

Transmission Control Protocol Accelerator on Packet Logic Real Time Enforcement (TCPA on PRE)

PREPARED BY
Shivi Sharma (2018BTechCSE118)

FACULTY GUIDE
Dr. S. Taruna



**Department of Compute Science Engineering
Institute of Engineering and Technology (IET)
JK Lakshmipat University Jaipur**

May 2022

**Transmission Control Protocol
Accelerator
on
Packet Logic Real Time Enforcement
(TCPA on PRE)**

Practice School-2

Submitted in partial fulfilment of the requirements

for the degree of

Bachelor of Technology in Computer Science Engineering

By:

Shivi Sharma (2018BTechCSE118)

Faculty Guide:

Dr. S. Taruna

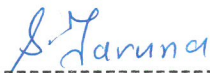


**Department of Computer Science Engineering
Institute of Engineering and Technology (IET)
JK Lakshmipat University Jaipur**

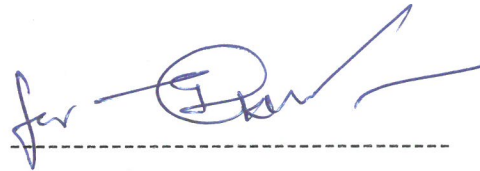
May 2022

CERTIFICATE

This is to certify that the Practice School-2 project work entitled “**TCP Accelerator on Packet Logic Real Time Enforcement**” submitted by **Shivi Sharma (2018BTechCSE118)**, towards the partial fulfillment of the requirements for the degree of **Bachelor of Technology in Computer Science Engineering** of JK Lakshmipat University Jaipur is the record of work carried out by them under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for Practice School-II examination.

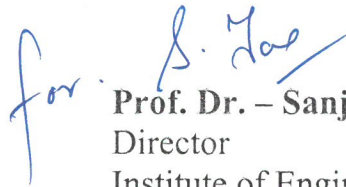


Dr. S Taruna
Associate Professor,
Department of Computer Science
Engineering
Institute of Engineering &
Technology (IET)
JK Lakshmipat University, Jaipur



Dr. Sonal Jain
Co-ordinator and Associate Professor,
Department of Computer Science
Engineering
Institute of Engineering &
Technology (IET)
JK Lakshmipat University, Jaipur

Date of Submission: 13-05-2022



Prof. Dr. – Sanjay Goel
Director
Institute of Engineering & Technology
JK Lakshmipat University Jaipur

ACKNOWLEDGEMENTS

I would like to express my heartfelt gratitude to Dr. S. Taruna, Professor, Department of Computer Science & Engineering for her valuable time and guidance that made the project work a success.

My heartfelt thanks to Mr. Neelesh Diwedi, Mr. Ankit Chibber and Mr. Manoj Mallappa for their support during the project. I would also like to thank Mr. Nilanjan Sarkar, Mr. Sandesh Nayak and Mr. Bhargav Marithamanahalli Prabhakar for their valuable input to the project work. Thanking all my friends and all those who have helped me carrying out this work directly or indirectly without whom working on this project work was not possible.

I would also like to thank sincerely to Dr. Sonal Jain, Head of CSE Department and Dr. Sanjay Goel, Director, Institute of Engineering and Technology, JK Lakshmipat University for giving us a platform to carry out the project.



Sincerely yours,

Shivi Sharma (2018BTechCSE118)

ABSTRACT

TCP is the internet transport protocol standard, accounting for an estimated 75 percent of all fixed and mobile internet traffic. Considering that TCP/IP was developed in the 1970s, and adopted in 1983, it has been a continuously remarkable enabler of internet growth. However, TCP has also created network performance challenges that have only been magnified as networks and services have become increasingly diverse. Although TCP is highly reliable as a delivery mechanism, it is chronically inefficient as it cannot recognize and adapt to differences between networks, applications, endpoints, and conditions; this means today's largely TCP-reliant networks will not perform to the full extent of their potential unless a new approach is taken. [2]

TCP is not fast enough to take the advantage of additional bandwidth. A very high latency and bad quality is experienced when bandwidth is scarce and TCP is too fast. When there are multiple TCP connections, they don't interact to behave as a whole. TCP Accelerator ensures TCP doesn't become bottleneck and makes the collection of TCP connections on the network to behave efficiently taking control over the TCP. It acts as server to client and client to server. During downlink or uplink if TCP is too slow/fast it can accelerate/deaccelerate the traffic.

Sandvine's TCP Optimization exerts greater control over TCP, accelerating performance even in networks that have already deployed congestion, video streaming, or network link management solutions. Focusing solely on the transport layer, the solution demonstrates marked improvements in performance. Sandvine's approach makes TCP access-aware, optimizing it to address the unique characteristics of today's diverse access network landscape, including satellite, 3G, 4G, 5G, fiber, DOCSIS, and DSL. Importantly, Sandvine avoids the pitfalls of proxy-based optimization by employing the more robust and precise TCP "midpoint" approach, asserting control over TCP while remaining transparent end-to-end, to both the end-user device and the server. [4]

When the TCP Optimization solution is deployed, important network and radio access resources are freed up much faster, and networks and spectrum are far better utilized, supporting more concurrent users, delivering more throughput, and achieving overall greater efficiency. As a result, users are also positively impacted as they experience higher speeds and therefore better QoE. With acceleration: radio glitches don't cause slow-downs, so transfers complete much faster and RAN resources are more readily available. [4]

List of Figures

Figure 1.1 Sandvine Logo	1
Figure 1.2 TCP Accelerator	1
Figure 1.3 Increased transfer rates over an LTE network	2
Figure 2.1 Active Logic	6
Figure 2.2 Layers of OSI Model	6
Figure 2.3 TCP Header	7
Figure 2.4 Network Congestion	8
Figure 2.5 TCP New Reno	9
Figure 2.6 Selective Acknowledgement	10
Figure 2.7 TCP Three-way Handshake	11
Figure 2.8 (a) Pseudo Header	11
Figure 2.8 (b) Checksum Calculation	11
Figure 2.9 B-Tree Insertion	12
Figure 2.10 B-Tree Deletion	13
Figure 4.1 Accelerated and Non-accelerated Traffic	17
Figure 4.2 Average Round Trip Time	18
Figure 4.3 Retransmission Rate towards Mobile	18
Figure 4.4 Network Efficiency	18

List of Acronyms

- TCP : Transmission Control Protocol
- TCPA : Transmission Control Protocol Accelerator
- QoE : Quality of Engagement
- IP : Internet Protocol
- DOCSIS : Data over cable service interface specification
- DSL : digital subscriber line
- LTE : Long-Term Evolution
- HTTP2 : HyperText Transfer Protocol 2
- QUIC : Quick UDP Internet Connections
- Wi-Fi : Wireless Fidelity
- PRE : Packet Logic Real Time Enforcement
- AL : Active Logic
- OSI : Open Systems Interconnection Model
- DO : Data offset
- RSV : Reserved Field
- URG : Urgent Pointer
- ACK : Acknowledgement
- PSH : Push Function

- RST : Reset
- SYN : Synchronize
- FIN : Finish
- CAA : Congestive Avoidance Algorithms
- Ssthresh : Slow Start Threshold
- AIMD : Additive Increase Multiplicative Decrease
- SACK : Selective Acknowledgement
- PAR : Positive Acknowledgement with re-transmission
- PDU : Protocol Data Unit
- PuTTY : Popular SSH and Telnet Client
- SSH : Secure Shell
- TCL : Tool command language
- DOS : Disk Operating System
- NUMA : Non-uniform Memory Allocation
- DPDK : Data Plane Development Kit
- CPU : Central Processing Unit
- VLAN : Virtual Local Area Networks
- ARP : Address Resolution Protocol

TABLE OF CONTENTS

Topic	Page No.
CERTIFICATE	iii
ACKNOWLEDGEMENTS	iv
ABSTRACT	v
LIST OF FIGURES	vi
LIST OF ACRONYMS	vi
CHAPTER 1: INTRODUCTION	
1.1 About Organization	1
1.2 Problem Statement	1
1.3 Objectives	2
1.4 Solutions	2
1.5 Key Features	3
CHAPTER 2: METHODOLOGIES	
2.1 Packet Logic Real Time Enforcement	6
2.2 Transmission Control Protocol	6
2.3 Transmission Control Protocol – Header	7
2.4 Congestion Control	8
2.5 TCP New Reno – Congestion Control Algorithm	9
2.6 Selective Acknowledgement	10
2.7 TCP Three-way Handshake	11
2.8 Error Detection Mechanism – Checksum	11
2.9 B-Tree	12
CHAPTER 3: TECHNOLOGY STACK	
3.1 C Programming	15
3.2 Linux Shell Commands	15
3.3 Git	15
3.4 Docker	15
3.5 PuTTY	15
3.6 TCL – Tool Command Language	16
3.7 JIRA	16
3.8 NUMA – Non uniform memory allocation	16
3.9 DPDK – Data Plane Development Kit	16
3.10 Scapy	16
CHAPTER 4: RESULTS	
4.1 Conclusion	17
REFERENCES	19

CHAPTER 1 : INTRODUCTION

1.1 About Organization

As a leading authority on internet phenomena, Sandvine provides a comprehensive suite of application and network intelligence solutions to service providers, enterprises, and governments around the world. In support of our core values, we take full ownership of our responsibilities, work collaboratively inside and outside the company to make our customers successful, and cultivate a welcoming atmosphere with equality, inclusion and dignity for all. Sandvine's policy is to investigate all reports of potential misuse of its products to perpetuate human rights abuse. [1]

As network and application usage shifts on a daily basis, service providers are looking for new ways to analyze, optimize, and monetize applications so they can deliver high quality consumer and enterprise experiences. [1]



Figure 1.1: Sandvine Logo

1.2 Problem Statement

TCP is not fast enough to take the advantage of additional bandwidth. A very high latency and bad QoE is experienced when bandwidth is scarce and TCP is too fast. When there are multiple TCP connections, they don't interact to behave as a whole. [4]

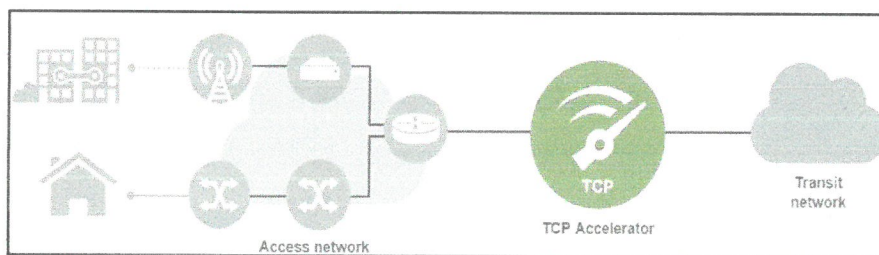


Figure 1.2: TCP Accelerator

For operators, these problems create significant consequences:

- Network efficiency is harmed, causing performance and return on investment to fall below expectations [4]
- Application QoE is impacted, decreasing customer satisfaction and potentially increasing churn [4]

TCPA ensures TCP doesn't become a bottleneck and makes the collection of TCP connections on the network to behave efficiently taking control over the TCP. It acts as server to client and client to server. During downlink or uplink if TCP is too slow/fast it can accelerate/deaccelerate the traffic. [4]

1.3 Objectives

OBJECTIVE 1: Increased Network Performance. Higher ratio of goodput to throughput, better resource utilization, reduced retransmissions, and extended legacy infrastructure lifetimes. [4]

OBJECTIVE 2: Improved Quality of Experience (QoE). Faster data transmissions and increased application performance lead to measured and consistent QoE, which tends to increase customer satisfaction. [4]

OBJECTIVE 3: Rapid Return on Investment: More efficient resource utilization defers capital and lets access network resources run 'hotter' without compromising quality of experience. [4]

1.4 Solutions

Sandvine's TCP Accelerator is a broadband optimization network function designed to resolve network congestion challenges. It is placed at the edge of the access network where it runs TCP/IP algorithms that are carefully tuned to react in a manner that is optimal for lossy high delay product networks, while still using simple algorithms towards the internet. [4]

Sandvine's TCP Accelerator allows network operators to dramatically improve network efficiency and quality of experience (QoE), without making changes to the network infrastructure. TCP Accelerator addresses the main issues associated with TCP, including the incompatibility of bandwidth availability and TCP speeds. It also addresses the inefficiencies caused by multiple concurrent TCP connections by transparently bridging the access network with the transit network; it is in a position to manage all the network's TCP connections as a collective whole. [4]

Sandvine's TCP Optimization exerts greater control over TCP, accelerating performance even in networks that have already deployed congestion, video streaming, or network link management solutions. Focusing solely on the transport layer, the solution demonstrates marked improvements in performance. Sandvine's approach makes TCP access-aware, optimizing it to address the unique characteristics of today's diverse access network landscape, including satellite, 3G, 4G, 5G, fiber, DOCSIS, and DSL. Importantly, Sandvine avoids the pitfalls of proxy-based optimization by employing the more robust and precise TCP "midpoint" approach, asserting control over TCP while remaining transparent end-to-end, to both the end-user device and the server. [4]

Transfer Size	10KB-50KB	50KB-100KB	100KB-500KB	500KB-1MB	1MB-5MB	5MB-10MB	>10MB
Uploads	+18%	+12%	+32%	+36%	+38%	+38%	+31%
Downloads	+11%	+17%	+24%	+43%	+43%	+33%	+24%

Figure 1.3: Increased transfer rates over an LTE network

1.5 Key Features

1.5.1 Transparency

TCP Accelerator behaves as a bridge and doesn't terminate the TCP connection, so the acceleration is completely transparent to the endpoints, yielding a number of benefits:

- Connection migration: connections can be migrated from optimized to non-optimized without affecting the connection from the perspective of either endpoint. [4]
- Connection resumption: long-lived but inactive connections time-out after a prolonged period of inactivity to recycle state memory; because of transparency, if a connection times out of TCP Accelerator, but then resumes, the connection is able to resume seamlessly. [4]
- Sequence number preservation: sequence numbers are untouched, which ensures that the TCP connection state in the client and server will be the same after the TCP handshake. [4]
- Step-out for easy upgrades: when acceleration is halted and the stores and forward buffers are cleared, TCP Accelerator can be stopped within minutes, without disrupting traffic. [4]
- TCP option transparency: if new TCP options or features are introduced, the endpoints can still negotiate its use (unlike terminating proxies, which would prevent its usage entirely). [4]

1.5.2 Powerful TCP Acceleration Techniques

TCP represents between 85% to 90% of fixed access internet traffic, and as much as 96% of mobile traffic, and TCP Accelerator applies acceleration to practically all of it, including:

- Uploads and downloads: acceleration is applied to upload and download traffic. [4]
- Any application: the default configuration accelerates traffic agnostic of application, protocol, service, etc. [4]
- Encrypted traffic: with more than 70% of traffic now encrypted, it's critical that solutions work with encrypted traffic, and TCP Accelerator does. [4]
- HTTP2: fully supported. [4]

The only TCP that isn't accelerated is the traffic that shouldn't be accelerated. For instance, if the traffic originates from a specialized device (e.g., a sensor or probe) that should be omitted from TCP acceleration, then TCP acceleration is not applied.

In addition to the buffer management features described below, TCP is accelerated with a combination of techniques, including:

- Two-sided acceleration: most configuration variables can be set separately for the access side and the internet side. [4]
- Reduced packet loss effect during slow-start: tunable system for mitigating the impact of TCP packet loss early in the connection, by allowing such packet loss to be ignored for the purposes of adjusting the congestion window. [4]
- Congestion control: distinguishes between genuine congestion events and radio glitches (in mobile networks) and handles each accordingly. [4]
- Fast retransmit: supports TCP Fast Retransmit, with configurable behaviour. [4]

- Improved retransmission handling: replaces the standard timeout-based TCP retransmit entirely for the access side of the connection, which has an enormous positive impact on mobile networks (where hand-overs and jitter can cause spurious retransmissions). [4]

Plus, TCP Accelerator is in a position to manage all the network's TCP connections as a collective whole, to ensure maximum efficiency.

1.5.3 TCP Buffer Management

When high-bandwidth downloads fill the buffers (queues) in the access network, other traffic gets stuck behind and the user experience suffers. [4]

To prevent this scenario—called bufferbloat—in the access network, round-trip times need to be carefully managed; at the same time, a minimum amount of data has to be sent so as not to starve the radio network. [4]

TCP Accelerator manages buffer queues by adjusting the sending rate to correspond to the level of buffered data in the access network. When TCP Accelerator determines that too much data is being queued for a particular user, it sends less data for the user in order to give DATASHEET time for the queue to shrink. TCP Accelerator learns the base latency for each user, which accounts for the varying radio access technologies and roaming users. [4]

Importantly, TCP Accelerator prevents latency-insensitive applications from being favored over latency-sensitive ones. This approach treats all traffic fairly and has an enormous positive impact on QoE because it ensures sensitive applications aren't starved. [4]

1.5.4 QUIC Traffic Control

QUIC is a UDP-based, Google provided protocol that has been designed to deliver data at lower latency rates than those of TCP. The protocol has gained some market traction with YouTube and other popular applications using it for transport. [4]

QUIC does not work in a coordinated fashion with TCP, so it can have a serious and negative impact on TCP traffic performance. Left unchecked, QUIC has a tendency to overwhelm the same network buffers used by TCP, degrading the performance of TCP-based services that are on their way to users. [4]

Before choosing a TCP optimization solution, make sure that the solution addresses both TCP and QUIC traffic management to ensure that applications running over both protocols can continue to perform as intended. [4]

1.5.5 Egress Burst Control

When a large amount of data is released at once from a server close to the core network, this traffic can overwhelm any switch where there is a speed differential (e.g., going from 10G to 1G); the buffers in the switch may be insufficient to accommodate micro-bursts resulting from the speed differential, and as a result they might be forced to simply drop packets. This behavior has a negative impact on the experience of all users on the impacted switch. [4]

To prevent this scenario from unfolding, Sandvine's TCP Accelerator prevents buffer overflow by limiting the amount of traffic sent to a user in a configurable fraction of a second (typically

a millisecond). This limiting allows the switch to handle the burst without exhausting its buffers, significantly reducing retransmission rates. [4]

This functionality can relieve a network operator from replacing a huge number of switches. [4]

1.5.6 Universal Access Support

TCP Accelerator can be deployed in any type of access network, with any combination of access technologies (e.g., Cable, DSL, 3G, 4G, WiFi, Satellite, etc.). [4]

1.5.7 Audit Records and Historic Reporting

TCP performance measurements and statistics are logged and can be used for audit purposes or examined for business and operational intelligence. [4]

CHAPTER 2 : METHODOLOGIES

2.1 Packet Logic Real Time Enforcement (Active Logic)

Operates between the internet and subscribers. All the traffic passes through active logic, and it parses and inspects the packet while bridging them. It can be configured to take various actions based on rules on specific traffic.

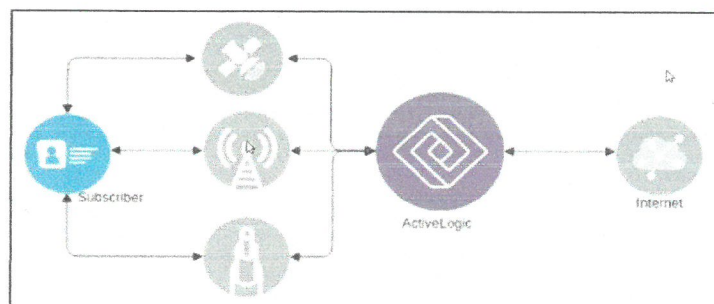


Figure 2.1: Active Logic

Sandvine's ActiveLogic equips service providers with a simplified, hyperscale, automation-ready data plane to tackle the next era of network challenges. With ActiveLogic, service providers have the flexibility, performance, and machine learning-powered traffic classification needed to keep up with internet phenomena. The data plane product – ActiveLogic – serves as the foundation for all Sandvine's Application and Network Intelligence solutions. [5]

2.2 Transmission Control Protocol

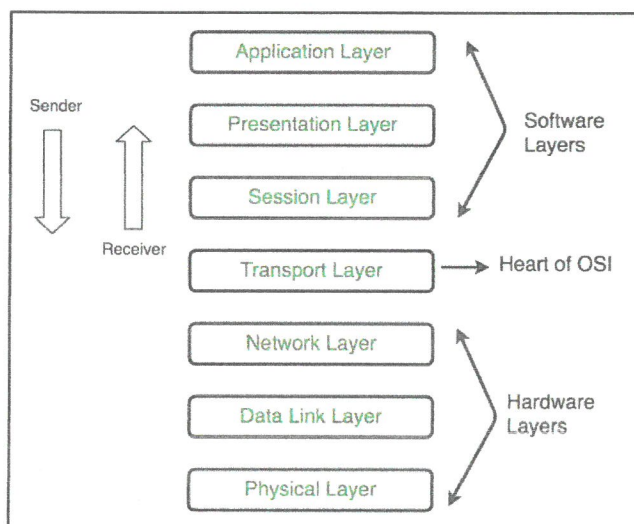


Figure 2.2: Layers of OSI Model

When we receive data from the application it is broken into smaller data parts as the whole data from the application cannot be sent through the network to the receiver host. The protocol we use in OSI (Open Systems Interconnection Model) in the Transport layer is TCP. So, after breaking the data from the application layer into smaller parts. This broken part form the body of the TCP. The TCP header usually varies from 20 Bytes(with no bits of option fields being used) to 60 Bytes (with all bits of options field being used). It has fields like Source and Destination Port addresses, urgent pointer, Checksum, etc. [3]

TCP stands for Transmission Control Protocol a communications standard that enables application programs and computing devices to exchange messages over a network. It is designed to send packets across the internet and ensure the successful delivery of data and messages over networks. [2]

TCP organizes data so that it can be transmitted between a server and a client. It guarantees the integrity of the data being communicated over a network. Before it transmits data, TCP establishes a connection between a source and its destination, which it ensures remains live until communication begins. It then breaks large amounts of data into smaller packets, while ensuring data integrity is in place throughout the process. [2]

2.3 Transmission Control Protocol - Header

The Transmission Control Protocol (TCP) header is the first 24 bytes of a TCP segment that contains the parameters and state of an end-to-end TCP socket. The TCP header is used to track the state of communication between two TCP endpoints. [6]

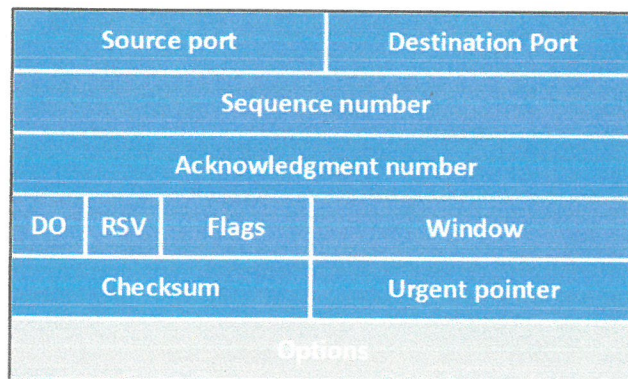


Figure 2.3: TCP Header

- Source port: this is a 16 bit field that specifies the port number of the sender.
- Destination port: this is a 16 bit field that specifies the port number of the receiver.
- Sequence number: the sequence number is a 32 bit field that indicates how much data is sent during the TCP session. When you establish a new TCP connection (3 way handshake) then the initial sequence number is a random 32 bit value. The receiver will use this sequence number and sends back an acknowledgment. Protocol analyzers like wireshark will often use a relative sequence number of 0 since it's easier to read than some high random number.
- Acknowledgment number: this 32 bit field is used by the receiver to request the next TCP segment. This value will be the sequence number incremented by 1.
- DO: this is the 4 bit data offset field, also known as the header length. It indicates the length of the TCP header so that we know where the actual data begins.
- RSV: these are 3 bits for the reserved field. They are unused and are always set to 0.
- Flags: there are 9 bits for flags, we also call them control bits. We use them to establish connections, send data and terminate connections:

- URG: urgent pointer. When this bit is set, the data should be treated as priority over other data.
 - ACK: used for the acknowledgment.
 - PSH: this is the push function. This tells an application that the data should be transmitted immediately and that we don't want to wait to fill the entire TCP segment.
 - RST: this resets the connection, when you receive this you have to terminate the connection right away. This is only used when there are unrecoverable errors and it's not a normal way to finish the TCP connection.
 - SYN: we use this for the initial three way handshake and it's used to set the initial sequence number.
 - FIN: this finish bit is used to end the TCP connection. TCP is full duplex so both parties will have to use the FIN bit to end the connection. This is the normal method how we end an connection.
- Window: the 16 bit window field specifies how many bytes the receiver is willing to receive. It is used so the receiver can tell the sender that it would like to receive more data than what it is currently receiving. It does so by specifying the number of bytes beyond the sequence number in the acknowledgment field.
 - Checksum: 16 bits are used for a checksum to check if the TCP header is OK or not.
 - Urgent pointer: these 16 bits are used when the URG bit has been set, the urgent pointer is used to indicate where the urgent data ends.
 - Options: this field is optional and can be anywhere between 0 and 320 bits.

2.4 Congestion Control

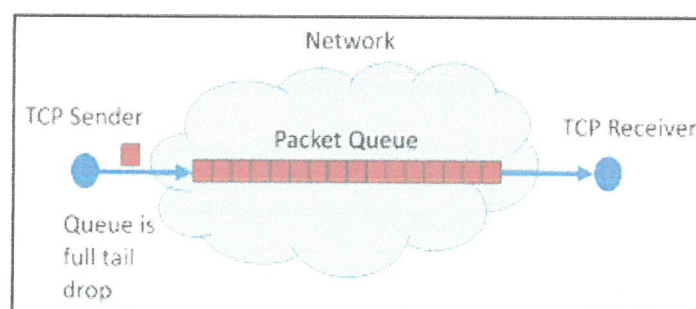


Figure 2.4: Congestion

Congestion is a state occurs in the network layer when the message traffic is so heavy that it slows down network response time.

Congestion Control is a mechanism that controls the entry of data packets into the network, enabling a better use of a shared network infrastructure and avoiding congestive collapse. Congestive-Avoidance Algorithms (CAA) are implemented at the TCP layer as the mechanism to avoid congestive collapse in a network. [7]

TCP uses a congestion window in the sender side to do congestion avoidance. The congestion window indicates the maximum amount of data that can be sent out on a connection without being acknowledged. TCP detects congestion when it fails to receive an acknowledgement for a packet within the estimated timeout. [7]

2.5 TCP New Reno – Congestion Control Algorithm

TCP Reno is the second variant of the TCP which came up with an in-built congestion algorithm. Congestion handling was not an integral part of the original TCP/IP suite. When packet loss occurs, the sender reduces the cwnd by 50% along with the ssthresh value for all the packets loss that happened in the same congestion window. This would allow the network to come out of the congestion state easily. It uses the concept of partial acknowledgment. When the sender receives the ACK of the first retransmitted packet then it does not consider it a “New ACK”. [8]

NewReno checks if all the previously transmitted packets of that particular window are ACKed or not. If there are multiple packets lost in the same congestion window then the receiver would have been sending the duplicate ACKs only even after receiving the retransmitted packet. This will make it clear to the sender that all the packets are not reached the receiver and hence sender will not consider that ACK as new. It will consider it a partial ACK because only a partial window is being ACKed not the whole. [8]

Slow Start phase: It lasts until congestion window size reaches up to “Slow Start Threshold (ssthresh)”. The slow start algorithm doubles the congestion window size(cwnd) in one RTT. Initially, ssthresh is set to infinite(∞). Subsequently, it adapts depending on the packet loss events. When cwnd becomes equal to ssthresh, slow start stops. After that AIMD phase takeovers.

AIMD phase: It starts when the slow start stops. Additive Increase increases cwnd by 1 and Multiplicative Decrease reduces ssthresh to 50% of cwnd. Note that cwnd is ‘not’ reduced by 50% but ssthresh. This is the point to note that cwnd again resets to the initial window size (10 in Linux kernel).

Fast Retransmit phase: It is the loss detection algorithm. It is triggered by 3 duplicate acknowledgments. On packet loss detection, it resets cwnd to initcwnd.

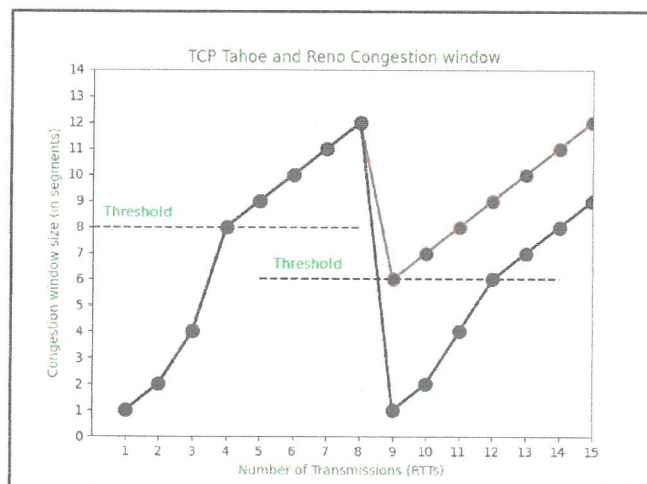


Figure 2.5: TCP New Reno

2.6 Selective Acknowledgement

TCP uses a cumulative acknowledgment scheme in which received segments that are not at the left edge of the receive window are not acknowledged. This forces the sender to either wait a roundtrip time to find out about each lost packet, or to unnecessarily retransmit segments which have been correctly received. Selective Acknowledgment (SACK) is a strategy which corrects this behavior in the face of multiple dropped segments. With selective acknowledgments, the data receiver can inform the sender about all segments that have arrived successfully, so the sender need retransmit only the segments that have actually been lost. [9]

SACK is a sender and receiver side optimization to TCP. Both sender and receiver should support the SACK feature then only it is possible to use this. SACK does not replace the original ACKs in the TCP header but adds another field in the TCP header's option field for SACK information. 'Options' field in the TCP header is used to send SACKs not the actual ACK field is tempered. A maximum of 32 bytes (out of 40 bytes) can be used for SACK in the 'Options' field. 8 bytes in the 'Options' field are already reserved for 'timestamps'. [10]

SACK feature helps the sender to identify 'gaps' in the receiver buffer. When the receiver does not receive packets in order then there exist some holes(missing points) in the buffer. SACK helps the sender to know about these holes in the receiver buffer at an early time. When the sender knows about the holes or lost packets, it retransmits them at the same time. [10]

A maximum of four SACK blocks can be reported by the receiver in one segment. One SACK block needs 8 bytes, so 8×4 SACK blocks = 32 bytes are available for SACK blocks. Because 8 bytes are reserved for the TCP Timestamp Option (Kind = 8, Length = 10). [10]

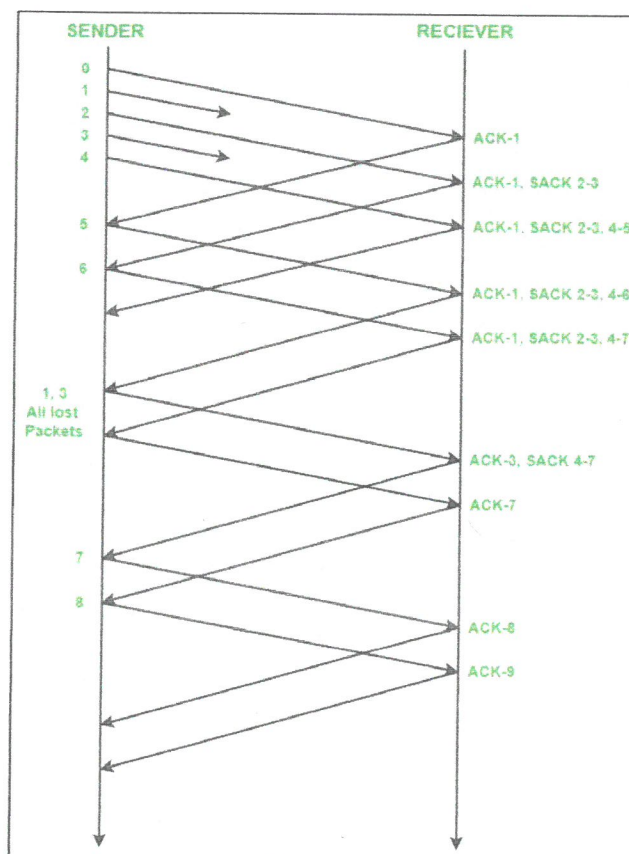


Figure 2.6: Selective Acknowledgement

2.7 TCP Three-way Handshake

TCP provides reliable communication with something called Positive Acknowledgement with Re-transmission (PAR). The Protocol Data Unit (PDU) of the transport layer is called a segment. Now a device using PAR resend the data unit until it receives an acknowledgement. If the data unit received at the receiver's end is damaged (it checks the data with checksum functionality of the transport layer that is used for Error Detection), the receiver discards the segment. So the sender has to resend the data unit for which positive acknowledgement is not received. You can realize from the above mechanism that three segments are exchanged between sender (client) and receiver (server) for a reliable TCP connection to get established. [11]

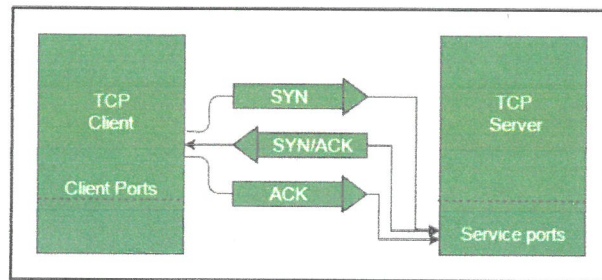


Figure 2.7: TCP Three-way Handshake

(SYN): In the first step, the client wants to establish a connection with a server, so it sends a segment with SYN(Synchronize Sequence Number) which informs the server that the client is likely to start communication and with what sequence number it starts segments with.

(SYN+ACK): Server responds to the client request with SYN-ACK signal bits set. Acknowledgement(ACK) signifies the response of the segment it received and SYN signifies with what sequence number it is likely to start the segments with.

(ACK): In the final part client acknowledges the response of the server and they both establish a reliable connection with which they will start the actual data transfer.

2.8 Error Detection Mechanism - Checksum

Checksum is a simple error detection mechanism to determine the integrity of the data transmitted over a network. Communication protocols like TCP implement this scheme in order to determine whether the received data is corrupted along the network. [12]

To calculate the TCP segment header's Checksum field, the TCP pseudo header is first constructed and placed, logically, before the TCP segment. The checksum is then calculated over both the pseudo header and the TCP segment. The pseudo header is then discarded. [12]

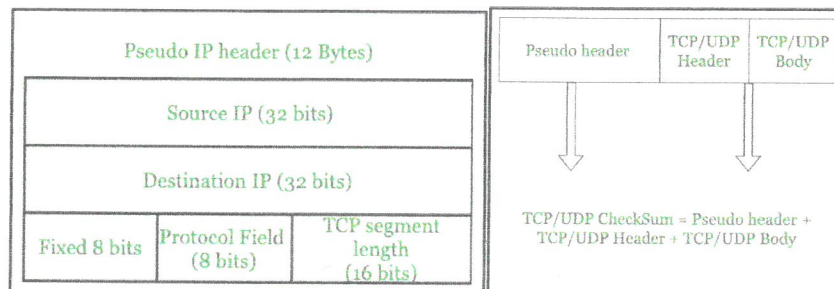


Figure 2.8: (a) Pseudo Header

(b) Checksum Calculation

2.9 B-Tree

B-Tree is a self-balancing search tree. To understand the use of B-Trees, we must think of the huge amount of data that cannot fit in main memory. When the number of keys is high, the data is read from disk in the form of blocks. Disk access time is very high compared to the main memory access time. The main idea of using B-Trees is to reduce the number of disk accesses. Most of the tree operations (search, insert, delete, max, min, ..etc) require $O(h)$ disk accesses where h is the height of the tree. B-tree is a fat tree. The height of B-Trees is kept low by putting maximum possible keys in a B-Tree node. [13]

1. All leaves are at the same level.
2. A B-Tree is defined by the term minimum degree 't'. The value of t depends upon disk block size.
3. Every node except root must contain at least $t-1$ keys. The root may contain minimum 1 key.
4. All nodes (including root) may contain at most $2*t - 1$ keys.
5. Number of children of a node is equal to the number of keys in it plus 1.
6. All keys of a node are sorted in increasing order. The child between two keys k_1 and k_2 contains all keys in the range from k_1 and k_2 .
7. B-Tree grows and shrinks from the root which is unlike Binary Search Tree. Binary Search Trees grow downward and also shrink from downward.
8. Like other balanced Binary Search Trees, time complexity to search, insert and delete is $O(\log n)$.
9. Insertion of a Node in B-Tree happens only at Leaf Node.

Search Operation in B-Tree:

Search is similar to the search in Binary Search Tree. Let the key to be searched be k . We start from the root and recursively traverse down. For every visited non-leaf node, if the node has the key, we simply return the node. Otherwise, we recur down to the appropriate child (The child which is just before the first greater key) of the node. If we reach a leaf node and don't find k in the leaf node, we return NULL. [14]

Insertion in B-Tree:

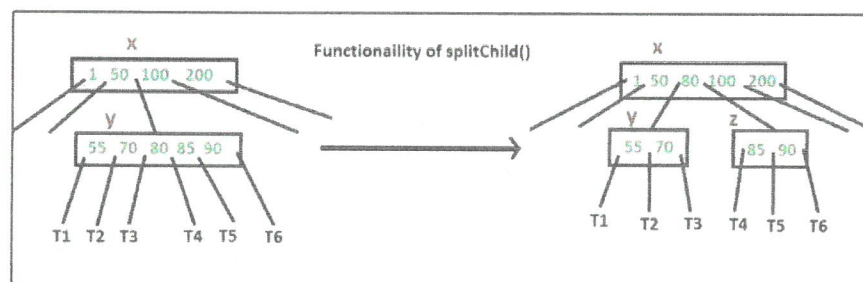


Figure 2.9: B-Tree Insertion

1. Initialize x as root.
2. While x is not leaf, do following:
 - a. Find the child of x that is going to be traversed next. Let the child be y .
 - b. If y is not full, change x to point to y .

- c. If y is full, split it and change x to point to one of the two parts of y . If k is smaller than mid key in y , then set x as the first part of y . Else second part of y .
When we split y , we move a key from y to its parent x .
3. The loop in step 2 stops when x is leaf. x must have space for 1 extra key as we have been splitting all nodes in advance. So simply insert k to x .

Deletion in B-Tree:

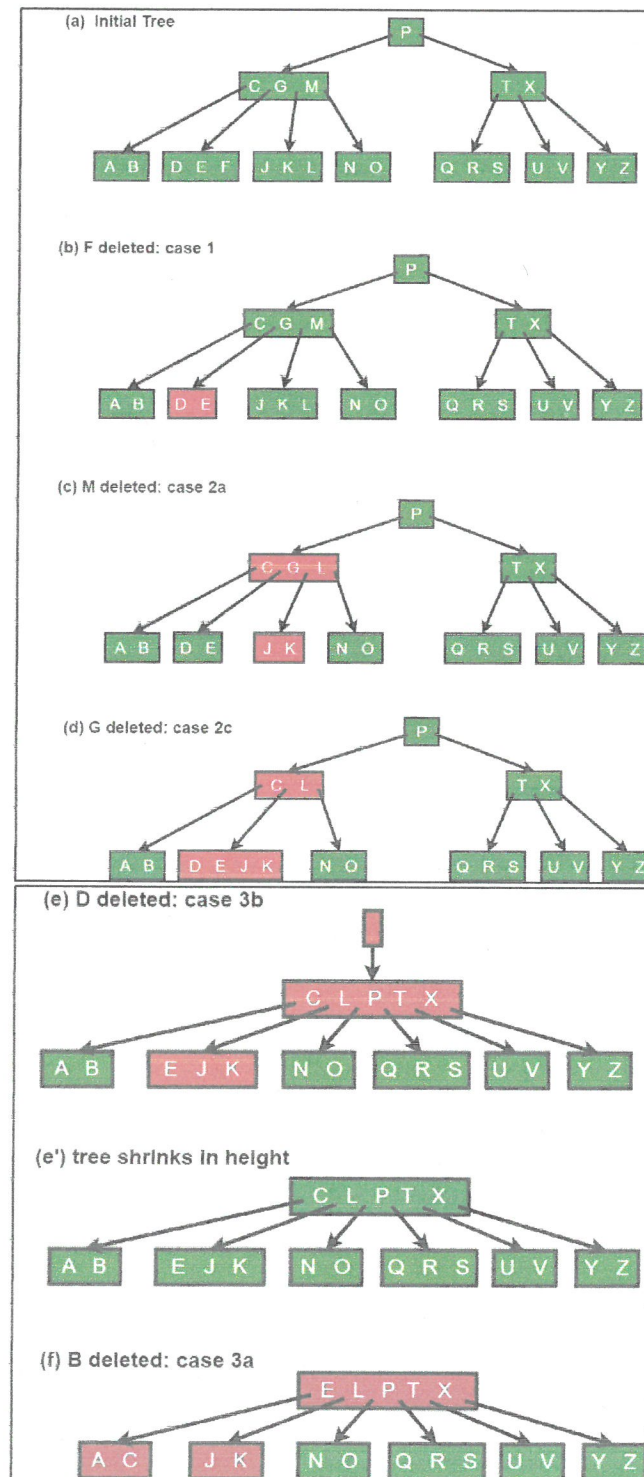


Figure 2.10: B-Tree Deletion

1. If the key k is in node x and x is a leaf, delete the key k from x .
2. If the key k is in node x and x is an internal node, do the following.
 - a. If the child y that precedes k in node x has at least t keys, then find the predecessor k_0 of k in the sub-tree rooted at y . Recursively delete k_0 , and replace k by k_0 in x . (We can find k_0 and delete it in a single downward pass.)
 - b. If y has fewer than t keys, then, symmetrically, examine the child z that follows k in node x . If z has at least t keys, then find the successor k_0 of k in the subtree rooted at z . Recursively delete k_0 , and replace k by k_0 in x . (We can find k_0 and delete it in a single downward pass.)
 - c. Otherwise, if both y and z have only $t-1$ keys, merge k and all of z into y , so that x loses both k and the pointer to z , and y now contains $2t-1$ keys. Then free z and recursively delete k from y .
3. If the key k is not present in internal node x , determine the root $x.c(i)$ of the appropriate subtree that must contain k , if k is in the tree at all. If $x.c(i)$ has only $t-1$ keys, execute step 3a or 3b as necessary to guarantee that we descend to a node containing at least t keys. Then finish by recursing on the appropriate child of x .
 - a. If $x.c(i)$ has only $t-1$ keys but has an immediate sibling with at least t keys, give $x.c(i)$ an extra key by moving a key from x down into $x.c(i)$, moving a key from $x.c(i)$'s immediate left or right sibling up into x , and moving the appropriate child pointer from the sibling into $x.c(i)$.
 - b. If $x.c(i)$ and both of $x.c(i)$'s immediate siblings have $t-1$ keys, merge $x.c(i)$ with one sibling, which involves moving a key from x down into the new merged node to become the median key for that node. [15]

CHAPTER 3 : TECHNOLOGY STACK

3.1 C programming

Network programming enables processes to communicate with each other over a computer network, but it is a complex task that requires programming with multiple libraries and protocols. With its support for third-party libraries and structured documentation, C is an ideal language to write network programs. [16]

In C, header files are simply files in which you can declare your own functions that you can use in your main program or these can be used while writing large C programs. [17]

Make - a tool that automatically determines which source files of a program need to be recompiled and/or linked. [18]

Makefile is a set of commands (similar to terminal commands) with variable names and targets to create object file and to remove them. In a single make file we can create multiple targets to compile and to remove object, binary files. You can compile your project (program) any number of times by using Makefile. [19]

3.2 Linux Shell Commands

A shell is a special user program that provides an interface to the user to use operating system services. Shell accepts human-readable commands from the user and converts them into something which the kernel can understand. It is a command language interpreter that executes commands read from input devices such as keyboards or from files. The shell gets started when the user logs in or starts the terminal. [20]

3.3 Git

Git is a DevOps tool used for source code management. It is a free and open-source version control system used to handle small to very large projects efficiently. Git is used to tracking changes in the source code, enabling multiple developers to work together on non-linear development. [21]

3.4 Docker

Docker is a software platform for building applications based on containers—small and lightweight execution environments that make shared use of the operating system kernel but otherwise run in isolation from one another. [22]

3.5 PuTTY

PuTTY is a free implementation of SSH (and telnet) for PCs running Microsoft Windows (it also includes an xterm terminal emulator). You will find PuTTY useful if you want to access an account on a Unix or other multi-user system from a PC. It is used to connect your local Windows system to NAS systems via SSH, as an alternative to OpenSSH. [23]

3.6 TCL – Tool Command Language

Tcl works by processing commands. In this regard, it's similar to an operating system shell like DOS or bash. Each line of a Tcl script is a command, and the first word of each line is the name of the command, and any other words are arguments for the command. [24]

3.7 JIRA

Jira Software is part of a family of products designed to help teams of all types manage work. Originally, Jira was designed as a bug and issue tracker. But today, Jira has evolved into a powerful work management tool for all kinds of use cases, from requirements and test case management to agile software development. [25]

3.8 NUMA

NUMA represents non-uniform memory access. In NUMA, all processes have access to all parts of main memory. Each process is closer to some memory locations than others. Memory is partitioned asymmetrically. [26]

3.9 DPDK

DPDK is the Data Plane Development Kit that consists of libraries to accelerate packet processing workloads running on a wide variety of CPU architectures. It is a set of libraries used to accelerate packet processing on multiple-core CPUs. DPDK facilitates the quicker expansion of high-speed data packet networking applications. Using instruction set architectures, DPDK enables more efficient computing than standard interrupt processing available in the kernel. [27]

3.10 Scapy

Scapy is a powerful interactive packet manipulation program. It is able to forge or decode packets of a wide number of protocols, send them on the wire, capture them, match requests and replies, and much more. It can easily handle most classical tasks like scanning, tracerouting, probing, unit tests, attacks or network discovery (it can replace hping, 85% of nmap, arpspoof, arp-sk, arping, tcpdump, tshark, p0f, etc.). It also performs very well at a lot of other specific tasks that most other tools can't handle, like sending invalid frames, injecting your own 802.11 frames, combining techniques (VLAN hopping+ARP cache poisoning, etc). [28]

CHAPTER 4 : RESULTS

4.1 Conclusion

TCP Accelerator is engineered to make TCP run better:

- When additional bandwidth is available, TCP Accelerator ensures TCP doesn't become a bottleneck by accelerating the slow-start phase and by making sure there is always enough data ready to be served.
- When too little bandwidth is available (e.g., from congestion, spotty mobile coverage, or a low-speed service plan), TCP Accelerator reduces effective latency by preventing bufferbloat in access network resources.
- By transparently bridging the access network with the transit network, TCP Accelerator is in a position to manage all the network's TCP connections as a collective whole, to ensure maximum efficiency

TCP DELIVERS:

1. More Efficient Data Transfers

This graph, showing a single TCP flow, illustrates how the accelerated traffic (blue) gets up to speed more quickly, is faster overall, is more consistent, and recovers more quickly after a genuine packet loss. [4]

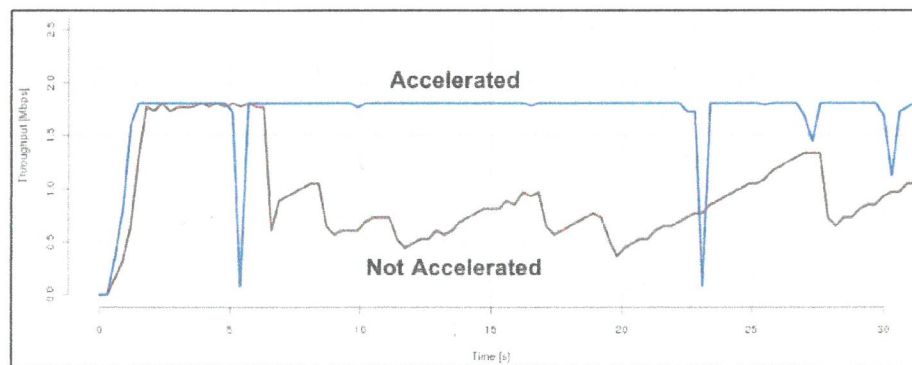


Figure 4.1: Accelerated and Non-accelerated Traffic

2. Faster and More Consistent Round-Trip Times

This graph shows the difference in flow round-trip times when TCP Acceleration (including buffer management) is enabled versus when it is not enabled. In this mobile network, the average round-trip time was cut in half (164ms average versus 320ms average), and round-trip times showed much greater consistency. [4]

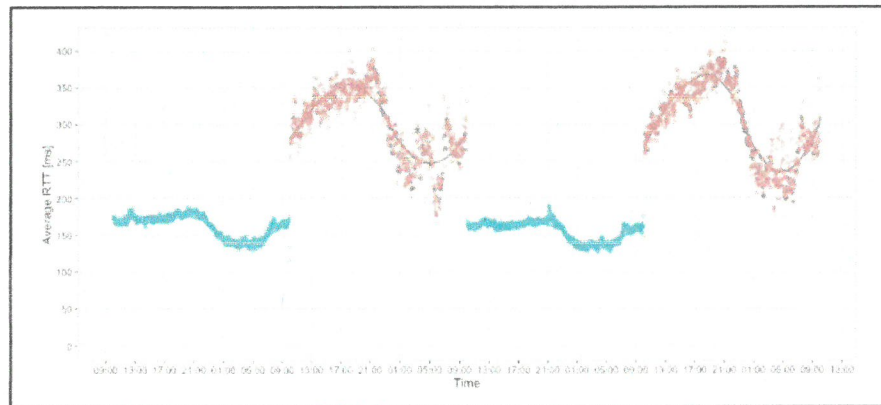


Figure 4.2: Average Round Trip Time

3. Fewer Retransmissions

This graph, shows the enormous positive effect on retransmission rate: when TCP acceleration is enabled, the retransmission rate is much lower, and much more consistent. [4]

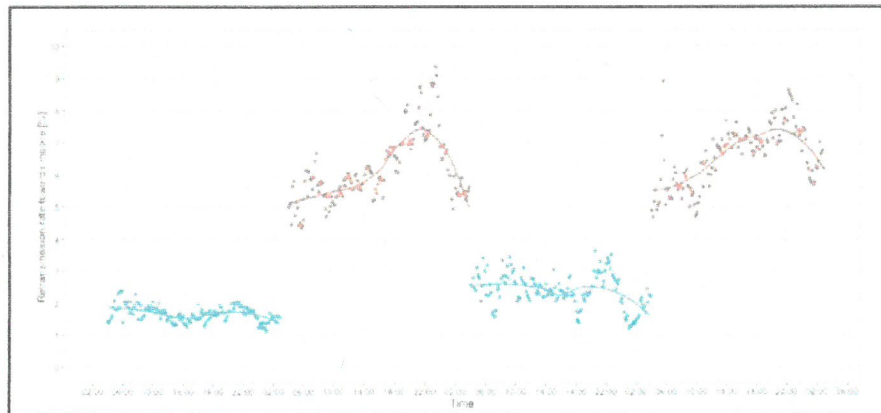


Figure 4.3: Retransmission rate towards mobile

4. More Efficient Networks

The graph, shows the overall positive impact on network efficiency as a result of TCP acceleration. In this example, network efficiency (goodput as a percentage of all throughput) improved by 5% and exhibited much tighter consistency. [4]

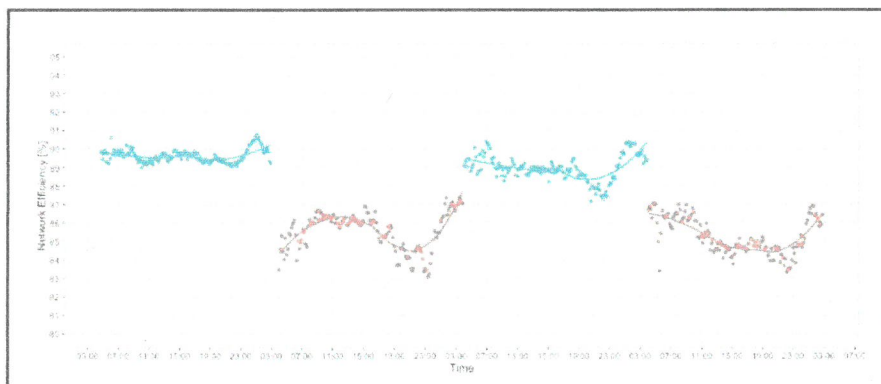


Figure 4.4: Network Efficiency

REFERENCES

- 1] <https://www.sandvine.com/>
- 2] <https://www.geeksforgeeks.org/what-is-transmission-control-protocol-tcp/>
- 3] <https://www.al0networks.com/glossary/what-is-layer-4-of-the-osi-model/>
- 4] https://www.sandvine.com/hubfs/Sandvine_Redesign_2019/Downloads/Datasheets/Sandvine_DS_TCP%20Accelerator%20AW%2020190509.pdf
- 5] <https://www.sandvine.com/products/data-plane>
- 6] <https://networklessons.com/cisco/ccie-routing-switching-written/tcp-header>
- 7] <https://www.geeksforgeeks.org/congestion-control-in-computer-networks/>
- 8] <https://www.geeksforgeeks.org/tcp-tahoe-and-tcp-reno/>
- 9] <https://wiki.geant.org/display/public/EK/SelectiveAcknowledgements>
- 10] <https://www.geeksforgeeks.org/selective-acknowledgments-sack-in-tcp/>
- 11] <https://www.geeksforgeeks.org/tcp-3-way-handshake-process/>
- 12] <https://www.geeksforgeeks.org/calculation-of-tcp-checksum/>
- 13] <https://www.geeksforgeeks.org/introduction-of-b-tree-2/>
- 14] <https://www.geeksforgeeks.org/insert-operation-in-b-tree/>
- 15] <https://www.geeksforgeeks.org/delete-operation-in-b-tree/>
- 16] <https://www.oreilly.com/library/view/hands-on-network-programming/9781789349863/>
- 17] <https://www.geeksforgeeks.org/write-header-file-c/>
- 18] https://www.dartmouth.edu/~rc/classes/soft_dev/make.html
- 19] <https://www.geeksforgeeks.org/how-to-use-make-utility-to-build-c-projects>
- 20] <https://www.geeksforgeeks.org/basic-shell-commands-in-linux/>
- 21] <https://www.simplilearn.com/tutorials/git-tutorial/what-is-git>
- 22] <https://www.infoworld.com/article/3204171/what-is-docker-the-spark-for-the-container-revolution.html>
- 23] <https://www.sussex.ac.uk/its/services/software/owncomputer/putty>
- 24] <https://www.geeksforgeeks.org/tcl-full-form/>
- 25] <https://www.atlassian.com/software/jira/guides/use-cases/what-is-jira-used-for>
- 26] <https://www.kernel.org/doc/html/v4.19/vm/numa.html>
- 27] <https://www.dpdk.org/>
- 28] <https://scapy.net/>

Date: 06 May 2022

Sandvine Technologies (I) Pvt. Ltd.
RMZ Ecoworld SEZ,
Building 1A, Ground Floor,
Devarabeesanahalli, ORR
Bengaluru 560103
India
t: +91 (80) 6774 3333
f: +91 (80) 6774 3232
e: info@sandvine.com
www.sandvine.com

INTERNSHIP CERTIFICATE

TO WHOMSOEVER IT MAY CONCERN

This is to certify that Shivi Sharma has done her internship as Trainee at Sandvine Technologies (I) Pvt Ltd., Bangalore, from 3rd Jan 2022 to 15th May 2022 for the duration of 4.5 months.

She has worked on a project titled TCP-A on Active logic. This project was aimed at Accelerating TCP flows. As part of the project, she has worked on the scrum activities mainly focusing on using data structures to store TCP-flow information, the transmission of flows and segments to the network, and window size update and acknowledging it to the peer and writing Unit tests.

During her internship, she has demonstrated her skills with self-motivation to learn new skills. Her performance exceeded our expectations and she was able to complete the project on time.

We wish her all the best for her upcoming career.



 *Manoj Mallappa*
Manoj Kavithala Mallappa
Manager, R&D
Sandvine Technologies (I) Pvt Ltd