WIRELESS NETWORK SLICING ON MAC SCHEDULING

A Thesis

by

LICHEN FU

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	I-Hong Hou
Committee Members,	Serap Savari
	Chao Tian
	Duncan M. "Hank" Walker
Head of Department,	Miroslav M. Begovic

December 2020

Major Subject: Electrical Engineering

ABSTRACT

Networks are constantly flooded by heterogeneous services, and while some services such as Virtual Reality (VR) and Voice over IP (VoIP) have strict requirements regarding real-time packet delivery, others such as downloading large files have non-real time overflow traffic.

In the past years, research community had sufficient study to provide optimal scheduling policies under different requirements on the network communication. However, in the studies, a distinct lack of network fluidity exists: there has only been one network base station supporting multiple services, and the optimal scheduling policy for one service doesn't necessarily serve the other services ideally.

In this thesis, I present a new software-defined MAC system that employs network slicing to support multiple heterogeneous scheduling policies in a uniform framework. After successfully constructing the testbed, I further integrated Software Defined Networking (SDN) into the testbed as a tool to manage slicing and scheduling configuration. With the help of SDN, I designed and implemented an algorithm to effectively approach and calculate the optimal resource allocation between slices. I'm able to demonstrate that the testbed outperforms the system with baseline single scheduling policy. In addition, the proposed algorithm can further increase the system performance.

DEDICATION

To my parents, and my friends in College Station.

ACKNOWLEDGMENTS

I would like to especially thank Dr. Hou for being my advisor and providing guidance along this adventurous journey. Working with you is my most precious experience in Texas A&M University.

I will thank Daojing Guo for his dedicated help whenever I'm in trouble and consistent support along the jouney, I couldn't have done it without you.

I'd also like to thank Dr. Tian, Dr. Savari and Dr. Walker for being my committee members.

Last but not least, I would like to thank Tao, David, Khaled and Siqi for their help in construction of SDN platform. I'm very fortunate to meet and work with you in Dr. Hou's group.

NOMENCLATURE

PULS	Processor-supported Ultra-low Latency Scheduling
WNV	Wireless Network Virtualization
USRP	National Instrument USRP Software Defined Radio Device
MAC	Medium Access Control
Wi-Fi	Wireless local area network based on IEEE 802.11 standard
AP	Access Point
ACK	Acknowledge packet
FPGA	Field Programmable Grid Array
VR	Virtual Reality
IIoT	Industrial Internet-of-Things
SDR	Software-Defined Radio
ms	Millisecond
μs	Microsecond
DBQ	Deficit Based Queueing Policy
LQF	Longest Queue First Scheduling Policy
RR	Round-Robin Scheduling Policy
PF	Proportional Fair Scheduling Policy

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

1. INTRODUCTION

1.1 Motivation

A critical challenge for next-generation wireless networking is the providing of stringent service guarantees for heterogeneous user cases. The 5G network concept is expected to support a wide range of emerging applications and devices with different characteristics. Some applications such as Virtual Reality (VR) and tactile Internet usages require strict, ultra-low latency packet delivery, while some applications like downloading and Internet of Things (IoT) sensors simply require high throughput over time. The modern research community has proposed numerous theoretical solutions for the differing service requirements. For example, [1] proposed and proved the optimal scheduling policy regarding traffic with per-packet deadline constraints. [2] proved that if a maximum weight matching algorithm is used, 100% throughput is achievable for both uniform and nonuniform arrivals. [3] provided a mathematical footing and analysis on proportional fair scheduler and showed that it corresponds to a reasonable maximization problem under elastic traffic. However, all of these different applications need to thrive on one network infrastructure. This heterogeneous situation generates a challenge when high quality services are needed to be provided to each one of the applications with one single server, because the optimal policy for one application doesn't necessarily serve the other applications well. Although one possible solution is to use a dedicated end-to-end network for every service; however, this will significantly increase deployment and operation cost.

1.1.1 PULS

PULS [4] is an implementation of processor-supported ultra low latency scheduling to simulate MAC Layer scheduling policies with ultra-low latency requirements. It consists of 2 parts, a host machine and a Field-Programmable Gate Array (FPGA) based software defined radio. The host machine is implemented with scheduling policies, while the FPGA only handles packet delivery at lower level with fast execution. This structure decouples the scheduling policy and traffic handling,

makes it an ideal testbed for scheduling algorithm experiments. PULS is able to transmit and schedule packets within one millisecond, which makes it even more powerful to test scheduling policies with strict time requirements. It made real experiment possible to evaluate individual theory-optimal scheduling policy due to its realistic packet arrival characteristics.

1.1.2 Wireless Network Virtualization

With fast development in different field of technology, vast application scenarios are demanding various new requirements to wireless network. For example, autopilot requires high security and low latency, where as IoT sensors only requires connectivity and fairness. All of these applications may use the same network infrastructure. This heterogeneous situation arises a challenge when we need to provide high quality service to each application, because the best scheduling policy for one application doesn't necessary serve well for other applications. Thus while PULS [4] is a powerful experimental testbed for many scheduling policies, but it can only implement one optimal scheduling policy under one certain scenario. In real world applications, there is a demand that we need to provide best scheduling policies for various services simultaneously.

In order to meet rapid growth of traffic demand over wireless network, wireless network virtualization (WNV) has been introduced and widely studied by researchers both in 5G and wireless ad hoc networks. In general, WNV refers to an abstraction layer concept created on top of the original physical network, upon which we can manage and program wireless network infrastructure for different functionality. Through multiple different virtual networks on same physical infrastructure, WNV became a useful tool for network resources sharing, so as to increase network utilization within limited capital and network resources.

Furthermore, in order to better serve different application with different network connection requirements, Network slicing is introduced to collect and organize different services to multiple network virtualization groups. The idea of network slicing is to abstract an logical network plane on top of physical hardware to serve different applications and services. In the wireless ad hoc networks, typically we define a set of flows to be one slice. [5] A flow is a specific packets streaming entity within wireless network infrastructure. A flow will have specific end user and

QoS requirement. Flows with same QoS requirement will be assigned to one slice. A slice can support multiple flows, meanwhile, a flow can join in different slices.

Slicing becomes an efficient way to deal with these various service requirements. Slicing is to cut network into slices with different resources and offer heterogeneous services to different client requirements. By assigning services with same requirements to one slice, we can design and allocated each slice according to its individual networking demand. Simply speaking, we use slicing to break network demands into several heterogeneous sub-demands, and provide tailored resources and mechanism on each slice, so as to achieve overall better satisfaction for everyone.

There are numerous work applying network slicing in different aspects of 5G network [6]. Some focus on physical network infrastructure [7] [8], some focus on network function layer [9] [10]. There are few consider slicing with scheduling. [11] theoretically proposed a scheme to achieve the optimal throughput according to various slicing constraints. However, there has been no implementation for slicing scheduling with multiple service constraints.

1.1.3 Network Management

The application and network requirements can change overtime. The slicing distribution between different services should also be able to configure for network administrators. Software Defined Network (SDN) is a powerful tool for network management. Generally speaking, SDN decouples the data and control planes of network, moving the control plane to a centralized location with global information. Thus the forwarding devices run and follow the forwarding rules programmed from the controller.

The testbed will even extend further if we can monitor and allocate system resources dynamically in a centralized manner through someone with global information such as a SDN controller. Furthermore, expanding and creating the slice towards an arbitrary novel application or service becomes possible, and thus brings flexibility and programmability towards wireless networking utilization.

1.2 Goal

For my thesis, I will concentrate on the last hop of wireless ad hoc network, where an access point (AP) forwards multiple flows to multiple client users. I will design and implement an experimental platform which supports slot-by-slot scheduling policy slicing and dynamic management over remote SDN controller. I performed experiments on the testbed and showed that slicing scheduling has great improvement over non-slicing scheduling. I further analyzed the resource allocation problem, proposed and implemented an algorithm on SDN controller towards automated optimal admission control. The rest of papers is organized as follows: Section 2 discusses some existing works and explain why they are differ from mine. Section 3 introduces the testbed in details. Section 4 discusses how to mathematically model and optimize the resource allocation problem, and the corresponding admission algorithm. Section 5 will show the experiment results and evaluation of proposed algorithm. And finally section 6 will conclude the paper.

2. RELATED WORK

Network slicing is a hot topic over network research community especially with development of 5G. However, many works are aiming to propose a theoretical model for network slicing towards applications or resource allocation. With few work has actual system implementation, here are some papers that are most similar to my work.

In [12] they implemented slicing on scheduling policy at ad hoc network last hop Access Point. However, within every slice, they schedule the flows in a same round-robin manner, which doesn't consider individual Quality of Service (QoS) requirements for each slice. Neither did they consider resource allocation optimization when assigning airtime to slices.

In [13] they performs network slicing over bandwidth and prioritization to meet differentiated services. They also use SDN to realize flexible slices allocation and configuration. However, they didn't consider scheduling within the network base station, and their work slices bandwidth on wired local wide area network (WAN) network, while our work focus on slicing scheduling resources on AP.

In [14] they design a scheduler within 5G network in the Radio Access Network (RAN). They achieved dynamically prioritize slices with higher weight while maintain coexistence of non-critical slices. However, similar to work [12] they only use weighted Round Robin (RR) scheduling policy, and merely give more weight to the critical slices. Without optimal policy for individual slices, their scheduler can not meet the requirements of heterogeneous services.

In [11] they proposed a slice-aware scheduling algorithm model that can handle aggregate rate targets and resource guarantees for possibly overlapping slices. Their algorithm can also make packet per packet resource allocation decisions for individual flows. However they only consider flows with no latency requirements, and doesn't have a system implementation to test their algorithm.

So far, to best of my knowledge, there is no previous work can include latency-sensitive traffic slicing with packet per packet scheduling within MAC Layer.

3. SYSTEM DESIGN

In this section we will provide an overview of the testbed, and explain the purpose of each major component.

Consider a scenario with 12 clients, and in which a group is formed by 4 consequent clients. Group 1 is handling the network load for VR (Virtual Reality) games, which routinely require network packets to be delivered with a very short deadline. Group 2 deals with videos, which have rather longer deadline requirements but do require continuous transmission over time. Group 3 is performing a general downloading application. In this scenario, if the interests of group 1 are considered exclusively, the Access Point (AP) should employ a time sensitive scheduling policy such as Deficit based Queuing (DBQ) [1]; DBQ enhances network performance for VR workloads. However, group 2 and 3 may experience a networking jam due to their low priority for deadlines. Despite this, if we use Largest Queue First (LQF) across the whole AP, group 1 will suffer due to the reason that the amount of transmission data is far less than video streaming and downloading. Our goal is to provide proper scheduling policy for each group simultaneously by slot-by-slot network slicing scheduling.

Fig. 3.1 is a blueprint of the system structure. The red workstation is an Access Point (AP) performing as a network base station. The AP provides downlink services to multiple clients shown as blue machines. While the AP generates groups of heterogeneous flows and performs slicing-scheduling, it also contains an SDN switch inside which communicates with an SDN controller located at another machine. This SDN controller will serve as a management tool with an accessible user interface.

3.1 Design Principle

The primary goal of our testbed is to provide slicing-scheduling policies simultaneously between heterogeneous services within the MAC layer. First, we need to provide flexible heterogeneous incoming traffic and support different policies to schedule this incoming traffic. Inherited

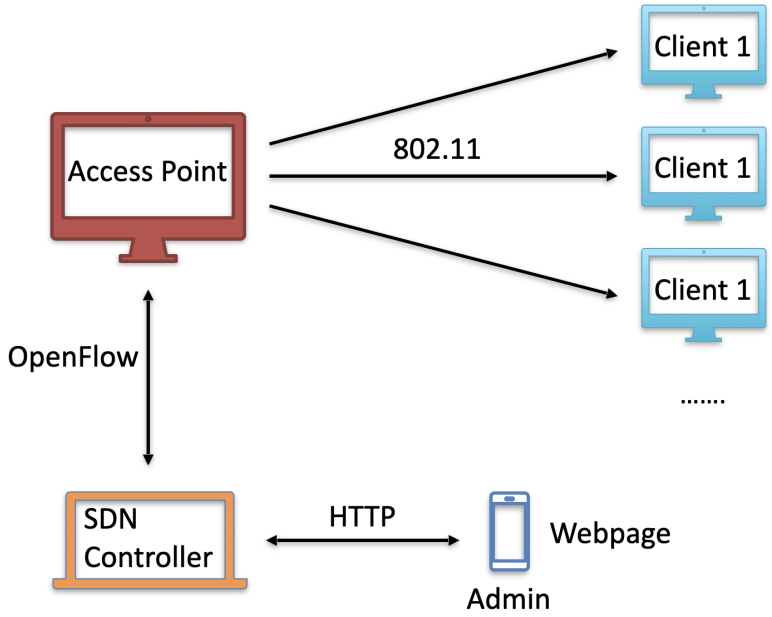


Figure 3.1: System Design

from PULS [4], we can generate both time sensitive and non-sensitive flows, and we are able to perform real-time scheduling on a host machine with feedback information from the physical layer in a packet-per-packet manner. We need to further enhance the incoming traffic characteristics and include slicing functionality; we address this with a custom scheduling algorithm that considers manageable weights to each policy without damaging the accuracy of the scheduling policy. Second, with the AP ready for slicing-scheduling, we implemented an SDN into our system for management purposes. With our SDN, we are able to not only implement admission control algorithms in the SDN controller but also retrieve and set flexible parameters in the MAC Layer based on contributing factors such as throughput and MCS.

One inherent and unique advantage of our testbed is flexibility. Incoming traffic is supported with real-time, non-real time elastic traffic, and non-real time inelastic traffic. Scheduling policy support includes DBQ (Deficit Based Queuing) [1], LQF (Largest Queue First), PF (Proportional Fair), RR (Round Robin), and with the potential to quickly and easily implement more policies. Our admission control algorithm is implemented within the SDN controller with a pragmatically designed two-way communication between the testbed and the SDN; this algorithm is also easy

to improve on and replace for administrator-specific situations. Furthermore, all three components listed above are independent of each other, bringing flexibility and ease of expansion for future research on network slicing-scheduling.

3.2 Packet Generation and Slices

Corresponding to the scenario proposed previously, there exists 12 flows independently generating packets. There are 4 significant parameters up for configuration: interval time, number of packets generated, probability of generation, packet deadline. Packets are generated within the interval time up to 1ms in resolution, and through 2 methods: deterministic generation, or uniformly distributed generation. In regards to deterministic generation, a certain number of packets are constructed according to a probability within the given time interval. For uniformly distributed generation, packets are uniformly constructed between 0 and the number of packets generated that is assigned to it within the given time interval. Each packet will prepend a timestamp calculated from the system time count at packet generation, plus a configurable deadline number. To support real-time flows, we set deadline numbers to be a few milliseconds while non-real time flows have larger deadlines, up to a few minutes. Each flow has an individual probability of generation, number of packets generated, and packet deadlines, thus realizing heterogeneous incoming traffic for their different client networking needs.

After the generation process, the packets will be queued and await scheduling. There are 12 queues representing 12 clients, and we assign 4 flows to every 1 slice. In the specified scenario, slice 1 contains the first 4 flows with real-time traffic, slice 2 contains flows 5-8 with non-real time inelastic traffic, and slice 3 contains the last 4 flows with non-real time elastic traffic.

The number of flows and slices are by no means exhaustive, but are sufficient to reveal the advancement in network performance due to slicing-scheduling. Packet generation and slice assignment can be easily adjusted according to the needs of future experimentation.

3.3 Slicing Scheduling

Once packets are generated and wait in their respective queues, scheduling is performed and repeated on a per-packet basis. Fig. 3.2 shows a flow diagram depicting the slicing-scheduling procedure.

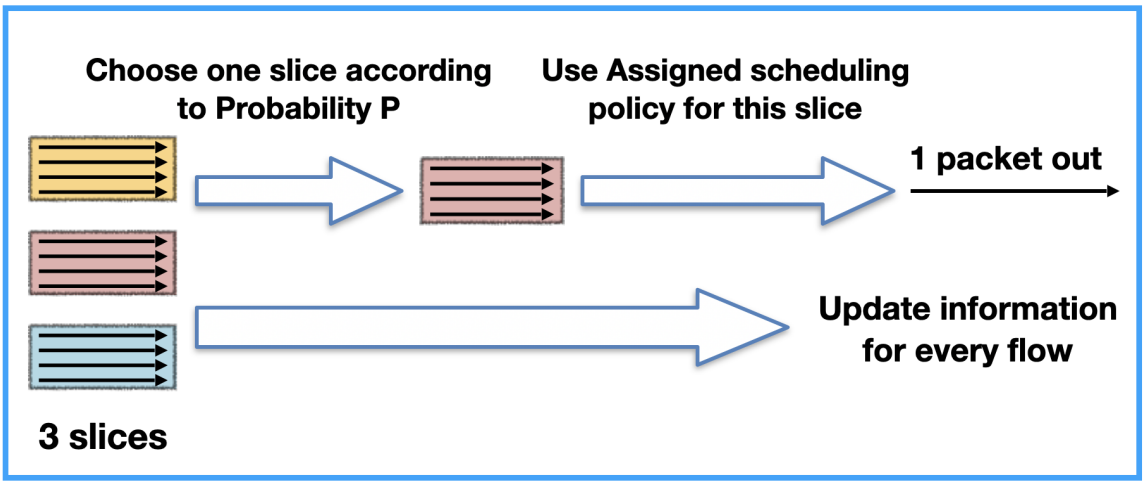


Figure 3.2: System procedure for scheduling 1 packet

There are 2 configuration parameters for the slicing-scheduling process: the scheduling policy assigned to each slice, and the probability of scheduling the slice. Different scheduling algorithms are embedded within the host machine and are readily available to be assigned to individual slices. The slicing mechanism will then be realized in the time division multiple access (TDMA) style. At each packet delivery time slot, the system chooses 1 out of 3 slices according to the probability set to this slice and uses the assigned scheduling policy for this slice to schedule 1 packet out of 4 flows. While scheduling packet 1 out, the system will also update the technical data for all the 16 flows. This includes examining the deadlines of the first elements in each queue and dropping the packet if the deadline has already expired. The system will also increment the flow index counter for the slices which are assigned the Round Robin policy. The probability for choosing a specific slice is determined through resource allocation necessities among slices, and in this case,

the probabilities for 3 slices should sum up to 1.

Channel reliability is set as the quotient of delivered packets over sent packets for an individual client over the last 1 second time interval. This gives 12 channel reliability indicators, each ranging from 0 to 1. Additionally, channel reliability can also be manually manipulated by intentionally dropping ACKs (acknowledgements) with a probability for a client. Based on the definition of deficit [1], Deficit Based Queuing policy schedules queues with the largest deficits. In our case, the system will schedule the maximum product of channel reliability and deficit between 4 flows. For largest queue first, the system will schedule the maximum product of channel reliability and queue length. With proportional fair, the system will schedule the maximum quotient of channel reliability over the throughput of each client. The throughput employed here is calculated over the previous 5 second time interval, adequately indicating the resources the client has gained over a relatively longer time period. Greater in-depth explanation on transmission procedure details and mechanism-policy separation can be found in PULS [4] paper.

3.4 SDN and Admission Control

Figure 3.3: Web page as user interface

We also added an SDN component into the testbed as a management platform to retrieve information from the MAC layer in the AP and send commands directed to the testbed. The testbed implements OpenFlow 1.3 Software Switch [15] and Ryu as supporting software for the controller. Ryu provides software components with well-defined API's that make creating new network management and control applications more manageable. The switch is located within the same machine as the AP, and communicates with the scheduling platform via localized UDP packets. The SDN controller is located on another laptop bundled with a web interface for administrator ease of use for information display and a configuration panel. This feature brings mobility and flexibility for the quick and remote control of the testbed.

Two-way communication exists between the SDN controller, switch and testbed in the same format of OpenFlow protocol, which aims to align with current communication standards and potential large-scale deployments. The MAC layer will automatically transmit flow statistic information at regular 5 second intervals to the SDN, and the SDN will interpret the message and refresh the statistic table displayed on the user interface. From the user interface, one can manually send administrating commands for management, such as changing MCS values, scheduling policies for each slice, and controlling the overall resource distribution to every slice. Later in this paper, we will introduce a algorithm embedded in the SDN controller with the purpose of calculating the optimal resource distributions for each distinct slice. An example of the user interface is shown in Fig. 3.3.

4. RESOURCE ALLOCATION

In addition to the previously described system implementations, this paper also further explores and addresses the notable resource allocation issue and proposes an algorithm stemming from the logic of gradient descent in order to determine optimal resource distribution between slices.

4.1 Evaluation Criteria

In order to find the best distribution, first we need to define what should be considered good. Let's continue with the example introduced in section 3. There are 12 clients consist of 3 slices with different service requirement. First slice is real-time inelastic flows with very short deadline and will be using DBQ. Second slice will have inelastic traffic and will be using LQF. Third slice will have elastic traffic, therefore this group have incoming traffic that system can never finish scheduling. We will use Proportional Fair (PF) for the third slice. It's obvious that a client will be satisfied when the number of delivered packets are equal or larger than the number of packets requested.

For real-time traffic, we define Y_n (Mbps) as target number of bits requested by client n within one second, which is the product between the arrival rate and delivery ratio, X_n (Mbps) as delivered timely throughput for client n, U_n as the utility of flow n, and W_1 as coefficient of real-time flows to keep consistent with the utility range of other type of traffic. We have:

$$U_n = W_1 * (\exp(Y_n) - \exp(Y_n - X_n)) \tag{4.1}$$

For a given target rate Y_n, U_n is a monotonically increasing function to the timely throughput X_n. U_n is zero when X_n is equal to zero.

Similar to real-time traffic, for non-real-time inelastic traffic, we define A_n (Mbps) as number of bits generated for the client n within one second, B_n (Mbps) as delivered throughput for client n, and W_2 as the coefficient of non-real-time inelastic flows. The utility of a flow can be written as:

$$U_n = W_2 * (\exp(A_n) - \exp(A_n - B_n)) \tag{4.2}$$

For non-real-time elastic traffic in group three, because the number of delivered packets can never reach the number of generated packets, and the applications seek to have the highest possible throughput, we use log throughput as the utility of the flow. we define C_n (Mbps) as the client throughput and W_3 as the coefficient:

$$U_n = W_3 * \ln(C_n + 1) \tag{4.3}$$

The utility of a slice i is defined as V_i to be the sum of utilities of all flows in that slice.

4.2 Gradient Ascent Algorithm

It's straightforward that the score of a slice is independent of the other slices and is only determined by the resource allocated to that slice. The more resources that are allocated to 1 slice, the greater the score of this slice will become. Another constraint also exists in that the slice allocation for all three slices should sum up to exactly 1; additionally, if the total score of the system performance is considered as defined by the sum of the three slice scores, a function between slice allocation and system performance can be determined. The domain of this function is a triangular intersection plane within a three-dimensional coordinate system, where the 3 axes represent the distribution of the three slices. Each point within the triangular plane represents a specific slicing allocation value for the three slices and corresponds to a total system score. Through the premise that there is only one maximum value in this function, we constructed an algorithm employing the concepts of gradient ascent in order to approach this optimal slice distribution question.

Our algorithm takes the three scores from each of the three slices as input, and outputs the slice distribution in each iteration. The algorithm will mathematically explore the triangular plane and calculate the gradient of each score at new positions, then use the three gradients as weight for three vectors pointing from the current position to the three vertices of the triangular plane. The next position is always the direction of a sum of three weighted vectors and a specified step size

related to size of current gradient. In the following algorithm, **U** represents the vector of current position, $\mathbf{U}_d$ represents the position very close to **U** in order to collect gradients of three scores. $\mathbf{U}_1$, $\mathbf{U}_2$, and $\mathbf{U}_3$ represent three vertices of triangular.

Algorithm 1 Gradient Descent Algorithm

Input: Score1, Score2, Score3

Output: **U**[0],**U**[1],**U**[2]

Initialisation :

Place **U** at center of triangular

Max steps = 100

LOOP Process

for $i < Max\ steps$ **do**

  Gradient = (**Score** $\mathbf{U}_d$-**Score U**)/($\mathbf{U}_d$-**U**)

  Step size = Max($|Gradient[0]|$,$|Gradient[1]|$,$|Gradient[2]|$) * Learning rate

  if ($Step\ size < 0.75$) **then**

    break

  end if

  New direction = ($\mathbf{U}_1$-**U**)*Gradient[0]+($\mathbf{U}_2$-**U**)

  Gradient[1]+($\mathbf{U}_3$-**U**)*Gradient[2]

  U = New direction * Step size + **U**

end for

return **U**

The algorithm is embedded in the SDN controller, and there will be five seconds delay between each allocation set command and utilities retrieving command in order to give testbed enough time for effective feedback of new bandwidth allocation.

5. EXPERIMENT RESULT

The objective of our system design and implementation is to drastically improve system performance when heterogeneous services requirements exist across a network. Traditionally, servers only execute one scheduling policy despite the numerous and diverse incoming traffic. These conventional systems perform well when incoming traffic only requires similar networking flow demands. However, in reality, when the difference in networking requirements between incoming traffic is appreciable, such as both real-time flows and non-real time flows flooding a network simultaneously, the efficiency of these traditional systems is seriously impaired.

We evaluated our system through two factors; first, we compare system performance between single scheduling policy and slicing-scheduling with multiple policies under the same heterogeneous incoming traffic pressure to exhibit the clear improvement of slicing-scheduling. Second, we also show how the embedded algorithm further increases the system performance under given incoming traffic, revealing the true flexibility and immense potential for even greater gains through algorithmic implementations in networking systems alongside slicing-scheduling.

5.1 Detailed Settings

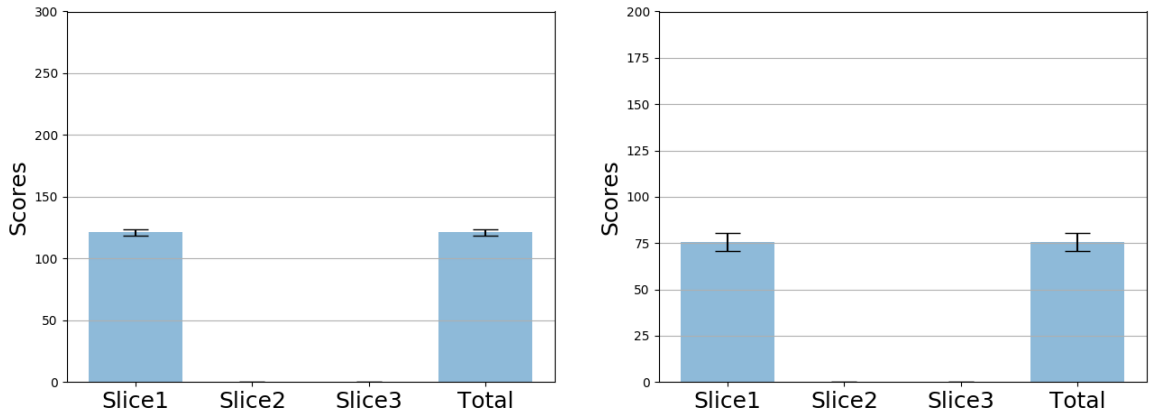
Despite the computational burden brought by the slicing mechanism, our system can still reach 32 Mbps under IEEE 802.11 ac 40 MHz, MCS 7, and with packet size equal to 4061 Bytes. The system can reach 5.2 Mbps under the same protocol with packet size to be 500 Bytes. For the convenience of the real-time flows, we attempt to keep round trip time (RTT) under 1 millisecond; as a result, we set packet size to be 2000 Bytes with maximum throughput of 19.5 Mbps under IEEE 802.11 ac 40 MHz and MCS 7.

In order to emphasize the heterogeneousness of incoming traffic, we set two experiments with different number of flows in each slice. First experiment we have four flows in slice one, three flows in slice two and slice three. The arrival rate for three slices are 8 Mbps, 6 Mbps and 30 Mbps. The deadline for real-time flows are 2 milliseconds. Under above incoming traffic, we set

W_1, W_2, W_3 to be 6, 7, and 21. The utility range is between 0 and 126.

For second experiments we assign two flows in slice one, and four flows in slice two and slice three. The arrival rate for three slices are 2 Mbps, 12 Mbps, and 40 Mbps. The deadline for real-time flows are 20 milliseconds. Under above incoming traffic, we set W_1, W_2, W_3 to be 24, 1, and 12. The utility range is between 0 and 72. The delivery ratios for both experiments are 0.97.

5.2 Result



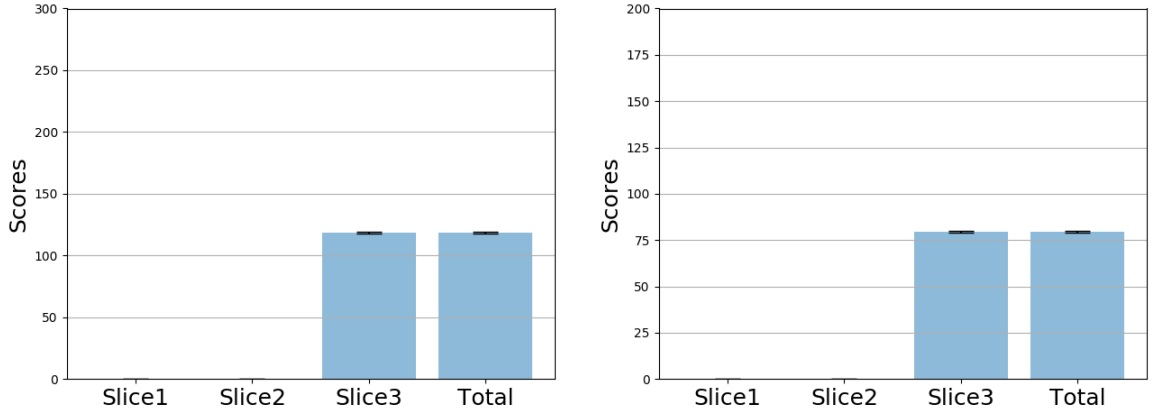
(a) First experiment (b) Second experiment

Figure 5.1: Deficit Based Queuing for All the Flows

When DBQ is set for all the flows (Fig. 5.1), real-time flows will perform well while non-real time flows will hardly get scheduled due to their longer deadlines.

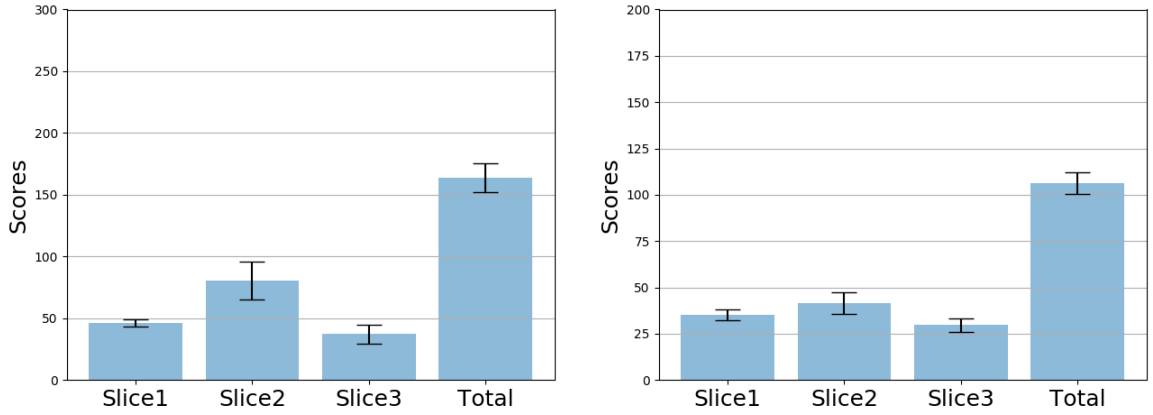
When LQF is set for all the flows (Fig. 5.2), only slice 3 will be scheduled because the over-loading incoming traffic ensures that its queue length will always be the longest.

When PF is set for all the flows (Fig. 5.3), every flow is able to share scheduling opportunities, and slice 1 and slice 2 will have greater priorities due to their lesser throughput values on average caused by smaller arrival rates. However, smaller arrival rates also result in a waste of scheduling time when there are no packets in the queue and yet a slice still continues to get scheduling opportunity due to its higher priority. PF is designed for elastic traffic and is not suitable for this



(a) First experiment (b) Second experiment

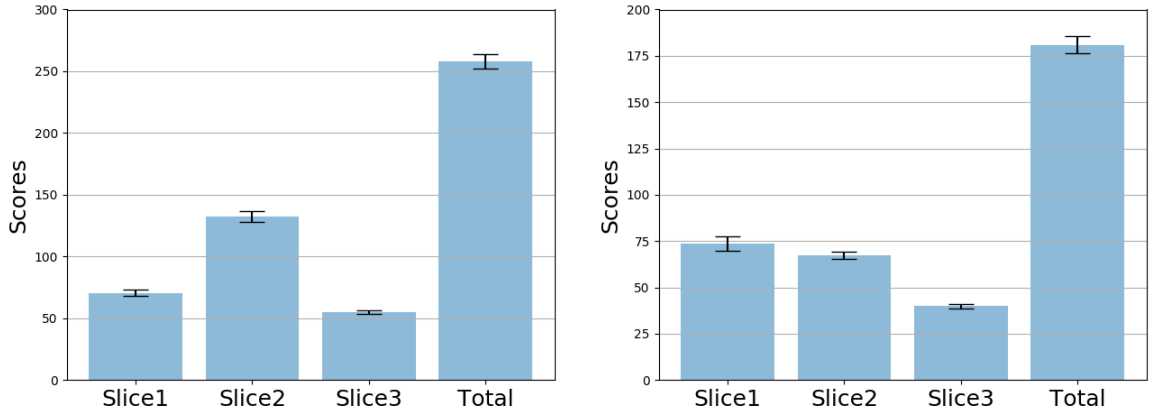
Figure 5.2: Largest Queue First for All the Flows



(a) First experiment (b) Second experiment

Figure 5.3: Proportional Fair for All the Flows

situation.



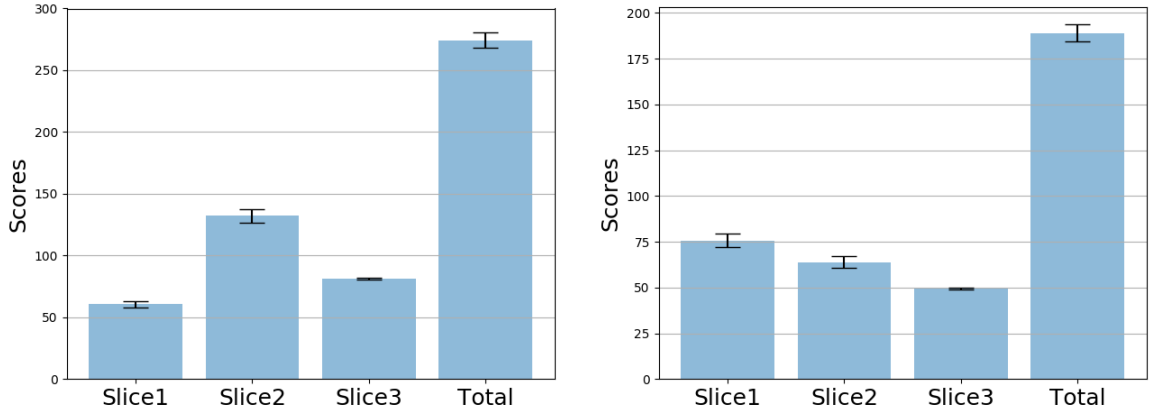
(a) First experiment (b) Second experiment

Figure 5.4: Round Robin for All the Flows

When Round Robin is set for all flows (Fig. 5.4), the best system performance by far can be observed with a total score of 258 for first experiment and 181 for second experiment. Indeed, RR shares scheduling resources equally to every flow and serves as a baseline for this heterogeneous incoming traffic.

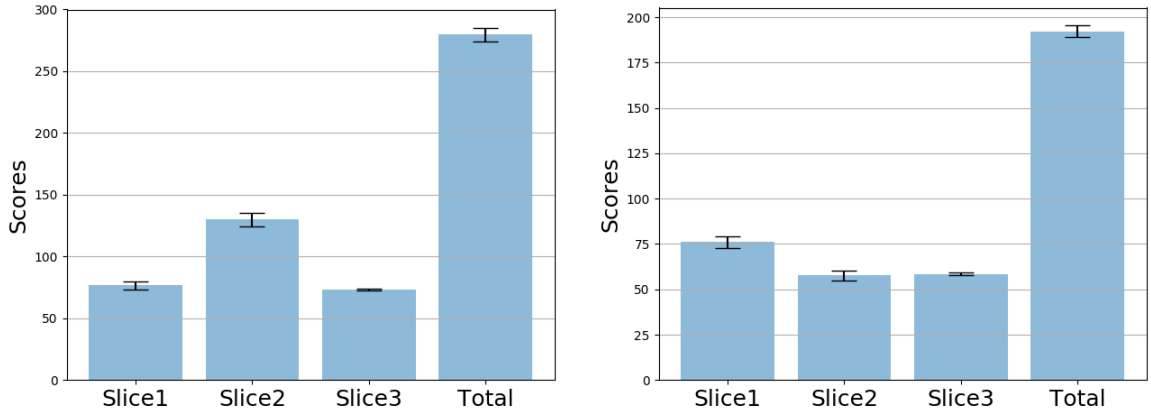
In Fig. 5.5 we use DBQ for slice one, LQF for slice two, PF for slice three and allocate bandwidth equally between three slices. The utility of slice three increases significantly in both experiments, while slice one in experiment one and slice two in experiment two slightly decrease. We suspect this is caused by the random process in the scheduler when choosing the slice. The randomness brings fluctuation to the allocated bandwidth and may impair the service to slice one and slice two. The total utility reaches 274 for the first experiment and 189 for the second experiment, indicating a better overall system performance.

Under the same networking traffic conditions, the iteration algorithm calculates the optimal resource allocation values to be 45%, 30%, 25% and 27%, 27%, 46% for two experiments and the results are shown in Fig. 5.6. For the first experiment, the total utility reaches 280, an improvement



(a) First experiment (b) Second experiment

Figure 5.5: Multiple Policies Scheduling for All the Flows



(a) First experiment (b) Second experiment

Figure 5.6: Improvement of Algorithm

of 8.5% compared with baseline Round Robin policy. The throughput of three slices increase by 0.3 Mbps, 0.2 Mbps, 0.7 Mbps respectively. For the second experiment, the total utility reaches 192, an improvement of 6% compared with baseline Round Robin policy. The throughput of slice one and slice three increase by 0.1 Mbps and 3.6 Mbps. The throughput of slice two decreases 2.8 Mbps due to the lower gradient during the iteration algorithm. The algorithm brings further improvement on overall system performance due to both the better scheduling protocol for each slice and the resource allocation optimization brought by the algorithm.

6. SUMMARY

6.1 Conclusion

With the rapid development in differing fields of technology requiring network plasticity, vast application scenarios demand unique requirements for wireless network performance. From this, an immense challenge arises for next-gen networks: providing custom-made and enhanced resource distribution for heterogeneous services. Although theoretical analysis and simulations for scheduling with the network slicing concept currently exist, there is no realized system of implementation, let alone one that combines both real-time and non-real time traffic slices. Our implementation effectively fills this gap between theoretical analysis and real-world experimentation, revealing great improvements as compared to non-slicing scheduling. The testbed is built with flexibility, versatility, and reprogrammability in mind; as a result, it can effectively serve as not only a realistic and practical networking system for next-gen networking augmentations, but also as a foundational framework for future research and implementation employing network slicing with sensible scheduling policies.

6.2 Future Work

This testbed can serve as a realistic experimental testbed on which network slicing and real-time scheduling policy research can occur. The testbed allows flexible configurations on the incoming traffic, scheduling algorithm embedding and resource allocation. Furthermore, with help of SDN platform, the testbed can serve as a real physical environment feedback to train an intelligent network slicing agent in the future.

REFERENCES

[1] I.-H. Hou, V. Borkar, and P. Kumar, *A theory of QoS for wireless*. IEEE, 2009.

[2] N. McKeown, A. Mekkittikul, V. Anantharam, and J. Walrand, "Achieving 100% throughput in an input-queued switch," *IEEE Transactions on Communications*, vol. 47, no. 8, pp. 1260–1267, 1999.

[3] H. J. Kushner and P. A. Whiting, "Convergence of proportional-fair sharing algorithms under general conditions," *IEEE transactions on wireless communications*, vol. 3, no. 4, pp. 1250–1259, 2004.

[4] S. Yau, P.-C. Hsieh, R. Bhattacharyya, K. K. Bhargav, S. Shakkottai, I.-H. Hou, and P. Kumar, "Puls: Processor-supported ultra-low latency scheduling," in *Proceedings of the Eighteenth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pp. 261–270, 2018.

[5] M. Richart, J. Baliosian, J. Serrat, and J.-L. Gorricho, "Resource slicing in virtual wireless networks: A survey," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 462–476, 2016.

[6] X. Foukas, G. Patounas, A. Elmokashfi, and M. K. Marina, "Network slicing in 5g: Survey and challenges," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 94–100, 2017.

[7] X. Zhou, R. Li, T. Chen, and H. Zhang, "Network slicing as a service: enabling enterprises' own software-defined cellular networks," *IEEE Communications Magazine*, vol. 54, no. 7, pp. 146–153, 2016.

[8] N. Nikaein, E. Schiller, R. Favraud, K. Katsalis, D. Stavropoulos, I. Alyafawi, Z. Zhao, T. Braun, and T. Korakis, "Network store: Exploring slicing in future 5g networks," in *Proceedings of the 10th International Workshop on Mobility in the Evolving Internet Architecture*, pp. 8–13, 2015.

[9] P. Rost, A. Banchs, I. Berberana, M. Breitbach, M. Doll, H. Droste, C. Mannweiler, M. A. Puente, K. Samdanis, and B. Sayadi, "Mobile network architecture evolution toward 5g," *IEEE Communications Magazine*, vol. 54, no. 5, pp. 84–91, 2016.

[10] M. R. Sama, X. An, Q. Wei, and S. Beker, "Reshaping the mobile core network via function decomposition and network slicing for the 5g era," in *2016 IEEE Wireless Communications and Networking Conference*, pp. 1–7, IEEE, 2016.

[11] S. Mandelli, M. Andrews, S. Borst, and S. Klein, "Satisfying network slicing constraints via 5g mac scheduling," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pp. 2332–2340, IEEE, 2019.

[12] M. Richart, J. Baliosian, J. Serrati, J.-L. Gorricho, R. Agüero, and N. Agoulmine, "Resource allocation for network slicing in wifi access points," in *2017 13th International conference on network and service management (CNSM)*, pp. 1–4, IEEE, 2017.

[13] J.-J. Chen, M.-H. Tsai, L. Zhao, W.-C. Chang, Y.-H. Lin, Q. Zhou, Y.-Z. Lu, J.-L. Tsai, and Y.-Z. Cai, "Realizing dynamic network slice resource management based on sdn networks," in *2019 International Conference on Intelligent Computing and its Emerging Applications (ICEA)*, pp. 120–125, IEEE, 2019.

[14] C. Bektas, S. Bocker, F. Kurtz, and C. Wietfeld, "Reliable software-defined ran network slicing for mission-critical 5g communication networks," in *2019 IEEE Globecom Workshops (GC Wkshps)*, pp. 1–6, IEEE, 2019.

[15] E. L. Fernandes, E. Rojas, J. Alvarez-Horcajo, Z. L. Kis, D. Sanvito, N. Bonelli, C. Cascone, and C. E. Rothenberg, "The road to bofuss: The basic openflow userspace software switch," *Journal of Network and Computer Applications*, p. 102685, 2020.