

**Evaluation of Background Transport Protocols in Production and  
Experimental LTE Networks.**

BY

SHIBIN MATHEW

B.Tech, Govt. Model Engineering College, Kochi, Kerala, India, 2014

THESIS

Submitted in partial fulfillment of the requirements  
for the degree of Master of Science in Computer Science  
in the Graduate College of the  
University of Illinois at Chicago, 2019

Chicago, Illinois

Defense Committee:

Balajee Vamanan, Chair and Advisor

Chris Kanich

Mark Grechanik

## ACKNOWLEDGMENT

I would like to thank my advisor, Balajee Vamanan, for introducing me to this project and guiding me throughout its lifetime. Balajee always inspired me to approach the challenges I faced, through the eyes of an avid researcher and pointed me in the right direction whenever it was needed. I am extremely grateful to him for the constant motivation and emotional support he has given me throughout my time here at UIC.

I am thankful to my other research collaborators, Emir Halepovic, Hulya Seferoglu, Shanyu Zhou, Muhammad Usama Chaudhry, and Vijay Gopalakrishnan. The diligence and hard-work put into this project by the team was always inspiring. It motivated in giving my best during the course of research. In their absence this project would not have been successful. I am grateful to have received an opportunity to work with this team.

I would also like to thank other members of my thesis committee, Chris Kanich and Mark Grechanik for their guidance and support.

SM

## CONTRIBUTIONS OF AUTHORS

This thesis involves proceedings to a conference submission that is a combined work of Shibin Mathew, Muhammad Usama Chaudhry, Shanyu Zhou, Vijay Gopalakrishnan, Emir Halepovic, Hulya Seferoglu, and Balajee Vamanan. The writing towards the conference submission is a combined effort of Balajee, Emir, Vijay and Hulya. Muhammad did the implementation of the work on Linux kernel, evaluation in ns3 and helped me with the evaluation in real network. Shanyu did the theoretical analysis of Proportional Fair (PF) scheduler. Designing and perfecting the algorithm was a combined effort of the entire team. I implemented PF algorithm in Open Air Interface (OAI), tested and verified the protocol's performance in the PhantomNet simulator, designed and developed an automated test-setup for real network testing which involves a traffic generator application for android phones and automation scripts for synchronization between servers and the phones. Various tests conducted involves, 1 on 1, mixed workload, mixed workload with randomized traffic pattern, mobility test, fairness study in real networks.

## TABLE OF CONTENTS

<u>CHAPTER</u>	<u>PAGE</u>
<b>1 INTRODUCTION . . . . .</b>	<b>1</b>
<b>2 BACKGROUND . . . . .</b>	<b>7</b>
2.1 Telecom Networks . . . . .	7
2.2 Popular transport protocol . . . . .	9
2.2.1 TCP Reno . . . . .	10
2.2.2 TCP CUBIC . . . . .	10
2.2.3 BBR . . . . .	11
2.3 Existing Background Flow Protocols . . . . .	11
2.3.1 TCP Low Priority (TCP-LP) . . . . .	11
2.3.2 LEDBAT . . . . .	12
2.3.3 TCP Vegas . . . . .	12
2.4 Why existing schemes do not work in cellular networks . . . .	13
<b>3 DESIGN AND IMPLEMENTATION . . . . .</b>	<b>15</b>
3.1 TCPLegilimens . . . . .	16
<b>4 TEST STRATEGIES AND EVALUATION . . . . .</b>	<b>19</b>
4.1 Emulated Test-bed . . . . .	20
4.1.1 One-on-one testing in emulator . . . . .	20
4.1.2 Mixed workload test in emulator . . . . .	22
4.1.2.1 Evaluation of emulated test results . . . . .	23
4.2 Real world tests . . . . .	26
4.2.1 Stationary one-on-one testing in commercial network . . . . .	27
4.2.2 Automated test framework for mixed workload test. . . . .	29
4.2.2.1 Evaluation of real world mixed workload experiment . . . . .	33
4.2.3 Mobile one-on-one testing in commercial network . . . . .	35
4.2.4 Mixed work-load experiment with randomized traffic pattern	37
4.2.5 Legilimens fairness study in real world network . . . . .	40
<b>5 PREVIOUS WORK . . . . .</b>	<b>42</b>
<b>6 CONCLUSION . . . . .</b>	<b>44</b>
6.1 Future Work . . . . .	45
<b>CITED LITERATURE . . . . .</b>	<b>46</b>

## TABLE OF CONTENTS (Continued)

<u>CHAPTER</u>	<u>PAGE</u>
APPENDICES . . . . .	50
VITA . . . . .	52

## LIST OF TABLES

<u>TABLE</u>		<u>PAGE</u>
I	Flow numbers for corresponding loads for measured capacity of 25 Mbps. . . . .	30
II	Data transmitted in randomized experiment Cubic vs Legilimens . .	39

## LIST OF FIGURES

<b><u>FIGURE</u></b>		<b><u>PAGE</u></b>
1	Existing protocols in wired vs wireless networks. . . . .	13
2	High-level overview . . . . .	16
3	Emulator test framework. . . . .	21
4	One on one background (Legilimens) vs foreground (CUBIC) . . . . .	22
5	Two on one with background (Legilimens) vs foreground (CUBIC) . . . . .	22
6	foreground short flow FCT in phantomnet . . . . .	24
7	foreground medium flow throughput in phantomnet . . . . .	24
8	foreground long flow throughput in phantomnet . . . . .	25
9	background flow throughput in phantomnet . . . . .	25
10	1-on-1 test between CUBIC and Legilimens in real world setup. . . . .	27
11	1-on1 test between CUBIC and BBR in real world setup. . . . .	28
12	Real world base station $U_{PRB}$ . . . . .	31
13	foreground short flow FCT. . . . .	31
14	foreground medium flow throughput. . . . .	31
15	foreground long flow throughput. . . . .	32
16	background flow throughput. . . . .	32
17	Mobile testing of one-one workload. . . . .	35
18	Randomized workload with background using CUBIC . . . . .	37
19	Randomized workload with background using Legilimens . . . . .	37

## LIST OF FIGURES (Continued)

<u>FIGURE</u>		<u>PAGE</u>
20	Fairness study of Legilimens in real world setup . . . . .	40

## LIST OF ABBREVIATIONS

IoT	Internet of Things
AIMD	Additive Increase Multiplicative Decrease
CSMA	Carrier Sense Multiple Access
FOTA	Firmware Over The Air
FIFO	First In First Out
OWD	One Way Delay
GAP	Gradually Aggressive Probing
FCT	Flow Completion Time
LEDBAT	Low Extra Delay Background Transport
LAN	Local Area Network
PF	Proportional Fair
PDCCP	Packet Data Convergence Protocol
RTT	Round-Trip Time
PRB	Physical Resource Block
SINR	Signal to Interference+Noise Ratio
TTI	Transmission Time Interval
OAI	Open Air Interface

## LIST OF ABBREVIATIONS (Continued)

RSRP	Reference Signal Received Power
RSRQ	Reference Signal Received Quality
BDP	Bandwidth Delay Product

## SUMMARY

As cellular networks are becoming faster, cheaper and wider in terms of reach, it is today the carrier of ever increasing bulk of network traffic. More and more foreground and background mobile applications are starting to depend on cellular networks than on wired or Wi-Fi networks for transmission. Based on time critical nature of traffic, it can be classified into high priority foreground and lower priority background traffic. Background traffic (software, firmware updates and cloud-sync), even though they can be considered as lower priority, is an important part of this mix. However they are time insensitive and it is highly undesirable if it competes with time sensitive foreground traffic like: live streaming, audio / video calls etc. Transport Congestion control schemes are responsible for preventing this contention.

Most congestion control techniques are end to end schemes with little or no intervention from the intermediate network elements. Therefore, it raises the need for efficient, end to end background congestion control schemes to deliver time insensitive background flows with little or no performance impact on foreground flows sharing the same bottleneck link. There are several protocols today like LEDBAT, VEGAS and TCP-LP that achieve this goal. However, these are designed for wired or Wi-Fi networks and do not perform well in cellular networks. This limitation can be attributed to the inability of cellular schedulers to differentiate between foreground and background flows. Wired and Wi-Fi networks use simple FIFO queues which does not have any scheduling policies other than tail dropping, whereas Cellular networks employ PF schedulers which provides weighted fairness among the competing flows in short

## SUMMARY (Continued)

time granularity. However, this scheduling policy of PF scheduler directly conflicts with the goals of end to end congestion control schemes which leads to undesirable results for protocols that depend on parameters like One Way Delay (OWD) or Round Trip Time (RTT) to detect contention.

In this thesis, we compare the performance of popular congestion control schemes like CUBIC and RENO with various state of the art background congestion schemes like LEDBAT, VEGAS, TCP-LP in cellular network and we perform comprehensive evaluation of our proposed scheme, Legilimens (1). Legilimens is specifically designed for LTE networks, it takes advantage of the cellular schedulers policies to accurately predict the presence of competing foreground flows in the network. Upon detecting contention, it pauses for random interval of time after which it uses a new probing technique (GAP) that incrementally probes the network for spare bandwidth. The probing technique provides an added advantage of being subtle on competing flows, compared to the traditional probing technique that relies on bulky probes which can choke competing foreground flows forcing them to back-off. Legilimens also provides added advantage of simplicity in deployment, as this does not need any explicit network feedback thereby obviating the need for infrastructural changes.

Our extensive evaluation involves one on one testing between competing protocols, mixed workload experiments, randomized mixed workload experiments, mobility tests and fairness tests for Legilimens. We perform the test in isolated and emulated test-bed, phantomnet. This makes sure that the test results are not tampered by any natural cross traffic that otherwise would have been present in commercial networks. We also compare the performance of Legili-

## SUMMARY (Continued)

mens with other state of the art background protocols in real world setups to provide proof of concept to its effectiveness. We evaluate Legilimens in terms of both foreground and background flow performance across different load and traffic settings and find that it performs well for foreground traffic at higher loads and for background traffic at lower loads. Legilimens accurately yields to foreground flows and is agile enough to capture spare bandwidth when available thereby maximizing resource utilization.

## CHAPTER 1

### INTRODUCTION

With more and more internet service providers (ISP) introducing unlimited data plans to customers at a cheaper rates, and due to its wider reach, mobile networks are being increasingly used to push software and firmware updates to billions of Internet of Things (IoT) and hand-held devices. One such major example is firmware updates for cars over cellular networks (2). With technologies like truly autonomous vehicles edging towards reality, the load on cellular networks due to time-sensitive and time-insensitive applications are bound to increase. According to Statistica, "by 2022, mobile data traffic worldwide is expected to reach 77.5 exabytes per month at a compound annual growth rate of 46 percent" (3). Technological advancements in health-care industry has enabled mobile diagnostics using a plethora of wearable devices which are connected via cellular networks, with the advent of 5G such real time applications will become more popular and widespread.

The above use cases suggests that, cellular networks are expected to cater to both low-latency and high throughput (video-on-demand, video chats) needs of modern day applications, requiring both time-sensitive (foreground) and time-insensitive (background) applications to share a common medium for transfer of vast amount of data with varying latency and throughput preferences. Large amount of data from background applications like cloud sync, firmware-over-the-air (FOTA) can result in significant congestion and performance degradation for users of typical real time applications such as web browsing, streaming, real time IoT data

and other interactive applications. This calls for techniques to prioritize time-sensitive foreground applications over the lower priority, time-insensitive background applications using the same shared medium. Rate-limiting specific flows is also possible. But naive rate-limiting can cause under-utilization at low loads (or allow too much data under high loads and overwhelm the network at high loads), while more sophisticated versions tend to be complex and expensive to realize. Wired and Wi-Fi networks depend on end-to-end congestion control mechanisms to cater to these network requirements.

Wired and Wi-Fi networks most commonly uses First-in-first-out(FIFO) queues with tail drop, simplest of all queuing algorithms. Basic FIFO queues cannot differentiate between the senders and therefore it pushes the responsibility of congestion detection and response to the edge devices, particularly the sender. Therefore prevalent congestion control algorithms are designed to perform independently without any aid from the network elements(4). Inability to differentiate between packets belonging to different flows is a major disadvantage of FIFO and Round Robin schedulers. It is also completely oblivious to how the edge to edge congestion control algorithms operates, this forms a gullible mechanism that cannot be policed. Its effectiveness depends on strict adherence of congestion control algorithms to principle of fairness and can be easily fooled by adversary protocols that can capture more than its share of bandwidth.

Cellular networks employ Proportional Fair (PF) scheduler, which combined with unique queues for each user in the base station, can achieve both fairness among users and high utilization of the radio spectrum(5). However, the cellular scheduler remains completely unaware of

the end to end objectives of the congestion control algorithm deployed at the server and therefore it's fairness objective can directly conflict with that of the congestion control algorithm.

There are existing TCP congestion control algorithms for delivery of bulk data (e.g., LEDBAT, TCP-LP) without affecting competing higher priority flows, these work well in wired and Wi-Fi networks. These algorithms introduce the concept of two-class service prioritization. The key idea is to have a "low-priority" mechanism for delivering large volume or time-insensitive data. This allows for typical user interactive applications (considered foreground flows) to have fast response times using a *fair-share* TCP variant like RENO or CUBIC, while simultaneously making progress on large volume transfers (considered background flows) using the lower priority TCP variant. However, results show that these protocols are not effective in cellular networks (in 2.4) due to variability in radio channel conditions and any attempts to correctly infer congestion from One Way Delay (OWD) is rendered futile because of the use of proportional fair (PF) schedulers in cellular base stations as stated above.

Drawing inspiration from TCP-LP protocols, we propose Legilimens (1), an agile Low Priority Transfer variant for cellular networks that delivers time-insensitive background traffic without adversely affecting time-sensitive foreground traffic. Legilimens has the main properties of other Low Priority Transfer protocols — to use all available bandwidth when no other traffic is present, and to yield quickly to standard TCP flows that share the same bottleneck link — but includes novel features that make it effective in cellular networks. Specifically, a well-designed Low Priority Transfer protocol for cellular networks must operate with minimal queuing, so that its packets do not compete with those of foreground flows at the scheduler.

When deployed Legilimens, considers all its generated flows as lower priority background flows and every other detected flow in the network as a higher-priority foreground flow. Therefore as its operation suggests it is highly suitable for time-insensitive background applications like FOTA, cloud backup and so on. Even though the approach can be helpful in both up-link and downlink traffic, we focus on Legilimens’s applications in downlink traffic(1). To that end, we design a novel algorithm that quickly estimates capacity and load based on packet inter-arrival times, not RTT or OWD. Our central *idea* lies in leveraging the downlink PF scheduler’s unique *strength* of providing fairness at short time scales to counteract its *weakness* of interfering with the operation of traditional Low Priority Transfer protocols(1). Based on the estimated capacity and load, Legilimens strives to deliver background traffic using *only* spare capacity. Legilimens operates in one of two modes: normal mode and probing mode. If the load is low and spare capacity is available, Legilimens operates in normal mode and quickly captures available bandwidth. If the load is substantial, Legilimens enters the probing mode, in which it yields all scheduling opportunities to other traffic most of the time while periodically sensing the network until spare capacity becomes available. In essence, we leverage the operational traits of the PF scheduler to drive a novel end-to-end congestion control algorithm, which otherwise does not work hand in hand with each other in cellular networks.

The following are the contributions of this thesis:

- Extensive testing and verification of Legilimens and competing background protocols in phantomnet emulator (6). We dive deeper into the working of PF scheduler and implement the same in open source software which is used in phantomnet.

- An automated test framework involving android applications that can request short, medium and large flows in fixed or random intervals based on configuration and server side code that tests various Low Priority Transfer congestion control schemes in an automated fashion.
- Extensive testing of Legilimens, (comparing it with various state of the art congestion control schemes), in real world setup using tests like 1-on-1 testing, mixed workload test, randomized mixed workload, mobility test and fairness study.
- We discuss the results obtained in both phantomnet and real world tests and show that Legilimens gives better foreground performance in the presence of competing background flows, at the same time captures spare bandwidth for background flows when available.

Deploying Legilimens in real network is a trivial task as it needs no interaction from the client nor the network. Legilimens functions solely using the meta-data available in ACKs therefore, it needs no modification at the client side or network components. A potential path for adoption could be deploying Legilimens on cellular proxy servers and TCP splitters that most cellular operators use today (7) or on CDN edge servers that specifically serve traffic to mobile devices over cellular links. Further, network and CDN operator would utilize existing traffic classification techniques to identify background traffic and direct its delivery via servers that run Legilimens.

The rest of the thesis is organized as follows. Chapter 2 delves deeper into the function of PF scheduler and the basics of cellular networks that is necessary for understanding Legilimens and also discusses the limitations of existing schemes. Chapter 3 gives an overview of the design

and operation of Legilimens. Chapter 4 discusses the test strategies used and we analyze the results obtained. Finally, Chapter 5 discusses related previous works, and Chapter 6 concludes the thesis.

## CHAPTER 2

### BACKGROUND

This chapter gives the necessary background information required to facilitate better understanding of the problem statement, motivation and design choices that drove the development of Legilimens. Section 2.1, gives an introduction to cellular networks and delves deeper into the working of PF scheduler, the operational characteristic of which forms the main design motivation for Legilimens. Section 2.2 introduces the various existing congestion control algorithms with similar design objectives and the design choices they made. Section 2.4 talks about why the existing schemes does not work in cellular networks and the challenges they face.

#### **2.1 Telecom Networks**

Long Term Evolution (LTE) forms the backbone of the telecommunication architecture and Legilimens was designed to operate primarily on top of it. The main requirements of LTE are "high spectral efficiency, high peak data rates, short round trip time as well as flexibility in frequency and bandwidth" (8). LTE access network can be defined as a network of base stations, evolved NodeB (eNB). LTE obviates the need for a central controller and distributes the intelligence among eNBs. It gives advantages like lower connection setup time and faster handover. The MAC protocol layer in LTE forms the brain of the base station as it houses the scheduler who handles resource allocation. Unlike in previous generation of cellular technologies, like in Universal Mobile Terrestrial System (UMTS), this obviates the dependence of scheduling

and MAC layer functionality on a central controller, leading to faster communication and decisions between the eNB and the UE(8).

MAC scheduler is the entity within the base station which allocates radio resources to User Equipment (UE) or phones. The smallest unit of radio resource that can be allocated to a user is called a Resource Block (RB) and it is 180 kHz wide in frequency. RBs can be allocated every epoch, which is 1 ms long in LTE and is referred to as Transmit Time Interval (TTI) (9). Based on the radio quality reported by the UE and its past state information stored in the eNB, the scheduler allocates sets of RBs called RB Groups to the most deserving UE on the same or different component carrier (CC). A CC is a part of the continuous frequency spectrum that is used to transmit or receive data between the eNB and UE and they can have a bandwidth of 1.4, 3, 5, 10, 15 or 20 MHz and each bands constitute of 6, 15, 25, 50, 75 and 100 RBs respectively every TTI.

One particular component of an eNB that is pivotal to its performance in terms of throughput and fairness is the PF scheduler and we dive deeper into its working. The objective of PF algorithm is to provide fairness among UEs at the same time, attain maximum utilization of the scarce radio resource. This is achieved by means of exploiting the multi-user diversity over temporally independent channel fluctuations. (10) In every TTI, the algorithm calculates the maximum achievable throughput or instantaneous rate of each connected UE for every sub-channel. This is influenced by the modulation scheme that can be used on the particular sub-channel. which is the direct consequence of the Signal to Noise Ratio(SINR) reported by the UE. The algorithm also keeps track of the average data rate of each connected user at time

t. The two data points mentioned above are used to generate a PF utility function which is the ratio of instantaneous throughput to average rate. In every scheduling instance, the UE with highest utility function is chosen and all available RBs are allocated to it. If the per-user queue of the UE does not have enough data to fill all RBs, the remaining RBs are allocated to UE with next larger utility value. Intuitively, the user with higher instantaneous rate has higher probability of getting picked over that with lower. This promotes better throughput and utilization of radio resources. Also those users with higher average rate are less probable to be chosen, which takes care of the fairness guarantee of PF scheduler PF.

## **2.2 Popular transport protocol**

Transmission Control Protocol (TCP) is the most widely adopted transport protocol, "designed to operate reliably over almost any transmission medium regardless of transmission rate, delay, corruption, duplication, or reordering of segments" (11). It provides services like congestion control, flow control, in-order delivery and reliable delivery in an end to end setup. TCP uses packet Acknowledgements (ACK) from receiver to attain necessary meta-data like ACK number for in order delivery, time-stamps that helps the sender calculate One Way Delay (OWD) and Round Trip Time (RTT) (12). Congestion control in TCP works by modulating congestion window depending on the presence or extend of congestion detected in the network. Initially the algorithm starts by setting congestion window to 10 Maximum Segment Size (MSS) (13) and then the window essentially doubles on being Acked, this phase is termed Slow-Start and once the algorithm detects congestion, it drops the window size then increases in a more linear fashion upon detecting more bandwidth. This is congestion avoidance. Almost all con-

gestion control algorithms out there essentially uses the same core idea. Let us look deeper into some of them.

### **2.2.1 TCP Reno**

TCP Reno (14) uses Additive Increase Multiplicative Decrease (AIMD) form of congestion window modulation. In the Slow-Start phase the algorithm increases the window size by 1 MSS for every received ACK, essentially doubling it. However, once it detects congestion through duplicate ACKs, it enters into Fast Re-transmit, drops the window size to half and continues with congestion avoidance. If packet loss is detected through a timeout, the window size is dropped to 1 MSS and Slow Start Threshold is set to half that of the window when it detected loss. Major disadvantage is that Reno depends on ACKs to increase the window size and therefore it is not suited for connections with high latency. It tends to under utilize the bandwidth. Regardless, it is one of the most popular congestion control scheme out there.

### **2.2.2 TCP CUBIC**

Unlike the usual TCP flavors, TCP CUBIC(15) does not depend on RTT for modulating the window size. As the name suggests, it uses a cubic function to increase the window size and therefore it quickly captures the available bandwidth. Once it encounters a congestion event, it sets the inflection point to the window size prior to the event, and drops the window size. The remaining operation is same as that of Reno. However, this is not suited for networks with low latency as the congestion window growth is no longer dependent on RTT. Cubic is the default scheme used in major Linux releases.

### 2.2.3 BBR

Bottleneck Bandwidth and Round-trip propagation time (BBR) (16) identifies loss based congestion control schemes as the major reason for buffer bloat which causes congestion. Instead of pushing the network into congestion and then detecting it, BBR proposes a clever solution to detect congestion even before it occurs. BBR keeps track of Bandwidth Delay Product (BDP) of each connection. The algorithm identifies two parameters : RTprop (round-trip propagation time) and BtlBw (bottleneck bandwidth), which controls the transport performance of a path (simplifying each path to a single link limited by the bottleneck bandwidth). BBR paces the packet to match the packet-arrival rate to the bottleneck links departure rate. The idea is to keep the packets in flight  $\approx$  BtlBw  $\times$  RTprop. However, BBR works poorly in tandem with loss based congestion control schemes. It aggressively captures the bandwidth and provides poor fairness when working in tandem with loss based protocols.

## 2.3 Existing Background Flow Protocols

Now we look at existing protocols that aim at the same end goal as Legilimens, back-off in the presence of high priority flows and make use of only the remaining bandwidth.

### 2.3.1 TCP Low Priority (TCP-LP)

TCP-LP (17), like the name suggests, TCP-LP is a scheme for delivering low priority traffic like FOTA or cloud updates. It uses One Way Delay (OWD) to infer congestion. It keeps track of the minimum and maximum OWD for the connection and also maintains a running average of the congestion window of the existing connection. The algorithm infers congestion if the running average is higher than a calculated threshold, which is influenced by the stored minimum and

maximum congestion window values and a threshold parameter that ranges between 0 and 1. Once it detects congestion, it drops the window to half and waits for a fixed amount of time. If it encounters congestion again, it drops the window to 1 MSS. This 2 stage congestion response is a key feature of TCP-LP.

### **2.3.2 LEDBAT**

LEDBAT(18), is a very commonly used scheme to deliver bulk of data across the internet without choking it. It does so by measuring the One Way Delay (OWD). For every connection it keeps track of the minimum OWD and assumes that to be one with minimum or no queuing. For all further ACKs, it calculates the queuing delay by comparing it with the stored minimum value. It comes up with a tolerable target delay and increases the congestion window if the difference is lesser than this value and decreases it if its higher.

### **2.3.3 TCP Vegas**

Vegas (19) is another flavor of protocols intended to serve background flows. The main aim of Vegas is to keep loss to the minimum and hence obtain higher good-put. Vegas calculates a target rate for each connection by making use of RTT. If the difference between target rate and current rate is lesser than a constant alpha then the congestion window is increased and if it is greater than a constant beta then the window is decreased. Congestion window remains unchanged if the value is between the two thresholds. The accuracy of this algorithm is highly dependent on the accurate measurement of RTT and the configured thresholds.

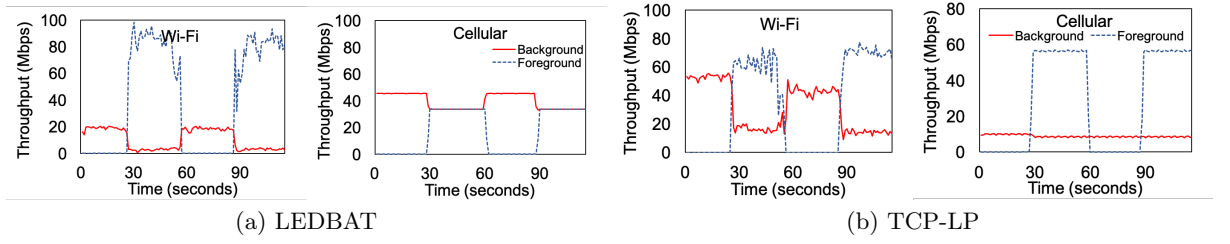


Figure 1: Existing protocols in wired vs wireless networks.

## 2.4 Why existing schemes do not work in cellular networks

Existing methods of congestion control greatly depends on the bottleneck queue to function. They expect the RTT and loss to accurately detect congestion related symptoms like buffer bloat. This expectation is met in the case of wired or Wi-Fi networks since they use FIFO queues. Delay or Loss based congestion control plus FIFO queues will form an efficient ecosystem. However, in the case of wireless networks this combination does not work. The reason being per user queues (20), the operation of PF scheduler and other factors like environmental conditions that lead to propagation delays (5) and interference in the radio front. As we discussed in the previous section the RTT and delay now includes the side effects of decisions made by PF scheduler. Such perturbations adversely affect the congestion inference in most delay based congestion control protocols. Therefore it is intuitive that such protocols are less efficient in determining the state of the network in cellular environment.

We conducted a simple experiment to prove this hypothesis (1). We use two of the most commonly used protocols to deliver background traffic, namely LEDBAT and VEGAS (21),

(22). The test-bed contains 2 servers situated in the lab environment, connected via high speed Local Area Network (LAN) to 2 test phones connected to the same Wi-Fi access point. One of the servers runs Cubic and serves high priority foreground flow to one of the phones while the other uses LEDBAT and VEGAS in subsequent runs to serve lower priority, bulk, continuous background flow to the other phone. The foreground flow joins in fixed intervals to form an on-off pattern (send for 30 seconds and then stay idle for the next 30). We repeat this experiment in a different setting where the phones are now forcefully connected to the same base station but uses the exact same traffic pattern. We expect TCP-LP protocols to back off whenever the foreground flow joins in, which is exactly what happens in the case of wired networks as seen in Figure 1(a). However, in the second iteration of this experiment in cellular settings, we find that both LEDBAT and TCP-LP struggle to accurately determine the state of network, demonstrated in Figure 1(b). TCP-LP calculates a very low decision threshold, thereby sending very less traffic and under-utilizing the network whereas LEDBAT ends up competing with the foreground flow.

This pilot study validates our hypothesis as to why existing TCP-LP protocols do not work well in cellular networks. Therefore, we see that there is need for a new transport protocol tailor made for cellular networks for sending lower priority background traffic. We developed such a protocol that determines congestion more accurately in the cellular environment when comparing to its counterparts.

## CHAPTER 3

### DESIGN AND IMPLEMENTATION

Design and implementation of this idea has been mentioned in detail in (1). We go through a brief overview of the operation and the design choices that we make. The prime motive of this algorithm is to favor higher priority foreground flows like video streaming, VOIP and text messages, when it shares the same bottleneck queue in the last hop, with lower priority background flows like FOTA or cloud updates. In such scenarios, limiting the throughput of background flows will free up extra bandwidth which can be utilized by the foreground flows. This will give better throughput and smaller Flow Completion Time (FCT) to the competing foreground flows. However, if the foreground flows suffers from bad signal strength or other factors limiting its achievable throughput, Legilimens should be agile enough to capture the spare capacity thereby generated, promoting better resource utilization. On a higher level, Legilimens is a foreground flow favoring protocol that sends background traffic utilizing only the spare capacity available, post prioritization of foreground flow. For this purpose, Legilimens identifies all other traffic competing with itself as higher priority foreground flows. Other key feature of this solution is that the only changes required to deploy this is in the sender stack. No other network element has to be modified, thereby making the deployment process easier.

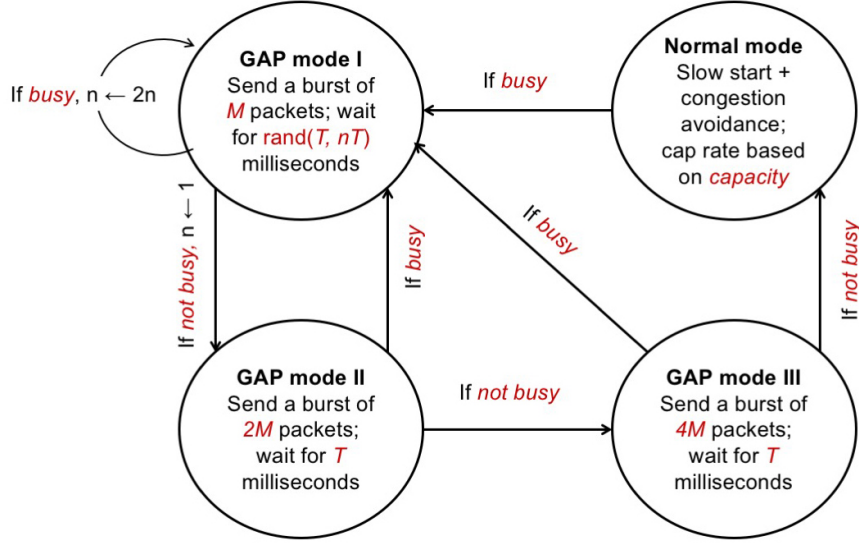


Figure 2: High-level overview

### 3.1 TCPLegilimens

Legilimens infers the presence of other flows in the end to end network path without any explicit feedback from network elements. It makes use of only the meta-data from the ACKs, specifically TCP timestamp and ACK number. TCP timestamp for a connection can be enabled without any changes needed from the client side (12). Legilimens has been designed and implemented on top of RENO and therefore it uses the same AIMD-style of congestion control during its regular operation. However unlike RENO, Legilimens limits the throughput to 80% of the estimated capacity. This is to ensure that we are leaving enough bandwidth for any new coming flows to get through and to avoid any form of buffer bloats or cell saturation that might lead to poor performance (2).

Figure (1) shows a complete state machine depicting the various operation modes of Legilimens. Essentially Legilimens has two operation modes, the normal mode and the Gradually Aggressive Probing (GAP) mode. In the normal mode there is no congestion in the network and Legilimens uses the AIMD style congestion control here. When Legilimens detects contention, it enters into GAP mode and pauses for a random interval of time thereby yielding to the competing foreground flow. Legilimens thereafter uses probing technique to determine spare bandwidth. However, we do not use the classic probing technique. Instead, Legilimens probes in 3 separate stages or GAP modes. In GAP mode 1, we send  $M$  packets and waits for a random interval of time, if we detect contention we wait for longer duration before we probe again with the same number of packets. However, if we detect no congestion, we probe with  $2M$  packets and waits for 250ms and if we detect spare capacity we send  $4M$  packets and wait for 250 ms again. This probing technique ensures that the trade-off between the accuracy of probing and its network impact is balanced. Legilimens uses regular data packets for probing, which does not impose any extra load (overhead) on the network. GAP mode prevents any unwanted oscillations between normal mode and dormant mode.

Legilimens estimates capacity by counting the number of packets serviced in every TTI. As PF scheduler picks the phone with highest utility value in a TTI, it is bound to get all the RBs in that TTI and therefore counting the number of packets serviced will give you a much accurate estimation of capacity. Legilimens detects business by analyzing scheduling gaps in the received timestamps of successive packets. If two successive packets were serviced by PF scheduler in the same TTI, then the gap in received timestamp will be smaller than 1 ms. However, if

there are competing flows in the base station, PF will not assign all RBs to the same receiver in successive TTIs and creating a visible gap of more than 1 ms in receiver timestamps. Legilimens uses this observation as an indication of busyness. Legilimens also measures the instantaneous throughput and cross checks it with the estimated capacity. A lower instantaneous capacity also indicates busyness.

## CHAPTER 4

### TEST STRATEGIES AND EVALUATION

In this chapter we introduce the various verification techniques used, the motivations towards using them, a detailed layout of the test setup, the metrics that we compare and also the test results obtained. We evaluate the performance of various background transport protocols, comparing them in particular with the proposed scheme Legilimens to evaluate its performance. Intuitive test strategy would be to run a mixture of foreground and background flows to phones connected to the same base station and use the background protocol in question as the underlying transport protocol for servers generating the background flows. However, using a live base-station during busy hours will give random results due to unpredictable cross traffic in the cell, making it near impossible to derive any conclusive performance metrics or justifications. Unpredictable response also makes it impossible to tune the algorithm to handle corner cases. We first evaluate the algorithms on an emulated LTE test platform with both 1 on 1 traffic pattern and also mixed workloads. Secondly, we evaluate them on real world test setup with 1 on 1, mixed workload and randomized mixed workload tests. We then perform a mobility test to make sure that Legilimens functions even when the phones suffer from varying signal strength. We evaluate each experiments in their respective subsections.

## 4.1 Emulated Test-bed

To evaluate the performance of algorithms in isolated test environment, we use the phantomnet (6) emulator. Phantomnet provides binary access to OpenEPC 3GPP framework implementation. It lets you configure an end to end experimental LTE test setup made of actual hardware or emulated network components configured using emulab profiles. The test setup we use contains a server running Open Air Interface (OAI) inside the LTE packet core and eNodeBs (base stations). eNodeB is based on Software-Defined Radio (SDR) running OAI(Intel NUC + USRP B210). PhantomNet also provides Nexus 5 phones accessible via Android Debug Bridge(ADB), target server and a GUI interface, Culebra. Log-distance path loss model was used to produce realistic varying Signal to Noise Ratio(SNR). We use 4 Nexus-5 phones connected to the same SDR thereby sharing the same PF scheduler and remotely connected servers generate test traffic both foreground and background. Since the setup is completely isolated, there is no concern of cross traffic altering the results. The scheduler in OAI is not standard PF scheduler, rather a flavor of it which gives RB to all connected devices in every TTI. This is a proprietary scheduler and does not represent those used in real-world base stations. Therefore, we implement a standard PF scheduler on top of OAI for verification purposes.

### 4.1.1 One-on-one testing in emulator

We do basic one-on-one testing as proof of concept to show the effectiveness of Legilimens. The workload involves 2 competing flows foreground and background, using the same bottleneck link serviced by PF scheduler. The background is one continuous flow generated by iperf3 and using Legilimens as the underlying transport protocol whereas the foreground follows an on-

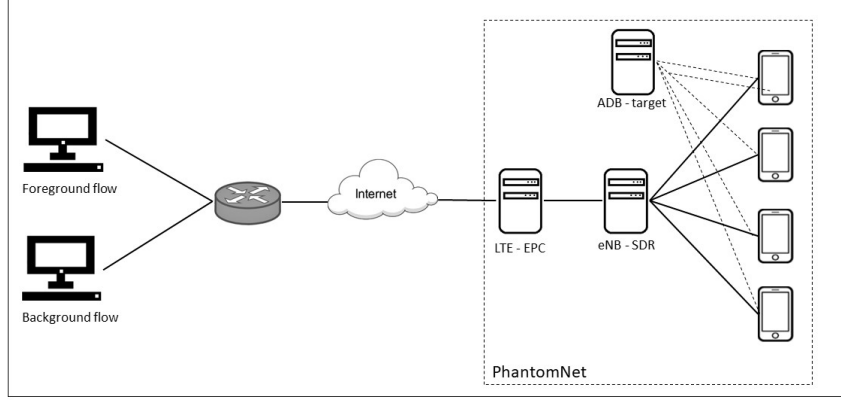


Figure 3: Emulator test framework.

off pattern (sends for 20 seconds then sleeps for 20 seconds) and uses CUBIC. Figure 4 is a per second throughput plot of both the flows. As expected, Legilimens detects contention and backs off whenever the foreground flow joins in and the background flow resumes once spare bandwidth is available.

To study the impact of signal strength of UE on the performance of Legilimens, we add another UE with a weaker or varying signal strength into the mix (Foreground Flow-1). We see in Figure 5 that background flow backs off completely for foreground flow-2 and the foreground is able to utilize this spare bandwidth made available. The UE receiving foreground flow-1 has varying signal strength and therefore is unable to utilize the resource efficiently. However, Legilimens is agile enough to sense spare bandwidth and sends enough background traffic to fill in. But once the signal strength of foreground UE revives, Legilimens backs-off. We see that

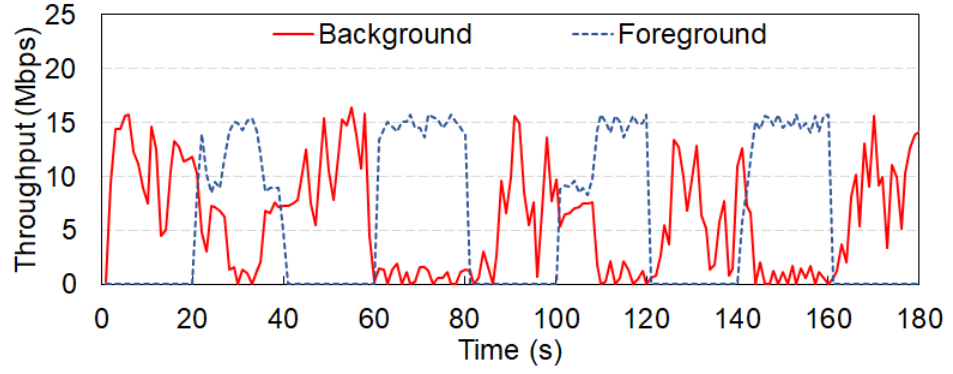


Figure 4: One on one background (Legilimens) vs foreground (CUBIC)

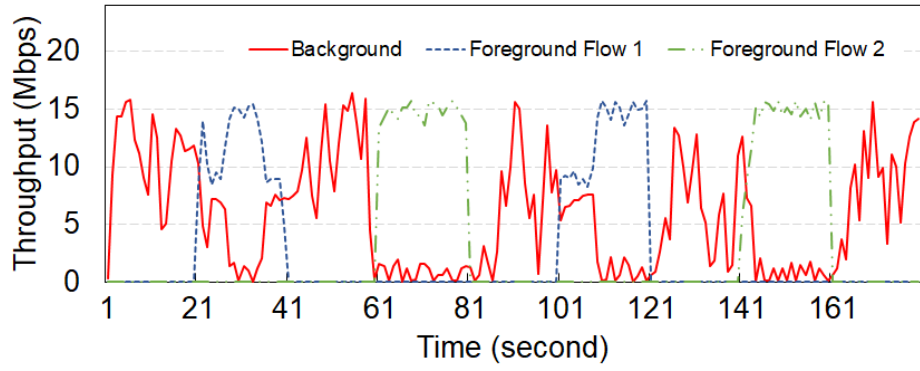


Figure 5: Two on one with background (Legilimens) vs foreground (CUBIC)

Legilimens promotes better link usage by preventing wastage of spare capacity, at the same time ensuring better throughput and FCT for competing foreground flows.

#### 4.1.2 Mixed workload test in emulator

Mixed workload tests are intended to mirror real world traffic scenarios and provide proof of concept for the algorithm in actual workload. Real world cellular traffic constitutes flows of

different sizes and duration, primarily categorized into 3 types. (1) Short flows: 32 KB flows that represent traffic like messages, web objects and other mobile application related data. (2) Medium flows: 1 MB represents content heavy pages, video segments from applications supporting adaptive video streaming, music downloads and so on (3) long Flows: 32 MB video downloads, software and firmware updates. We model the workload similar to observed traffic characteristics, which follows a heavy tailed distribution with majority of the traffic being generated by long flows (23). The short, medium and long flows constitute 10%, 30% and 60% of the total capacity respectively.

We generate 3 different loads using only the foreground flows, measured in terms of physical resource block  $U_{PRB}$  utilization. Low load  $U_{PRB} = 30\%$ , medium load  $U_{PRB} = 60\%$  and high load  $U_{PRB} = 70\%$  we use the combined maximum achievable throughput of the phones as an indication for capacity. Stable tests at higher loads were limited to 70% capacity due to limitations in phantomnet. The above mentioned workload forms the foreground flows in the test setup and they use CUBIC as the underlying transport protocol, whereas the background flow is one continuous long flow that runs throughout the duration of the experiment and uses competing protocols like Cubic, Reno, Vegas, Ledbat, LP and Legilimens.

#### **4.1.2.1 Evaluation of emulated test results**

Figure 6 shows the 50th, 99th and 99.99th percentile Flow Completion Time (FCT) of foreground short flows in the mixed workload experiment, Figure 7, Figure 8 and Figure 9 shows the throughput attained for foreground medium, foreground long and background flow respectively. We see that Legilimens achieves the lowest FCT for short flows across all loads and

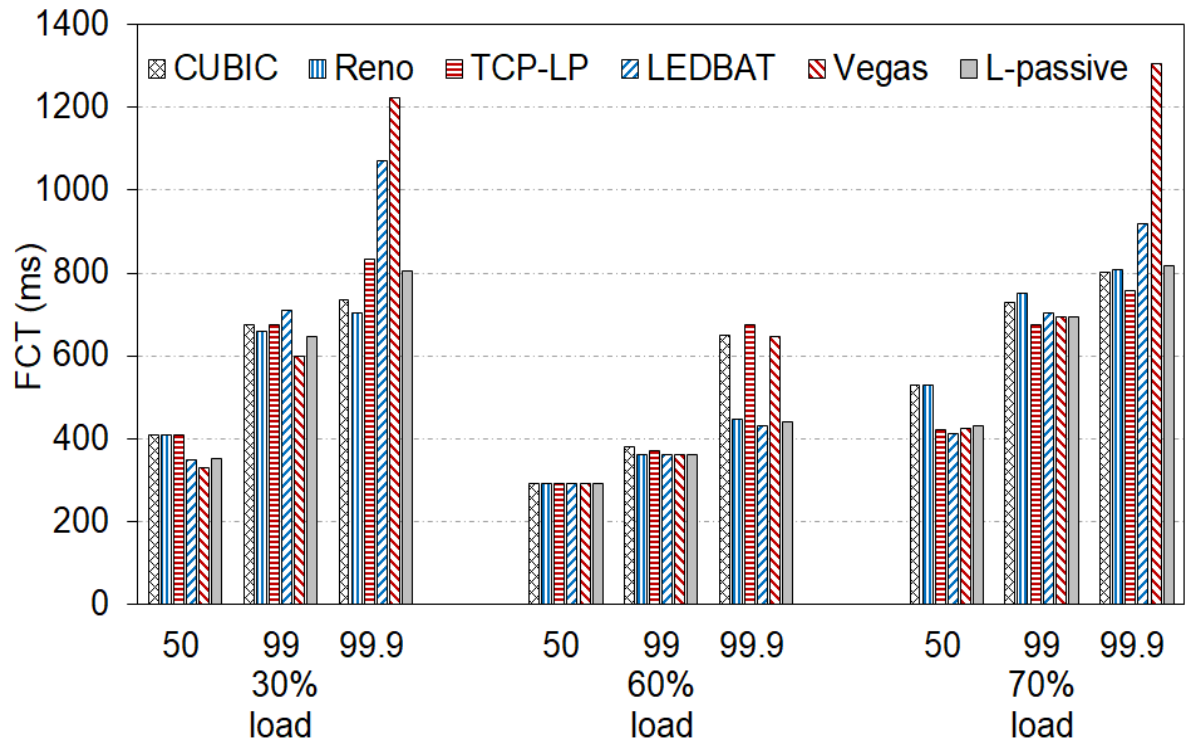


Figure 6: foreground short flow FCT in phantomnet

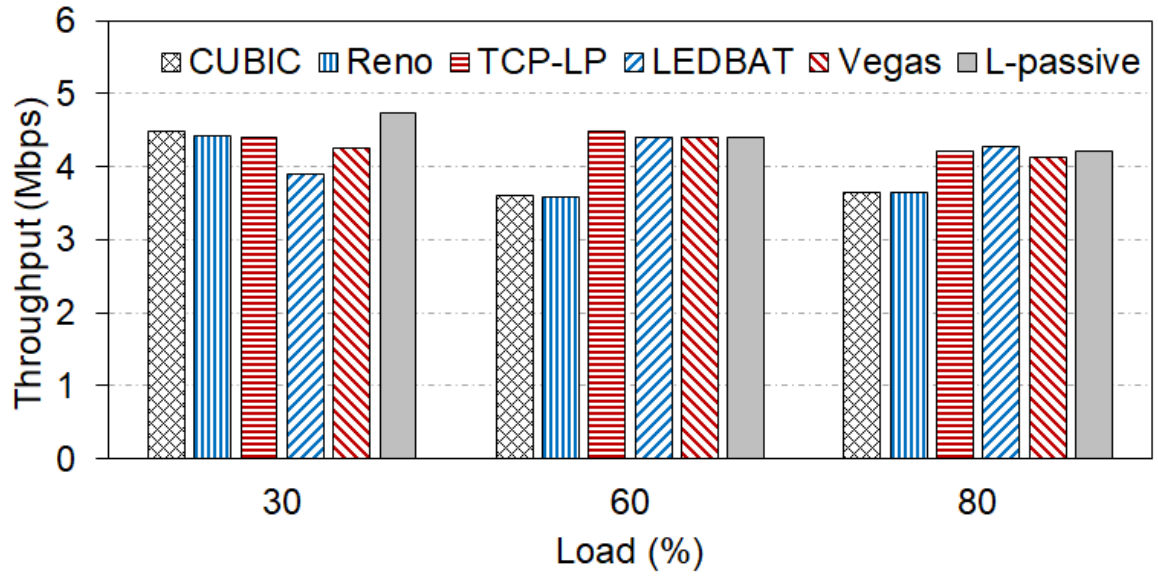


Figure 7: foreground medium flow throughput in phantomnet

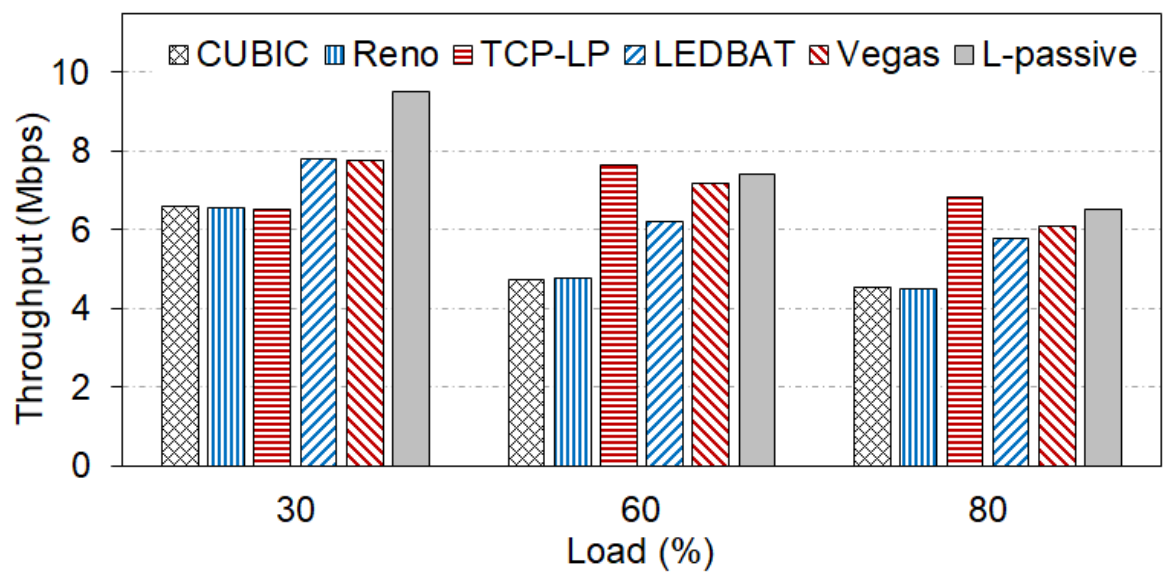


Figure 8: foreground long flow throughput in phantomnet

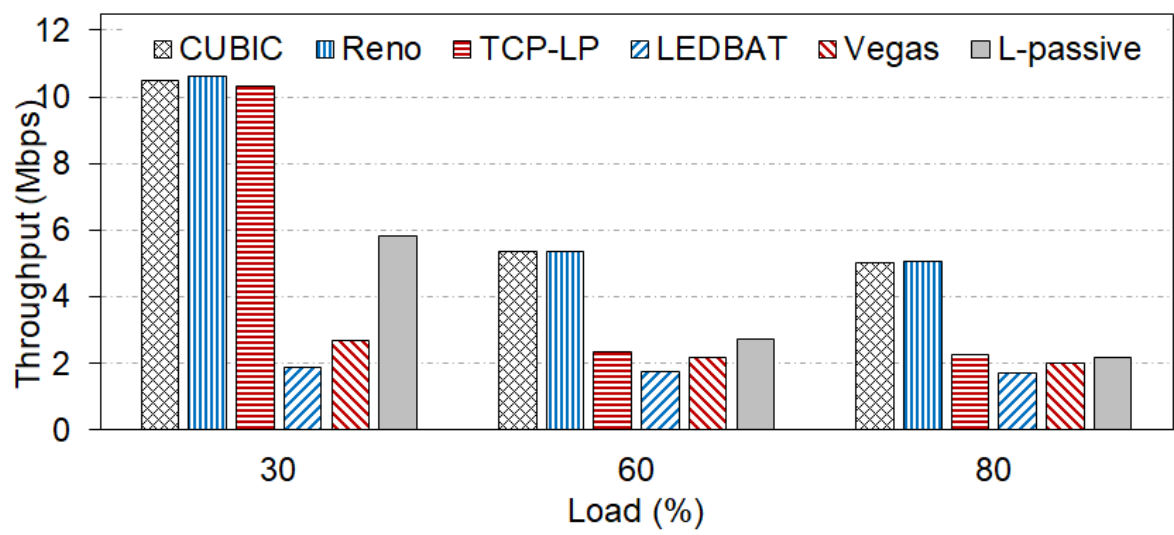


Figure 9: background flow throughput in phantomnet

plotted percentiles, whereas RENO and CUBIC gives consistent performance, VEGAS struggles in the tail 99.9th across all loads which indicates that it gives the worst tail performance of the lot. It is intuitive that since Legilimens limits itself at 80% capacity, the short flows are able to get through unaffected. Medium flow performance is almost identical across all the competing protocols, with a slight inclination towards RENO and CUBIC. This might be because the flow sizes are not large enough to exploit the spare bandwidth made available, further investigation is needed into this observation to confirm this claim.

All background protocols show significantly better performance in long flows, with TCP-LP and Legilimens being the best of them. VEGAS and LEDBAT performs consistently worse than others in terms of background throughput across all loads. This suggests that in cellular networks, the capacity thresholds determined by these protocols are erroneous and so they force the background flow to send less, regardless the availability of spare capacity. Intuitively, corresponding long flow throughput is higher. However an ideal background protocol should attain a balance between the two, back-off for foreground flow at the same time capture spare capacity when available and Legilimens comes closer to achieving this goal.

## 4.2 Real world tests

Simulated and emulated test bed gives you a controlled environment which attempts to mimic the real world traffic scenarios. This is an ideal test method for preliminary modelling, evaluation and bug fixes as the reaction of the algorithm is completely predictable in this scenario. However, it is necessary to deploy the scheme in commercial network to evaluate its performance in real world. This section talks of the real world test setup we use, the deployment

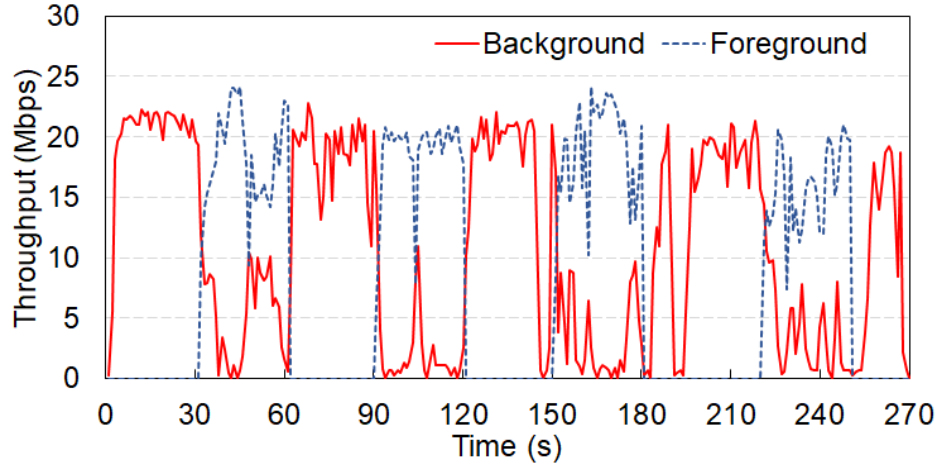


Figure 10: 1-on-1 test between CUBIC and Legilimens in real world setup.

model, the automated framework used and the evaluation of test results hence obtained. We consider 2 tests.

- One-on-one testing between foreground and background.
- Mixed workload testing.

#### 4.2.1 Stationary one-on-one testing in commercial network

We replicate the 1 on 1 Legilimens vs CUBIC experiment in real world setup and the results are in 4.2. Both the phones receiving foreground and background flows are stationary located approximately 150 meters from the macro cell, we also repeat this experiment in mobile test setup in section 4.2.3. We attempt to provide proof of concept for our claim that Legilimens backs off when it detects the presence of foreground flow in the bottleneck link and is agile

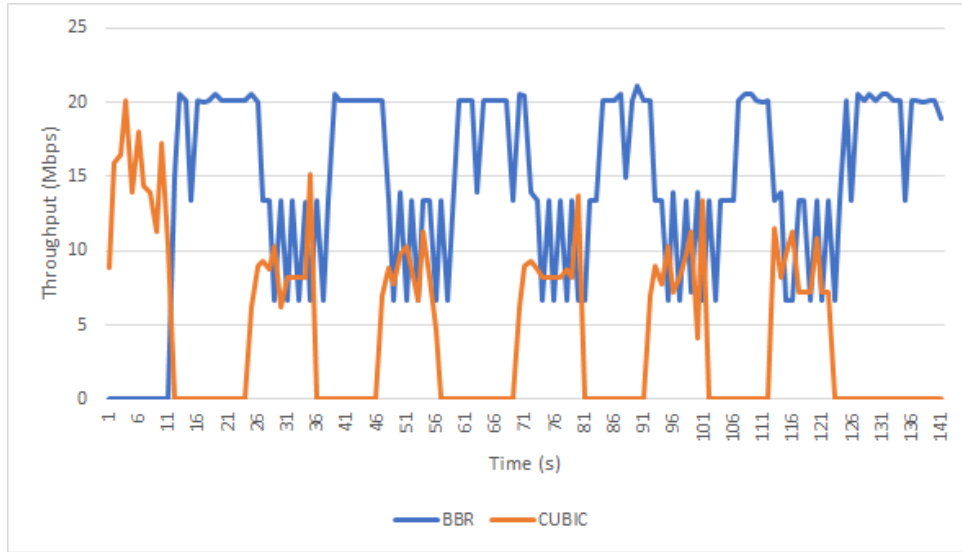


Figure 11: 1-on1 test between CUBIC and BBR in real world setup.

enough to detect when spare capacity becomes available and utilizes it. We see in figure 4.2 (1) that Legilimens lives up-to this expectation.

We also perform 1 on 1 testing between CUBIC and BBR and the results are plotted in 4.2.1. The traffic follows a similar on-off pattern that we have seen throughout with BBR being one continuous flow and CUBIC follows an on-off pattern. We notice that BBR competes with CUBIC and it even is more aggressive in capturing the bandwidth. It is a known limitation of BBR that it does not work well in tandem with loss based protocols. Therefore, we see that BBR is not suited to function as a background protocol and we do not use it in any other mixed workload tests.

#### 4.2.2 Automated test framework for mixed workload test.

The test framework constitutes a lab server running Linux version 4.15 generating foreground traffic, a virtual machine with the modified TCP stack generating background flows and 4 phones physically located in a different state, 3 of which are Samsung J7s and one S6. The phones are running on Android 7.0 (Nougat), band locked to the same carrier and frequency band (10 MHz) on the same macro cell using Gnet tracker pro android application (24). The signal quality represented by "Reference Signal Received Power" (RSRP) fluctuated between -92 and -95 dBm and "Reference Signal Received Quality" (RSRQ) between -10 to -13 dB. The phones use a custom android application to request foreground flows of configurable sizes, numbers and inter arrival times.

The entire test-bed was automated and synchronization among these components were achieved using socket communication. Each run would last for 15 minutes with the background flow remaining ON throughout the run. The number of foreground flows and the time interval between them are configured based on the capacity and load. The framework automatically cycles between different background protocols every 15 minutes by loading the respective patch in the kernel and captures the tcpdump at the server interface. We use tstat (25) to extract relevant metrics as below.

- 99 and 99.9 percentile of flow completion times (FCT) of short flows representing median and tail.
- foreground medium flow throughput
- foreground long flow throughput

- background continuous flow average throughput

TABLE I: Flow numbers for corresponding loads for measured capacity of 25 Mbps.

Load	Short flows	Medium flows	Long flows
30%	1080	203	13
60%	2160	406	26
80%	2880	540	30

The traffic pattern used is identical to the emulated test-bed. With short, medium and long flow constituting 10%, 30% and 60% of the foreground flow traffic respectively (1). We use three different loads for foreground traffic: low load  $U_{PRB} = 30\%$ , medium load  $U_{PRB} = 60\%$  and high load  $U_{PRB} = 80\%$ . The corresponding flow numbers can be found in Table I, these number of flows correspond to the respective contribution to load in terms of bytes transmitted. We spread out the flows equally across the 15 minute test period. We use the combined, maximum achievable throughput of the phones as an indication for capacity, generate traffic of respective load levels using this capacity as an index and verify it by measuring the performance monitoring (PM) counters at the base station for the corresponding 15 minute interval. The load values were determined after monitoring the  $U_{PRB}$  of the macro cell (used in the tests) on a typical week day as shown in figure Figure 12. To minimize the cross traffic affecting the experiment, the tests were conducted during quiet hours post midnight until 6 am when the typical  $U_{PRB}$  is less than 5% on average.

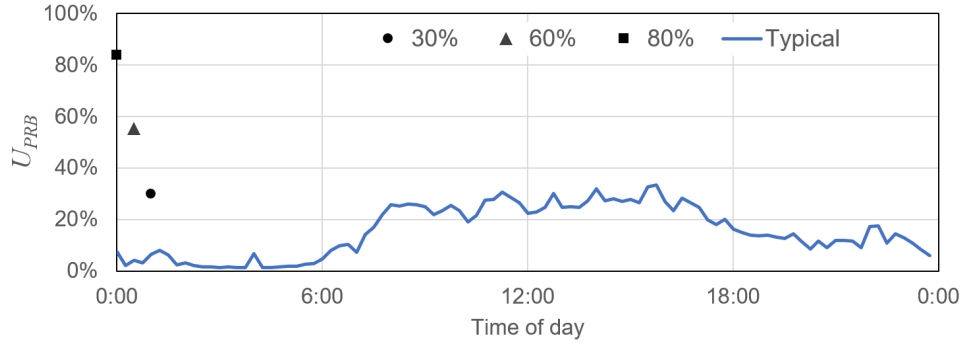
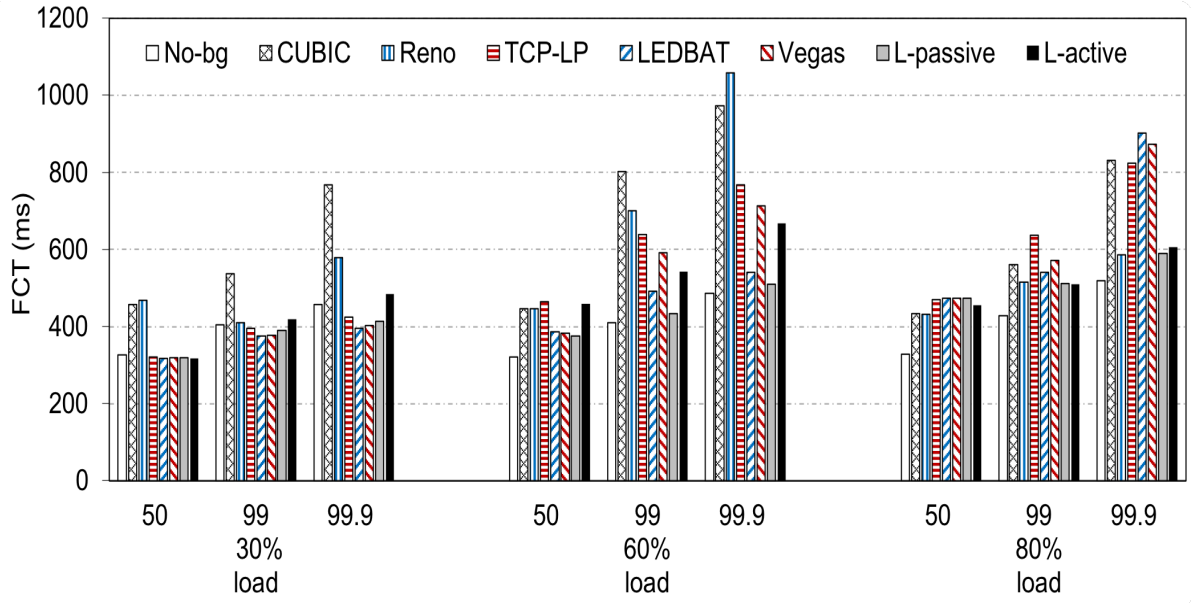
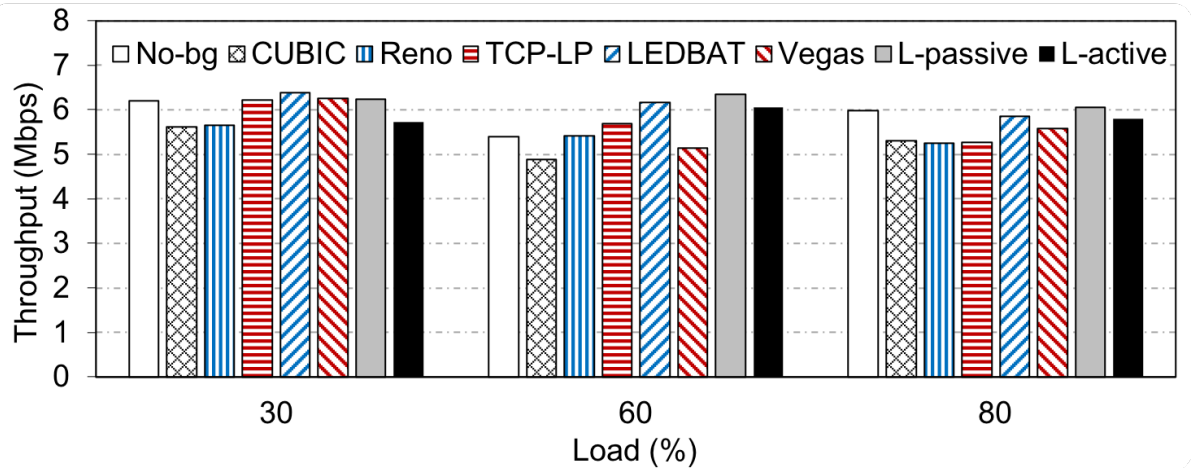
Figure 12: Real world base station  $U_{PRB}$ 

Figure 13: foreground short flow FCT.



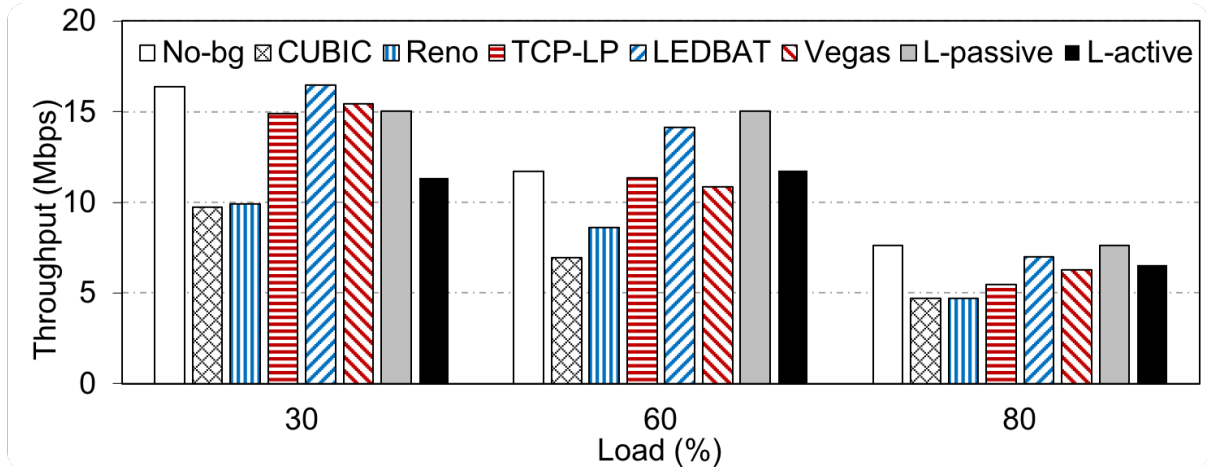


Figure 15: foreground long flow throughput.

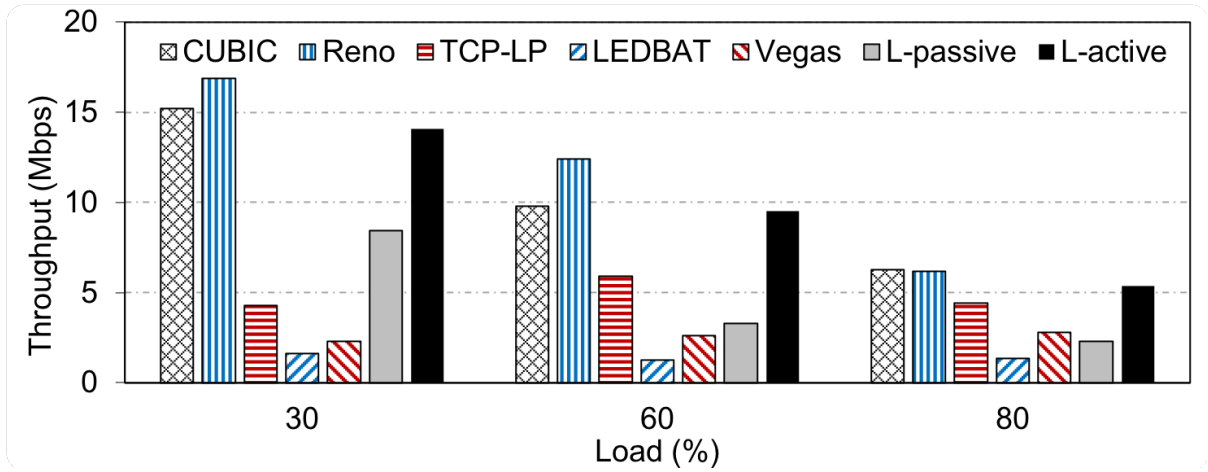


Figure 16: background flow throughput.

#### 4.2.2.1 Evaluation of real world mixed workload experiment

We evaluate the results of mixed work flow experiments in the real world setup. Figure 13, Figure 14, Figure 15 represents the foreground flow performance and Figure 16 represents the background flow performance (1). The plots compares protocols like CUBIC, RENO, LEDBAT, TCP-LP and Legilimens. We use a no background flow run as the yardstick to compare their performances. No background test uses the same foreground test setup the only difference being the absence of background flow which thereby can be considered as an ideal case.

We see that for foreground short flows the FCT is consistently better for the passive version of Legilimens, while the active version achieves better background flow performance at the expense of slightly worse short flow FCT and foreground throughput. CUBIC and RENO aim at fairness among competing flows and therefore are not intended as background protocols. As expected, the corresponding foreground flows performs the worst. Since these protocol aim at maximum utilization, it does not leave any headroom for foreground flows to get through. background protocols like Legilimens, utilizes only 80% of the capacity, thereby leaving enough room for foreground flows. This has significant impact on short flows FCT. This is also the reason why the tail (99.9 %) is so much worse for other protocols compared to Legilimens.

Medium flow performance does not show drastic differences across all protocols. We see that protocols like RENO and CUBIC perform worse in their corresponding foreground medium flow throughput as the load increases. This is because the background flows do not back off, rather tries to attain its fair share of bandwidth. Low Priority Transfer protocols detect the

presence of foreground flow and backs off. Both the active and passive versions of Legilimens performs comparatively better which indicates that it possesses more accurate algorithms in detection of foreground flow, back-off and is more agile in capturing the spare bandwidth when made available. We talk about the background flow performance in this section. Let us Keep in mind that in the ideal world, the aim of Low Priority Transfer protocols is to prioritize foreground flows that compete with itself in the bottleneck link. Therefore, their corresponding background flow performance will degrade with increase in the foreground load. This is desirable because it opens up more bandwidth for foreground flows leading to better foreground flow performance. We see an expected trend in protocols like CUBIC and RENO as the background flow throughput converges to fair share. We see a proportional decrease in throughput with increasing load and we expect a much sharper decline for Low Priority Transfer protocols. However, for Low Priority Transfer protocols that depends on accuracy of the calculated capacity and thresholds like LEDBAT, TCP-LP and VEGAS, the performance is near identical even across drastic changes in foreground loads. This is not desirable, as it indicates significant under-utilization in low loads and competition in higher loads. We see a favorable trend in background flow performance of Legilimens where the throughput degrades with increasing foreground load opening up spare capacity for foreground flows. We see a performance trade-off between the active and passive version of Legilimens, with the active version giving more aggressive background flow performance while passive Legilimens background flow is more conservative. This shows that Legilimens is configurable in terms of the background and foreground performance. Passive version can blindly prioritize the foreground

flows whenever they are present, giving considerably better performance for foreground flows compared to protocols that promote fair sharing of available bandwidth. Active version of Legilimens attains better background performance while slightly sacrificing the foreground flow performance (still better than all compared protocols).

#### 4.2.3 Mobile one-on-one testing in commercial network

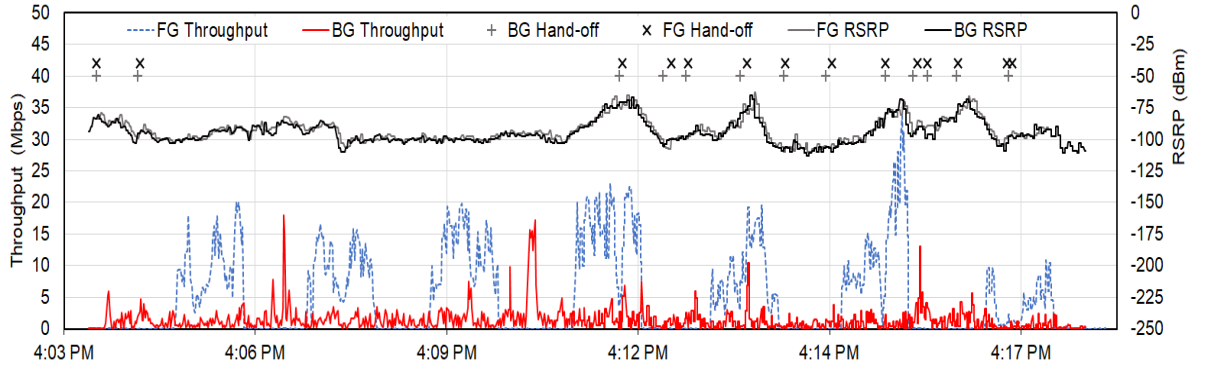


Figure 17: Mobile testing of one-one workload.

To evaluate Legilimens in a mobile scenario, where changing radio signal and different cell loads can be frequently encountered, we conduct a mobile test. The test lasts for about 37 minutes, starting with walking for about 8 minutes, then driving at moderate speed (30-70 km/h) for about 3 minutes, followed by freeway speed (70-110 km/h) for 10 minutes, and

completing the remainder of the test at moderate driving speed. The total distance is about 24 km on freeway and residential area roads. For clarity, we show the first 14 minutes of the test. Two J7 phones are used, placed in a laptop bag, which is carried and then placed on the vehicle seat. One phone runs a continuous Legilimens background flow in passive mode, and another phone runs a foreground CUBIC flow with 1-minute ON/OFF pattern.

Figure 17 shows the results and the mobility profile of this test. Signal strengths, measured as RSRP, varied from -111 dBm to -68 dBm. A total of 13 hand-offs were performed. While the two devices sometimes perform a hand-off a few seconds apart, their signal strength is almost the same, with small but expected deviation.

Performance-wise, we do not expect to see a clear pattern of yielding by Legilimens, simply because we only see two flows among a large volume of regular traffic in the network. However, as expected, we see that Legilimens performs significantly more conservative than CUBIC, which competes with other traffic. Legilimens also does not appear to suffer major disruption by changing signal strength, especially in the second half of the time series, where driving causes frequent hand-offs.

#### 4.2.4 Mixed work-load experiment with randomized traffic pattern

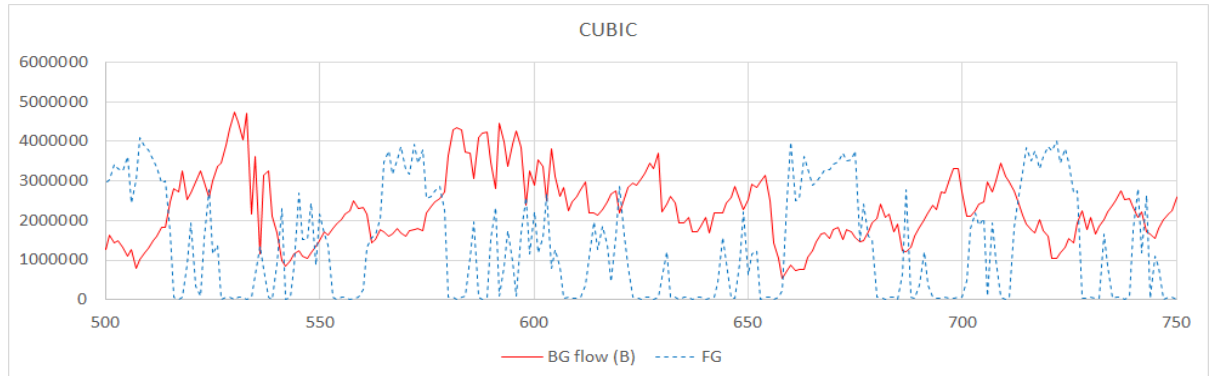


Figure 18: Randomized workload with background using CUBIC

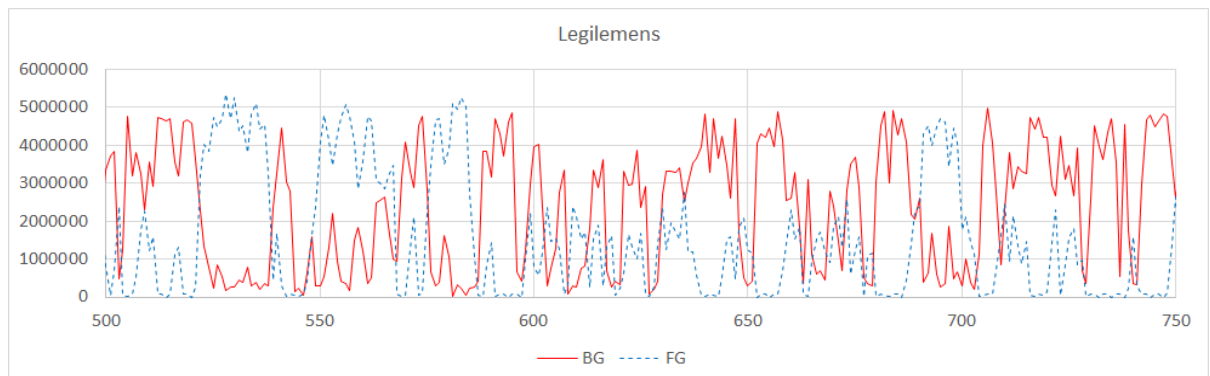


Figure 19: Randomized workload with background using Legilimens

The mixed work flow experiments uses constant number of short, medium and long flows for the same capacity, with inter arrival time of flows changing based on the completion of previous flow. However we see that the flows follows a pattern which might not ideally mimic the real world traffic pattern. To analyze if this provides any performance aid to the protocols, we perform randomized mixed work-flow experiment. The hardware components in the test setup are the same, however the flow sizes and flow inter arrival times of foreground flows are no longer uniform. They are picked by the sender from a uniform bounded distribution. foreground short flows follows the same pattern, 1 short flow every second. The medium flow sizes can vary randomly between 0.5 and 5 MB, and long flow between 10 and 64 MB. Flow size is calculated using the formula:

$$\text{flowsize} = (u * ((\text{max} - \text{min}) + 1)) + \text{min}$$

, where  $u$  is uniform random number in  $[0,1)$ . Inter arrival time for the next flow is calculated using the formula:

$$\text{InterArrivalTime(IAT)} = -\text{mean}/\ln(1 - u)$$

, where mean is 4 for medium flows. We bound the IAT for medium flow to 15 seconds to supply a healthy number of medium flows. Similarly for long flows, we use a mean of 70 and an upper bound of 160 seconds. background flow is a bulky long flow that runs throughout the duration of the experiment.

Figure 18 shows the results of foreground vs background flows in the randomized experiment where both background and foreground flows use CUBIC as the underlying transport protocol. The foreground flow plot is an accumulated representation of short, medium and long foreground flows in the mix. We see that the competing flows attempt to converge to fair share while the foreground is the bulkier long flow. In case of smaller medium flows, we see that the foreground flows are not large enough for the flows to converge to fair share. Therefore, for medium flows, the throughput is significantly lesser. Figure 19 represents the run where background protocol was Legilimens. We see a much desirable trend here, the background flows yield for foreground flows regardless the flow sizes, at the same time it is agile enough to capture spare capacity. This also shows that Legilimens is robust to varying traffic conditions. Both the experiments were run for same duration in identical hardware settings. Legilimens sent a combined 1.175 GB of foreground traffic and 2.246 GB of background traffic while CUBIC sent only 1.031 GB of foreground traffic and 2.079 GB of background traffic. A detailed flow wise statistics (data transmitted) can be found in table II. Even though the traffic pattern followed is not identical, the flow sizes and IAT are chosen from the same uniform bounded distribution thereby producing comparable tests. This also shows that Legilimens promotes better link utilization in this experiment.

TABLE II: Data transmitted in randomized experiment Cubic vs Legilimens

<b>BG Protocol</b>	<b>FG Short (MB)</b>	<b>FG Medium (MB)</b>	<b>FG Long (MB)</b>	<b>BG (MB)</b>
Cubic	38.84	557.86	434.53	2079.90
Legilimens	41.17	675.85	457.62	2246.33

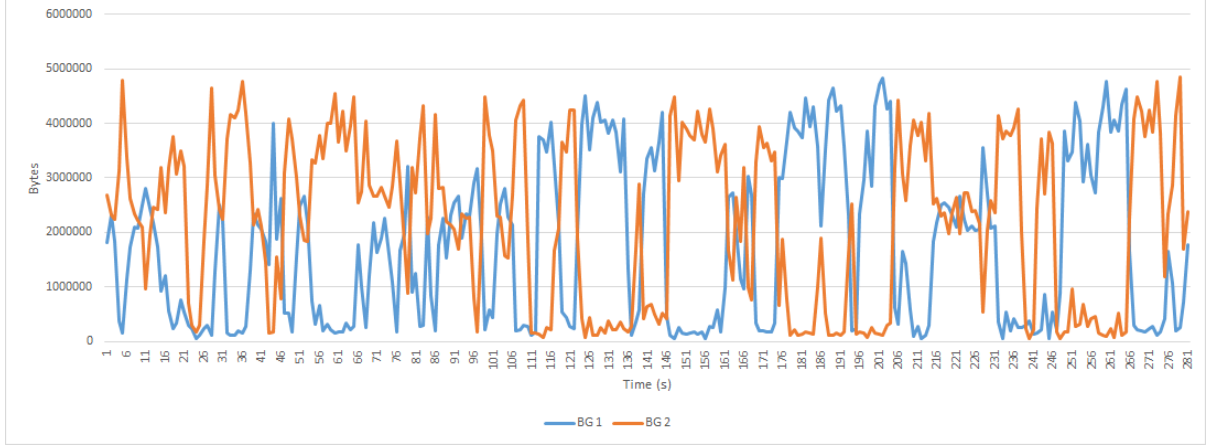


Figure 20: Fairness study of Legilimens in real world setup

#### 4.2.5 Legilimens fairness study in real world network

The effectiveness of Legilimens lies in its ability to accurately detect the presence of competing foreground flows in the bottleneck link. Legilimens assumes every other flow in the network as foreground flow to achieve this, which raises a very important question. What if there are only Legilimens generated flows in the base station, will they fair against each other? To verify this, we perform a fairness study in real world setup. The test was performed during quiet hours, so that no cross traffic interferes with the results. We run 2 bulk backgroundLegilimens flows and capture the tcpdump to analyze per second throughput as seen in Figure Figure 20. We notice that Legilimens achieves fairness among the competing flows. This attributes to the operational characteristics of PF scheduler and Legilimens. Once a UE starts getting contin-

uous opportunity at the base station, the PF utility index of that UE decreases and that of the competing UE increases. Which forces PF scheduler to pick the latter UE over the former and Legilimens of the former UE detects this and backs off for a random interval of time. This random back-off combined with PF scheduler ensures that Legilimens flows achieves fairness among themselves,

We have performed a comprehensive evaluation of the various competing background protocols in isolated as well as real world settings. Traditional background protocols like TCP-LP, LEDBAT and VEGAS have proved to be effective in wired settings however they do not adapt well in cellular networks primarily due to the PF scheduler characteristics. An ideal background protocol will pause or reduce the throughput when it detects a competing foreground flow sharing the bottleneck link and diligently capture spare bandwidth once it is available. Our evaluation suggests that: an accurate foreground flow detection algorithm, appropriate back-off mechanism, subtle probing technique and accurate capacity estimation form the pillars of an ideal background protocol. We see that Legilimens comes closer to these objectives particularly at higher loads by accurately predicting the presence and yielding to foreground flows and also at lower loads by capturing the spare bandwidth, unlike competing background protocols that suffer from severe under-utilization of radio resource. On those regards, Legilimens outperforms various state of the art transport protocols.

## CHAPTER 5

### PREVIOUS WORK

Congestion control schemes for networks is a very heavily researched area, with high throughput and low latency for flows being the main goal. Some of the earliest among the TCP variants include CUBIC (15), RENO (26), NEWRENO (14) and TAHOE (27). CUBIC uses an AIMD style congestion window evolution to cater to high latency networks. We use this flavor of congestion window adaptation. Loss based congestion control protocols face severe performance degradation in cellular networks due to buffer bloat (28), since it has higher tolerance to delays, it keeps pushing the network further deeper into congestion even if small delays in RTT might indicate the onset of congestion. Latest state of the art protocols like BBR, (16) aims to control the congestion window using Bandwidth Delay Product (BDP). It makes use of both RTT and capacity estimate to make use of any spare bandwidth available. BBR aims at achieving the same input rate at the bottleneck queue as is its output rate.

A solution to the limitations of loss based systems is to use delay based protocols which depend on OWD or RTT to infer congestion, but these can be very sensitive to competing flows. Hence, such delay based protocols are ideal choice for protocols that service background flows. Prior research into this field has led to development of protocols like VEGAS (19), LEDBAT (18), TCP-LP (17) and TCP-NICE (29). These protocols are effective in wired and Wi-Fi networks, but does not work well in cellular networks, primarily due to PF scheduler which is not aware of congestion control objectives of these end to end protocols and follows its on

scheduling policies and ends up conflicting with each other. Also cellular links degrade the performance of such protocols (30),(31), (32),(33),(34).

Realizing the above conflict, there has been previous research into congestion control schemes for cellular networks. Sprout (5) and Verus (35) aims at accurately predicting the congestion window by observing the arrival times of packets. Legilimens uses the same observation by QProbe (36). Qprobe uses the operational characteristics of the cellular PF scheduler to identify bottleneck links, whereas Legilimens uses the same technique to identify busyness in the bottleneck link and use this information to scheduler background traffic in such a way that it makes use of only the spare bandwidth available. PropRate (37) uses OWD to achieve better throughput for flows. All the above mentioned protocols, though they work in cellular networks, their goal is to achieve better throughput, FCT and fairness. However, Legilimens is an ideal protocol for background flows and it does not try to compete with other flows in the network and rather backs down to let them through. Legilimens achieves good fairness among other background flows as seen in 4.2.5. There are other techniques like passive estimation of channel power and pilot signal as used in LoadSense (38). But unlike Legilimens which does not require any support from client side devices, it does need support from the phones to make these measurements and hence makes deployment more complicated.

## CHAPTER 6

### CONCLUSION

With the advent of faster network speeds, better coverage, increase in number of IoT and connected mobile devices, the load on cellular networks is only set to increase. Multimedia and other time sensitive traffic still bears higher priority than background time insensitive applications, therefore it is imperative that the services catering to background traffic use an effective, accurate and agile congestion control technique that prioritizes foreground flow over itself and promotes better link utilization by capturing any spare bandwidth available. There are several congestion control algorithms in existence designed to achieve this goal, however they are efficient on cellular networks. The main aim of this thesis was to test the various congestion control techniques out there and compare it against new proposed scheme, the Legilimens. Our test methods indicate that an accurate foreground flow detection algorithm, appropriate back-off mechanism, subtle probing technique and accurate capacity estimation form the pillars of an ideal background protocol. Through extensive evaluation in real world as well as in emulated (and isolated) test setups, we not only provide proof of concept to the effectiveness of Legilimens but also introduce various verification techniques that can be used to prove the effectiveness of any future research works with the same goal.

## 6.1 Future Work

Future work in terms of verification involves using parallel foreground flows in the mixed workload experiments. Serializing the foreground flows affects the actual load on the network. Also, testing various background protocols with multiple background flows in the mixed workload experiments in real networks, which is more closer to realistic scenarios. I intend on completing an ongoing verification test, where I track the congestion window evolution of Legilimens in milli second scale, while running mixed or 1-on-1 foreground workload. This will serve as proof of concept to the effectiveness of Legilimens in smaller time granularity.

Legilimens is developed particularly for downlink traffic. It needs to be verified and if needed modified to work in uplink traffic scenarios as well. Also, it will be beneficial in exploring the deployment possibilities of Legilimens in 5G networks.

## CITED LITERATURE

1. Realizing provider policies on cellular networks, 2018.
2. Andrade, C. E., Byers, S. D., Gopalakrishnan, V., Halepovic, E., Majmundar, M., Poole, D. J., Tran, L. K., and Volinsky, C. T.: Managing massive firmware-over-the-air updates for connected cars in cellular networks. In Proceedings of the 2Nd ACM International Workshop on Smart, Autonomous, and Connected Vehicular Systems and Services, CarSys '17, pages 65–72, New York, NY, USA, 2017. ACM.
3. Global mobile data traffic from 2017 to 2022 (in exabytes per month), 2019.
4. Peterson, L. and Davie, B.: Computer networks: A systems approach. In Computer Networks: A Systems Approach, pages 41–46, 2012.
5. Winstein, K., Sivaraman, A., and Balakrishnan, H.: Stochastic forecasts achieve high throughput and low delay over cellular networks. In Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation, nsdi'13, pages 459–472, Berkeley, CA, USA, 2013. USENIX Association.
6. Banerjee, A., Cho, J., Eide, E., Duerig, J., Nguyen, B., Ricci, R., Van der Merwe, J., Webb, K., and Wong, G.: Phantomnet: Research infrastructure for mobile networking, cloud computing and software-defined networking. GetMobile: Mobile Computing and Communications, 19(2):28–33, 2015.
7. Xu, X., Jiang, Y., Flach, T., Katz-Bassett, E., Choffnes, D. R., and Govindan, R.: "investigating transparent web proxies in cellular networks". In PAM, 2015.
8. Lte overviewe, 2019.
9. Lte in a nutshell.
10. Barayan, Y. and Kostanic, I.: Performance evaluation of proportional fairness scheduling in lte. In Proceedings of the World Congress on Engineering and Computer Science 2013 Vol II, volume 2, page 1, October 2013.

11. Zhang, L., Braden, R. T., and Jacobson, V.: TCP Extension for High-Speed Paths. RFC 1185, October 1990.
12. Jacobson, V., Braden, B., and Borman, D.: Tcp extensions for high performance. RFC 1323, RFC Editor, May 1992.
13. Dukkupati, N., Refice, T., Cheng, Y., Chu, J., Herbert, T., Agarwal, A., Jain, A., and Sutin, N.: An argument for increasing tcp's initial congestion window. Computer Communication Review, 40(3):26–33, 2010.
14. Henderson, T., Floyd, S., Gurtov, A., and Nishida, Y.: The newreno modification to tcp's fast recovery algorithm. RFC 6582, RFC Editor, April 2012.
15. Rhee, I., Xu, L., Ha, S., Zimmermann, A., Eggert, L., and Scheffenegger, R.: Cubic for fast long-distance networks. RFC 8312, RFC Editor, February 2018.
16. Cardwell, N., Cheng, Y., Gunn, C. S., Yeganeh, S. H., and Jacobson, V.: Bbr: Congestion-based congestion control. ACM Queue, 14, September-October:20 – 53, 2016.
17. Kuzmanovic, A. and Knightly, E. W.: Tcp-lp: A distributed algorithm for low priority data transfer. In INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies, volume 3, pages 1691–1701. IEEE, 2003.
18. Shalunov, S., Hazel, G., Iyengar, J., and Kuehlewind, M.: Low extra delay background transport (ledbat). RFC 6817, RFC Editor, December 2012.
19. Brakmo, L. S. and Peterson, L. L.: Tcp vegas: End to end congestion avoidance on a global internet. IEEE Journal on selected Areas in communications, 13(8):1465–1480, 1995.
20. Bodrog, L., Horváth, G., and Vulkán, C.: Analytical tcp throughput model for high-speed downlink packet access. IET software, 3(6):480–494, 2009.
21. Rossi, D., Testa, C., Valenti, S., and Muscariello, L.: Ledbat: The new bittorrent congestion control protocol. In ICCCN, pages 1–6, 2010.
22. Kuzmanovic, A. and Knightly, E. W.: Tcp-lp: low-priority service via end-point congestion control. IEEE/ACM Transactions on Networking (TON), 14(4):739–752, 2006.

23. Huang, J., Qian, F., Guo, Y., Zhou, Y., Xu, Q., Mao, Z. M., Sen, S., and Spatscheck, O.: An in-depth study of lte: Effect of network protocol and application behavior on performance. In Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM, SIGCOMM '13, pages 363–374, New York, NY, USA, 2013. ACM.
24. Net monitor and drive test tool application for umts/gsm/lte/cdma/evdo network, 2016.
25. Tcp statistic and analysis tool, 2016.
26. Fall, K. and Floyd, S.: Simulation-based comparisons of tahoe, reno and sack tcp. SIGCOMM Comput. Commun. Rev., 26(3):5–21, July 1996.
27. Jacobson, V.: Congestion avoidance and control. In Symposium Proceedings on Communications Architectures and Protocols, SIGCOMM '88, pages 314–329, New York, NY, USA, 1988. ACM.
28. Jiang, H., Wang, Y., Lee, K., and Rhee, I.: Tackling bufferbloat in 3g/4g networks. In Proceedings of the 2012 Internet Measurement Conference, IMC '12, pages 329–342, New York, NY, USA, 2012. ACM.
29. Chen, J., Mahindra, R., Khojastepour, M. A., Rangarajan, S., and Chiang, M.: A scheduling framework for adaptive video delivery over cellular networks. In Proceedings of the 19th Annual International Conference on Mobile Computing & Networking, MobiCom '13, pages 389–400, New York, NY, USA, 2013. ACM.
30. Wang, J., Huang, A., WeiWang, Zhang, Z., and Lau, V. K. N.: On the transmission opportunity and tcp throughput in cognitive radio networks. Int. J. Commun. Syst., 27(2):303–321, May 2012.
31. Lu, F., Du, H., Jain, A., Voelker, G. M., Snoeren, A. C., and Terzis, A.: Cqic: Revisiting cross-layer congestion control for cellular networks. In Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications, pages 45–50. ACM, 2015.
32. Ludwig, R. and Katz, R.: The eifel algorithm: Making tcp robust against spurious retransmissions. ACM Computer Communication Review, 30:30–36, January 2000.
33. Gurtov, A. and Ludwig, R.: Responding to spurious timeouts in tcp. In Proc. of IEEE INFOCOM, volume 3, pages 2312–2322, 2003.

34. Liu, X., Sridharan, A., Machiraju, S., Seshadri, M., and Zang, H.: Experiences in a 3g network: Interplay between the wireless channel and applications. In MOBICOM, pages 211–222. ACM, 2008.
35. Zaki, Y., Pötsch, T., Chen, J., Subramanian, L., and Görg, C.: Adaptive congestion control for unpredictable cellular networks. In Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication, SIGCOMM '15, pages 509–522, New York, NY, USA, 2015. ACM.
36. Baranasuriya, N., Navda, V., Padmanabhan, V. N., and Gilbert, S.: Qprobe: Locating the bottleneck in cellular communication. In Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies, CoNEXT '15, pages 33:1–33:7, New York, NY, USA, 2015. ACM.
37. Leong, W. K., Wang, Z., and Leong, B.: Tcp congestion control beyond bandwidth-delay product for mobile cellular networks. In Proceedings of the 13th International Conference on Emerging Networking EXperiments and Technologies, CoNEXT '17, pages 167–179, New York, NY, USA, 2017. ACM.
38. Chakraborty, A., Navda, V., Padmanabhan, V. N., and Ramjee, R.: Coordinating cellular background transfers using loadsense. In Proceedings of the 19th annual international conference on Mobile computing & networking, pages 63–74. ACM, 2013.

## APPENDICES

7/15/2019

Muhammad Usama Chaudhry

I am writing to request permission to use the following materials from your publication (Realizing Provider Policies on Cellular Networks, 2018-11-20) in my thesis "Evaluation of Background Transport Protocols in Production and Experimental LTE Networks". This material will appear with changes noted below. Unless you request otherwise, I will use the conventional style of the Graduate College of the University of Illinois at Chicago as acknowledgment.

1. Behavior of existing protocols (Figure 4, Page 15)
2. TCP Chicago congestion control (Figure 8, Page 23).
3. Typical UPRs and workload impact (Figure 11, Page 33).
4. FCT for short flows in the real network (Figure 12, Page 34) – there will be both active and passive versions of the protocol TCP-Chicago.
5. Throughput in the real network (Figure 13, Page 34) – there will be both active and passive versions of the protocol TCP-Chicago.

A copy of this letter is included for your records

Thank you for your kind consideration of this request.

Sincerely,

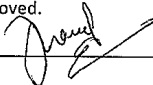
Shibin Mathew

2434 W Flourney Street, Unit 2, Chicago, Illinois, 60612

---

The above request is approved.

Approved by: \_\_\_\_\_



Date: 07/15/2019

## VITA

<b>NAME</b>	Shibin Mathew
<b>EDUCATION</b>	B.Tech., Electronics and Communication, Government Model Engineering College Thrikkakara, Kochi, Kerala, India. 2014
<b>TA</b>	Computer Algorithms (CS401, Spring 2018)
<b>PUBLICATIONS</b>	