3

Table 9: *Bandwidth usage for competing hosts in egalitarian fairness setup.*
Host 1 minimum rate guarantee: 100 Mbps
Host 2 minimum rate guarantee: 200 Mbps
Total available bandwidth: 600 Mbps

The competing (Table 9) results show that actual bandwidth was within 6% of the expected bandwidth, and the high priority bandwidth was within 5% of the minimum rate for each host for UDP traffic. This indicates that for UDP traffic under egalitarian fairness with sum bandwidth demands over the link capacity, theoretical expected fairness can be met in full, minimum rates are given priority, and link capacity is fully utilized. As in the non-competing scenario, TCP performance was poor. Host 2's actual usage was 20% above the expected with its minimum rate maintained, but Host 1 was 66% below its expected bandwidth, and its high priority traffic fell 27% below its minimum rate. This disparity is likely caused by retransmissions due to TCP sequences being separated into different queues, elaborated upon further in the overall results analysis.

From the dynamic adaptivity test for egalitarian allocation (Figures 8, 9, 10), it can be seen that the controller's egalitarian fairness algorithm is responsive to changes in

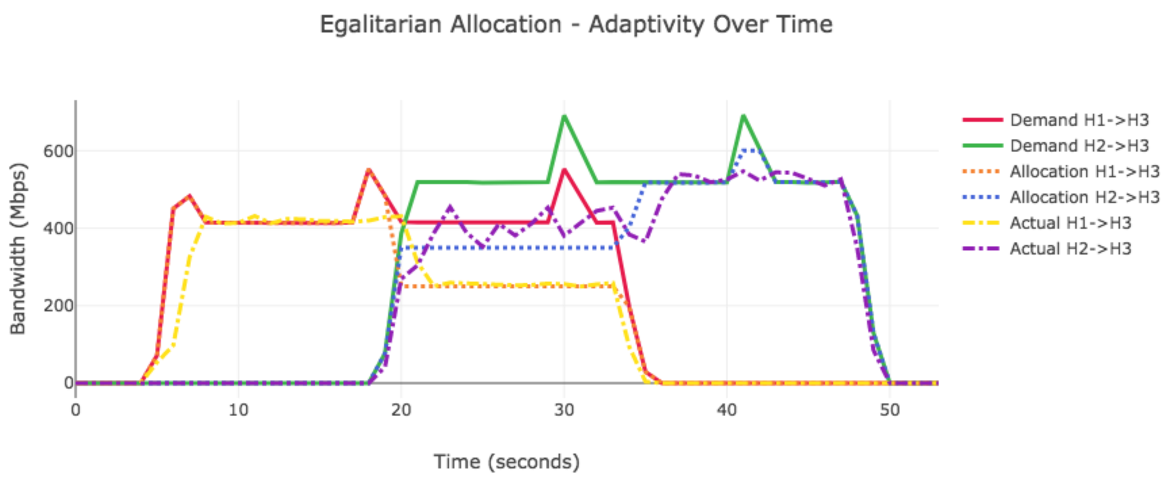

Figure 8: *Egalitarian allocation - adaptivity over time - Host 1 and Host 2.*

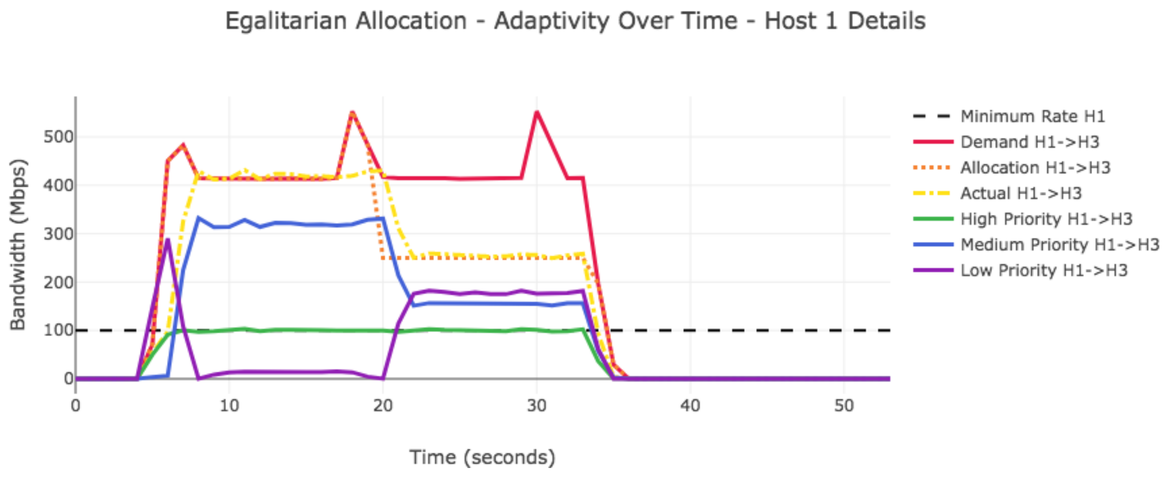

Figure 9: *Egalitarian allocation - adaptivity over time - Host 1 details.*

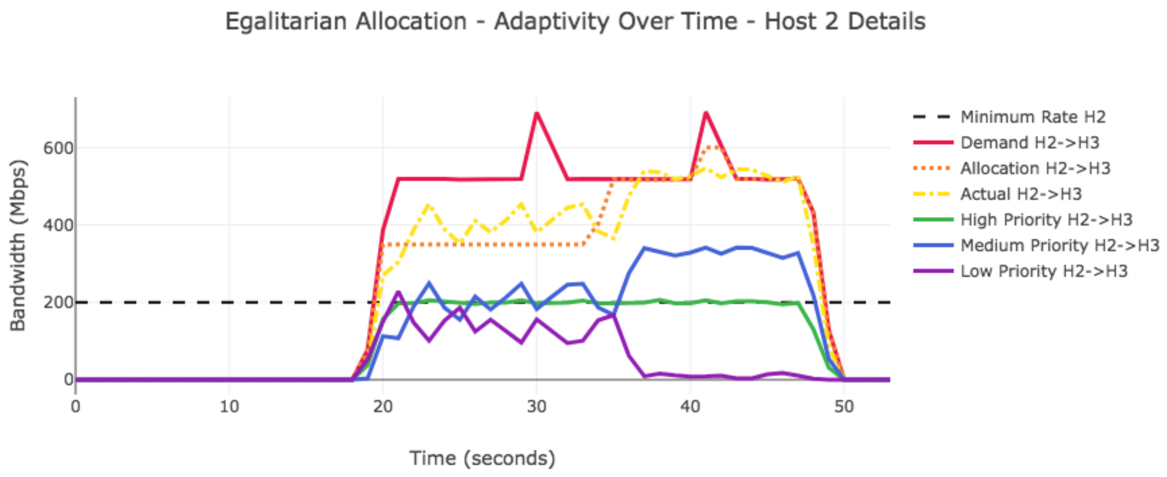

Figure 10: *Egalitarian allocation - adaptivity over time - Host 2 details.*

bandwidth use by each host. Initially while Host 1 demanded 400 Mbps, it was allocated 400 Mbps as there was no competition. When Host 2 created competition by demanding 500 Mbps, Host 1 was allocated 250 Mbps and Host 2 was allocated 350 Mbps. This is correct according to the egalitarian algorithm, as the excess bandwidth of 300 Mbps was split evenly between the two hosts and added to their minimum rates, resulting in 100 + 150 = 250 Mbps for Host 1, and 200 + 150 = 350 Mbps for Host 2. Finally when Host 1 ceased to transmit and competition ended Host 2 was allocated the full 500 Mbps it was demanding. These results indicate that the egalitarian fairness algorithm is properly responsive to changing bandwidth requirements.

4.6 Proportional Fairness

Both non-competing (Table 10) and competing (Table 11) scenarios were run on the minimum guarantee setup for both UDP and TCP. The results recorded were the demand for each host (Demand), the theoretical expected throughput according to the proportional allocation algorithm (Expected), the actual bandwidth achieved according to iperf (Actual), and the bandwidth going to the high priority (High), medium priority (Mid), and low priority (Low) egress queues according to the controller. Additionally, the dynamic adaptivity test (Figures 11, 12, 13) was run for this setup.

Protocol	Host	Demand	Expected	Actual	High	Mid	Low
UDP	Host 1	200	200	199	98	109	4
	Host 2	300	300	308	198	118	2
TCP	Host 1	200	200	86	78	5	15
	Host 2	300	300	314	197	96	19

Table 10: *Bandwidth usage for non-competing hosts in proportional fairness setup.*
Host 1 minimum rate guarantee: 100 Mbps
Host 2 minimum rate guarantee: 200 Mbps
Total available bandwidth: 600 Mbps

The non-competing (Table 10) results show that actual bandwidth was within 5% of the expected bandwidth, and the high priority bandwidth was within 5% of the minimum rate for each host for UDP traffic. This indicates that for UDP traffic under proporional fairness with sum bandwidth demands under link capacity, theoretical expected fairness can be met in full, and minimum rates are given priority. As in egalitarian fairness, TCP performance was very poor. Host 2 saw an acceptable 5% deviation from actual to expected with minimum rate maintained, but Host 1 fell 57% below the target expected bandwidth, and its high priority traffic fell 22% below its minimum rate. This disparity is likely caused by retransmissions due to TCP sequences being separated into different queues, elaborated upon further in the overall results analysis.

Protocol	Host	Demand	Expected	Actual	High	Mid	Low
UDP	Host 1	400	200	198	98	102	223
	Host 2	400	400	397	197	207	13
TCP	Host 1	400	200	91	73	6	14
	Host 2	400	400	419	197	235	2

Table 11: *Bandwidth usage for competing hosts in proportional fairness setup.*
Host 1 minimum rate guarantee: 100 Mbps
Host 2 minimum rate guarantee: 200 Mbps
Total available bandwidth: 600 Mbps

The competing (Table 11) results show that actual bandwidth was within 5% of the expected bandwidth, and the high priority bandwidth was within 5% of the minimum rate for each host for UDP traffic. This indicates that for UDP traffic under proportional fairness with sum bandwidth demands over the link capacity, theoretical expected fairness can be met in full, minimum rates are given priority as expected, and link capacity is fully utilized. As in the non-competing scenario, TCP performance was very poor. Host 2's actual usage was 4% above the expected with its minimum rate maintained, but Host 1 was 55% below its expected bandwidth, and its high priority traffic fell 27% below its minimum rate. This disparity is likely caused by retransmissions due to TCP sequences being separated into different queues, elaborated upon further in the overall results analysis.

From the dynamic adaptivity test for proportional allocation (Figures 11, 12, 13), it can be seen that the controller's proportional fairness algorithm is responsive to changes in bandwidth use by each host. Initially while Host 1 demanded 400 Mbps, it was allocated 400 Mbps as there was no competition. When Host 2 created competition by demanding 500 Mbps, Host 1 was allocated 200 Mbps and Host 2 was allocated 400 Mbps. This is correct according to the proportional algorithm, as the excess bandwidth of 300 Mbps was split proportionally based on minimum rate between the two hosts (33% for Host 1 and 66% for Host 2) and added to their minimum rates, resulting in 100 + 100 = 200 Mbps for Host 1, and 200 + 200 = 400 Mbps for Host 2. Finally when Host 1 ceased to transmit and competition ended Host 2 was allocated the full 500 Mbps it was demanding. These results indicate that the proportional fairness algorithm is properly responsive to changing bandwidth requirements.

4.7 Hybrid Proportional Fairness

Both non-competing (Table 12) and competing (Table 13) scenarios were run on the minimum guarantee setup for both UDP and TCP. The results recorded were the demand for each host (Demand), the theoretical expected throughput according to the hybrid proportional alloca-

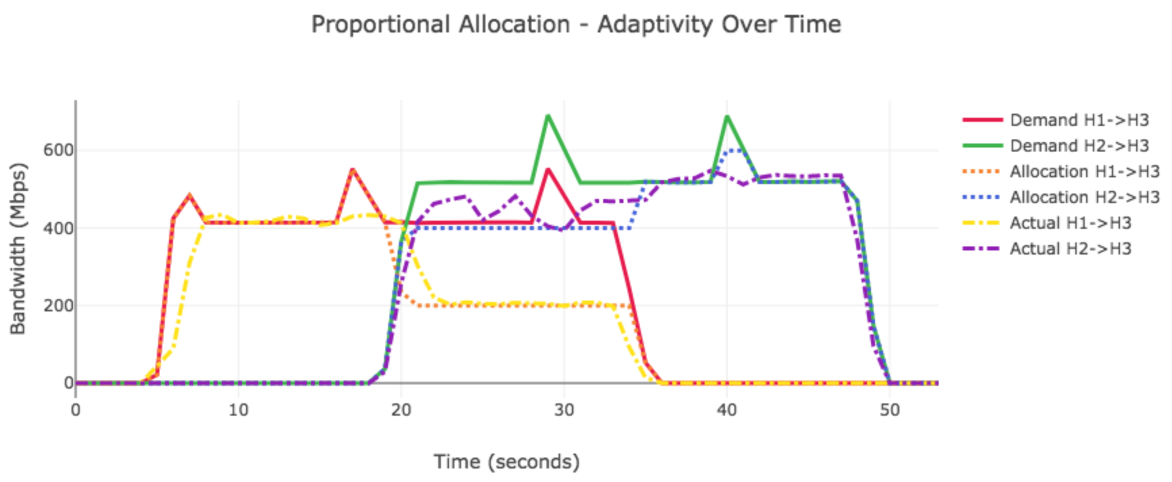

Figure 11: *Proportional allocation - adaptivity over time - Host 1 and Host 2.*

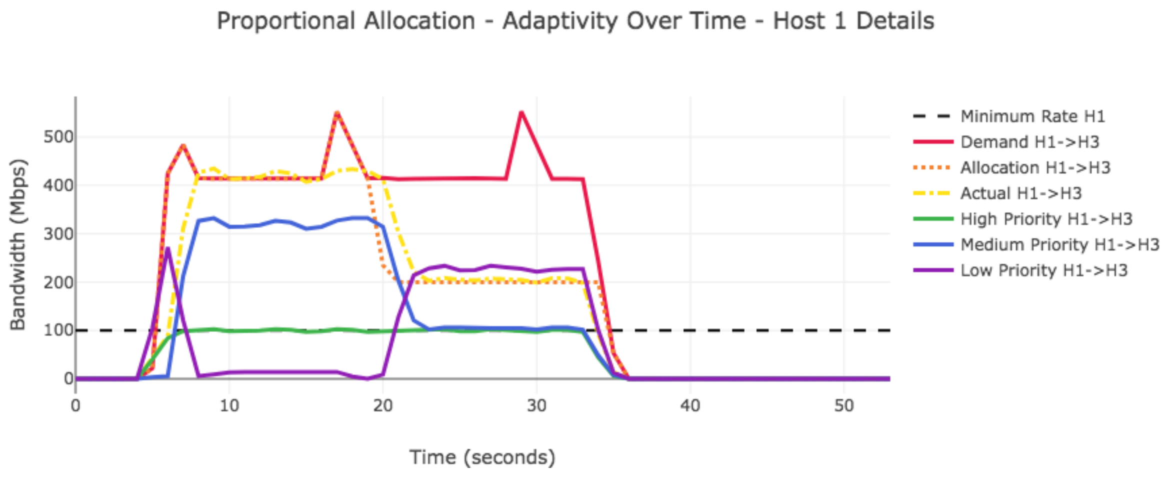

Figure 12: *Proportional allocation - adaptivity over time - Host 1 details.*

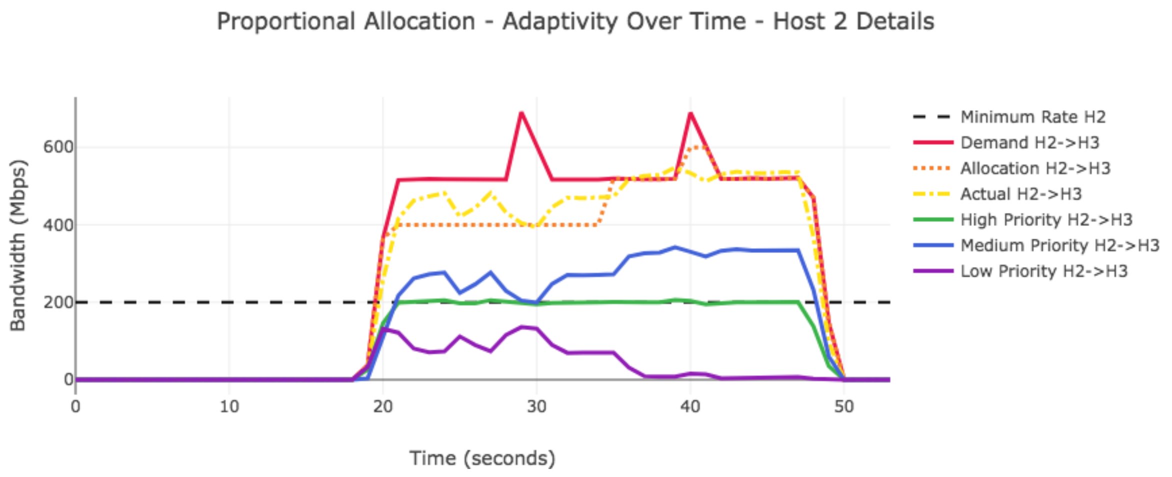

Figure 13: *Proportional allocation - adaptivity over time - Host 2 details.*

tion algorithm (Expected), the actual bandwidth achieved according to iperf (Actual), and the bandwidth going to the high priority (High), medium priority (Mid), and low priority (Low) egress queues according to the controller. Additionally, the dynamic adaptivity test (Figures 14, 15, 16) was run for this setup.

Protocol	Host	Demand	Expected	Actual	High	Mid	Low
UDP	Host 1	200	200	201	98	110	3
	Host 2	300	300	309	199	120	1
TCP	Host 1	200	200	103	75	15	17
	Host 2	300	300	315	197	128	0

Table 12: *Bandwidth usage for non-competing hosts in hybrid proportional fairness setup.*
1/10th of minimum prioritized
Host 1 minimum rate guarantee: 100 Mbps
Host 2 minimum rate guarantee: 200 Mbps
Total available bandwidth: 600 Mbps

The non-competing (Table 12) results are once again positive for UDP traffic, showing that the actual bandwidth was within 5% of the expected bandwidth, and the high priority bandwidth was within 5% of the minimum rate for each host. This indicates that for UDP traffic under hybrid proporional fairness with sum bandwidth demands under link capacity, theoretical expected fairness can be met in full, and minimum rates are given priority. Just as in the previous two algorithms, TCP performance was poor. Host 2 saw an acceptable 5% deviation from actual to expected with minimum rate maintained, but Host 1 deviated 48% below its expected bandwidth, and its high priority traffic fell 25% below its guaranteed minimum rate. This disparity is likely caused by retransmissions due to TCP sequences being separated into different queues, elaborated upon further in the overall results analysis.

Protocol	Host	Demand	Expected	Actual	High	Mid	Low
UDP	Host 1	400	245	238	98	149	180
	Host 2	400	355	369	197	170	37
TCP	Host 1	400	245	80	67	5	13
	Host 2	400	355	419	198	236	4

Table 13: *Bandwidth usage for competing hosts in hybrid proportional fairness setup.*
1/10th of minimum prioritized
Host 1 minimum rate guarantee: 100 Mbps
Host 2 minimum rate guarantee: 200 Mbps
Total available bandwidth: 600 Mbps

The competing (Table 13) results show that actual bandwidth was within 3% of the expected bandwidth, and the high priority bandwidth was within 5% of the minimum rate for each host for UDP traffic. This indicates that for UDP traffic under hyrbid proportional fairness with sum bandwidth demands over the link capacity, theoretical expected fairness can be met in full, minimum rates are given priority as expected, and link capacity is fully utilized. As in the non-competing scenario and previous two competing scenarios, TCP performance was overall very poor. Host 2's actual usage was 18% above the expected with its minimum rate maintained, but Host 1 was 67% below its expected bandwidth, and its high priority traffic fell 33% below its minimum rate. This disparity is likely caused by retransmissions due to TCP sequences being separated into different queues, elaborated upon further in the overall results analysis.

From the dynamic adaptivity test for hybrid proportional allocation (Figures 12, 13, 14), it can be seen that the controller's hybrid proportional fairness algorithm is responsive to changes in bandwidth use by each host. Initially while Host 1 demanded 400 Mbps, it was allocated 400 Mbps as there was no competition. When Host 2 created competition by demanding 500 Mbps, Host 1 was allocated 245 Mbps and Host 2 was allocated 355 Mbps. This is correct according to the hybrid proportional algorithm, as up to 1/10th of the minimum rate for each host was allocated to them (10 and 20 Mbps respectively), and the remaining 270 Mbps of excess bandwidth was split evenly between the two hosts and added to their minimum rates, resulting in 100 + 10 + 135 = 245 Mbps for Host 1, and 200 + 20 + 135 = 355 Mbps for Host 2. Finally when Host 1 ceased to transmit and competition ended Host 2 was allocated the full 500 Mbps it was demanding. These results indicate that the hybrid proportional fairness algorithm is properly responsive to changing bandwidth requirements.

4.8 Overall Result Analysis

Overall UDP performance was very close to expectations, while TCP performance was very poor. This is likely due to TCP sequences and retransmissions. The dynamic allocation system does not have the ability to distinguish TCP packets based on their sequence numbers, and thus may (and very frequently does) split TCP sequences into separate queues as soon as traffic goes above a meter trigger rate. These packets are often very delayed or dropped entirely if there is sufficient congestion, causing frequent timeouts and retransmissions from the client not receiving acknowledgements. This leads to a shrinking of the TCP window size, and a lower overall throughput. This also explains why TCP performance is noticeably higher for one of the two competing hosts. The host with the poor performance is always the host with a lower minimum rate and a higher demand. This means more TCP packets are split into disparate queues earlier, along with a lowered window size and consequently lower throughput earlier. Since the worse perfoming host's throughput drops, the better performing host is allocated more

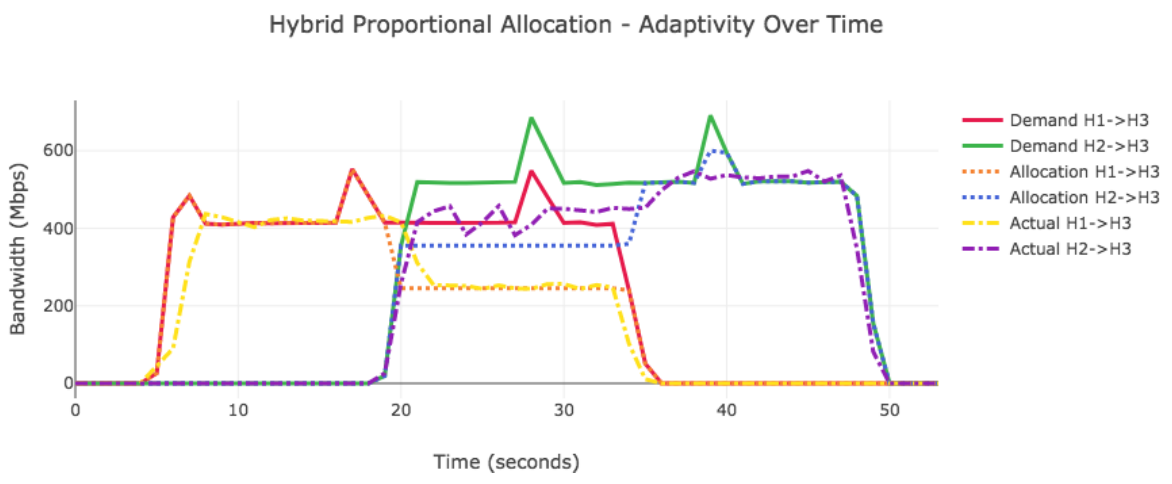

Figure 14: *Hybrid proportional allocation - adaptivity over time - Host 1 and Host 2.*

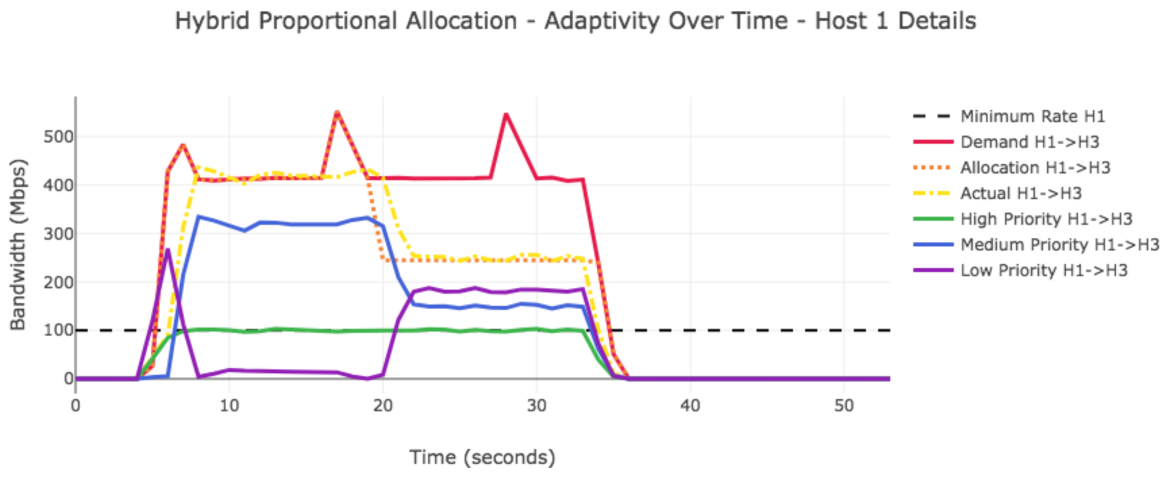

Figure 15: *Hybrid proportional allocation - adaptivity over time - Host 1 details.*

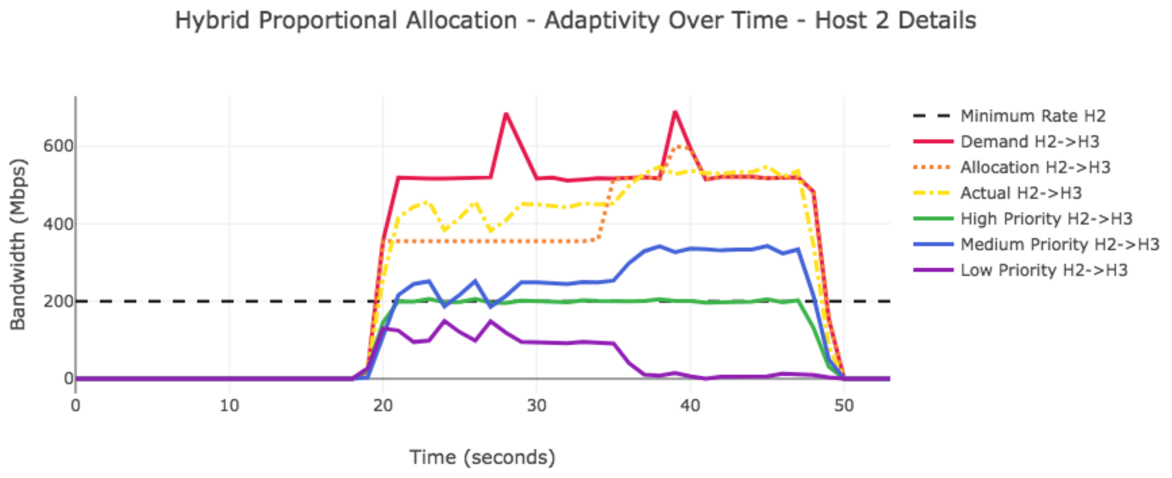

Figure 16: *Hybrid proportional allocation - adaptivity over time - Host 2 details.*

bandwidth by the dynamic allocation algorithms, leading to one host having very poor performance and the other having abnormally high performance. UDP traffic was not affected by this problem since the UDP protocol does not make use of sequencing or retransmission, so packets being dropped due to congestion has no greater effect on throughput.

5 Discussion

Overall, the SDN switch approach to dynamic bandwidth allocation is promising. It provides a platform for a variety of algorithms to operate on, and while the current egalitarian, proportional, and hybrid proportional distribution algorithms may not be ideal, their ability to all run on the same platform provides opportunity for future superior algorithms to improve upon current performance problems.

A superior variant of the project would be one that runs entirely on one switch using multiple flow tables as originall designed. Using multiple switches greatly complicates the implementation of any dynamic allocation platform and instroduces bottlenecks in terms of the links between the switches, which will never be as fast as the the internal connections within one switch. While this was impossible with current hardware, this variant would be feasible should hardware more closely adhering to the OpenFlow standard should enter production.

The logical follow-up research for this project would be to pursue the improvement of TCP performance. There are some mechanisms unused in this project that have the potential to improve TCP performance, such as dynamically changing the meter burst sizes, which dictate the size of the blocks of packets held at the meter before performing the meter operation on them. Higher burst sizes or burst sizes proportional to demand could reduce the amount of dropped packets in a TCP sequence, and minimize the amount of required retransmissions. Weighted Random Early Detection (WRED) egress queues could also improve TCP performance, as they selectively drop packets from the queue during congestion to minimize the amount of packets dropped, again reducing retransmissions.

The key learning gained from this project is that with the current SDN hardware ecosystem, no assumptions can be made about the implementability of a theoretical solution on real hardware, no matter if it would be possible according to the OpenFlow standard. There are many possible avenues for failure, and anyone developing for SDN enabled hardware must be prepared to modify or completely reassess their solutions upon discovering relevant hardware limitations.

6 Conclusion

In conclusion, this project has shown that it is possible to use SDN and an OpenFlow enabled switch to enable dynamic bandwidth allocation of excess bandwidth while maintaining minimum rate guarantees for UDP traffic in a high volume network. This can be done by using the OpenFlow meter construct along with DSCP remark to create a traffic tiering system, in which minimum rates can be guaranteed, and excess bandwidth can be allocated fairly and dynamically. Unfortunately to do so requires comprimising on many design choices due to hardware limitations, as the SDN hardware ecosystem has yet to reach full compliance with the OpenFlow standard. While such a system has good performance with UDP traffic, TCP traffic is far less performant due to the lack of methods to avoid dropping or reordering the sequence of TCP packets, causing significant retransmissions and low effective bandwidth usage. With the current hardware limitations and low ability to interact with TCP packets, this solution cannot currently be recommended for production use in a real network, though with the rapid advance of the SDN field it may become feasible in the near future.

References

[1] iperf. `https://sourceforge.net/projects/iperf/`. Accessed: 2019-04-10.

[2] T. Balogh, D. Luknrov, and M. Medveck. Performance of round robin-based queue scheduling algorithms. In *2010 Third International Conference on Communication Theory, Reliability, and Quality of Service*, pages 156–161, June 2010.

[3] S. Dwarakanathan, L. Bass, and L. Zhu. Cloud application ha using sdn to ensure qos. In *2015 IEEE 8th International Conference on Cloud Computing*, pages 1003–1007, June 2015.

[4] Open Networking Foundation. What is openflow? definition and how it relates to sdn. [Online]. Available: `https://www.opennetworking.org/sdn-definition/`. [Accessed 08-Oct-2018].

[5] Paul Goransson and Chuck Black. Chapter 4 - how sdn works. In Paul Goransson and Chuck Black, editors, *Software Defined Networks*, pages 59 – 79. Morgan Kaufmann, Boston, 2014.

[6] B. G. Kim. The soft qos service (sqs) in the internet. In *Joint 4th IEEE International Conference on ATM(ICATM'01) and High Speed Intelligent Internet Symposium. ICATM 2001 (Cat. No.00EX486)*, pages 56–60, April 2001.

[7] H. Krishna, N. L. M. van Adrichem, and F. A. Kuipers. Providing bandwidth guarantees with openflow. In *2016 Symposium on Communications and Vehicular Technologies (SCVT)*, pages 1–6, Nov 2016.

[8] SDxCentral. Software-defined networking (sdn) definition. [Online]. Available: `https://www.sdxcentral.com/sdn/definitions/what-is-openflow/`. [Accessed 08-Oct-2018].

[9] S. Tomovic, N. Prasad, and I. Radusinovic. Sdn control framework for qos provisioning. In *2014 22nd Telecommunications Forum Telfor (TELFOR)*, pages 111–114, Nov 2014.

[10] N. L. M. van Adrichem, C. Doerr, and F. A. Kuipers. Opennetmon: Network monitoring in openflow software-defined networks. In *2014 IEEE Network Operations and Management Symposium (NOMS)*, pages 1–8, May 2014.

[11] Ryan Wallner and Robert Cannistra. An sdn approach: Quality of service using big switch's floodlight open-source controller. *Proceedings of the Asia-Pacific Advanced Network*, 35:14, 06 2013.