Annals of R.S.C.B., ISSN:1583-6258, Vol. 25, Issue 6, 2021, Pages. 5357 - 5369
Received 25 April 2021; Accepted 08 May 2021.

Performace Evaluation of TCP Cubic , Compound TCP and NewReno under Windows 20H1 , via 802.11n Link to LTE Core Network

Muhammad Ahsan[1], Mazhar Javed Awan[2], Awais Yasin[3], Saeed Ali Bahaj[4], Hafiz Muhammad Faisal Shehzad[5]
[1,2 *]Department of Software Engineering, University of Managenemt and Technology, Lahore 54770, Pakistan
[3]Department of Computer Engineering, National University of Technology, Islamabad 44000, Pakistan
[4]MIS Department College of Business Administration Prince Sattam bin Abdulaziz University, Alkharj Saudi Arabia
[5]Department of Computer Science, University of Sargodha, Sargodha 40100, Pakistan
* muhammadahsan@umt.edu.pk, mazhar.awan@umt.edu.pk

ABSTRACT

The transmission control protocol (TCP) is one of the major protocols in TCP/IP suite that ensure process to process error and flow control. It provides reliability via use of acknowledgements and sequence numbers. It makes sure that the packets are delivered in order to the processes. It ensures that it deals with the congestion a packet faces as it traverses the internet nodes to its destination using a variety of congestion control algorithms. This paper focuses on a comparative study on some of these congestion control algorithms namely, New Reno, Compound TCP (CTCP) and TCP Cubic. We analyzed their performance on Microsoft's latest operating system build, named Windows 20H1 and have analyzed the throughput and latency occurred while using these algorithms under 20H1 build of Windows 10 operating system. We were connected via WIFI to Long-Term Evolution (LTE) IP packet core-based network to access a remote file server based in Europe for downloading big size files and measuring the throughput and latency involved with these three congestion control algorithms. Respective algorithm was configured one by one via Power shell of Windows 20H1 in administrative mode. The access network mode deployed was dual network band 2.4 Ghz IEEE 801.11n. The analysis only starts via each network band's co-channel interference has been minimized, RSSI and signal strength is optimum and wireless security has been set to appropriate levels. The results show that TCP Cubic gives an appreciable increase in throughput while downloading big size files (4-5 GB) through LTE network when compared to Compound TCP and NewReno.

Keywords
Congestion Control, Compound TCP, New Reno, RTT Fairness, TCP Cubic, Windows 20H1, LTE

INTRODUCTION

Transmission Control Protocol is one of the backbone protocols of today's internet, it has its special place in the TCP/IP suite. Along with IP as transporter or carrier of its forwarded payload a.k.a segments, it plays a crucial role in providing process to process reliable communication(Amezcua Valdovinos, Perez Diaz, Garcia Villalba, & Kim, 2017).The reliability is achieved using acknowledgements and sequence numbers in the segments. Thus, it provides reliable, in-order delivery of stream of bytes. Since it is an FDX protocol each TCP socket supports a pair of byte streams. Flow control is provided by it so that the sender does not overwhelm the receiver with streams of bytes. It provides multiplexing and de-multiplexing services to the end processes running on respective end hosts systems. So that an in-order byte stream is delivered to multiple applications that are currently using Internet browsers, video conferencing applications, chatting programs, network tools requiring internet access all can simultaneously access the internet using TCP, which intelligently multiplexes and de-multiplexes each byte stream to respective process of the application for which the byte stream is destined for. HTTP, FTP, HTTPS, SNMP, Telnet and may more protocols require the reliable services of TCP to ensure data integrity and error and flow control on process-to-process basis (Xylomenos, Polyzos, Mahonen, & Saaranen, 2001).The information extraction and recommendation system also improved the various processes in the network field as well (Alam & Awan, 2018; Rehma, Awan, & Butt, 2018).

TCP is not limited in providing the above discussed services only, but it also plays a critical role in providing congestion-control mechanism. Congestion occurs when network links are overburdened by IP packets and the routers buffers starts getting piled up and eventually dropping the access packets that they can't handle due to lack of buffer storage at their input and output buffers. So, congestion control and congestion avoidance techniques are key for a smooth working of network. Some of them are exponential back off in protocols such as CSMA/CA in 802.11 and CSMA/CD in the classic ethernet, congestion and receive window tuning in TCP, and fair queuing in devices such as routers and network switches (Kwon, Seo, Kim, & Lee, 2009). There are other techniques as well that avoids congestion by prioritizing the packets and use of admission control to for allocation of resources to specific flows. It is pertinent to mention that TCP maintains two important state variables for its connections, namely congestion window (cwnd) and receive window (rwnd). The former is maintained by the sender and its key role is to prevent the link between the sender and receiver to be overwhelmed by too many IP packets. The other is the sliding window that is maintained by the receiver to prevent from being overloaded. Every TCP algorithm till now have tried its best to get the most optimizing cwnd values, so that the link is not congested and the high bandwidth available in todays networks be utilized optimizing. Cwnd is usually setup to some TCP initial maximum segment size (MSS) and later it is enhanced using different formulae (Chen, Gerla, Lee, & Sanadidi, 2008).

The flow of data octets over TCP is controlled by is controlled by the advertised receive window. The sender compares its own congestion window with the receive window to know how much data can be sent at the time keeping the links unchoked (Postel, 1983).

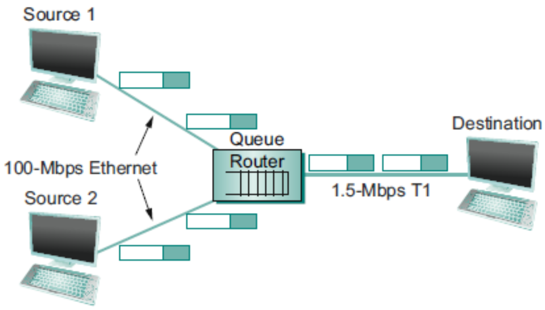


Figure 1: Congestion Ethernet Frames

The above figure gives a simple network scenario in which two sources each of 100 Mbps are sending Ethernet frames to the router. The router is connected to the destination via T1 leased connection of 1.5 Mbps. The diagram very clearly shows that when both sources (1&2) are sending Ethernet frames at their full link utilization the router buffer will soon overflow and Ethernet packets will be dropped due to congestion at the node (An et al., 2003).

TCP provides a vast variety of congestion control mechanisms to prevent congestion and also to reduce it if it has already occurred at the first place.

In this literature we will discuss three of such congestion control algorithms of TCP, namely TCP NewReno, Compound TCP and TCP Cubic.

In section II background study has been performed on the three techniques TCP NewReno, CTCP and TCP Cubic showing how they work, and some formulae involved in calculation the congestion window when the network is going to be in congestion phase.

In section III platform is setup before starting the comparative analysis by providing an optimum 2.4 GHz platform free from co-channel interference and improved security to Wireless Access Networks to access the LTE IP core network of M/S Zong.
Furthermore, a file server in Europe is selected to perform the throughput analysis, its latency results are shared along with its whose information.
In section IV the latest operating systems of Microsoft, named Windows 10 with its latest build, known as 20H1 is used as the modern operating system to conduct this performance experiment (Miao et al., 2009). All the required patches and necessary modules have been loaded for the platform to work error free and optimally. Powershell of 20H1 is invoked to access the current running TCP profiles in the operating system. Optimizations are proposed in this section on the current TCP stack. RWIN of TCP is tweaked accordingly for better throughput. In Section V TCP NewReno has been imported in the internet profile as the default congestion control algorithm in 20H1, after its imports and necessary state variables changes, throughput analysis has been performed and latency involved has also been recorded in three different times of the day i.e Morning, Evening and at Night to get a fair picture of its performance under varying load conditions of internet. Although the file server is taken as a constant IP source for providing the downlink data.

BACKGROUND STUDY

Congestion can occur anytime in a network or on internet. The indicators that can help detect congestion are the increased packet losses, increased RTTs, Duplicate Acks, and timer time outs.
Since 1980s there are many TCP algorithms that were implemented , starting with TCP Slow Start and congestion avoidance by Stevens in 1997 , TCP Tahoe , TCP Reno and TCP Vegas were the earliest , they were mostly developed in the times when we were using dialup connections 14.4kbps to roughly 56kbps PPP/SLIRP based connectivity to access the internet with MTU=576 bytes and MSS=536 bytes (Allman, Paxson, & Stevens, 1999).
With broadband connections out in the market and then especially the wireless Long Fat Networks, which exhibit longer delays and have high bandwidth, these methods were very conservative and were not able to unleash the full bandwidth of the LFNs, as a result faster algorithms came into existence that can provide much large initial ‘Cwnd’ and similarly a higher receiver side windows ‘RWIN’ to provide much larger throughput than their old traditional predecessors (Corral, Zaballos, Fulgueira, & Abella, 2012). The TCP NewReno, Compound TCP and especially the groundbreaking Cubic TCP algorithm which was provided a dramatic increase in throughputs when it was first introduced on Linux based systems in 2005(Munir, Welzl, & Damjanovic, 2007).

A. NewReno

In TCP NewReno, which is the modified form of TCP Reno, is good at detecting multiple packet losses as compared to Reno. TCP NewReno also enters fast-retransmit when multiple packet losses are encountered but the main difference from Reno is in the fact that it does not leave the fast-recovery phase until and unless all the outstanding packets are acknowledged since it entered fast-recovery. As a result, the issue that we faced in Reno of reducing the CWND multiple times is resolved (Floyd & Henderson, 1999).

B. CTCP

Compound TCP is a TCP congestion control mechanism deveopled by Microsoft , it tried to improve the congestion control mechanisms by maintainning two congestion windows at the same time in order to provide better performance over LFN. CTCP improves the RTT fairness when compared with older TCP congestion control algorthims , thanks to the deployment of delay based component in CTCP.

It was first introduced in Windows Vista by Microsoft and it did performed well but only till the arrival of much more sophisticated and intelligent algorthims like Bic and TCP Cubic (Oura & Yamaguchi, 2012).

C. *Cubic TCP*

a. Cubic is an innovation in TCP congestion control algorithms for Long Fat Networks to achieve high bandwidth connections over internet much more quickly then the previous TCP CC algorithms (Wang et al., 2013).

Cubic was first released in 2006 in Linux kernel 2.6.19 for its community. MacOS adopted in in 2014 in its OS X code named Yosemite. Microsoft Windows added in its TCP/IP stack starting 2017 and in its Server version from 2018.

In cubic window size is a cubic function of time starting from the time the last congestion event occurred, and the inflection point is set as to the window size before the last congestion event occurred.

Being a cubic function, it has two parts, namely the concave and convex portion shown in Figure.2 below (Ha, Rhee, & Xu, 2008)

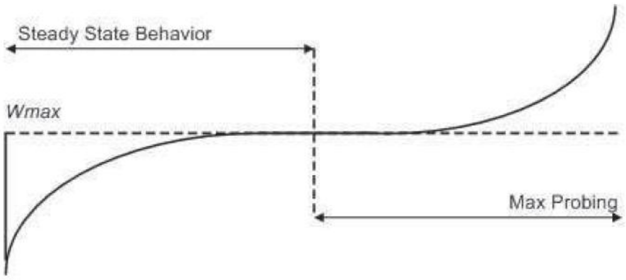


Figure 2: Cubic Window Growth Function

In the concave portion the window quickly increases to the same size since the last congestion event occurred, next is the convex phase where the window is looking for more bandwidth, initially somewhat conservatively, but then very rapidly. Cubic algorithm waits for some time in the middle of the above graph i.e around the point of inflection waiting for network stabilization before it goes out for its hunt for more bandwidth.

b. *RTT Fairness in TCP Cubic*

One important feature of TCP Cubic is the fairness between the flows , in TCP NewReno , flows with shorter RTTs will receive ACKs much faster than flows with increased RTTs and have their congestion windows grow more faster than the flows with longer RTTs. This shows dependence of TCP NewReno on RTT, whereas in TCP Cubic this is not the case and it is RTT independent and Cubic window size is only dependent on last congestion event. As a result, Cubic allows more RTT fairness among the flows as it is not dependent on RTTs.

TCP Cubic algorithm works as follows:

1. When the congestion event occurs , the window size will be recorded as Wmax or the maximum window size.
2. This Wmax will be used as the inflection point of the cubic function and it will set the pace for the growth of the congestion window.
3. The data transfer will then again start with a smaller window and if no congestion occurs , the congestion window will rise according to the concave portion of the curve (Ahmad, Jabbar, Ahmad, Piccialli, & Jeon, 2018).

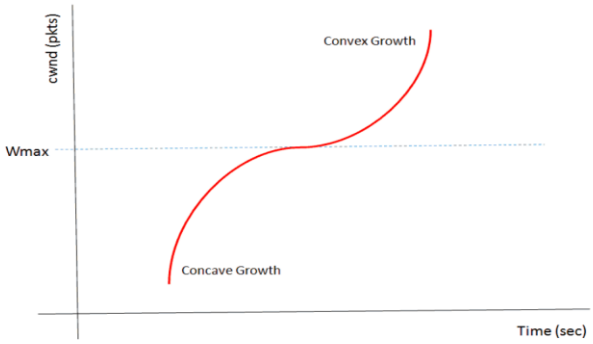


Figure 4: The WMax and time concave curve

4. It will slow down as it reaches W_{max}
5. Once the inflection point is reached, the congestion window will continue to increase discreetly.
6. Now if the network is not experiencing congestion the window size will continue to increase according to the convex portion of the function as show in the Figure 8.
7.

Methodology

The methodology of our network system is as follow in the figure 5.

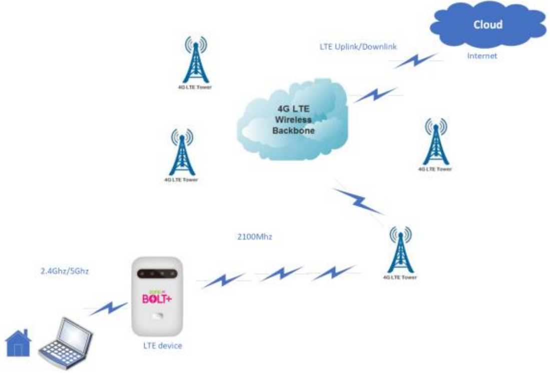


Figure 5: Network Diagram

Network Scenario: Wireless Access Network to LTE core network.

Adapter: Qualcomm 802.11n
SSID: Mr. President
Network Band : 2.4 Ghz
File server IP : 163.172.251.201
# of Hops away = 17 hops
Avg. RTT for file server 200 msec

Since the access network is wireless in a 2.4 Ghz band in this scenario, which is used to access the internet via LTE based packet core network of Zong Pvt. Limited. It is worth sharing the channels involved and the Fig.6 showing their co-channel interference.

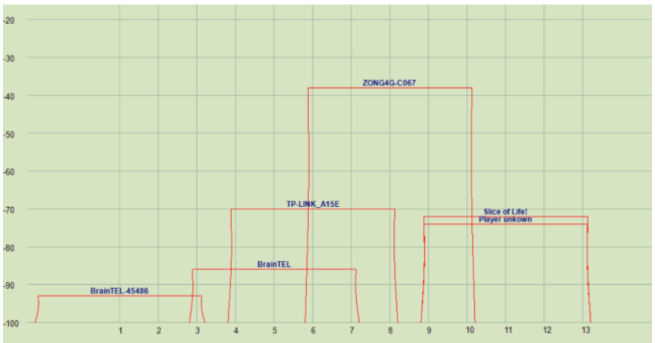
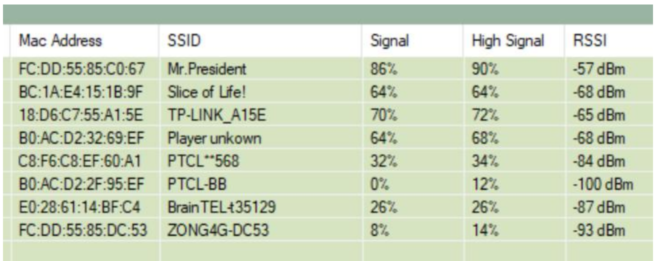


Figure 6: 2.4Ghz RF environment

The below Fig.7 shows the Mac Addresses, SSIDs, Signal strengths and RSSI of the 2.4 Ghz devices currently in active state around our SSID i.e Mr. President.



Mac Address	SSID	Signal	High Signal	RSSI
FC:DD:55:85:C0:67	Mr.President	86%	90%	-57 dBm
BC:1A:E4:15:1B:9F	Slice of Life!	64%	64%	-68 dBm
18:D6:C7:55:A1:5E	TP-LINK_A15E	70%	72%	-65 dBm
B0:AC:D2:32:69:EF	Player unkown	64%	68%	-68 dBm
C8:F6:C8:EF:60:A1	PTCL**568	32%	34%	-84 dBm
B0:AC:D2:2F:95:EF	PTCL-BB	0%	12%	-100 dBm
E0:28:61:14:BF:C4	BrainTEL-t35129	26%	26%	-87 dBm
FC:DD:55:85:DC:53	ZONG4G-DC53	8%	14%	-93 dBm

Figure 7: RSSI levels

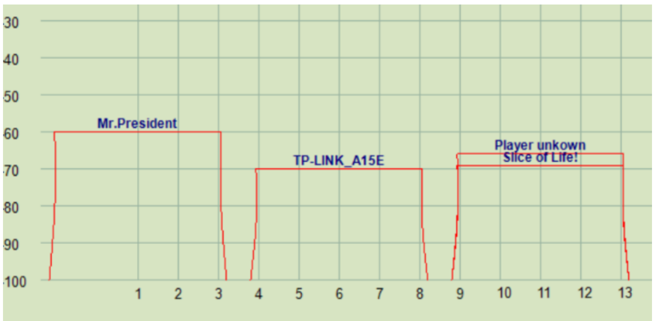


Figure 8: Channel 1 at 2.4 Ghz

Before start of this experiment the SSID was the default SSID of the device Zong4G-C067 and channel selection was auto in the configuration of the LTE device ROM.

Using the RF tool we carefully monitored the 2.4Ghz band and found that the channel being assigned again and again by the LTE device on every bootup was 8 , as you can see from the Fig.9 there were four other devices who are interfering in the channel 8 , this contention of channels in 802.11n at 2.4 Ghz resulted in delay as more and more RTS/CTS messages had to be exchanged between the device to get hold of the free channel slot and send data through uplink. So as an initial step we configured the LTE device via its browser admin control mode and manually changed the SSID to Mr. President and the channel from auto selection to 1 , as we know now that the other channel selections will interefere with ours.

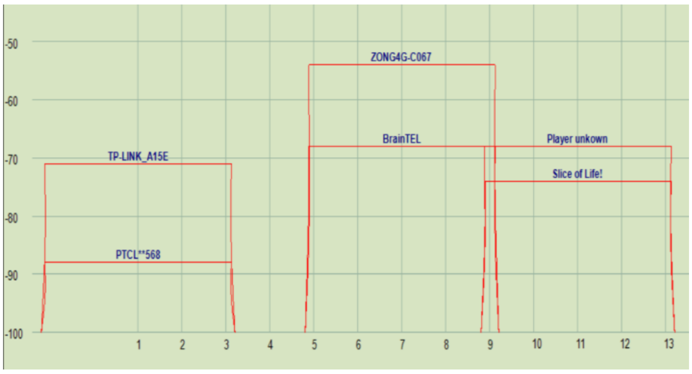


Figure 9: Channel interference graph

The above Fig.9 is showing five other devices at 2.4 Ghz with four of them interfering the bandwidth allocated to us at auto channel 8 given by the device automatically.

Now from Fig.10 you can clearly see that after making the suggested changes and carefully selecting channel 1 in this scenario , the co-channel interference is almost not there.

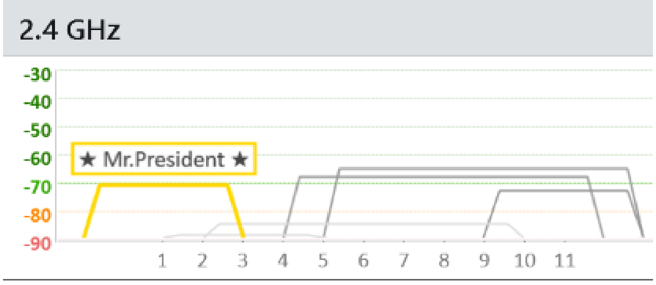
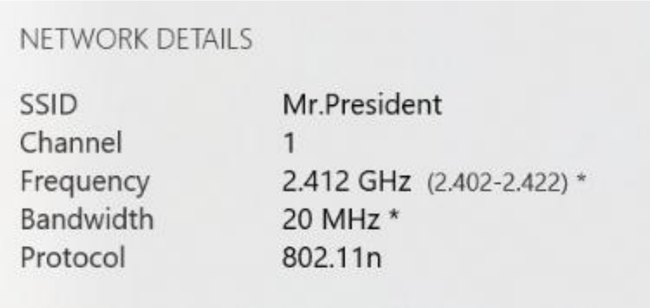


Figure 10: Channel 1 at 2.4 Ghz

There are still two SSIDs at the bottom in grey lines but their RSSI is too low to interfere with our device strong signal. The above Fig.8 taken via inSSIDer tool confirms that the co-channel interference has been almost removed.

Now we have an almost interference free 2.4Ghz network band access at channel 1, which is of 20 Mhz , centered at 2.412 MHz precisely , using IEEE 802.11n protocol

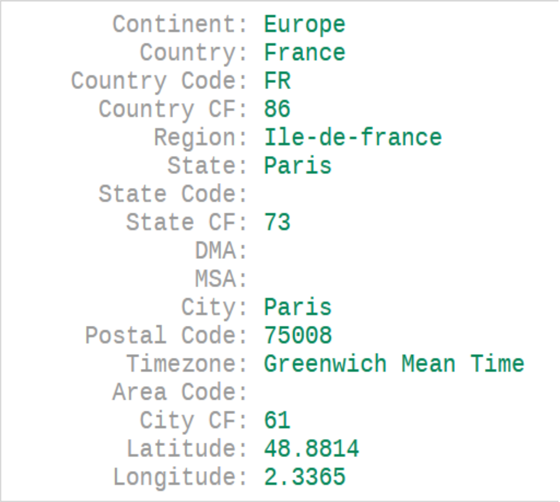


NETWORK DETAILS

SSID	Mr.President
Channel	1
Frequency	2.412 GHz (2.402-2.422) *
Bandwidth	20 MHz *
Protocol	802.11n

Figure 11: Network Details

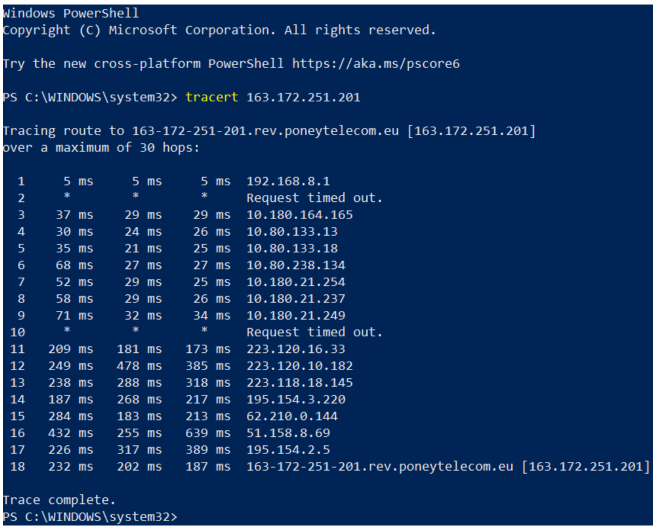
In the next step, we picked a file server located at France at IP address 163.172.251.201



```
   Continent: Europe
     Country: France
Country Code: FR
  Country CF: 86
      Region: Ile-de-france
       State: Paris
  State Code:
    State CF: 73
         DMA:
         MSA:
        City: Paris
 Postal Code: 75008
    Timezone: Greenwich Mean Time
   Area Code:
     City CF: 61
    Latitude: 48.8814
   Longitude: 2.3365
```

Figure 12: Whois Info.

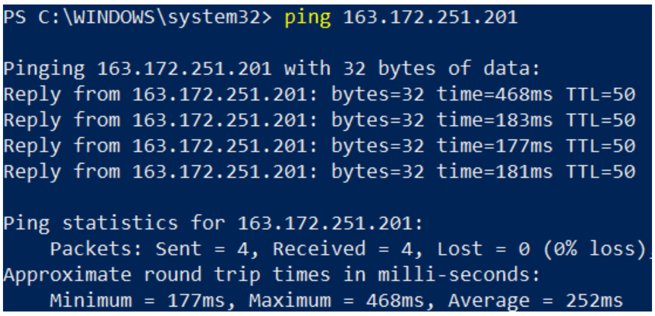
This particular file server was chosen as it has large repository of ISO images to be downloaded from. The bigger files downloads make the average analysis more accurate.
The traceroute results to the file server from our location
Is shown in the figure below:



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\WINDOWS\system32> tracert 163.172.251.201

Tracing route to 163-172-251-201.rev.poneytelecom.eu [163.172.251.201]
over a maximum of 30 hops:

  1     5 ms     5 ms     5 ms  192.168.8.1
  2     *        *        *     Request timed out.
  3    37 ms    29 ms    29 ms  10.180.164.165
  4    30 ms    24 ms    26 ms  10.80.133.13
  5    35 ms    21 ms    25 ms  10.80.133.18
  6    68 ms    27 ms    27 ms  10.80.238.134
  7    52 ms    29 ms    25 ms  10.180.21.254
  8    58 ms    29 ms    26 ms  10.180.21.237
  9    71 ms    32 ms    34 ms  10.180.21.249
 10     *        *        *     Request timed out.
 11   209 ms   181 ms   173 ms  223.120.16.33
 12   249 ms   478 ms   385 ms  223.120.10.182
 13   238 ms   288 ms   318 ms  223.118.18.145
 14   187 ms   268 ms   217 ms  195.154.3.220
 15   284 ms   183 ms   213 ms  62.210.0.144
 16   432 ms   255 ms   639 ms  51.158.8.69
 17   226 ms   317 ms   389 ms  195.154.2.5
 18   232 ms   202 ms   187 ms  163-172-251-201.rev.poneytelecom.eu [163.172.251.201]

Trace complete.
PS C:\WINDOWS\system32>
```

Figure 13: Traceroute

The traceroute result shows that the experimental file server is located 17 hops from our test network. Fig. 5.
Multiple traces were taken at different times of the day, but the average # of hops come to be around 17.
Next important stet was to calculate the average RTT to this file server. The averages were taken at different times of the day, morning, evening and at night and the below average was finally agreed upon. Which turns out to be 252msec.

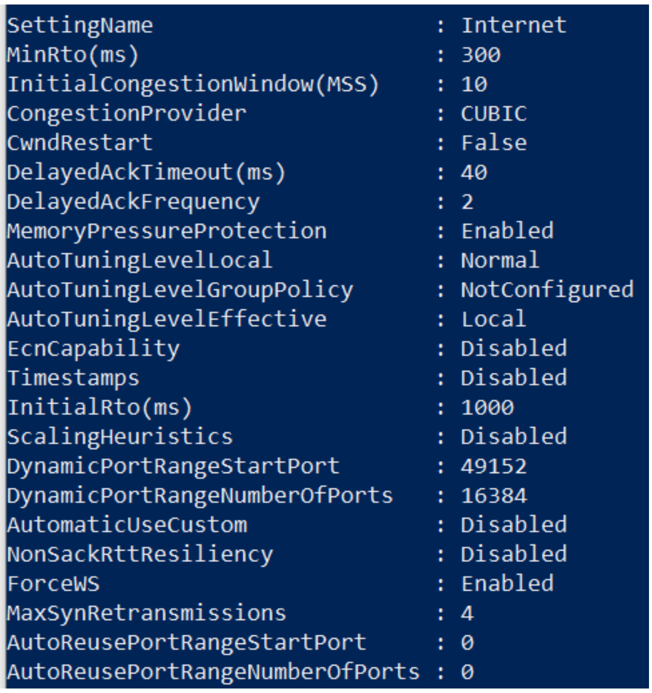


```
PS C:\WINDOWS\system32> ping 163.172.251.201

Pinging 163.172.251.201 with 32 bytes of data:
Reply from 163.172.251.201: bytes=32 time=468ms TTL=50
Reply from 163.172.251.201: bytes=32 time=183ms TTL=50
Reply from 163.172.251.201: bytes=32 time=177ms TTL=50
Reply from 163.172.251.201: bytes=32 time=181ms TTL=50

Ping statistics for 163.172.251.201:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 177ms, Maximum = 468ms, Average = 252ms
```

Figure 14: Ping statistics

RESULTS

The windows 20h1 build of latest OS by Microsoft contains some specific profiles for TCP, below is the one active on our system.



```
SettingName                     : Internet
MinRto(ms)                      : 300
InitialCongestionWindow(MSS)    : 10
CongestionProvider              : CUBIC
CwndRestart                     : False
DelayedAckTimeout(ms)           : 40
DelayedAckFrequency             : 2
MemoryPressureProtection        : Enabled
AutoTuningLevelLocal            : Normal
AutoTuningLevelGroupPolicy      : NotConfigured
AutoTuningLevelEffective        : Local
EcnCapability                   : Disabled
Timestamps                      : Disabled
InitialRto(ms)                  : 1000
ScalingHeuristics               : Disabled
DynamicPortRangeStartPort       : 49152
DynamicPortRangeNumberOfPorts   : 16384
AutomaticUseCustom              : Disabled
NonSackRttResiliency            : Disabled
ForceWS                         : Enabled
MaxSynRetransmissions           : 4
AutoReusePortRangeStartPort     : 0
AutoReusePortRangeNumberOfPorts : 0
```

Figure 15: Profile snapshot

Invoking the Power shell in admin mode in windows 20H1
and using registry and scripting technique the TCP/IP stack was modified to use NewReno and CTCP at first and results were recorded. Later, after a TCP/IP stack full reset the drill was performed with cubic as the main congestion provider and throughput monitored again.
There was an indeed improved performance while using Cubic on our system. The results with 2.4 Ghz wireless access network to access the LTE IP core were recorded and they showed improvement when using Cubic as the main congestion control provider in Windows 20H1.
The cubic algorithm outperforms the delay based NewReno and CTCP algorithm with its more intelligent and proactive management of the bandwidth and handling of the Cwnd state variable.

TABLE I. THROUGHPUT COMPARISON

163.172.251.201	**NewReno**	**CTCP**	**Cubic**
802.11n	Mbps	Mbps	Mbps
2.4 Ghz	6	6.9	7.7
Channel 1	5.5	6.8	7.6
11 (Mbps)	5.7	6.7	7.6
File size=5.1GB	5.3	7	7.2
Hops Away=16	5.9	6.9	7.1
Avg RTT 252 ms	5.2	6.2	7.3

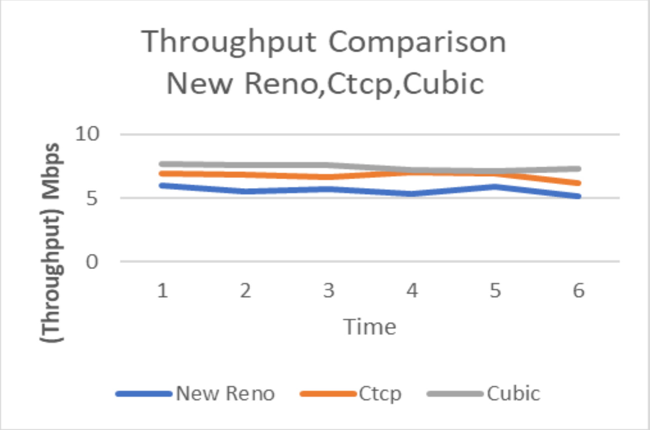


Figure 16: Throughput Comparison Graph

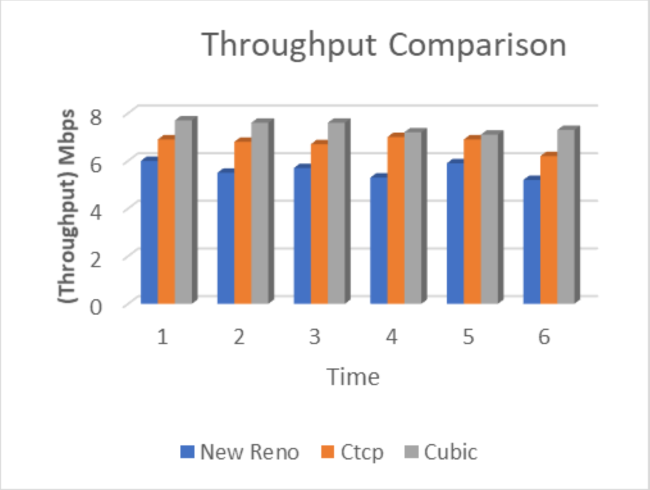


Figure 17: Throughput Comparison Chart

FUTURE WORK

Our future work will focus on using TCP Cubic for the upcoming build from Microsoft 20H2 for normal browsing and downloading activities and using a new TCP congestion control

variant to work in background without clogging the network, it is known as the Low Extra Delay Background Transport (LEDBaT)
Although TCP Cubic currently works well and is being used in the most modern operating systems i.e Linux, MacOS and Microsoft Windows 10 build 20H1, there is still room to add more bling to it as already LEDBaT++ has jumped in. LEDBaT is being used by BitTorrent (in their P2P content transfers), it is a delay-based congestion control algorithm.
[RFC 8617] describes it (Ong, 2020).
Microsoft feature updates and as well as cumulative updates can be downloaded with more ease using the broadband links with LEDBaT++ while the user browsing experience is not suffered. Microsoft is already using this combination with its new Edge browser. Apple, another giant is using it in their software update infrastructure. The p2p client BitTorrent uses it as its main congestion control protocol.
There is still a lot of work that can be done with LEDBaT (Abbasloo, Yen, & Chao, 2020). We surely plan to big data and IOT. Recent work inspired through big data Spark based models (Aftab, Awan, Khalid, Javed, & Shabir, 2021; Ahmed, Javed Awan, Khan, Yasin, & Faisal Shehzad, 2021; Mazhar Javed Awan et al., 2021; Javed Awan, Shafry Mohd Rahim, Nobanee, Munawar, et al., 2021; Javed Awan, Shafry Mohd Rahim, Nobanee, Yasin, et al., 2021), machine learning based bimodal approaches (Abdullah, Awais, Awan, Shehzad, & Ashraf, 2020; Ali et al., 2019; Anam et al., 2021; Gupta et al., 2021; Nagi, Awan, Javed, & Ayesha, 2021) and latest deep learning approaches (Awan, 2019; M. J. Awan et al., 2021; Javed, Saba, Humdullah, Jamail, & Awan, 2021; Mujahid et al., 2021) will also be implemented.

CONCLUSION

We observe from Table 1. that cubic performs better than the other two TCP CC algorithms, NewReno and Compound TCP. Since the results were gathered at different times of the day and the source file server was kept same for all the three TCP CC variants, so the average RTTs can be assumed to be constant for all the three variants. We see from the throughput comparison graph in Fig.16 and fig. 17 that cubic outperforms and wins the throughput race.The % increase in avg. throughput when compared TCP Cubic with CTCP = 9.7%. The % increase in avg. throughput when compared TCP Cubic with NewReno= 32.3 %. Moreover, Cubic non-dependency on RTT results in much fairer allocation of bandwidth to the flows than NewReno and CTCP. Having a strong networking stack is very crucial to handle modern day web applications and connection requests coming from a disparate network. Having a strong networking stack is very crucial to handle modern day web applications and connection requests coming from a disparate network.

References

1. Abbasloo, S., Yen, C.-Y., & Chao, H. J. (2020). *Classic meets modern: A pragmatic learning-based congestion control for the Internet.* Paper presented at the Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication.
2. Abdullah, Awais, Y., Awan, M. J., Shehzad, M. F., & Ashraf, M. (2020). Fake News Classification Bimodal using Convolutional Neural Network and Long Short-Term Memory. *International Journal of Emerging Technologies in Learning 11*(2), 209-212.

3. Aftab, M. O., Awan, M. J., Khalid, S., Javed, R., & Shabir, H. (2021). *Executing Spark BigDL for Leukemia Detection from Microscopic Images using Transfer Learning.* Paper presented at the 2021 1st International Conference on Artificial Intelligence and Data Analytics (CAIDA).
4. Ahmad, M., Jabbar, S., Ahmad, A., Piccialli, F., & Jeon, G. (2018). A sustainable solution to support data security in high bandwidth healthcare remote locations by using TCP CUBIC mechanism. *IEEE Transactions on Sustainable Computing, 5*(2), 249-259.
5. Ahmed, H. M., Javed Awan, M., Khan, N. S., Yasin, A., & Faisal Shehzad, H. M. (2021). Sentiment Analysis of Online Food Reviews using Big Data Analytics. *Elementary Education Online, 20*(2), 827-836.
6. Alam, T. M., & Awan, M. J. (2018). Domain analysis of information extraction techniques. *International Journal of Multidisciplinary Sciences and Engineering, 9*, 1-9.
7. Ali, Y., Farooq, A., Alam, T. M., Farooq, M. S., Awan, M. J., & Baig, T. I. (2019). Detection of Schistosomiasis Factors Using Association Rule Mining. *IEEE Access, 7,* 186108-186114. doi:10.1109/access.2019.2956020
8. Allman, M., Paxson, V., & Stevens, W. (1999). TCP congestion control.
9. Amezcua Valdovinos, I., Perez Diaz, J. A., Garcia Villalba, L. J., & Kim, T.-h. (2017). BATCP: Bandwidth-aggregation transmission control protocol. *Symmetry, 9*(8), 167.
10. An, F., Bae, H., Hsueh, Y., Rogge, M., Kazovsky, L., & Kim, K. (2003). *A new media access control protocol guaranteeing fairness among users in Ethernet-based passive optical networks.* Paper presented at the Optical Fiber Communication Conference.
11. Anam, M., Ponnusamy, V. a. p., Hussain, M., Nadeem, M. W., Javed Awan, M., Goh, H. G., & Qadeer, S. (2021). Osteoporosis prediction for trabecular bone using machine learning: a review. *Computers, Materials & Continua (CMC), 67*(1), 89-105.
12. Awan, M. J. (2019). Acceleration of Knee MRI Cancellous bone Classification on Google Colaboratory using Convolutional Neural Network. *International Journal of Advanced Trends in Computer Science and Engineering, 8*(1.6), 83-88. doi:10.30534/ijatcse/2019/1381.62019
13. Awan, M. J., Khan, R. A., Nobanee, H., Yasin, A., Anwar, S. M., Naseem, U., & Singh, V. P. (2021). A Recommendation Engine for Predicting Movie Ratings Using a Big Data Approach. *Electronics, 10*(10), 1215. doi:10.3390/electronics10101215
14. Awan, M. J., Rahim, M. S. M., Salim, N., Mohammed, M. A., Garcia-Zapirain, B., & Abdulkareem, K. H. (2021). Efficient Detection of Knee Anterior Cruciate Ligament from Magnetic Resonance Imaging Using Deep Learning Approach. *Diagnostics (Basel), 11*(1). doi:10.3390/diagnostics11010105
15. Chen, J., Gerla, M., Lee, Y. Z., & Sanadidi, M. (2008). TCP with delayed ack for wireless networks. *Ad Hoc Networks, 6*(7), 1098-1116.
16. Corral, G., Zaballos, A., Fulgueira, T., & Abella, J. (2012). Simulation-based study of TCP flow control mechanisms using OPNET Modeler. *algorithms, 2,* 8.
17. Floyd, S., & Henderson, T. (1999). *The NewReno Modification to TCP's Fast Recovery Algorithm RFC 2582 (Experimental).* Retrieved from
18. Gupta, M., Jain, R., Arora, S., Gupta, A., Javed Awan, M., Chaudhary, G., & Nobanee, H. (2021). AI-enabled COVID-9 Outbreak Analysis and Prediction: Indian States vs. Union Territories. *Computers, Materials & Continua, 67*(1), 933-950. doi:10.32604/cmc.2021.014221
19. Ha, S., Rhee, I., & Xu, L. (2008). CUBIC: a new TCP-friendly high-speed TCP variant. *ACM SIGOPS operating systems review, 42*(5), 64-74.

20. Javed Awan, M., Shafry Mohd Rahim, M., Nobanee, H., Munawar, A., Yasin, A., & Mohd Zain Azlanmz, A. (2021). Social Media and Stock Market Prediction: A Big Data Approach. *Computers, Materials & Continua, 67*(2), 2569-2583. doi:10.32604/cmc.2021.014253
21. Javed Awan, M., Shafry Mohd Rahim, M., Nobanee, H., Yasin, A., Ibrahim Khalaf, O., & Ishfaq, U. (2021). A Big Data Approach to Black Friday Sales. *Intelligent Automation & Soft Computing, 27*(3), 785-797. doi:10.32604/iasc.2021.014216
22. Javed, R., Saba, T., Humdullah, S., Jamail, N. S. M., & Awan, M. J. (2021). *An Efficient Pattern Recognition Based Method for Drug-Drug Interaction Diagnosis.* Paper presented at the 2021 1st International Conference on Artificial Intelligence and Data Analytics (CAIDA).
23. Kwon, H., Seo, H., Kim, S., & Lee, B. G. (2009). Generalized CSMA/CA for OFDMA systems: protocol design, throughput analysis, and implementation issues. *IEEE Transactions on Wireless Communications, 8*(8), 4176-4187.
24. Miao, X., Feng, Q., Ping, D., Yajuan, Q., Sidong, Z., & Hongke, Z. (2009). *Fairness evaluation of the default highspeed TCP in common operating systems.* Paper presented at the 2009 2nd IEEE International Conference on Broadband Network & Multimedia Technology.
25. Mujahid, A., Awan, M. J., Yasin, A., Mohammed, M. A., Damaševičius, R., Maskeliūnas, R., & Abdulkareem, K. H. (2021). Real-Time Hand Gesture Recognition Based on Deep Learning YOLOv3 Model. *Applied Sciences, 11*(9). doi:10.3390/app11094164
26. Munir, K., Welzl, M., & Damjanovic, D. (2007). *Linux beats windows! or the worrying evolution of TCP in common operating systems.* Paper presented at the PFLDnet Workshop. pp.
27. Nagi, A. T., Awan, M. J., Javed, R., & Ayesha, N. (2021). *A Comparison of Two-Stage Classifier Algorithm with Ensemble Techniques On Detection of Diabetic Retinopathy.* Paper presented at the 2021 1st International Conference on Artificial Intelligence and Data Analytics (CAIDA).
28. Ong, K. (2020). *Evaluation and optimisation of Less-than-Best-Effort TCP congestion control mechanisms.* Murdoch University,
29. Oura, R., & Yamaguchi, S. (2012). *Fairness comparisons among modern TCP implementations.* Paper presented at the 2012 26th International Conference on Advanced Information Networking and Applications Workshops.
30. Postel, J. (1983). *The TCP maximum segment size and related topics*. Retrieved from
31. Rehma, A. A., Awan, M. J., & Butt, I. (2018). Comparison and Evaluation of Information Retrieval Models. *VFAST Transactions on Software Engineering, 6*(1), 7-14.
32. Wang, J., Wen, J., Han, Y., Zhang, J., Li, C., & Xiong, Z. (2013). CUBIC-FIT: A high performance and TCP CUBIC friendly congestion control algorithm. *IEEE Communications Letters, 17*(8), 1664-1667.
33. Xylomenos, G., Polyzos, G. C., Mahonen, P., & Saaranen, M. (2001). TCP performance issues over wireless links. *IEEE communications magazine, 39*(4), 52-58.