TCP PEP

TCP Performance Enhancing Proxy to Support
Non-interactive Applications

Joe Bayer

Informatics: Programming and System Architecture
60 ECTS study points

Department of Informatics
Faculty of Mathematics and Natural Sciences

Spring 2023

Joe Bayer

TCP PEP

TCP Performance Enhancing Proxy to
Support Non-interactive Applications

Supervisors:
Michael Welzl
Kristjon Ciko

Contents

Chapter 1

Intro

Chapter 2

Background

In this chapter we will present some of the required background knowledge to understand the concepts presented in this paper. Mostly topics that are outside the common understanding of programming and networking are covered, especially details of certain congestion controllers and network protocols will be discussed.

2.1 TCP/IP

(Get inspired by intro of tcp transport converter) Perhaps the most well known internet transport protocol is the Transmission Control Protocol (TCP). It provides reliable and in order delivery of packets using acknowledgements and re-transmissions [5]. However, as the demands of the internet has changed, TCP has not. Even though TCP has changed with minor extensions over the years, such as increased initial windows or new options, the core ideas have stayed the same [2]. Concepts as the end-to-end argument still play a vital role in how TCP is used in the modern internet. Especially congestion control is suffering under the illusion that all logic should be placed on the endpoints, even if it spans multiple different domains (wireless / wired). The topology and demands of a network can vary, especially between wired and wireless domains.

- **Wireless Domain**: A wireless communication domain refers to the transmission of data over a wireless medium without the use of physical connections such as wires or cables between devices. This domain covers a variety of technologies, including 3G, 4G, and 5G for mobile communication, Bluetooth and Wi-Fi for close-range communication, and satellite communication for worldwide communication.

- **Wired Domain**: Unlike a wireless domain, a wired domain provides a steady and reliable bandwidth with low error rates and high throughput. The use of Ethernet and Fiber are typical for wired networks, they enable the transmission of large amount of data over long distances and low signal noise.

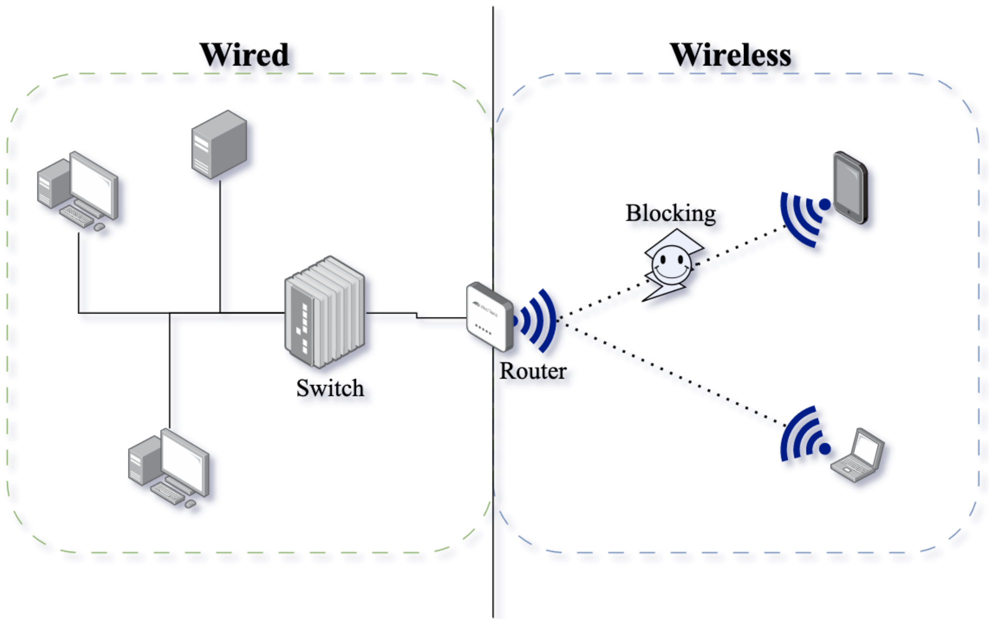

Figure 2.1: 5G bandwidth fluctuations from humans

Each domain has different requirements that a single TCP connection cannot provide. Fig. 2.1 shows the two domains and their differences.

Interactive traffic uses TCP? source End to end argument. Bad? TCP handshake, reduce RTTs but using TCP Fast Open. Mostly short flows (cite) End to End congestion controller not very suited for highly fluctuating bandwidth.(cite David Hayes?)

.slow reactiveness of TCP [7].

2.1.1 Congestion Control

https://datatracker.ietf.org/doc/draft-irtf-iccrg-welzl-congestion-control-open-research/08/

Congestion occurs in the internet when a network's resources, such as routers, are overloaded to the point that they diminish quality of the network. Packet loss and high delays are common issues associated to high congestion in the network. To solve the problem of congestion, a distributed algorithm is used: Congestion Control. The main goal of congestion control is to maintain a stable network, while still utilizing the available bandwidth shared among all flows. This is achieved by additively increasing the sending rate, and multiplicatively reducing the sending rate when detecting congestion. Congestion can be detected by packet loss, changes in delay, but also by explicit notifications.

Explain slow start, etc?

Over time different variations of congestion controller have emerged. Although their goal is the same, reduce congestion in the network, their approaches vary.

Different congestion control algorithms

- **TCP Reno**: Reno embodies the traditional approach to congestion control. Slowly increasing the sending rate while the network is stable and drastically reducing it on packet loss. TCP Reno was designed for unstable and dynamic networks, where the rapid response rate is crucial to prevent network overloading. However, the slow start rate and aggressive reduction of the sending rate make it suboptimal for more stable networks, where packet loss is less frequent and predictable. Consequently, TCP Reno's reliance on packet loss may lead to unnecessary rate reductions and decreased network throughput.

- **New Vegas**: New Vegas is similar to TCP Reno is most aspects, the main difference is the use of delay to detect congestion instead of packet loss. This makes New Vegas able to react faster to congestion, however it also introduces some interesting side effects. If New Vegas competes with TCP Reno flows, it will start reducing its sender rate before TCP Reno does, this leads to New Vegas losing out on possible bandwidth.

- **Cubic**: Cubic improves on the idea of TCP Reno by using a cubic function for to adjust its (congestion window) sending rate in order to achieve higher throughput in a fast manner. Cubic is very efficient in highspeed networks and known for handling large data transfer over long distances. However, Cubic is not as reliable and robust as more traditional congestion controllers like TCP Reno.

In summary, the main differences between TCP Reno, New Vegas and Cubic are their approach to congestion control, their performance in different types of networks, and their trade-off between efficiency and reliability.

2.1.2 3 Way handshake (0 RTT)

For TCP to establish a connection it uses a three-way handshake. Initially, it transmits a synchronization (SYN) packet to the desired endpoint. The endpoint responds with a acknowledgement and a synchronization packet of its own (SYN/ACK). Finally, the client responds with a acknowledgment (ACK). At this point both endpoints have confirmed that they are ready for further communication. For any connection to be established this handshake has to be done. For short flows that terminate in just a few round trips the initial TCP handshake can be a bottleneck, which is made worse if the connection is using a proxy and has to exchange additional information.

2.1.3 TCP Options

A TCP connection can be configured with optional header extensions called TCP Options [3].. Short flows terminating in a few round-trips. Meaning the "bottleneck" is the required initial TCP handshake.
TCP Fast Open allows data being exchanged during the handshake.

Allow "syn fowarding" with TCP Fast Open creating a 0RTT increase when connecting through a proxy. 0RTT Transport Converter [2].

2.2 Future of wireless communication.

The future of wireless communication has seen a lot of improvements such as... ... highly increased bandwidth ... using Millimetre frequency bands ... but at the cost of Highly fluctuating bandwidth with wireless networks, especially with higher frequencies.

2.2.1 5G Millimetre Wave

The emergence of 5G Millimeter wave communications has opened the doors for low latency networks with multiple gigabit bandwidth. This is achieved by using higher millimetre wave (mmWave) frequencies in the range of 30GHz to 300GHz, which as a lot of benefits. [1] A wider spectrum of frequencies to choose from and higher data transfer rates are just some of the many benefits mmWave provides. But along side the benefits, mmWave has also introduced a lot of new challenges.

A big problem with millimetre wave communication is signal path blocking also called "Line of sight blocking" [8]. Its caused by the use of Beam-forming to increase the bandwidth and range of milimeter wave signals. Beam-forming focuses the signal in a certain direction making any blocking of the signal path devastating for the bandwidth. Even the human body can create enough blockage to drastically reduce the bandwidth. This causes huge fluctuations in the bandwidth whenever the signal is blocked. PICTURE?

Fluctuating bandwidths lead to unstable TCP connections with a worst case of losing packets. Current TCP congestion controllers such as CUBIC, New Reno or New Vegas struggle when reacting to sudden fluctuating changes. They are simply not able to utilize the high bandwidth when it is available. Simply increasing the aggressiveness of a congestion controller is

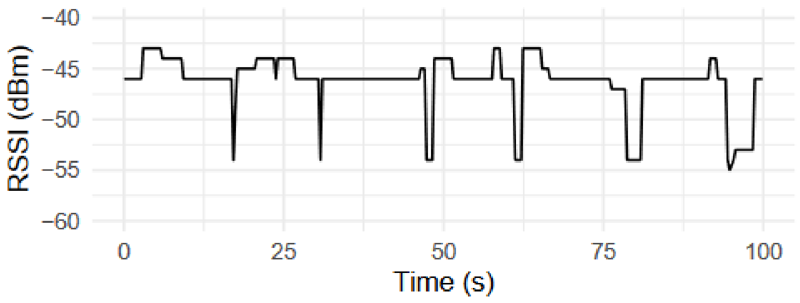

Figure 2.2: 5G bandwidth fluctuations from humans

not a option either as it would disrupt the internet and not be TCP-friendly. A possible solution could be to buffer packets at the 5G (base) stations, having the data ready for when the bandwidth is high, this however creates a new problem bufferbloat.

2.2.2 Buffering

Buffer bloat

The buffer bloat problem occurs when the systems between the endpoints buffer so many packets that the latency drastically increases and the reliability of the network as a whole goes down.https://lwn.net/Articles/507065/ The increased latency is detrimental for interactive (latency sensitive) applications. Generally its preferred to drop packets and keep buffers small to avoid buffering time sensitive packets such as synchronization packets. Although this works in most cases, its far from a optimal solution.

The increased bandwidth and low latency promises of new technology such as 5G has but a lot of pressure on the efficient forwarding of packets. Small buffers are therefore the standard, But at the same time, fluctuating bandwidth has shown the potential need to buffer packets for non-interactive traffic.

Most focus has been on (helping? Supporting?) latency sensitive applications like virtual reality or remote surgery to name a few. This thesis will explore non-interactive applications where latency is not that critical and more buffering is acceptable and most likely desirable. By splitting traffic into interactive and non-interactive we can improve the performance of both. By having very small buffers for interactive applications we avoid bufferbloat problems, while utilizing the benefits of big buffers for non-interactive applications.

Packet Scheduling

A method of reducing the effects off bufferbloat is packet scheduling. A system should not send more packets than the weakest link can handle [?], this idea is built into TCP in the form of congestion control. However, when buffers grow to the point of causing bufferbloat, TCPs congestion control algorithms are unable to confidently determine a sending rate. Packet scheduling can solve this problem as it usually controls the size of the buffers. It makes sure queues can grow when needed, but keep the overall state of the buffers low. Packet scheduling has a lot more to offer than simple queue management, this will be explored later?

Proposed packet scheduling algorithms:

- **FQ CoDel**: The Flow Queue Controlled Delay algorithm, FQ CoDel for short, was partly developed to deal with the bufferbloat problem. Its main goal is to reduce the impact of head-of-line blocking and give a fair share of bandwidth by mixing packets from multiple flows [6]. Internally FQ CoDel uses a FIFO queue, classifying packets into different flows to provide a fair share of bandwidth.
- **HTB**: Hierarchical Token Bucket is a queuing discipline based on assgining different classes a certain amount of bandwidth and sending rate. Because of its extensive bandwidth and delay mangement its a good option for testing, espeically in a virtual environment.

2.2.3 Non-Interactive Applications

Non-Interactive applications such as Web traffic, File transfers and Videos? can benefit from larger buffering, especially with fluctuating bandwidths. Being able to have packets buffered for when the bandwidth is high will decrease delay times. (need citation or prove it myself?). At the same time, interactive applications will not suffer under large queue delays.

2.3 PEPs

A performance enhancing proxy (PEP) is a (connection splitting) proxy designed to increase performance of applications using it. The idea behind the PEP is putting more logic inside the network,

2.3.1 PEP for wireless communication

Performance enhancing proxies are already deployed and in use for a lot of wireless communication, especially satellites and radio access networks [7]. They have an inherent performance increase just by splitting the connection between the wireless and wired domains. These PEPs are therefor often installed at the base stations. However they are unable to distinguish between interactive or non-interactive traffic, meaning their buffers need to be low and still suffer from fluctuating bandwidth problems. Already in use! (satelites, radio access networks, cite). Inherently increases performance? [7].

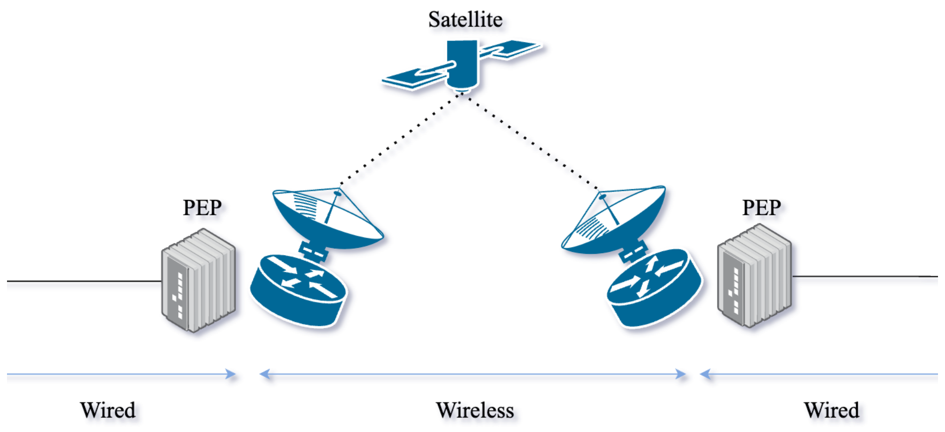

Figure 2.3: PEP installed to support Wireless traffic over satellite.

2.3.2 Transparent vs Non-Transparent

A big discussion regarding PEPs has been if they should be transparent or non-transparent. Transparent PEPs are not visible to the applications that use it. They silently split the connections and spoof the IP-address of both the client and server [4]. This is prone to cause uninteded side effects, such as certain TCP options not being forwarded and security concerns. Non-Transparent PEPs on the other hand are explicitly chosen by either the client or the server, and the sender is aware of the proxy splitting the original connection. This approach can be seen as more ethical and "correct" (FIND BETTER WORDING), but modifications at the sender side become necessary to utilize the PEP.

2.4 Linux

2.4.1 Kernel Modules

LKM (Loadable Kernel Modules), "program" running inside the Linux kernel. Userspace vs Kernel. Reduce system call overhead.

2.4.2 System Calls

Reduce over head from userspace -> kernel system calls.

Chapter 3

Implementation | Design

```c
int pep_tcp_receive(struct socket *sock, u8* buffer, u32 size)
{
  struct msghdr msg = {
    .msg_flags = MSG_DONTWAIT,
  };

  struct kvec vec;
  int rc = 0;

  vec.iov_base = buffer;
  vec.iov_len  = size;

  printk(KERN_INFO "[PEP] kernel_recvmsg: calling recvmsg \n");
pep_tcp_receive_read_again:
  rc = kernel_recvmsg(sock, &msg, &vec, 1, vec.iov_len,
   MSG_DONTWAIT);
  if (rc > 0)
  {
    tlv_print(buffer);
    printk(KERN_INFO "[PEP] kernel_recvmsg: recvmsg returned %d
   \n", rc);
    return rc;
  }

  if(rc == -EAGAIN || rc == -ERESTARTSYS)
  {
    goto pep_tcp_receive_read_again;
  }

  printk(KERN_INFO "[PEP] kernel_recvmsg: recvmsg returned %d\n
   ", rc);
  return rc;
}
```

Listing 3.1: kernel_recvmsg wrapper for receiving for TCP msgs

3.1 TLVs

3.2 PEP

3.3 PEP Connect

Since are PEP is explicitly addressed we need to change how we connect on the client side. The default approach is to use the **connect** function, our goal is to imitate this functions behavior, return code and parameters. By keeping our connect function as similar to the original we reduce the overhead of switching over to it. The only change to the original connect that we need is to specify type of traffic, interactive or non-interactive.

```
int socket, ret;
struct sockaddr_in s_in;
bzero((char *)&s_in, sizeof(s_in));
s_in.sin_family = AF_INET;
s_in.sin_addr.s_addr = inet_addr(IP);
s_in.sin_port = htons(PORT);

socket = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);

ret = connect(server, (struct sockaddr*) &s_in, sizeof(s_in));
ret = pep_connect(server, (struct sockaddr*) &s_in, sizeof(s_in
    ), PEP_INTERACTIVE);
```

Listing 3.2: PEP connection

Table of design decisions based on different PEP implementations compared to ours. 0RTT, Transparent, Using TLV, Special ACKS, connection splitting.

PEP List

Implementation	0RTT	Connection Splitting	Special ACKs	Transparent
milliProxy	AF	AFG	004	x
PEPDNA	AX	ALA	248	x
SnoopTCP	AL	ALB	008	x
Our PEP	DZ	DZA	012	x
Transport Converter	AS	ASM	016	x
...	AD	AND	020	x
...	AO	AGO	024	x

Chapter 4

Evaluation

Chapter 5

Conclusion

Bibliography

[1] S. K. Agrawal and Kapil Sharma. 5g millimeter wave (mmwave) communications. In *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, page 3630–3634, Mar 2016.

[2] Olivier Bonaventure, Mohamed Boucadair, Sri Gundavelli, SungHoon Seo, and Benjamin Hesmans. 0-RTT TCP Convert Protocol. RFC 8803, July 2020.

[3] David Borman. TCP Options and Maximum Segment Size (MSS). RFC 6691, July 2012.

[4] Kristjon Ciko, Michael Welzl, and Peyman Teymoori. Pep-dna: A performance enhancing proxy for deploying network architectures. In *2021 IEEE 29th International Conference on Network Protocols (ICNP)*, pages 1–6, 2021.

[5] Wesley Eddy. Transmission control protocol (tcp). Request for Comments RFC 9293, Internet Engineering Task Force, Aug 2022.

[6] Toke Høiland-Jørgensen, Paul McKenney, dave.taht@gmail.com, Jim Gettys, and Eric Dumazet. *The Flow Queue CoDel Packet Scheduler and Active Queue Management Algorithm.* Number RFC 8290. Jan 2018.

[7] Michele Polese, Marco Mezzavilla, Menglei Zhang, Jing Zhu, Sundeep Rangan, Shivendra Panwar, and Michele Zorzi. milliproxy: A tcp proxy architecture for 5g mmwave cellular systems. In *2017 51st Asilomar Conference on Signals, Systems, and Computers*, pages 951–957, 2017.

[8] Cristian García Ruiz, Antonio Pascual-Iserte, and Olga Muñoz. Analysis of blocking in mmwave cellular systems: Application to relay positioning. *IEEE Transactions on Communications*, 69(2):1329–1342, Feb 2021.