CSE 610
Special Topics on Mobile Sensing & Mobile Networks
Project Report

Project members details:

First Name	Last Name	Email	Person Number
Mohiuddeen	Khan	mohiudde@buffalo.edu	50464453
Yashwanth	Rama Krishna Reddy	ramakri4@buffalo.edu	50478775

Problem Statement & Abstract

- This project focuses on evaluating the performance of different congestion control algorithms within cellular networks using Pantheon, a specialised network measurement platform.

- The experiment entails establishing a remote server while employing a mobile phone as the client, thereby facilitating the evaluation of UDP/TCP connections between the client and server under distinct congestion control algorithms. The key metrics used to assess performance encompass throughput, which measures the amount of data successfully transmitted, and delay, which quantifies the time taken for data to travel between the client and server.

- Through a meticulous examination of the collected data, this study aims to provide an extensive analysis elucidating the strengths and weaknesses of the tested congestion control algorithms in the context of cellular networks.

- This research endeavour holds the potential to enhance our understanding of congestion control mechanisms and contribute valuable insights towards optimising network performance in mobile communication scenarios.

Acknowledgement

Primarily we would like to thank our supervisor Prof. Yaxiong Xie, whose valuable guidance has been the one that helped us patch this project and make it full proof success. His suggestions and his instructions have served as the major contribution in the project.

Mohiuddeen Khan, mohiudde@buffalo.edu
Yashwanth Rama Krishna Reddy, ramakri4@buffalo.edu

Chapter 1: Introduction

1.1 Pantheon

Pantheon is a platform that is open to the public and created to aid in the study and creation of Internet congestion control schemes. With a variety of network settings and scenarios, Pantheon offers a platform for evaluating congestion management methods under controlled circumstances. It includes a suite of realistic network conditions that emulate a variety of Internet paths, as well as a set of metrics for evaluating the performance of congestion control algorithms. By providing a standardized platform for evaluating congestion control algorithms, Pantheon enables researchers and developers to compare the effectiveness of different approaches and improve the overall quality of Internet congestion control.[1]

Pantheon-tunnel comprises software controlling a virtual network device (TUN) at each endpoint. The software captures all IP datagrams sent to the local TUN, assigns each a unique identifier (UID), and logs the UID and a timestamp. It then encapsulates the packet and its UID in a UDP datagram, which it transmits to the other endpoint via the path under test. The receiving endpoint decapsulates, records the UID and arrival time, and delivers the packet to its own Pantheon-tunnel TUN device.

Chapter 2: Literature Review

2.1 Congestion Control

Congestion control refers to the techniques and mechanisms employed in computer networks to manage and regulate the flow of data traffic, preventing network congestion and ensuring efficient and reliable data transmission. Network congestion occurs when the demand for network resources exceeds its capacity, leading to performance degradation, packet loss, increased latency, and reduced throughput. Congestion control mechanisms aim to detect and alleviate congestion to maintain network stability and quality of service [2].

Throughput

Throughput refers to the amount of data that can be transmitted over a network in a given period of time.
In the context of the paper, the authors use throughput as a metric for evaluating the performance of different congestion control schemes. They measure the average throughput achieved by each scheme under different network conditions, such as varying levels of congestion and path dynamics.

Delay

Delay refers to the amount of time it takes for data to travel from one point in a network to another.
In the context of the paper, the authors use delay as a metric for evaluating the performance of different congestion control schemes. They measure the average delay experienced by each scheme under different network conditions, such as varying levels of congestion and path dynamics.

2.2 Congestion Control Schemes [3]

The following are the congestion control schemes we used in the experiment:

1. Cubic: Cubic is a popular TCP congestion control algorithm that aims to achieve high network utilisation while maintaining fairness. It uses a cubic function to adjust the congestion window size based on network conditions, gradually increasing the sending rate during periods of low congestion and reducing it during periods of high congestion.

2. WebRTC: WebRTC (Web Real-Time Communication) is a communication protocol that includes its own congestion control mechanism. It utilizes a combination of

algorithms such as Google Congestion Control (GCC) and NADA (Network-Assisted Dynamic Adaptation) to adapt the transmission rate based on network conditions and prioritize real-time media streams.

3. SCReAM: SCReAM (Self-Clocked Rate Adaptation for Multimedia) is a congestion control algorithm specifically designed for real-time multimedia communication, such as video conferencing. It uses feedback from both the sender and receiver to estimate the available network capacity and adjust the sending rate accordingly, aiming to maintain low latency and high video quality.

4. QUIC: QUIC (Quick UDP Internet Connections) is a transport protocol developed by Google that incorporates its own congestion control mechanism. QUIC uses a variant of the TCP Cubic algorithm for congestion control, but operates over UDP instead of TCP. It also includes features like packet pacing and loss recovery to optimise performance for applications such as web browsing and streaming.

5. Sprout: Sprout is a congestion control scheme designed for interactive real-time applications like online games. It utilises a feedback loop and a model of the network path to estimate the available bandwidth and adjust the sending rate. Sprout aims to provide low latency, responsiveness, and fairness by adapting to changing network conditions.

6. Indigo: Indigo is a congestion control scheme that focuses on providing high throughput in data centre networks. It employs a combination of packet pacing and Explicit Congestion Notification (ECN) feedback to control the congestion window size and regulate the sending rate, optimizing the network utilization and reducing packet loss.

7. Taova: Taova (Topology-aware Overlay Assisted Video Adaptation) is a congestion control scheme for video streaming over overlay networks. It combines information from both the overlay network and the underlying network to estimate the available bandwidth and adapt the video bitrate accordingly, aiming to provide a smooth streaming experience and reduce congestion-related issues.

8. Fillp: Fillp (Forward Interactive Low Latency Protocol) is a congestion control algorithm designed for real-time interactive applications like video conferencing and gaming. It utilizes a combination of delay-based and loss-based congestion control mechanisms to estimate the available network capacity and adjust the sending rate, aiming to minimize both latency and packet loss.

9. PCC: PCC (Performance-oriented Congestion Control) is a congestion control scheme that uses machine learning techniques to adaptively adjust the sending rate. It learns the characteristics of the network path and dynamically modifies the congestion control policy to achieve high throughput, low latency, and fairness.

10. LEDBAT: LEDBAT (Low Extra Delay Background Transport) is a congestion control algorithm specifically designed for background transfers, such as file downloads and software updates. It prioritizes the responsiveness of interactive traffic by limiting its impact on the network, adapting its sending rate based on delay and ECN feedback, and providing fair sharing of available bandwidth with other traffic.

Chapter 3: Proposed Work

As told earlier in the project, we leveraged pantheon to perform our experiment. We are going to perform 3 experiments in this study. The 3 experiments are:

1) Comparison of schemes with different locations. (Performed using T-Mobile)
2) Comparison of schemes with different timings. (Performed using AT&T)
3) Comparison of schemes with different amounts of concentrated population. (Performed using AT&T)

Note: Some of the congestion control schemes ran only once or twice despite attempting multiple attempts and some schemes failed for some networks.

The experiments are described in detail later. The set-up instructions for the software are:

<u>**3.1 Set-up Instructions [2]**</u>

Step 1: Downloading the Pantheon Open Source Code
Unleash the power of Pantheon by acquiring its exceptional open source code from the renowned platform, GitHub. Delve into the wonders of collaborative development and cutting-edge technology.

Step 2: Following the Enlightening Readme.md Instructions
Embark on your Pantheon journey by carefully adhering to the enlightening guidance provided in the readme.md file. This comprehensive resource will serve as your compass, ensuring a smooth and seamless installation process.

Step 3: Tailoring Pantheon to Python 2.7 and Ubuntu 18.04
Crafting an optimized environment for Pantheon is crucial. Prepare your local Ubuntu desktop and remote AWS instance, both set to Python 2.7 and Ubuntu 18.04. This harmonious alignment will guarantee the flawless execution of Pantheon's extraordinary capabilities.

Step 4: Generating a Captivating RSA Key Pair
Ignite the security aspect of your Pantheon experience by generating a captivating RSA key pair. This cryptographic duo will serve as the fortress protecting your valuable data. Let your local desktop unveil its mastery by producing this exceptional key pair.

Step 5: Elevating Access to the Remote AWS Instance
Unlock the gateway to the remote AWS instance by bestowing upon it your locally-generated RSA key pair. By adding this unparalleled key to the authorized_keys file, you empower Pantheon to grant seamless login access to authorized users. Witness the seamless interplay between the local and remote Pantheon instances.

Step 6: The Dance of the Client and Server
Immerse yourself in the captivating roleplay of Pantheon's client and server components. Your local Pantheon, adorned with grace and agility, takes center stage as the client. Meanwhile, the remote Pantheon, exuding strength and dependability, gracefully performs as the server. Together, they forge an unbreakable bond, amplifying the power of Pantheon.

Step 7: Unveiling the Pantheon's Potential
Navigate to the heart of Pantheon's magic by traversing into the pantheon-director directory. Here, the crescendo awaits as you execute the awe-inspiring test.py script located in the src/experiments folder. Watch as Pantheon unfolds its magnificent capabilities, bridging the gap between your local and remote instances.

Step 8: The Symphony of Analysis
With the successful execution of Pantheon's test script, the stage is set for an enchanting symphony of analysis. Engage the src/analysis/analyze.py script, empowering it to perform its data-driven magic. Provide the designated data directory (DIR) and let Pantheon's analytical prowess shine through. Prepare to be astounded.

Step 9: The Grand Finale: Pantheon_Report.pdf
As the curtains draw to a close, bask in the glory of Pantheon's final act. Within the designated directory (DIR) mentioned, behold the majestic Pantheon_Report.pdf, a masterpiece forged from the depths of data and analysis. Capture its essence, preserve its wisdom, and relish in the pride of your Pantheon adventure.

3.2 Experiment places

Write performed our experiments at 3 different places in the UB campus. The 3 places were:

1) UB Silverman Library:

Located in the center of UB's North Campus, the Oscar A. Silverman Library in Capen Hall is an innovative 21st-century learning environment. This newly-renovated space is a vibrant hub of activity, discovery and connectivity for students, faculty and staff.

2) UB Student Union:

The main Student Union is the North Campus community center, featuring a large open lobby with a welcome center, restaurants, retail, meeting space and a theater.

3) UB Flickinger Court:

Flickinger Court is one of UB's graduate student housing options. The village contains 115 units in thirteen townhouse apartment clusters arranged on 11.5 acres of lawns and landscaping. There is also a community building.

Chapter 4: Results & Analysis

Experiment I

Comparison of schemes with different locations

The design and implementation of Pantheon, which automatically evaluates the performance of several transport protocols and congestion-controlling strategies over a wide range of network pathways, are covered in this section. We will present the results and analysis of the software implemented in different locations (Silverman Library, Student Union, Flickinger Court) in Section 3.2 and will compare them in terms of their performance.

4.1 UB Silverman Library

We performed the experiment as showed in earlier steps and deployed 11 congestion control schemes over the network. We tested the experiment from a local linux machine with cellular network of T-Mobile in the UB Silverman Library to AWS server for 3 runs of 30 seconds. Each test that we did lasted for 1 flow. We carefully examined the relative performances because comparative analyses provide insight into which protocol end hosts should run in a particular setting. Figure 1(a) shows

shows the throughput and delay of 11 congestion control algorithms between a remote AWS server in Northern Virginia, USA and a cellular network operated client located in UB Silverman Library.

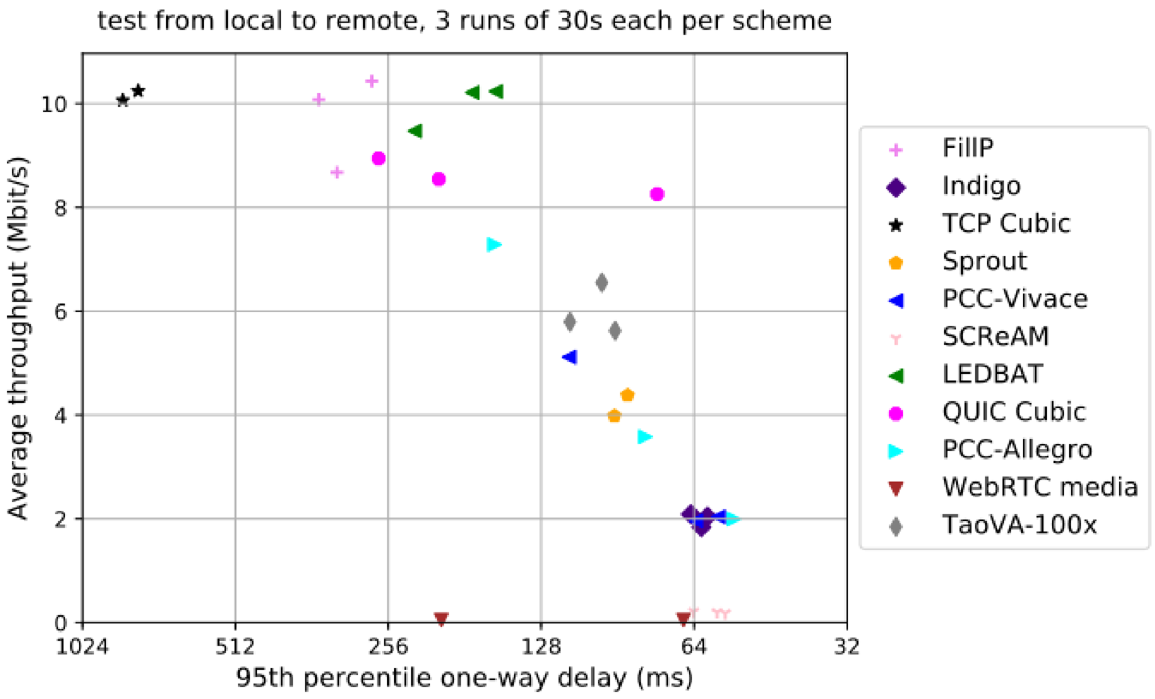


Fig 1(a) AWS to UB Silverman Library cellular network

The below figure, i.e. Fig 1(b) shows the same figure as Fig 1(a) but by taking the mean of all test runs by schemes.

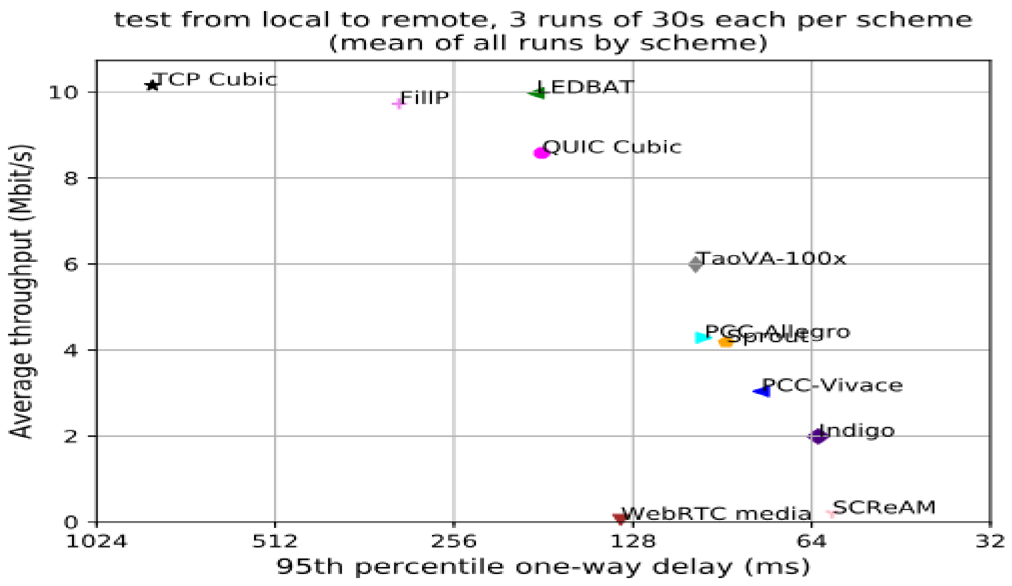
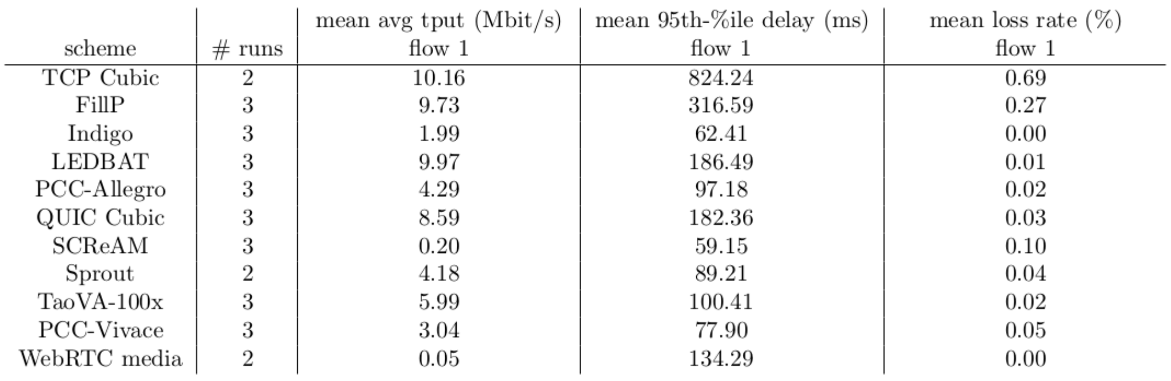


Fig 1(b) AWS to UB Silverman cellular network (Mean)

Firstly, we can see that TCP Cubic is having the best throughput but has a delay much higher than the lowest performing schemes namely SCReAM and WebRTCmedia. We can see that QUIC Cubic is one of the schemes where the scheme has a high average throughput and not a very high one-way delay resulting in a good performance. We also conclude that SCReAM and WebRTC media has the lowest throughput among all the congestion control schemes but has a very low one way delay (ms). We can also see that delays of TCP Cubic, Flip and EDBAT have a significant difference in their delays but have almost the same throughput. The other schemes dont vary a lot in terms of their delay excluding QUIC Cubic. Cubic has shown results similar to EDBAT in terms of delay and has a very small difference when it comes to throughput.

Based on these findings we cannot simply conclude which scheme is better than the other, however we will see how changing the path will affect the congestion control schemes in the experiment. We will perform the comparison later in the report.



		mean avg tput (Mbit/s)	mean 95th-%ile delay (ms)	mean loss rate (%)
scheme	# runs	flow 1	flow 1	flow 1
TCP Cubic	2	10.16	824.24	0.69
FillP	3	9.73	316.59	0.27
Indigo	3	1.99	62.41	0.00
LEDBAT	3	9.97	186.49	0.01
PCC-Allegro	3	4.29	97.18	0.02
QUIC Cubic	3	8.59	182.36	0.03
SCReAM	3	0.20	59.15	0.10
Sprout	2	4.18	89.21	0.04
TaoVA-100x	3	5.99	100.41	0.02
PCC-Vivace	3	3.04	77.90	0.05
WebRTC media	2	0.05	134.29	0.00

Table 1. Statistics of experiment

Table 1 above summarizes the statistics of the experiment which clearly shows that the scheme with:

Highest mean average throughput: TCP Cubic - 10.16 Mbit/s
Lowest mean average throughput: WebRTC media - 0.05 Mbit/s
Highest mean 95th percentile delay: TCP Cubic - 824.24 ms
Lowest mean 95th percentile delay: SCReAM - 59.15 ms
Highest mean loss rate (%): TCP Cubic - 0.69

Lowest mean loss rate (%): Indigo and WebRTC media - 0.00

4.2 UB Student Union

For this experiment we changed our location from Silverman Library to UB Student Union located in the UB North Campus. For three runs totaling 30 seconds, we tested the experiment using a local Linux computer connected to a T-mobile network in the UB Silverman Library which was remotely connected to AWS server. We conducted one flow of testing for each test. We carefully considered the relative performances since comparisons might reveal which protocol end hosts should be used in a given situation. Figure 2(a) displays the throughput and delay of 11 congestion control schemes between a client running on a cellular network in the UB Silverman Library and a distant AWS server in Northern Virginia, USA.

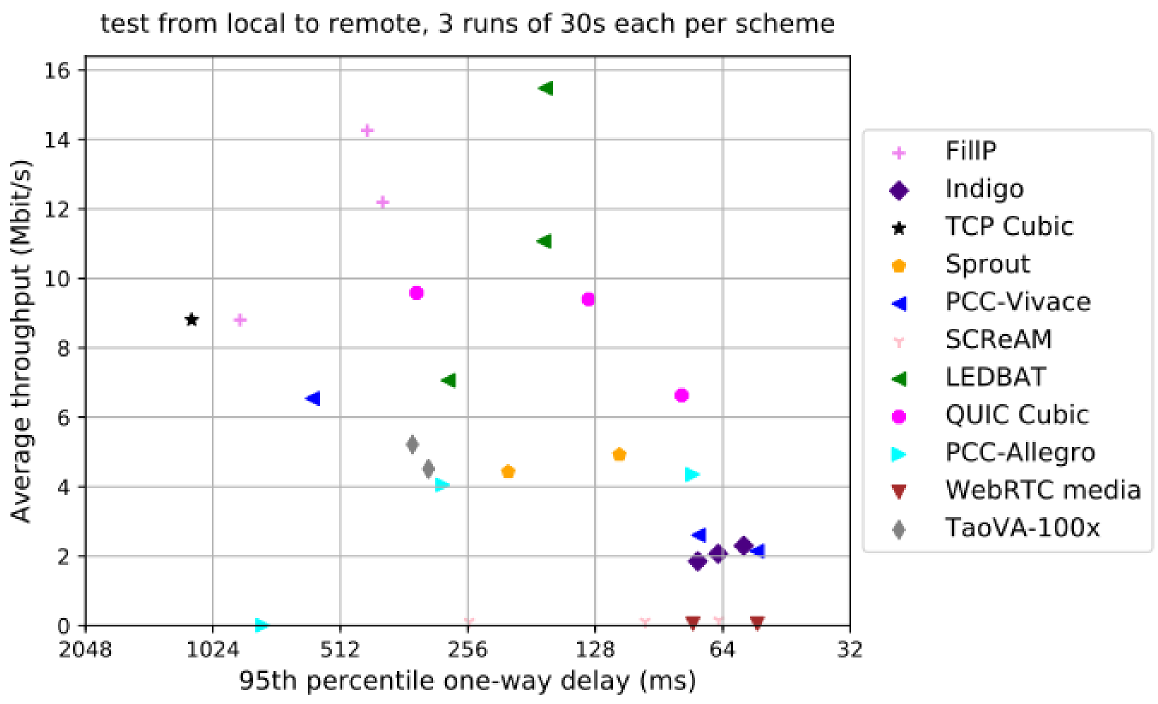
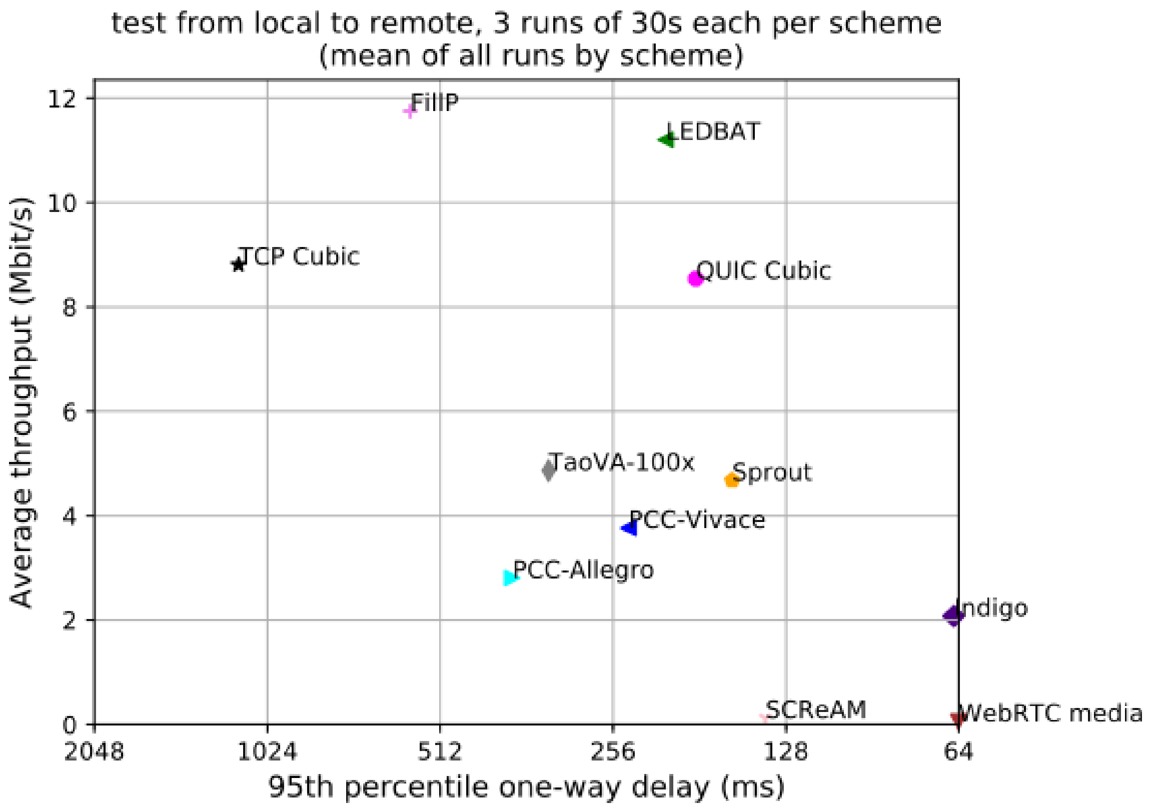


Fig 2(a) AWS to UB Student Union cellular network

Like again from the previous experiment, the below figure, i.e. Fig 2(b) shows the same figure as Fig 2(a) but by taking the mean of all test runs by schemes.



<u>Fig 2(b) AWS to UB Student Union cellular network (Mean)</u>

The results this time are quite interesting. We performed the experiment 1 hour later after the first experiment to see how the schemes and network varies with time also. Compared with TCP Cubic scheme in the last experiment, this time, FillP got the highest average throughput among all the congestion control schemes. Also, FillP doesn't have the worst one-way delay among all of the algorithms. The lowest average throughput is given again by WebRTC media scheme followed by SCReAM. LEDBAT scheme in this case if performs good by having a good throughput and a little high one way delay but almost 5 times less delay compared to TCP Cubic and almosthalf of that of Fillp.

scheme	# runs	mean avg tput (Mbit/s) flow 1	mean 95th-%ile delay (ms) flow 1	mean loss rate (%) flow 1
TCP Cubic	1	8.81	1149.05	1.11
FillP	3	11.75	577.35	1.79
Indigo	3	2.08	65.32	0.00
LEDBAT	3	11.20	207.45	0.01
PCC-Allegro	3	2.80	383.42	33.31
QUIC Cubic	3	8.54	183.68	0.00
SCReAM	3	0.12	139.02	0.02
Sprout	2	4.67	158.86	3.43
TaoVA-100x	2	4.87	331.53	0.00
PCC-Vivace	3	3.77	240.80	0.01
WebRTC media	2	0.06	64.12	0.00

<u>Table 2. Statistics of experiment</u>

Table 2 above summarizes the statistics of the experiment which clearly shows that the scheme with:

Highest mean average throughput: FillP - 11.75 Mbit/s
Lowest mean average throughput: WebRTC media - 0.06 Mbit/s
Highest mean 95th percentile delay: TCP Cubic - 1149.05 ms
Lowest mean 95th percentile delay: WebRTC media - 64.12 ms
Highest mean loss rate (%): PCC-Allegro - 33.31
Lowest mean loss rate (%): QUIC Cubic and WebRTC media - 0.00

4.3 UB Flickinger Court

The same experiment was done with 11 congestion control schemes lasting for 3 runs with each run for 30 seconds. We can see Figure 3(a) that PCC-Allegro performed the worst as having an average throughput value but a very high one way delay. We can conclude from the observations that LEDBAT performed the best among all as having the highest throughput among all the congestion control schemes and having a not bad one way delay. SCReAm performed the worst as having the lowest throughput and at the same time having the highest one way delay.

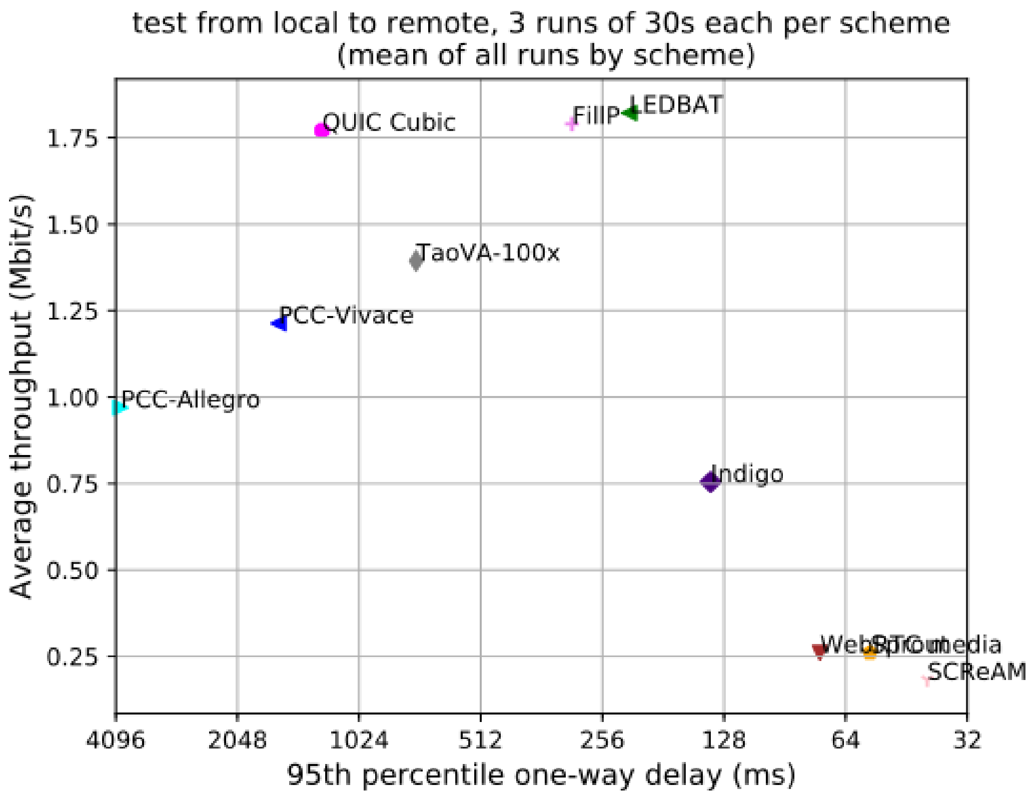
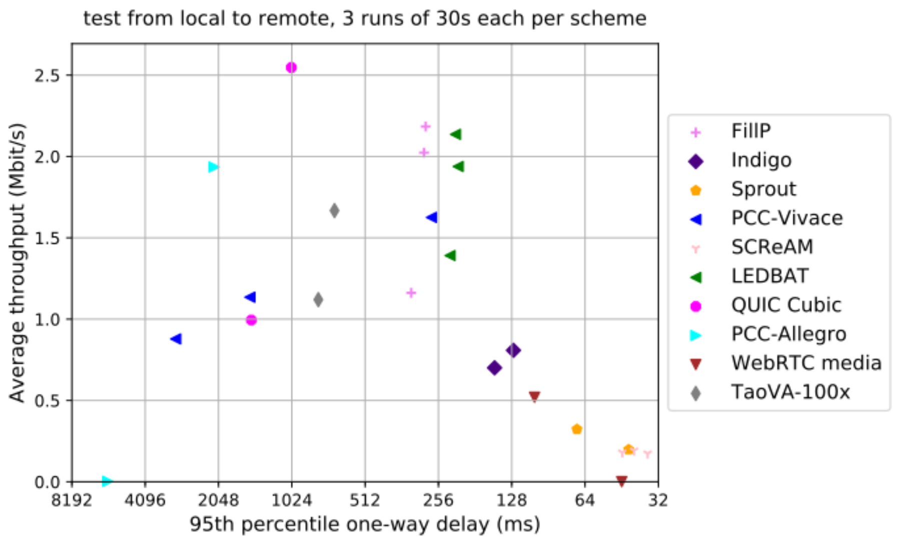


Fig 3(a) AWS to UB Flickinger court cellular network

Like again from the previous experiment, the below figure, i.e. Fig 3(b) shows the same figure as Fig 3(a) but by taking the mean of all test runs by schemes.



<u>Fig 3(b) AWS to UB Student Union cellular network (Mean)</u>

scheme	# runs	mean avg tput (Mbit/s) flow 1	mean 95th-%ile delay (ms) flow 1	mean loss rate (%) flow 1
TCP Cubic	0	N/A	N/A	N/A
FillP	3	1.79	304.38	0.58
Indigo	2	0.76	138.22	0.03
LEDBAT	3	1.82	219.87	0.00
PCC-Allegro	2	0.97	3986.59	52.48
QUIC Cubic	2	1.77	1264.97	1.33
SCReAM	3	0.18	40.23	0.05
Sprout	2	0.26	55.73	14.68
TaoVA-100x	2	1.40	740.39	1.42
PCC-Vivace	3	1.21	1621.83	17.09
WebRTC media	2	0.26	74.19	47.64

<u>Table 3. Statistics of experiment</u>

Table 3 above summarizes the statistics of the experiment which clearly shows that the scheme with:

Highest mean average throughput: LEDBAT - 1.82 Mbit/s
Lowest mean average throughput: SCReAM - 0.06 Mbit/s
Highest mean 95th percentile delay: PCC Allegro - 3986.59 ms
Lowest mean 95th percentile delay: SCReAM - 40.23 ms
Highest mean loss rate (%): PCC-Allegro - 52.48
Lowest mean loss rate (%): LEDBAT- 0.00

We can see from the observations that LEDBAT gave the highest throughput among all the different schemes while SCReAM gave the lowest throughput among all the schemes. SCReAM on the other hand, has one good advantage of having the lowest one-way delay. PCC Allegro gave the highest one-way delay among all schemes

Findings:
Based on experiment 1 we cannot conclude that a particular scheme is best for all the places and the performances of the schemes may vary. However, one can conclude that a particular scheme will always have a high/low throughput and a high/low one-time delay.

.

Experiment II

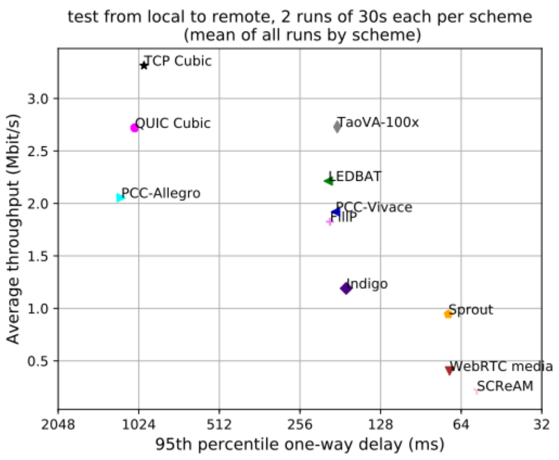
Comparison of schemes with different timings

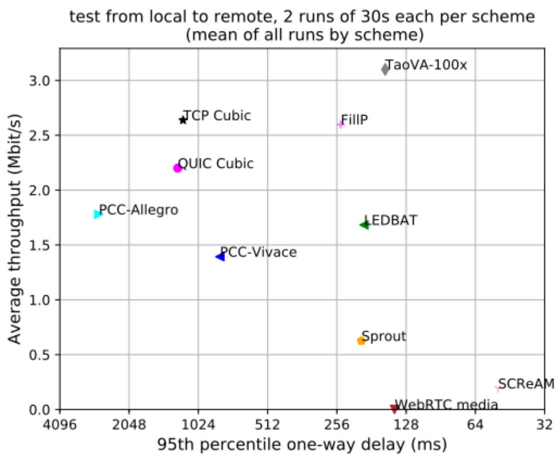
After the feedback from the professor, we did a different experiment to see how the congestion control schemes are varying with the time of the experiment performed. We wanted to see how the network and the tunnel perform in the morning, afternoon, and night time. This experiment was performed at Flickinger Court during the 3 times 9 AM in the morning, 3 PM in the afternoon, and 12 AM for the night time. We tested the experiment using a local Linux PC linked to the AT&T - mobile network in the UB Flickinger Court that was remotely connected to an AWS server for two runs totaling 30 seconds. For each test, we only used one flow of testing.

The Internet Speed during different timings were:

Timing	Speed
Morning	5.54 Mbps
Afternoon	3.31 Mbps
Night	5.62 Mbps

The same experiment was done with 11 congestion control schemes lasting for 2 runs with each run for 30 seconds. The results of the experiment are showed below in Fig 4(a). The: Fig 4(a) shows the throughput and delay of 11 congestion control schemes for morning, afternoon and evening respectively.





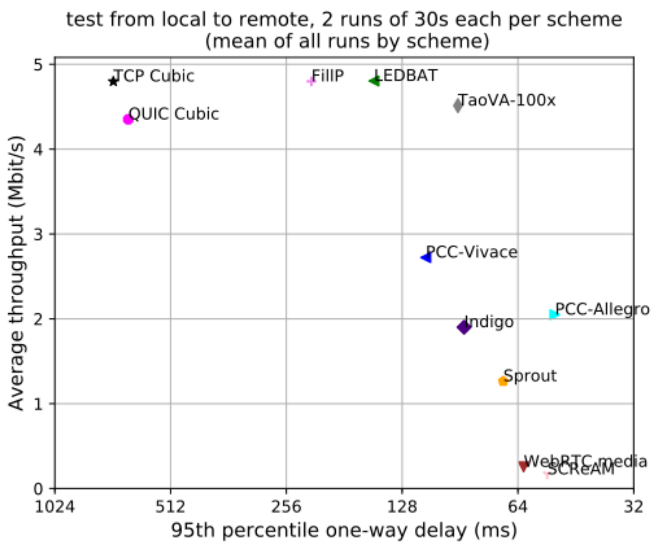
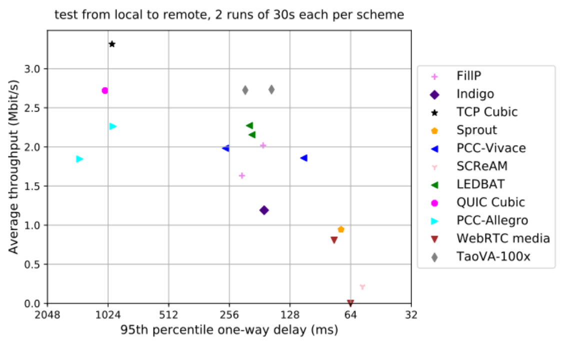
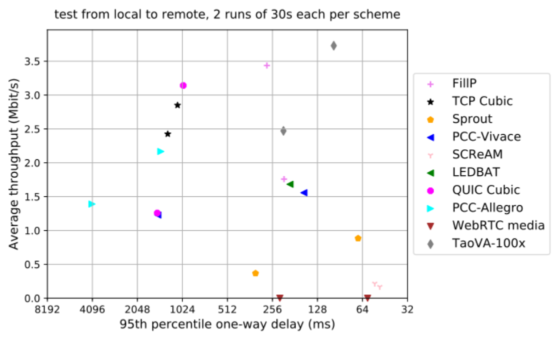


Fig 4(a). Results from different timings, the top left image is generated in morning time, the right corner in the afternoon, and the last one in the night time





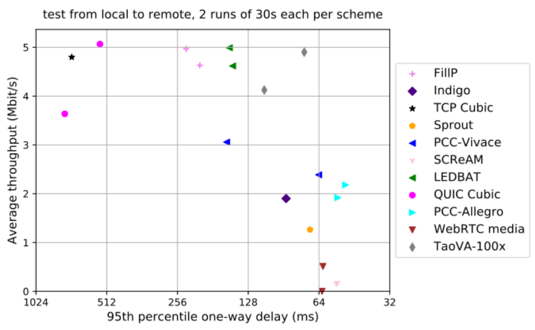


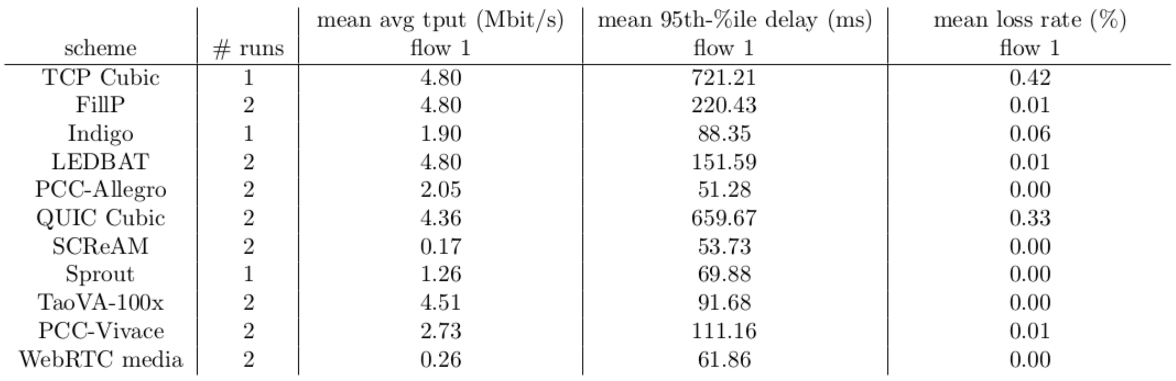
Fig 4(b). Results from different timings, the top left image is generated in morning time, the right corner in the afternoon, and the last one in the night time

scheme	# runs	mean avg tput (Mbit/s) flow 1	mean 95th-%ile delay (ms) flow 1	mean loss rate (%) flow 1
TCP Cubic	1	3.31	978.79	2.00
FillP	2	1.82	197.89	0.00
Indigo	1	1.19	172.03	0.00
LEDBAT	2	2.21	200.73	0.02
PCC-Allegro	2	2.05	1191.24	15.38
QUIC Cubic	1	2.72	1059.33	0.03
SCReAM	2	0.22	55.95	0.00
Sprout	1	0.94	71.45	0.00
TaoVA-100x	2	2.73	185.77	0.01
PCC-Vivace	2	1.92	188.56	0.00
WebRTC media	2	0.41	70.72	0.00

Table 4. Statistics of morning experiment

scheme	# runs	mean avg tput (Mbit/s) flow 1	mean 95th-%ile delay (ms) flow 1	mean loss rate (%) flow 1
TCP Cubic	2	2.64	1193.43	1.77
FillP	2	2.60	246.42	0.00
Indigo	0	N/A	N/A	N/A
LEDBAT	1	1.68	195.29	0.00
PCC-Allegro	2	1.78	2774.17	17.68
QUIC Cubic	2	2.20	1259.46	5.43
SCReAM	2	0.20	51.01	0.00
Sprout	2	0.62	200.31	-0.14
TaoVA-100x	2	3.10	157.81	0.01
PCC-Vivace	2	1.40	825.73	8.12
WebRTC media	2	0.00	143.76	0.00

Table 5. Statistics of the afternoon experiment



scheme	# runs	mean avg tput (Mbit/s) flow 1	mean 95th-%ile delay (ms) flow 1	mean loss rate (%) flow 1
TCP Cubic	1	4.80	721.21	0.42
FillP	2	4.80	220.43	0.01
Indigo	1	1.90	88.35	0.06
LEDBAT	2	4.80	151.59	0.01
PCC-Allegro	2	2.05	51.28	0.00
QUIC Cubic	2	4.36	659.67	0.33
SCReAM	2	0.17	53.73	0.00
Sprout	1	1.26	69.88	0.00
TaoVA-100x	2	4.51	91.68	0.00
PCC-Vivace	2	2.73	111.16	0.01
WebRTC media	2	0.26	61.86	0.00

Table 6. Statistics of the night experiment

4.4 Observations:

Scheme with highest throughput during:
Morning: TCP Cubic - 3.31 Mbit/s
Afternoon: TaoVA-100x - 3.10 Mbit/s
Night: TCP Cubic, LEDBAT and Fillp - 4.80 Mbit/s
Scheme with lowest throughput during:
Morning: SCReAM - 0.22 Mbit/s
Afternoon: SCReAM - 0.20 Mbit/s
Night: SCReAM - 0.17 Mbit/s

Scheme with highest one way delay during:
Morning: PCC-Allegro - 1191.24 ms
Afternoon: PCC-Allegro - 2774.17 ms
Night: TCP Cubic - 721.21 ms

Scheme with lowest one way delay during:
Morning: SCReAM - 55.95 ms
Afternoon: SCReAM - 55.95 ms
Night:PCC : 51.28 ms

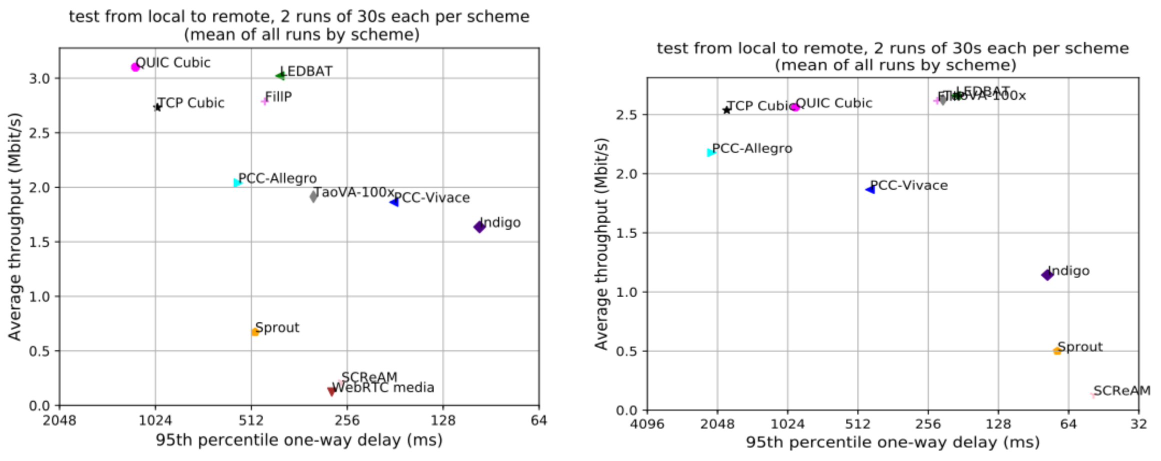
We can conclude from the above observations that no scheme is perfect for every scenario as different schemes top different categories when it comes to performance. We saw that although TCP-Cubic gave a very high throughput during morning and night times, it was unable to give the best throughput during the afternoon time. Although we see that SCReAM always gave the lowest throughput during different timings. Lableing SCReAM as the worst scheme won't be justice as it gave the lowest one way delay most of the times. We also saw that PCC-Allegro got the highest one way delay most of the times the experiment was performed. According to our observations, we can say that LEDBAT performed the best as having a good throughput and low one way delay at the same time. Different people can have different perspectives though.

<u>Experiment III</u>

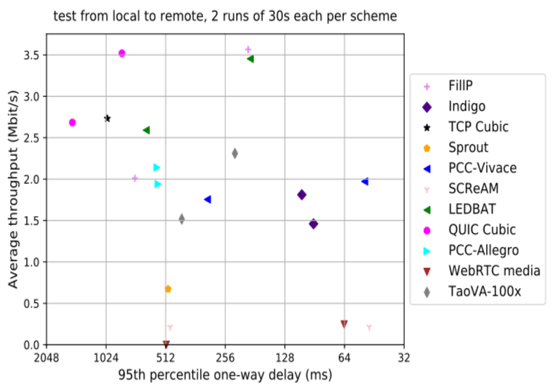
<u>Comparison of schemes with different amounts of concentrated population</u>

With another feedback from the professor, we did another different experiment in the UB Silverman Library to see how the congestion control schemes are varying with the amount of population concentrated within the library. To achieve this, we first performed the experiment during the evening in the SIlverman library as these days the library is fully packed during the day times due to the exam season. To compare it with the less concentrated population, we performed the same experiment at midnight in the Silverman library when the student population concentration was very less as most of the students leave the library in night. Using a local Linux computer at the UB Flickinger Court running on AT&T network that was remotely linked to an AWS server for two runs totaling 30 seconds, we tested the experiment. For each test, we ran a single flow of tests. The Internet Speed during different times were:

Population Concentration	Speed
High Concentration (daytime)	6.25 Mbps
Low Concentration (Night time)	8.01 Mbps



<u>Fig 5(a). Results from different timings, the top left image is generated in morning time, the right corner in the afternoon, and the last one in the night time</u>



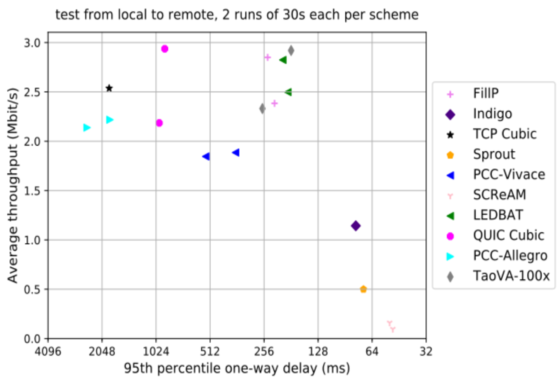


Fig 5(b). Results from different timings, the top left image is generated in morning time, the right corner in the afternoon, and the last one in the night time

scheme	# runs	mean avg tput (Mbit/s) flow 1	mean 95th-%ile delay (ms) flow 1	mean loss rate (%) flow 1
TCP Cubic	1	2.73	1009.23	0.96
FillP	2	2.79	464.34	0.00
Indigo	2	1.64	98.43	0.01
LEDBAT	2	3.02	416.85	0.28
PCC-Allegro	2	2.04	563.03	0.01
QUIC Cubic	2	3.10	1183.31	0.01
SCReAM	2	0.21	267.03	0.00
Sprout	1	0.67	498.27	0.00
TaoVA-100x	2	1.92	326.79	0.00
PCC-Vivace	2	1.86	182.88	0.00
WebRTC media	2	0.12	286.25	0.00

Table 7. Statistics of the experiment with high population concentration

scheme	# runs	mean avg tput (Mbit/s) flow 1	mean 95th-%ile delay (ms) flow 1	mean loss rate (%) flow 1
TCP Cubic	1	2.54	1869.04	0.39
FillP	2	2.62	234.18	0.00
Indigo	1	1.14	78.91	0.00
LEDBAT	2	2.66	195.06	0.01
PCC-Allegro	2	2.18	2165.53	0.14
QUIC Cubic	2	2.56	949.43	0.00
SCReAM	2	0.13	50.09	0.00
Sprout	1	0.50	71.53	0.00
TaoVA-100x	2	2.62	221.22	0.01
PCC-Vivace	2	1.87	455.29	0.01
WebRTC media	0	N/A	N/A	N/A

Table 8. Statistics of the experiment with low population concentration

4.5 Observations:

Scheme with highest throughput during:
High population concentration: QUIC Cubic - 3.10 Mbit/s
Low population concentration: LEDBAT - 2.66 Mbit/s

Scheme with lowest throughput during:
High population concentration: WebRTC media - 0.12 Mbit/s
Low population concentration: SCReAM - 0.13 Mbit/s

Scheme with highest one way delay during:
High population concentration: TCP Cubic - 1183.31 ms
Low population concentration: Indigo - 2165.53 ms

Scheme with lowest one way delay during:
High population concentration: Indigo - 98.43 ms
Low population concentration: SCReAM - 50.09 ms

We can conclude a lot of observations from this scenario. Firstly, we see that if there is a high population in an environment with high traffic, QUIC Cubic gives the highest throughput among all different congestion control schemes. Secondly, we see that if the population is less concentrated in an area then LEDBAT gives the highest throughput among all the different schemes. Similarly, WebRTC media gives the lowest throughput among all the schemes when the traffic is high and the population is concentrated and SCReAM gives the lowest throughput with low concentration of population.

Another observation is that although TCP Cubic gives a good throughput it gives the highest one way delay during high traffic with high population concentration. Indigo on the other hand gives the highest one way delay with the lowest population concentration.

There is one very strange observation while we calculate the lowest one way delay among different population concentrations. While Indigo gave the highest one way delay with low population concentration, it gave the lowest one way delay with a high population concentration. Also, SCReAM gave the lowest one way delay among all the schemes with low population concentration.

5.0 Conclusion

We concluded in these experiments that no scheme is perfect for all the scenarios and there are various factors that affect the congestion control including the location, signal strength, and population concentration. The results clearly show how the throughput and the delay changes with respect to different scenarios affecting the time of the experiment, the location of the experiment, the cellular network strength etc.

REFERENCES

[1] https://pantheon.stanford.edu/

[2] El-Sayed, Ayman & HAGGAG, Shimaa & EL-FESHAWY, Nawal. (2011). A Survey of Mechanisms for TCP Congestion Control. International Journal of Research and Reviews in Computer Science (IJRRCS). 02. 676-682.

[3] Gupta, A & Garg, Vishal & Lal, Mohan. (2010). Congestion Control Schemes in Wireless Communication Networks: A Review.

[4] https://groups.google.com/g/pantheon-stanford?pli=1