Packet Scheduling techniques based on Software Defined Networking

Runge Yan, 906252597, yrunge@vt.edu
Yiqing Liu, 906297543, yqliu@vt.edu

Abstract
Software Defined Networking (SDN) inspires technical innovations for wired and wireless network resource management. It's a good approach to achieve Quality of Service (QoS) for diverse applications with different QoS requirements. In this project, we enable SDN switches with multiple package scheduling techniques, which can diversify the QoS provisioning based on SDN. We described and implemented four packet scheduling techniques supported by legacy switches, including SFQ (Stochastic Fairness Queueing), PQ (Priority Queueing), CBWFQ (Class-Based Weighted Fair Queueing) and LLQ (Low Latency Queueing). We set up network environment using Mininet on Ubuntu to conduct experiment and analyze the performance of the techniques. Turned out the four techniques give different performance on same packet flow. Each of them has its strength and weakness.

Introduction
SDN enhances the flexibility of resource management for wired and wireless networks [1]. With centralized control, global network view and programmability, SDN is considered to be a better approach to guarantee QoS compared with previous proposed service models, e.g., Differentiated Service (Diffserv) [2] and Integrated Services (IntServ) [3].

Durner et al. [4] study the realization of QoS mechanism for OpenFlow-enabled SDN switches in examining dynamic TCP QoS support of OpenFlow. Dwarakanathat et al. [5] propose a module in Floodlight controller [6] to ensure QoS by reserving bandwidth for individual applications. The module also supports dynamic queue configuration on OpenFlow based switches. Caba et al. [7] argue that the mechanisms of QoS enforcement in the data plane devices needs more consideration. They propose an architecture which incorporates data plane QoS mechanism and control plane management logic and present details for prototype implementation of such a platform. Morin et al. [8] highlight that as opposed to queues, meter as an OpenFlow tool can also be used to share the bandwidth among multiple flows, although it is not widely supported yet. Chato et al. [9] examine rudimentary Class-Based Queueing (CBQ) scheduling algorithm in OpenFlow environment and find that leaf switches provide better performance compared to core switches. Sieber et al. [10] introduce a unified data model to manage and configure both SDN and legacy devices in order to achieve QoS. Tsung-Feng et al. [11] present an approach to support video streaming with QoS by treating different layer packets in different manner.

Existing work have proved the benefits of SDN to achieve QoS provisioning for different applications. We implement and analyze a series of packet scheduling techniques for end-to-end QoS routing and multipath routing.

In this project:
(1) We implement four typical packet scheduling techniques in SDN switches, including SFQ, PQ, CBWFQ and LLQ. We create multiple parallel queues for output interfaces in SDN switch.

(2) We evaluate these techniques on Open vSwitch. Experimental results show that SFQ treats the packets in the queues equally, while PQ gives priority to forwarding the packets in the queue with a higher priority. CBWFQ allocates the bandwidth to each queue proportionally

when the network is congested, while LLQ provides a strict guarantee on QoS for the packets in the queue with the highest priority.

(3) According to the OpenFlow switch specifications [12], an OpenFlow switch can only provide limited QoS support with queueing technique. Packets, after mapped to a specific queue, will be scheduled according to the queue configuration (e.g., min rate), and the process is switch dependent. In this paper, we implement multiple queues on the output port of the switch. Those queues can be used to schedule packets exiting that port with different algorithms, which can diversify the QoS provisioning based on SDN.

Packets Scheduling Techniques

(1) Stochastic Fairness Queueing
SFQ [13] treats all the data flows equally without discrimination. When the network is congested, the bandwidth is evenly assigned to all the flows. That is to say, SFQ achieves bandwidth friendly fairness, but it cannot guarantee the bandwidth and real-time performance for the flows.

(2) Priority Queueing
PQ [14] gives preference to forwarding the packets in the queue with a higher priority. However, for lower priority packets in the queue, they may encounter the starvation problem. PQ is applicable to the flows marked with different priorities and it can guarantee the real-time performance for the flows.

(3) Class-Based Weighted Fair Queueing
CBWFQ [15] separates the traffic into different classes, each of which corresponds to a queue. It determines the transmission speed of the queues, and then weights the priority of each queue. The weighted value of each queue is set in proportion to the bandwidth allocated to the flows.

(4) Low Latency Queueing
LLQ [16] brings a strict priority queue compared to CBWFQ. It queues delay-sensitive application packets to the priority queue so that they can be forwarded as fast as the network allows. LLQ can provide QoS guarantee for the real-time applications.

Table 1 Four typical packet scheduling techniques

Name	Priority	Weight	Characteristics
SFQ	No	No	assigns the bandwidth on average, but the bandwidth and real-time performance is not guaranteed
PQ	Yes	No	gives preference to forwarding the packets in the queue with a higher priority
CBWFQ	No	Yes	separates the traffic into different classes and assigns the bandwidth proportionally
LLQ	Yes	Yes	brings a strict priority queue to CBWFQ and provides QoS guarantee for the real-time applications

Implementation of Packet Scheduling Techniques

(1) SFQ
SFQ is a typical fair queue scheduling technique, which requires only a small amount of computation to achieve a higher degree of fairness. SFQ operates by maintaining multiple First-In-First-Out (FIFO) queues. Each queue corresponds to a stream or a connection. Each stream or connection gets sent in order. This scheduling technique ensures that every flow or connection will not be overwhelmed by other traffic.

Implementation:

SFQ checks if the round-robin pointer points to an empty queue. If true, increment the pointer, if false, schedule the packet out of the queue and increment the pointer.

(2) PQ
PQ schedules the packets from the queues with the same priority according to the principle of first come, first serve. However, for the queues with different priorities, the packets are selected according to the queue priorities. Packets in the queue with the highest priority are first scheduled out. When the queue with the highest priority is empty, packets in the queue with the second highest priority will be served, and so on.

Implementation:
PQ checks if the queue(s) of the highest priority level is empty. If not, schedule all the packets in this queue out; otherwise, move on to the next priority level. If all the queues on the same level are empty, move on to the lower priority levels.

(3) CBWFQ
CBWFQ divides the traffic into different classes and each class corresponds to a queue. The transmission speed of each queue is determined by a weighted value. The weighted value of each queue can be calculated from its minimum packet sending rate, which is specified by the operator. When a certain type of queue is empty, the bandwidth assigned to this queue can be allocated to other queues proportionally according to the predefined weights. Thus, the bandwidth utilization can be improved greatly.

Implementation:
Packets are classified into classes based on its QoS feature. With SFQ as a base, if there's congestion in the interface, CBWFQ allocates bandwidth for each queue according to the weighted value.

(4) LLQ
LLQ is a combination of PQ and CBWFQ. Typically, there is one Priority Queue and some Class-Based Weighted Fair Queues. LLQ queues real-time application packets to the priority queue all the other packets are queued into the weighted fair queues. The priority queue is served first, so the real-time application packets can be processed as fast as the interface allows. When the priority queue is empty, the scheduler will serve the weighted fair queues. However, if a packet is queued into the priority queue, the scheduler will process the priority queue immediately after scheduling the current packet out of a weighted fair queue. Therefore, LLQ can provide a strict QoS guarantee for the real-time applications.

Implementation:
If the priority queue is not empty, schedule all the packets in it out of the output interface. Otherwise, if there's congestion in the interface, LLQ allocates bandwidth to weighted fair queue according to the weighted values. All the other rules follow the SFQ.

Experiment Analysis

1. Experiment Design
The network topology for experiment is shown in Fig. 1. An Open vSwitch (OVS) connects to an SDN controller as well as eight hosts. We implement four kinds of packet scheduling techniques on OVS. The queue parameters are shown in Table 2. We created four queues at each interface. The queue of SFQ are set with two parameters of Min-rate and Max-rate, while the queues of PQ, CBWFQ and LLQ are set with one more parameter of Priority.

We deploy four client hosts sending UDP flows to four server hosts respectively. In order to observe how each packet scheduling technique works, we send the flows at different times. Table 3 shows the names of each flow and the sending time of each flow between a client host and a server host. These four flows will be queued into the queues we created.

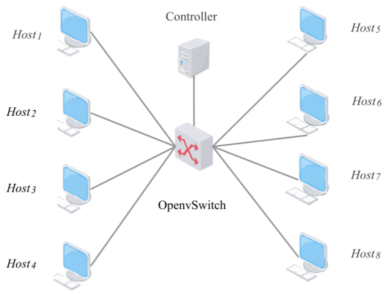


Fig. 1 Network Topology

Table 2 parameters of the four queues

	Parameters	q0	q1	q2	q3
SFQ	Min-rate(kbit/s)	80	80	80	80
	Max-rate(kbit/s)	8000	8000	8000	8000
PQ	Min-rate(kbit/s)	80	80	80	80
	Max-rate(kbit/s)	8000	8000	8000	8000
	Priority	1	2	2	3
CBWFQ	Min-rate(kbit/s)	800	1600	2400	3200
	Max-rate(kbit/s)	8000	8000	8000	8000
LLQ	Min-rate(kbit/s)	8000	80	160	240
	Max-rate(kbit/s)	8000	8000	8000	8000
	Priority	1	2	2	2

Table 3 The four experimental flows

Type	Flows	Sending times(s)
flow_1	*Host1-Host5*	0
flow_2	*Host2-Host6*	10
flow_3	*Host3-Host7*	20
flow_4	*Host4-Host8*	30

2. Experiment Analysis

The rules of the flow tables are set through the controller. Iperf is used to evaluate the performance from the aspects of bandwidth, jitter and packer-loss rate. At last, performance is analyzed, and the results are shown in Fig. 2 ~ Fig. 4.

(1) Bandwidth:

Fig. 2 shows the bandwidth utilization of the queues when using different packet scheduling techniques. It can be seen that when the sending rate is greater than the limited rate and there are more than one packet arriving, SFQ forwards the packets in turn. And when the load reaches to total bandwidth limitation, SFQ forwards the packets on average.

In the test of SFQ, the sending rate of the four flows are all 8 Mbit/s. Fig. 2(a) shows that when flow_3 enters the interface at 10s and the load reaches to bandwidth limitation, the bandwidth of flow_4 decreases to 3.88 Mbit/s, sharing the bandwidth with flow_3 on average. When flow_2 enters the interface at 20s, the bandwidth of flow_4 and flow_3 decreases to 2.60 Mbit/s, sharing the bandwidth with flow_2 on average. Finally, flow_1 enters the interface at 30s, which leads to the equal bandwidth allocated for the four queues.

In the test of PQ, the sending rate of the four flows are all 5 Mbit/s. As shown in Fig. 2(b), flow_4 first enters the interface and flow_3 enters the interface at 10s. It does not lead to interface congestion until flow_2 enters the interface at 20s. Since the priority of flow_4 is the lowest, its bandwidth decreases to 0.07 Mbit/s. Then at 30s, flow_1 with the highest priority enters the interface. It causes the bandwidth decrease of flow_3 and flow_4. The bandwidth of flow_3 decreases to 1.35 Mbit/s, and the bandwidth of flow_4 decreases to 0.07 Mbit/s. In addition, since the priority of flow_4 is the lowest, bandwidth assigned to it is also the least among the queues.

In the test of CBWFQ, the sending rate of the four flows are all 8 Mbit/s. As shown in Fig. 2(c), flow_4 enters the interface first, and flow_3 enters the interface at 10s. The bandwidth assigned to flow_3 and flow_4 is limited because they encounter interface congestion. The bandwidth of flow decreases to 3.29 Mbit/s, and the bandwidth of flow_4 decreases to 4.68 Mbit/s. In CBWFQ, the bandwidth is assigned according to the ratio of the minimum sending rate of the packets in the queue. As depicted in Fig. 2(c), the ratio is around 1:2:3:4. And when flow_1 enters the interface at 30s, the bandwidth of the four flows turns to 3.15 Mbit/s, 2.33 Mbit/s, 2.56 Mbit/s and 0.80 Mbit/s.

In the test of LLQ, the sending rate of the. Four flows are all 5 Mbit/s. Fig. 2(d) shows the test results of LLQ. The bandwidth in LLQ is limited as predefined. The packets come afterwards will be dropped until there is enough available bandwidth. As shown in Fig. 2(d), flow_4 enters the interface first, and flow_3 enters the interface at 10s, which leads to interface congestion. As a result, the bandwidth of flow_4 decreases to 4.66 Mbit/s. When flow_2 enters the interface at 20s, it cannot get too much bandwidth due to that its priority is equal to flow_3 and flow_4. However, when flow_1 with the highest priority arrives at 30s, it has priority to the bandwidth. As a result, the bandwidth assigned to flow_2, flow_3 and flow_4 decreases. Results show that LLQ serves the priority queue first. And bandwidth assigned to other three weighted fair queues is based on the ratio of the minimum sending rate of the flows.

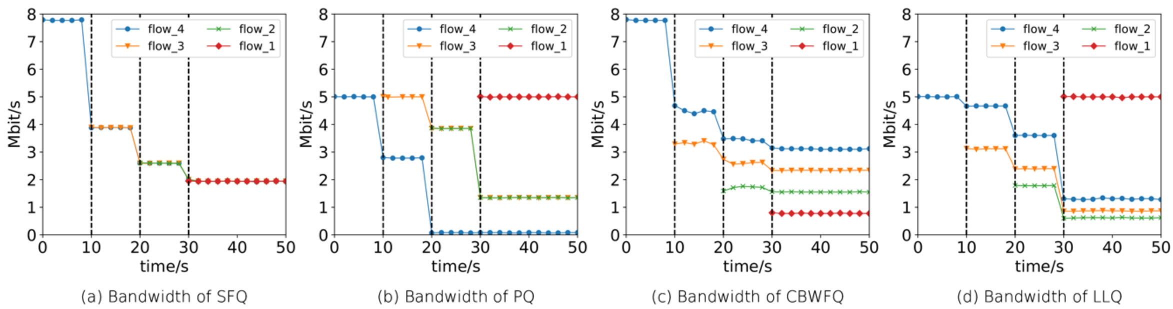


Fig. 2 Bandwidth performance

(2) Jitter

Long queues in a data processing time often lead to jitter – a variation of delay. If the interface is congested, queueing delay will affect end-to-end delay and cause packet delays transmitted over the same connection to be different. And jotter is used to describe the extent of such a delay change.

As shown in Fig. 3, SFQ is suitable for the data flow sensitive to jitter such as video applications, because the jitter change in each queue remains stable, and the state of the transmission link is stable. The PQ gives priority to flow_1 to meet its requirement when the interface is congested, because the priority of flow_1 is the highest. Jitter change is not obvious when it is in high-speed forwarding state. Since the priority of flow_2 is same to that of flow_3, they present the similar jitter changes. While flow_4 has the lowest priority, it occupies the least amount of bandwidth resource and has a low transmission rate with an obvious jitter. In

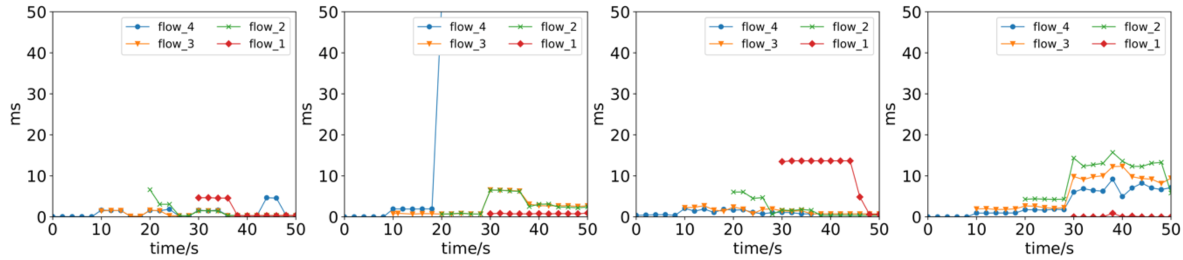
CBWFQ, the jitter varies in a wide range when the interface is congested, especially in the low-speed link. So, it is more suitable for the delay-insensitive and jitter-insensitive data flows such as data applications. In LLQ, flow_1 has the highest priority. To meet the required bandwidth, the jitter change is almost 0. In addition, for flow_2, flow_3 and flow_4, the jitter changes in a similar way because they have the same priorities.

In order to more intuitively analyze the characteristics of these four kinds of packet scheduling techniques, we calculate the average jitter to conduct a comparative analysis. The results are shown in Table 4.

According to the results shown in Table 4, it can be seen that both SFQ and CBWFQ present a certain degree of jitter changes. The jitter in SFQ changes within a certain range from 0.66ms to 4.07ms. It is suitable for the delay-sensitive and jitter-sensitive data transmission. While the jitter in CBWFQ is larger, ranging from 0.97ms to 6.89ms. So, it is suitable for data stream transmission. In the comparison of PQ and LLQ, the priority of flow_1 is the highest. In PQ, it appears slightly jitter change and the jitter of flow_1 is 0.80ms. While in LLQ, it nearly has no jitter and the jitter of flow_1 is 0.02ms. That means LLQ has a good delay guarantee and it is more suitable for the network applications that require high real-time performance.

Table 4 average jitter of the four queues

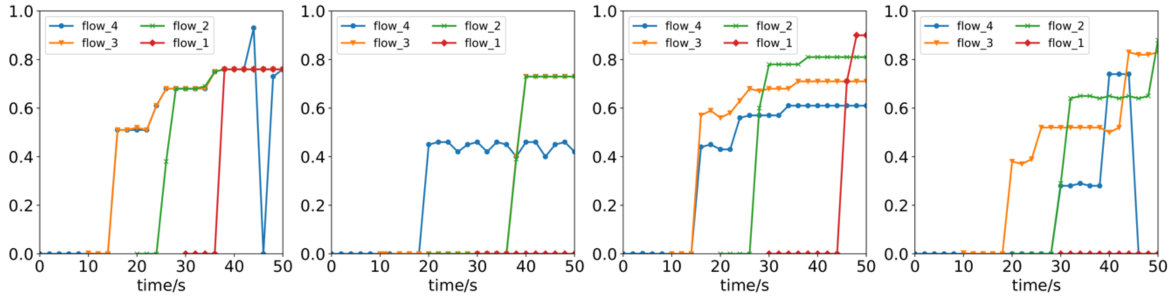
average of jitter(ms)	SFQ	PQ	CBWFQ	LLQ
flow_1	1.26	0.80	6.89	0.02
flow_2	4.07	3.00	1.94	9.28
flow_3	1.27	3.88	2.00	15.84
flow_4	0.66	93.60	0.97	3.78



(a) Jitter of SFQ (b) Jitter of PQ (c) Jitter of CBWFQ (d) Jitter of LLQ

Fig. 3 Jitter performance

(3) Packet-loss Rate: Packet-loss rate is the ratio of the lost packets to the total number of packets. The results of packer-loss rate are shown in Fig. 4. It shows that when the interface is in congestion, the packet-loss rate is high in SFQ. SFQ cannot guarantee all the packets are forwarded. In PQ, flow with the highest priority is loss free in forwarding. In CBWFQ, bandwidth is allocated according to the ratio of the minimum sending rate of the packets in the queues. At the same time, the packet-loss rate is higher when less bandwidth is allocated. In LLQ, it is guaranteed that the flow with higher priority will be forwarded with less delay.



(a) Packet-loss Rate of SFQ (b) Packet-loss Rate of PQ (c) Packet-loss Rate of CBWFQ (d) Packet-loss Rate of LLQ

Fig. 4 Packet-loss Rate Performance

3. Analysis

Four scheduling techniques show strength and weakness under certain network condition.

(1) SFQ treats all the data flow equally and shows fairness in bandwidth and packet-loss rate, but it does not guarantee the bandwidth and real-time performance. Therefore, SFQ is suitable for the streaming media application, such as video on demand.

(2) PQ ensures that the prioritized traffic is forwarded with the least delay and all other traffic can be handled when the highest priority queue is empty. PQ is applicable to streams with distinct priorities and provides a real-time guarantee for different network applications, such as online games and VoIP.

(3) CBWFQ does not guarantee the real-time performance. However, compared with SFQ, it has obvious bias in bandwidth allocation. Therefore, CBWFQ is suitable for background applications such as email transmission, FTP downloading, etc.

(4) LLQ guarantees the delay-sensitive traffic, typically the voice traffic. It provides the top priority class low latency, while the classes with lower priorities will tolerate a certain degree of delay. Compared with PQ, there is almost no jitter in the highest priority queue. So LLQ is often applied to remote surgery in hospitals, IP telephony in enterprise and video conference.

Future Work

(1) Our program runs well under ipv4, ipv6. However, we didn't come up with specific method for ipv6. As it gradually takes over ipv4, we can focus on further improvement under ipv6 network

(2) Although we come up with the characteristics of the four scheduling techniques, certain recognition method is needed to identify the type of application data so that we can map the data flow to our queues.

Reference

[1] Kreutz, Diego, et al. "Software-defined networking: A comprehensive survey." *arXiv preprint arXiv:1406.0440* (2014).

[2] S. Blake, D. Black, M. Carlson, et al. An Architecture for Differentiated Service. IETF FC2475, (1998) December, pp. 2-9.

[3] R. Braden, D. Clark, S. Shenker. Integrated Services in the Internet Architecture: an Overview. IETF RFC1633, (1994) June, pp. 2-3.

[4] R. Durner, A. Blenk, W. Kellerer. Performance study of dynamic QoS management for OpenFlow-enabled SDN switches. 2015 IEEE 23rd International Symposium on Quality of Service (IWQoS), 2015, pp. 177-182.

[5] S. Dwarakanathan, L. Bass, L. Zhu. Cloud Application HA Using SDN to Ensure QoS. 2015 IEEE 8th International Conference on Cloud Computing, 2015, pp. 1003-1007.

[6] Floodlight Home Page, http://docs.projectFloodlight.org.

[7] Cosmin Caba, Jos Soler. SDN-based QoS Aware Network Service Provisioning. Mobile, Secure, and Programmable Networking. Springer, 2015. pp. 119-133 (Lecture Notes in Computer Science, Vol. 9395).
defined networks. 2017 IEEE International Symposium on Broadband

[8] C. Morin, G. Texier, C. Phan. On demand QoS with a SDN traffic engineering management (STEM) module. 2017 13th International Conference on Network and Service Management (CNSM), 2017, pp. 1-6.

[9] O. Chato,W. E. S. Yu. An exploration of various quality of service mechanisms in an OpenFlow and software defined networking environment. 2016 3rd International Conference on Systems and Informatics (ICSAI), 2016, pp. 768-776.

[10] C. Sieber, A. Blenk, D. Hock, M. Scheib, T. H¨ohn, S. K¨ohler, W. Kellerer. Network configuration with quality of service abstractions for SDN and legacy networks. 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), 2015: 1135-1136.

[11] Tsung-Feng Yu, Kuochen Wang and Yi-Huai Hsu. Adaptive routing for video streaming with QoS support over SDN networks. 2015 International Conference on Information Networking (ICOIN), 2015, pp. 318-323.

[12] OpenFlow Switch Specification. https://www.opennetworking.org

[13] P. E. McKenney. Stochastic fairness queueing. In INFOCOM'90, Ninth Annual Joint Conference of the IEEE Computer and Communication Societies. The Multiple Facets of Integration. Proceedings, 1990: 733-740.

[14] R. J. T. Morris. Priority queuing networks. The Bell System Technical Journal, 60(8), 1981: 1745-1769.

[15] S. Badr, F. Bayoumi, G. Darwesh. QoS adaptation in real time systems based on CBWFQ. In 2011 28th National Radio Science Conference (NRSC), 2011: 1-8.

[16] B. Dekeris, T.Adomkus, A. Budnikas. Analysis of QoS assurance using weighted fair queueing (WQF) scheduling discipline with low latency queue (LLQ). In 28th International

Appendix

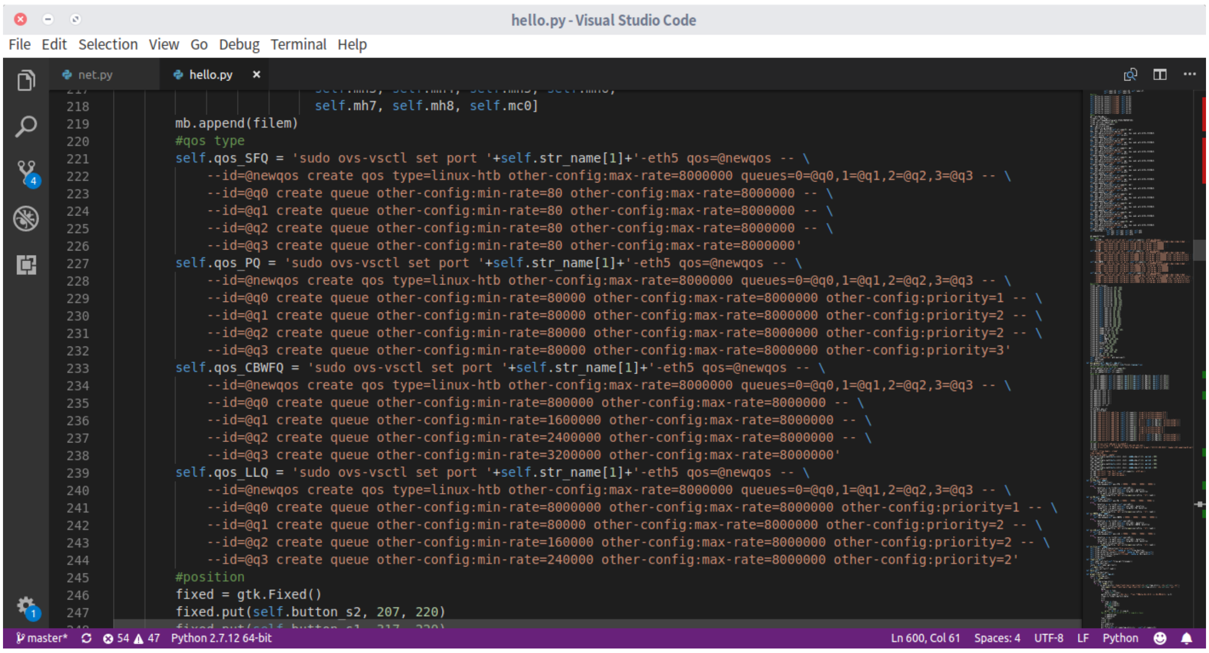


Fig. 1 Parameters for the four scheduling techniques

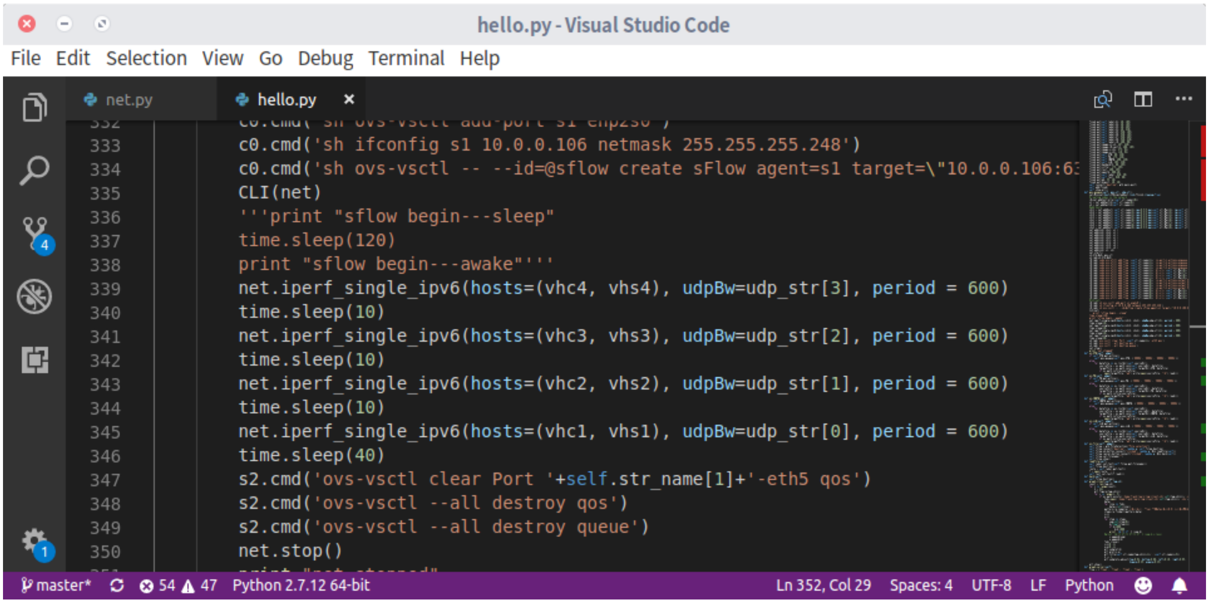


Fig. 2 Parameters for Iperf measurement