Managing Multi-Class Traffic in Cellular Networks

BY

ANUSHA CHANDRASHEKAR KUMBLE
B.E., Visvesvaraya Technological University (VTU), 2016

THESIS

Submitted as partial fulfillment of the requirements
for the degree of Master of Science in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Chicago, 2021

Chicago, Illinois

Defense Committee:

Hulya Seferoglu, Chair and Advisor
Besma Smida
Rashid Ansari

To my beloved parents, Gayathri and Chandrashekar.

ACKNOWLEDGMENTS

I would like to express my sincere thanks to my advisor Professor Dr. Hulya Seferoglu for the immeasurable moral and financial support I have received from her throughout this journey and for imparting her knowledge and expertise in this study. It has been a great learning experience for me to produce this piece of work under her guidance. I would like to extend my gratitude to Dr. Rashid Ansari and Dr. Besma Smida for serving on my thesis committee and providing their valuable feedback on my research.

I would also like to thank Mojtaba from BITS Lab, Department of CSE, UIC for his support and kindness.

I also acknowledge with a deep sense of reverence, my gratitude towards my parents and members of my family, who have always been my strength through their endless love, support, and sacrifice.

AK

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

dBm	decibel-milliwatts
Gbps	Gigabits per second
KB	Kilobyte
LEDBAT	Low Extra Delay Background Transport
LTE	Long-Term Evolution
MAC	Media access control
MB	Megabyte
Mbps	Megabits per second
MHz	Megahertz
MIMO	Multiple-Input and Multiple-Output
ms	millisecond
NS3	Network Simulator-3
PDCP	Packet Data Convergence Protocol
PFS	Proportional Fair Scheduler
QCI	QoS Class Identifier
RED	Random Early Detection
RLC	Radio Link Control

LIST OF ABBREVIATIONS (Continued)

RR	Round Robin
TCP-LP	Transmission Control Protocol – Low Priority

SUMMARY

In this thesis, we investigate cellular networks, and how resources are allocated to competing time-sensitive and insensitive TCP traffic. We divide traffic into multiple classes depending on their urgency. For example, video conferencing traffic is considered as time-sensitive, while software updates are not time-sensitive, and web-browsing is mid-level time-sensitive. Our goal is to design a mechanism to give priority to more time-sensitive traffic while allocating resources.

We first focus on the scenario that there are two traffic classes in the system; foreground (time-sensitive) and background (time-insensitive). In this setup, we build on an existing work named "Sneaker", which develops an active queue management mechanism. Sneaker drops packets from background traffic at the base station so that foreground traffic gets priority while background traffic still survives. Sneaker is designed for Proportional Fair Scheduler (PFS). In this thesis, we designed the same idea behind Sneaker for Round-Robin (RR) scheduler. We implemented our algorithm in ns3, and evaluated in a large-scale setup.

Next, we focus on a scenario that there are more than two traffic classes. We designed a packet dropping mechanism similar to Sneaker, but for multiple class scenario. We implemented our mechanism in ns3 by employing Round Robin(RR) as a cellular network scheduler. We verify that our packet dropping mechanism achieves the desired traffic packet prioritization.

CHAPTER 1

INTRODUCTION

1.1 Motivation

Today's cellular network consists of a large variety of traffic that compete for scarce cellular bandwidth. These large volumes of traffic are a combination of time-sensitive traffic such as mobile-to-mobile calls, video conferencing, web-browsing, video streaming, etc., and time-insensitive traffic such as software updates, cloud sync, etc., Although present networks allow to prioritize the flows based on QCI, they offer limited traffic classes that are specific to the services provided by the operator. Also, existing low-priority, end-to-end protocols do not perform well for the cellular networks due to per-device queues. Thus, it is important to design an approach to facilitate classification of flows that is application independent and at the same time can perform well with existing transport protocols while not requiring changes to existing schedulers as well as TCP. First, we implement and evaluate our design for time-sensitive, foreground traffic and time-insensitive, background traffic. We then implement and evaluate for multiple-class traffic having varying time-sensitivities.

1.2 Thesis Organization

This thesis is organized into seven chapters. Descriptions of each chapter of the thesis are as follows:

Chapter 1: This chapter provides the motivation for our work and explains our approach to address some of the research challenges related to our work.

Chapter 2: In this chapter, we discuss background work related to this thesis.

Chapter 3: In this chapter of the thesis, we discuss the implementation of Sneaker using round-robin scheduler to handle foreground and background flows.

Chapter 4: This chapter evaluates our implementation presented in Chapter 3.

Chapter 5: This chapter explains the implementation of Sneaker for flows of multiple classes of traffic with different time sensitivities.

Chapter 6: This chapter provides the evaluation of our design implemented in Chapter 5.

Chapter 7: We present a concluding summary of the thesis in this chapter.

CHAPTER 2

BACKGROUND AND NOTATION

2.1 Theory of Dropping Rate Probability

This work is an inspiration from and extension of design of Sneaker (1), which addresses the issue of congestion control in cellular networks. Sneaker develops an active queue management mechanism by classifying the traffic into foreground and background flows. This design of Sneaker is implemented on Proportional Fair Scheduler(PFS). Figure 1 represents the sample cellular system setup using Sneaker as discussed in (1). The incoming TCP traffic from the remote hosts is a set of N flows denoted by S_1,...,S_N, is a combination of foreground and background traffic. As this traffic enters the base station, they get queued by the queuing mechanism supported by Sneaker. These per flow queues range from Q_1,...,Q_N. These queues of traffic at the base station are received by K end users represented by R_1,...,R_K, based on the scheduling algorithm of the traffic scheduler implemented by the cellular network. This approach helps to randomly drop packets from the background flows that arrive at the base station when the network is congested. In order to drop the packets from background flows and achieve per-flow prioritization, practical packet dropping probability is formulated using NUM framework.

After analyzing the interaction between TCP and common base station schedulers, optimal and practical dropping rates are derived. Optimal dropping rate is derived for dropping the packets from background flows during periods of congestion in the network and allowing foreground flows. It makes sure that the spare capacity is recaptured when the load of the network subsides. Practical dropping rate

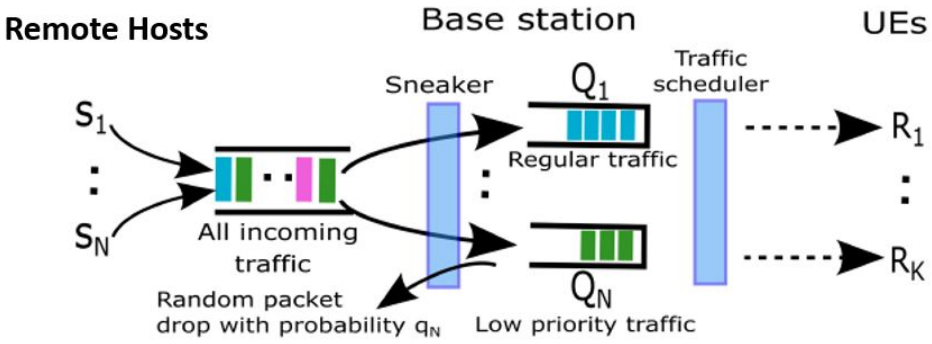


Figure 1: Sample cellular system setup using Sneaker.

This figure is from(1)

helps to achieve the prioritization and fairness among foreground and background flows. The equation for the practical dropping rate thus derived is given as

(1)
$$q_l^{PR} = 1 - \frac{\gamma\,(\,c_l\,T_l \max\{\,1-\,\tau_F\,,\quad \varepsilon\,\})^2}{\rho_l\left(\,1+\,\gamma\,\,(\,c_l\,T_l \max\{\,1-\,\tau_F\,,\quad \varepsilon\,\})^2\right)}\,,\ \forall l\ \in\ \mathscr{L} \tag{2.1}$$

where $\gamma = \frac{\beta}{L^2 B^2}$ (1) and approximate time share of foreground traffic is represented as $\tau_F \approx \sum_{m \in \mathscr{M}} \frac{Q_m(t)}{R_m(t)}$ (1). The packet dropping probability is calculated when the scheduler measures packet scheduling probability based on scheduling algorithm and passes it onto Sneaker. In the implementation, classification of flows based on their priority into foreground(high priority) and background(low priority) traffic is handled by using a tag. This tag in TCP header which is added by the sender is of one bit. TCP header is extracted at the base station by Sneaker where it learns if the flow is background(low priority) or not.

Proportional Fair Scheduler(PFS) which is the default scheduler used by Sneaker, is an algorithm that aims to maximize the total throughput of the network along with allowing minimal service to each user. Each data flow is assigned a data rate or scheduling priority that is inversely proportional to its anticipated resource consumption (2) (3). Channels are scheduled for queues with maximum priority function that is denoted by using the equation, $P = T^{\alpha}/R^{\beta}$, where T represents the data rate achievable for a queue at a current time slot, R represents past average data rate at the queue. α and β which are approximately equal to 1 (4), helps to tune the fairness of the scheduler. Round Robin(RR) scheduler, which is used in our current design implementation of Sneaker, is an algorithm in which every active data flow of a queue that has packets is allowed to take turns to transfer the packets on a shared channel in a periodically repeated order. If no packets exists in a particular flow, next data flow with packets will take its place. Thus, the scheduler helps to prevent link resources from being unused.

2.2 Notation

In this section we list the variables that we will keep referring to in the following chapters:

q_l^{PR}, q_{l_i}	practical drop rate
L	number of background/low priority flows
M	number of foreground/high priority flows
B	typical TCP packet size
c_l	channel capacity of background flows
c_{l_i}	channel capacity of low priority traffic

T_l, T_{l_i}	Round trip time
τ_F	time share of foreground traffic
τ_{F_j}	time share of high priority traffic
ε	minimum rate
ρ_l, ρ_{l_i}	packet scheduling probability at the base station
l, l_i	background/low priority user
m, l_j	foreground/high priority user
t	transmission time interval
$Q_m(t)$	queue size of foreground user m
$Q_{l_j}(t)$	queue size of high priority user
$R_m(t)$	maximum packets transmitted by a foreground user m
$R_{l_j}(t)$	maximum number of packets that can be transmitted from high priority user
$\mathscr{L}$	sets of background flows
$\mathscr{M}$	sets of foreground flows
Γ	total number of classes
i	defines all low-priority flows
j	defines all high-priority flows

CHAPTER 3

SNEAKER IMPLEMENTATION ON ROUND-ROBIN SCHEDULER

Sneaker(1) is implemented on top of the PDCP layer of the existing protocol stack in LTE eNodeB. Packets get buffered through the PDCP and RLC layers and are finally read by the MAC layer as per the algorithm of the round-robin scheduler. Sneaker obtains local signaling information from the MAC and RLC layers in order to calculate the packet dropping probability provided by Eq.2.1 in Section 2.1 of Chapter 2. Thus, Sneaker drops packets according to the packet dropping probability before they arrive at the PDCP layer. Figure 2 below shows how Sneaker is built on top of the existing LTE protocol stack.

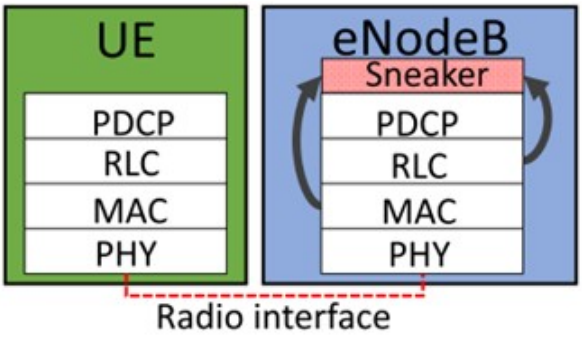


Figure 2: Sneaker on LTE protocol stack.

This figure is from(1)

Also, in order for the packets to carry end-to-end information, no modifications are made to TCP.

CHAPTER 4

PERFORMANCE EVALUATION OF SNEAKER IMPLEMENTED ON ROUND ROBIN SCHEDULER

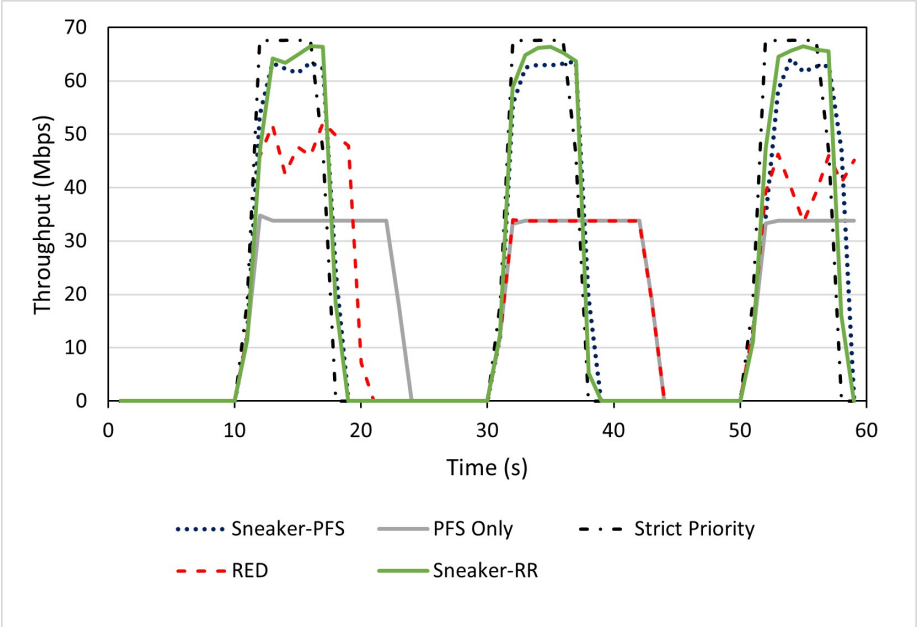
Our design of Sneaker on round robin scheduler is evaluated by creating the network topology having the following specifications which is developed using NS3 simulator (5). It consists of a downlink band of 751 MHz and bandwidth of 10 MHz at the base station with 50 resource blocks. The base station supports MIMO transmission with 47.78 dBm transmission power and size of RLC buffer is 512 KB. The speed of the link connecting the base station to the packet gateway and service gateway is 300 Mbps. The packet gateway connects to the remote servers through 1Gbps multi-hop wired links with delay of 10 ms in the links. We make use of log-distance path loss model having loss exponential parameter of 3.52.

4.1 Evaluation of priority

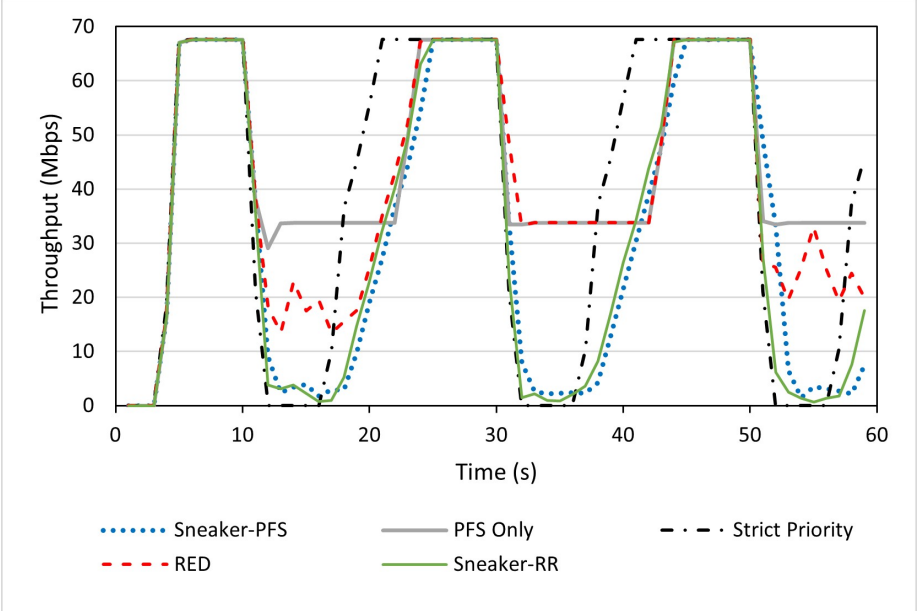
We evaluate Sneaker on round robin in this experiment based on its ability to differentiate between foreground and background traffic. We send traffic across two remote senders and two end users. Traffic is sent in such a way that, one of the senders sends continuous background traffic while the other sends foreground traffic at every 10 seconds interval in an on-off pattern. A 70 Mbps cellular link is shared among all the senders which use TCP Reno. Our design of Sneaker on round robin (RR) scheduler is compared with the following baselines: Sneaker on PFS, PFS Only, Strict Priority, RED. Proportional

fair scheduler utilizes network resources efficiently while maintaining a balance with the fairness among the users. The priority of each user is calculated at each resource block and then the user with maximum priority gets assigned with the resource block. Round robin scheduler employs equal time-sharing for allocating resources to each user. Every active data flow that has data packets in queue take turns in transferring packets on a shared channel in a periodically repeated order. RED uses predictive models to pre-emptively drop packets from buffer before the buffer becomes full. Strict Priority serves high priority traffic queue, until it is empty and then moves to lower priority queues.

Packets are dropped according to the practical dropping rate as in Eq.2.1.The throughput achieved by both foreground and background flows are as shown in Fig.3.We notice that PFS Only does not actively drop packets, every packet is dropped from the background flow if there exists a foreground flow in case of Strict Priority and packets are dropped according to the policy in (6) by RED. We observe that Sneaker on RR, Sneaker on PFS and Strict Priority quickly yield to foreground traffic when they exist and serve background traffic at remaining times. This ability is not supported by PFS Only and RED. Strict Priority is not ideal as it blocks background traffic completely until foreground traffic finishes which causes timeouts and disconnections. Further, we evaluate the impact of increasing number of background flows on foreground traffic rate by sending foreground flow to one user. We send background flows in various numbers to another user connecting the same cellular base station. It is observed that Sneaker on round-robin is able to isolate the foreground flows better than Sneaker on PFS and PFS Only as the degradation is significantly smaller with our design. Fig.4 shows the average throughput achieved by the foreground flows using PFS Only, Sneaker on PFS and Sneaker on RR. It can be observed that there is degradation in the throughput of foreground flows when the number of



(a) Foreground flows



(b) Background flows

Figure 3: Priority using different approaches

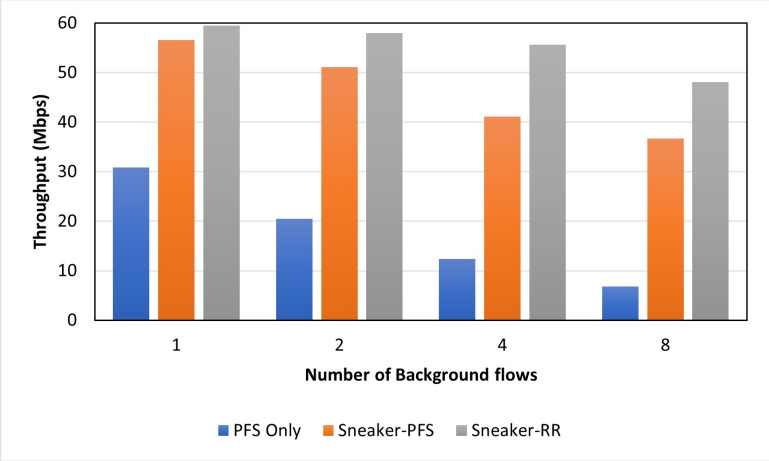


Figure 4: Benefit of foreground flows using Sneaker-RR

background flows increases, which is expected. A foreground flow achieves 1.9 times higher throughput for a background flow, with Sneaker on RR than with PFS Only and an increase in throughput by seven times when the number of background flows increases to 8.

4.2 Evaluation of fair sharing of bandwidth

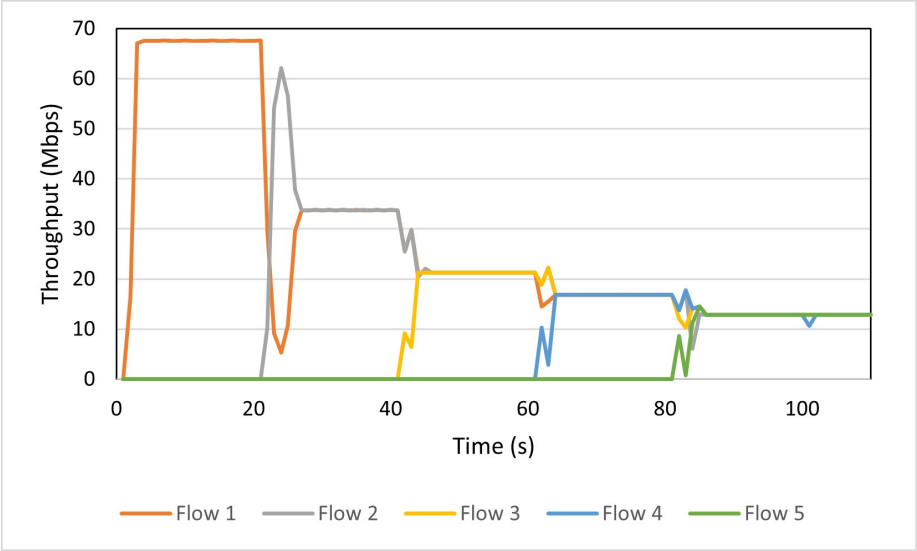
Through this experiment, we evaluate the fair sharing of available bandwidth among foreground flows and fair sharing of spare bandwidth among background flows. We verify this by considering the following two scenarios:

Scenario 1 : There are five background flows with no foreground flows. New flows join the system after every 20 seconds. We can observe that, using Sneaker on RR, the background flows converge quickly to their fair-share throughput when new flows get added from Fig.5a.

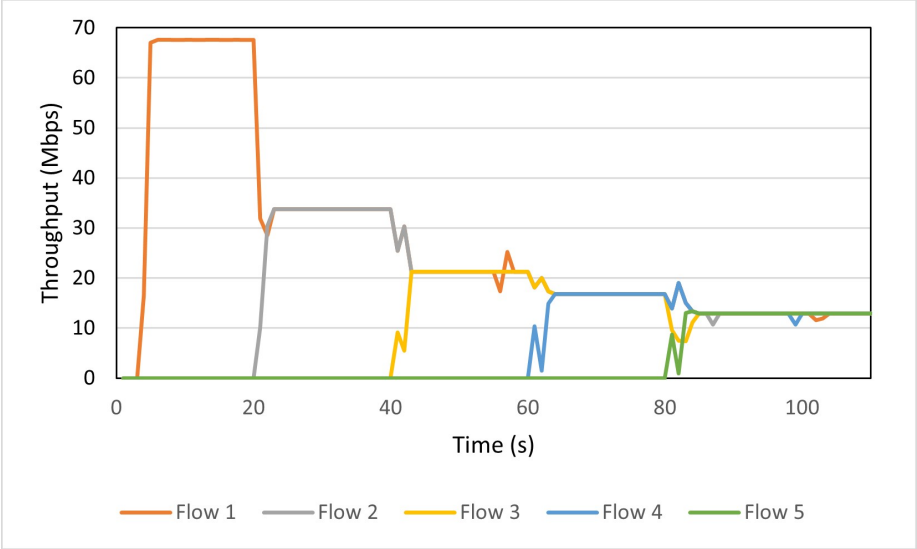
Scenario 2 : There are five foreground flows with one background flow. New flows join the system after every 20 seconds. We observe that the throughput is fairly shared by foreground flows when using Sneaker on RR and this is shown in Fig.5b.

4.3 Performance evaluation for moderate and low loads through large-scale simulations

In order to evaluate for realistic large workloads, we create a large topology consisting of 100 users connecting to a same base station. We generate mixed traffic that consists of a combination of short, medium, and long flows. Short flows which are of size 64 KB contributes to 10% of overall foreground traffic. Medium flows which are of size 1 MB contributes to 40% of the load. The remaining 50% of the load is contributed by long flows of size 32 MB and this constitutes for the background flows that are sent continuously. We compare Sneaker implemented on round-robin scheduler using TCP-Reno, alongside of well-known end-to-end congestion control mechanisms like TCP-LP, LEDBAT and Sneaker on PFS scheduler, that are designed to yield to foreground traffic. We compare the throughput achieved by both foreground and background flows for all the schemes by generating traffic for 120 seconds. We evaluate the performance of the four schemes by setting the overall network load due to foreground flows to 50%. We observe that the foreground traffic has a bursty nature under all the schemes as shown in Fig. 6a. TCP-LP achieves better throughput when compared to LEDBAT. Sneaker on PFS and RR schedulers achieves the best throughput of all the schemes. The throughput achieved by background flows vary drastically among all the schemes as shown in Fig. 6b. We observe that TCP-LP and LEDBAT does not yield to foreground flows, whereas Sneaker on PFS and RR schedulers modulates its throughput to allow foreground traffic to use most of the capacity that is available. This



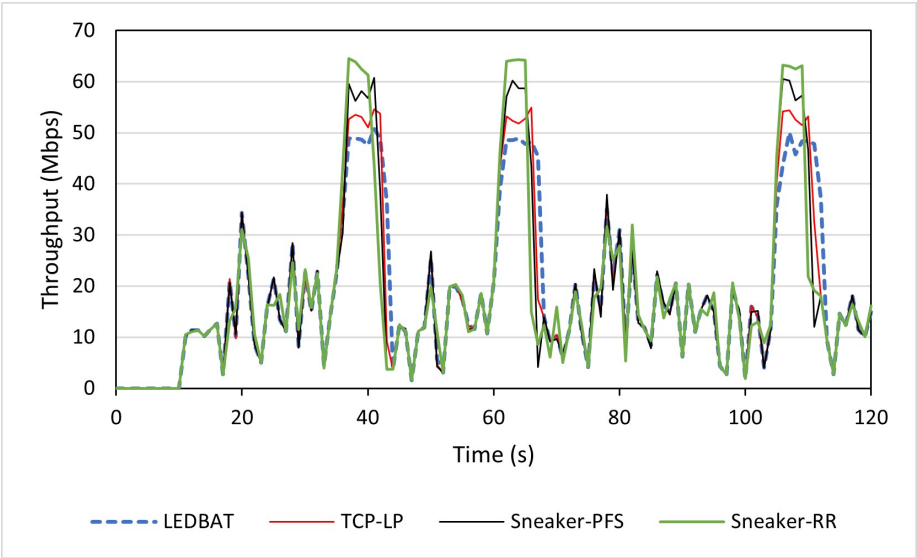
(a) Fair sharing among background flows



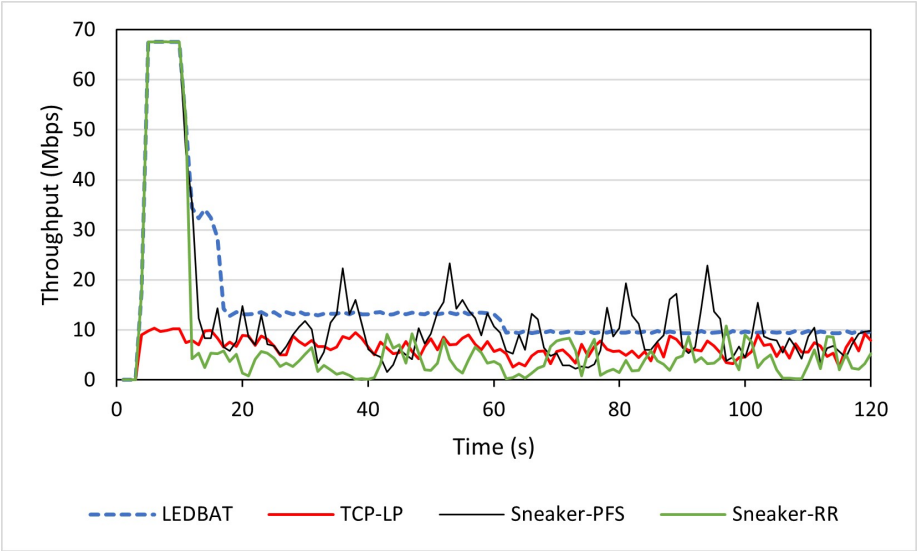
(b) Fair sharing among foreground flows

Figure 5: Fair sharing of bandwidth

helps to utilize the spare capacity effectively. We evaluate the performance for low loads by considering the same experiment with 20% load. Fig. 7a shows the throughput achieved by foreground flows under all schemes which are similar. The throughput achieved by background flows with all the schemes is shown in Fig. 7b. We observe that TCP-LP and LEDBAT achieve lower throughput and are unable to utilize the spare capacity completely. Sneaker on RR scheduler utilizes the spare capacity efficiently by achieving higher throughput similar to Sneaker on PFS scheduler.

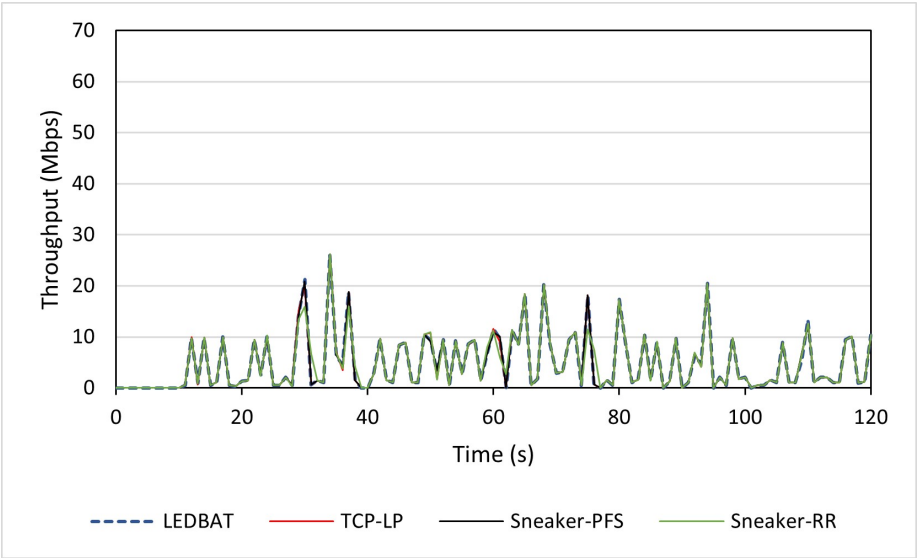


(a) Foreground flows

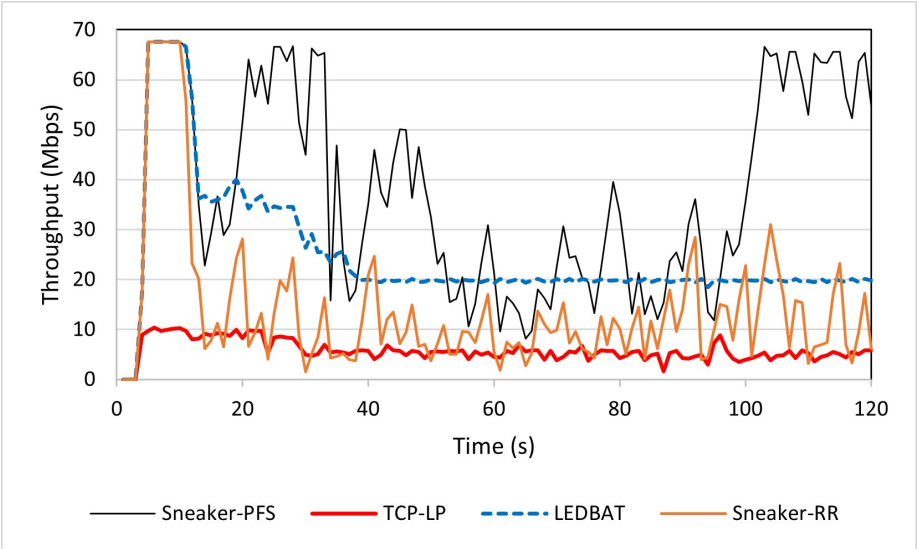


(b) Background flows

Figure 6: Performance evaluation at moderate load (50%) using large-scale simulations



(a) Foreground flows



(b) Background flows

Figure 7: Performance evaluation at low load (20%) using large-scale simulations

CHAPTER 5

IMPLEMENTATION OF SNEAKER FOR MULTI-CLASS FLOWS

The implementation of Sneaker for multi-class flows focuses on managing more than two classes of traffic. We designed a packet dropping mechanism for multiple classes which is similar to that of Sneaker, but can handle a total of Γ traffic classes. We mainly consider three classes of traffic namely high priority time-sensitive traffic, mid-level time-sensitive traffic and low priority time-insensitive traffic for this thesis. This design drops packets at the base station from low priority and mid-level priority traffic when all three traffic classes exists. This helps to sustain both mid-level and low priority traffic while allowing high priority traffic. In a similar way, during the occurrence of mid-level and low priority traffic, packets from low-priority traffic are dropped, so that mid-level traffic gets priority while low priority traffic still survives. We implement Sneaker for multi-class as a slim layer on top of PDCP layer at the eNodeB, similar to Sneaker implementation for handling foreground and background flows as discussed in Chapter 3. Sneaker extracts the end-to-end information of all the incoming packets. Packet dropping probability derived to handle multi-class traffic flows is as shown in Eq. 5.1. The priorities of the flows for all the Γ classes are represented as $l_1 > l_2 > \ldots > l_\Gamma$.

$$q_{l_i} = 1 - \frac{\gamma_i (c_{l_i}\ T_{l_i}\ \max\{\ 1 - \tau_{F_j}\ ,\ \varepsilon\ \})^2}{\rho_{l_i}(1\ +\ \gamma_i (c_{l_i}\ T_{l_i}\ \max\{\ 1 - \tau_{F_j}\ ,\ \varepsilon\ \})^2}\ ,\ \forall l_i \in \mathscr{L}_i, i \in \{2, \ldots,\ \Gamma\} \tag{5.1}$$

where, $\gamma_i = \frac{\beta}{L_i^2 B^2}$ and approximate time share of high priority traffic is represented as $\tau_{F_j} \approx \sum_{j \in \{1,\ldots,\Gamma-1\}}$

$\sum_{\forall l_j} \frac{Q_{l_j}(t)}{R_{l_j}(t)}$. Packets are dropped according to the value of packet dropping probability calculated before they arrive at the PDCP layer. We implement our Sneaker for multi-class traffic design on NS3 simulator(5) by making use of round robin (RR) scheduler.

CHAPTER 6

PERFORMANCE EVALUATION OF SNEAKER FOR MULTI-CLASS FLOWS

We evaluate the performance of our Sneaker design implemented for multi-class flows using the same network topology used for the evaluation of Sneaker implemented on round-robin scheduler in NS3(5). Remote servers connect to the Packet Gateway through multi-hop wired links of 1Gbps speed and 10ms delay. The packet and service gateways connect to the base station with links having speed of 300 Mbps. The base station supports MIMO transmission with transmission power of 47.78 dBm, and RLC buffer size of 512 KB. Base station has downlink band of 751 MHz with bandwidth of 10 MHz and 50 resource blocks. The log-distance propagation model having loss exponential of 3.52 is used as path loss model.

We evaluate the priority achieved for flows of multiple classes in this experiment by using Sneaker for multi-class. We use three remote senders to send traffic to three different end users. The first sender sends least priority i.e. time-insensitive traffic with Flow Priority 3 continuously. The second and third senders send traffic in on-off pattern at every 40 seconds and 30 seconds respectively. The time-sensitivity of flows for second sender with Flow Priority 2 is higher than that of first sender i.e. Flow Priority 3. The third sender sends traffic of Flow Priority 1, which are highly time-sensitive(i.e. priority more than that of senders one and two). Thus we have, Sender 3's Flow Priority 1 $>$ Sender 2's Flow Priority 2 $>$ Sender 1's Flow Priority 3. Cellular link of 70 Mbps is shared among all senders with TCP-Reno. Packets are dropped by Sneaker implemented for multi-class according to the dropping probability determined by Eq. 5.1. The throughput achieved by all the three traffic classes is as shown in

Fig. 8. Sneaker for multi-class, drops packets from senders one and two when all the three traffic classes exist, there by allowing high priority traffic from sender three to flow, while flows from other two traffic classes which are less time-sensitive still survive. In the absence of traffic from sender three, packets from sender one is dropped, allowing the flow of traffic from sender two. We verify that, using Sneaker for multi-class, sender one with traffic of least priority yields to higher priority traffic from senders two and three when the network is congested and quickly recaptures the spare capacity when the network becomes lightly loaded.

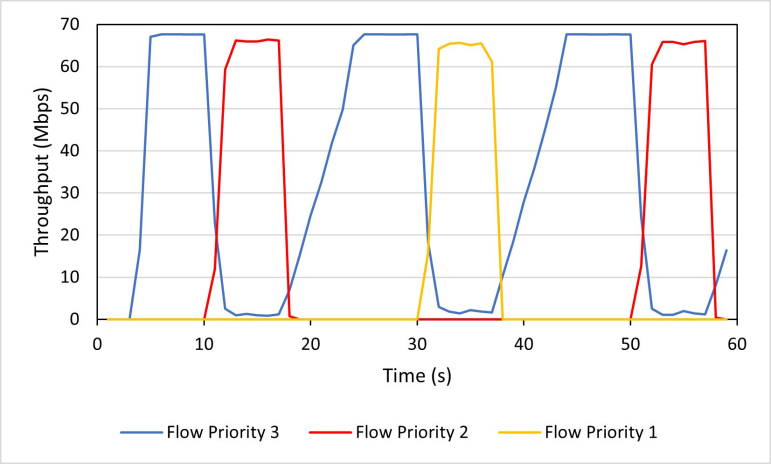


Figure 8: Priorities of multi-class flows where Flow Priority 1 > Flow Priority 2 > Flow Priority 3

CHAPTER 7

CONCLUSION

Through this thesis, we provide an approach to traffic management that offers per-flow prioritization and also works well over the scheduled links in cellular networks. It enables differentiation between flows of different classes and priorities. This approach helps to maintain fairness within each class of traffic while providing high performance in cellular networks. We have implemented our design for Sneaker using round-robin scheduler and also to handle multi-class traffic by considering packet dropping probability. Further, we evaluated our implementations for both small-scale and realistic large-scale traffic through extensive simulations and verified that our design of Sneaker achieves the desired optimality.

CITED LITERATURE

1. S. Zhou, M. U. Chaudhry, V. Gopalakrishnan, E. Halepovic, B. Vamanan, and H. Seferoglu, "Managing background traffic in cellular networks," in *2019 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*. IEEE, 2019, pp. 1–6.

2. 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.

3. G. Miao, J. Zander, K. W. Sung, and S. B. Slimane, *Fundamentals of mobile data networks*. Cambridge University Press, 2016.

4. J. Yang, Z. Yifan, W. Ying, and Z. Ping, "Average rate updating mechanism in proportional fair scheduler for hdr," in *IEEE Global Telecommunications Conference, 2004. GLOBECOM'04.*, vol. 6. IEEE, 2004, pp. 3464–3466.

5. "Ns-3 network simulator." [Online]. Available: http://www.nsnam.org/

6. P. Kuusela, P. Lassila, J. Virtamo, and P. Key, "Modeling red with idealized tcp sources," *Proceedings of IFIP ATM & IP*, pp. 155–166, 2001.

7. S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on networking*, vol. 1, no. 4, pp. 397–413, 1993.

8. K. Winstein, A. Sivaraman, and H. Balakrishnan, "Stochastic forecasts achieve high throughput and low delay over cellular networks," in *10th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 13)*, 2013, pp. 459–471.

9. A. Kuzmanovic and E. W. Knightly, "Tcp-lp: low-priority service via end-point congestion control," *IEEE/ACM Transactions on Networking*, vol. 14, no. 4, pp. 739–752, 2006.

10. F. Capozzi, G. Piro, L. A. Grieco, G. Boggia, and P. Camarda, "Downlink packet scheduling in lte cellular networks: Key design issues and a survey," *IEEE communications surveys & tutorials*, vol. 15, no. 2, pp. 678–700, 2012.

11. J. Padhye, V. Firoiu, D. F. Towsley, and J. F. Kurose, "Modeling tcp reno performance: a simple model and its empirical validation," *IEEE/ACM transactions on Networking*, vol. 8, no. 2, pp. 133–145, 2000.

12. A. Gurtov and R. Ludwig, "Responding to spurious timeouts in tcp," in *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No. 03CH37428)*, vol. 3. IEEE, 2003, pp. 2312–2322.

13. T. E. Klein, K. Leung, and H. Zheng, "Enhanced scheduling algorithms for improved tcp performance in wireless ip networks," in *IEEE Globecom*, vol. 4. Citeseer, 2004, pp. 2744–2759.

14. D. Rossi, C. Testa, S. Valenti, and L. Muscariello, "Ledbat: the new bittorrent congestion control protocol," in *2010 Proceedings of 19th International Conference on Computer Communications and Networks*. IEEE, 2010, pp. 1–6.

15. R. Pan, B. Prabhakar, and K. Psounis, "Choke-a stateless active queue management scheme for approximating fair bandwidth allocation," in *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No. 00CH37064)*, vol. 2. IEEE, 2000, pp. 942–951.

16. R. Ludwig and R. H. Katz, "The eifel algorithm: making tcp robust against spurious retransmissions," *ACM SIGCOMM Computer Communication Review*, vol. 30, no. 1, pp. 30–36, 2000.

17. F. Lu, H. Du, A. Jain, G. M. Voelker, A. C. Snoeren, and A. Terzis, "Cqic: Revisiting cross-layer congestion control for cellular networks," in *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications*, 2015, pp. 45–50.

18. W.-c. Feng, K. G. Shin, D. D. Kandlur, and D. Saha, "The blue active queue management algorithms," *IEEE/ACM transactions on networking*, vol. 10, no. 4, pp. 513–528, 2002.

19. S. H. Low, "A duality model of tcp and queue management algorithms," *IEEE/ACM Transactions On Networking*, vol. 11, no. 4, pp. 525–536, 2003.

20. B. Holfeld, D. Wieruch, T. Wirth, L. Thiele, S. A. Ashraf, J. Huschke, I. Aktas, and J. Ansari, "Wireless communication for factory automation: An opportunity for lte and 5g systems," *IEEE Communications Magazine*, vol. 54, no. 6, pp. 36–43, 2016.

VITA

ANUSHA CHANDRASHEKAR KUMBLE

Education	M.S. Electrical and Computer Engineering University of Illinois at Chicago	2019 – 2021
	B.S. Electrical and Electronics Engineering Visvesvaraya Technological University	2012 – 2016
Experience	Software Engineer at Tech Mahindra Associate Software Engineer at Tech Mahindra	