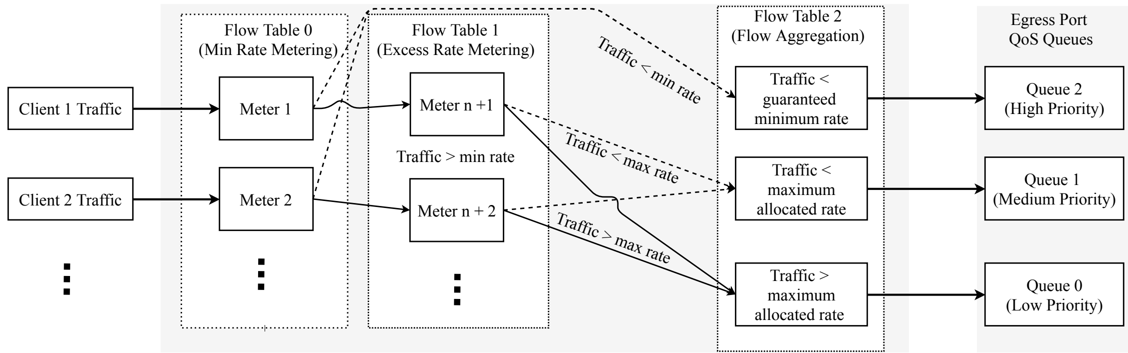

Figure 1: Three-tier flow aggregation and metering for fair and efficient dynamic bandwidth allocation

Table 1: Results for bandwidth utilization of competing TCP traffic from two clients. (Link capacity 600, rates in Mbps)

Algorithm	Client	Min-Rate	Demand	Fair Share	Delivered
Max-Min Fair Exact-Allocation	H1 H2	100 200	400 600	250 350	85 519
Max-Min Fair Over-Allocation	H1 H2	100 200	400 600	250 350	234 324

3 DBA FAIR UTILIZATION

The DBA algorithm is implemented within the SDN controller. The controller monitors the real-time bandwidth usage for each client through the flow statistics at each tier, and dynamically adjusts the metering rate at the second tier in order to maximize utilization. Different allocation algorithms are possible through this design. We have implemented a max-min fair allocation algorithm, which gives an equal share of the spare bandwidth to all clients upon demand, regardless of their committed purchased rate.

There are subtleties in how the spare bandwidth is allocated. With exact-allocation, where the sum of all allocated bandwidth is no more than the link capacity, either the link remains underutilized (when allocation is based on equal share of the spare bandwidth), or the TCP congestion control for some clients adapts to a lower rate than their fair share (when allocation is based on the perceived demand by the controller). To achieve maximum utilization and maintain max-min fairness, we use an over-allocation scheme, which allocates the equal share to clients with low perceived demand (even though they may not utilize it), and allocates the fair share of the unutilized spare bandwidth to clients with high demand. The over-allocation stops when traffic reaches the fair allocation rate for all clients.

4 IMPLEMENTATION AND RESULTS

We have implemented a proof-of-concept testbed using SDN enabled Pica8 (P-3290) switches and Ryu SDN controller with REST API (source code available on GitHub [8]). These switches only implement one stage of the OpenFlow pipeline, and therefore do not support pipeline processing instructions such as `goto_table`. Therefore, we used three switches to implement the three-tier flow aggregation and metering.

Table 1 shows utilization results for competing TCP traffic. With exact-allocation based on perceived demand, TCP congestion control for the client with the lower min guaranteed rate adapts to a lower rate than its fair share, and the client with higher min rate gets more than its fair share. With over-allocation, both clients get close to their fair share. Figure 2 illustrates the adaptivity of our allocation scheme over time. The TCP congestion control quickly adapts to the max-min fair rate when over-allocation is applied.

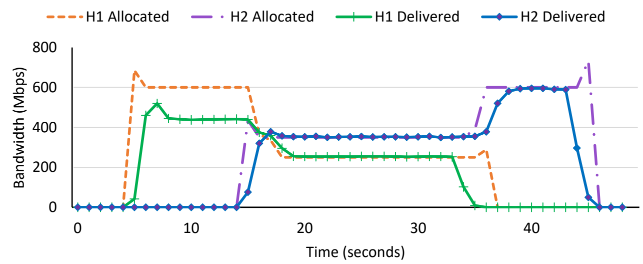

Figure 2: Adaptivity of the max-min fair DBA algorithm over time. Client H1 with guaranteed rate 100 Mbps tries to transmit at rate 400 Mbps for 30s. H1 initially utilizes close to 400 Mbps through the link without competition. Client H2 with guaranteed rate 200 Mbps starts competing for bandwidth with 10s delay, and tries to transmit at rate 600 Mbps for 30s. The algorithm quickly adjusts the rate for H1 and H2. When H1 stops transmission, H2 is able to utilize the available spare bandwidth and transmit at near 600 Mbps.

5 CONCLUSIONS

We present a three-tier flow-aggregation and metering design using OpenFlow that provides fair and efficient DBA for networks with minimum bandwidth guarantees. Initial evaluation on an SDN testbed shows that to achieve max-min fair utilization of the spare bandwidth by TCP flows, an over-allocation scheme is required. Our ongoing work include investigating the interplay between the TCP congestion control and other fair allocation algorithms (e.g. proportional fair), and evaluating the scalability of our solution.

REFERENCES

[1] W. Aljoby, X. Wang, T. Fu, and R. Ma. 2018. On SDN-enabled online and dynamic bandwidth allocation for stream analytics. In *Proc. of IEEE ICNP*. 209–219.
[2] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. 1998. RFC 2475: An architecture for differentiated services.
[3] D. Clark, R Braden, and S. Shenker. 1994. RFC 1633: Integrated services in the internet architecture: an overview.
[4] T. Flach, P. Papageorge, A. Terzis, L. Pedrosa, Y. Cheng, T. Karim, E. Katz-Bassett, and R. Govindan. 2016. An Internet-Wide Analysis of Traffic Policing. In *Proc. of ACM SIGCOMM Conference* (Florianopolis, Brazil). 468–482.
[5] H. Krishna, N. van Adrichem, and F. Kuipers. 2016. Providing bandwidth guarantees with OpenFlow. In *Proc of SCVT*. 1–6.
[6] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. 2008. OpenFlow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review* 38, 2 (2008), 69–74.
[7] S. Nickolay, E. Jung, R. Kettimuthu, and I. Foster. 2018. Bridging the gap between peak and average loads on science networks. *Future Generation Computer Systems* 79 (feb 2018), 169–179.
[8] J. van Egmond and M. Elahi. 2019. Ryu REST Controller for Fair Dynamic Bandwidth Allocation. https://github.com/bmelahi/ryuRestDBA.git.