Master's thesis

TCP PEP

Extension of a 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

Extension of a TCP Performance
Enhancing Proxy to Support
Non-interactive Applications

Supervisors:
Michael Welzl
Kristjon Ciko

Contents

Chapter 1

Intro

Chapter 2

Background

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 has 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).

A networks topology and demands will vary a lot, especially between wired and wireless domains:

- **Wireless Domain**: Fluctuating bandwidth, high error rate, blocking, radio signals.
- **Wired Domain**: Steady bandwidth, low error rate, high throughput, Ethernet / fiber.

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 [6].

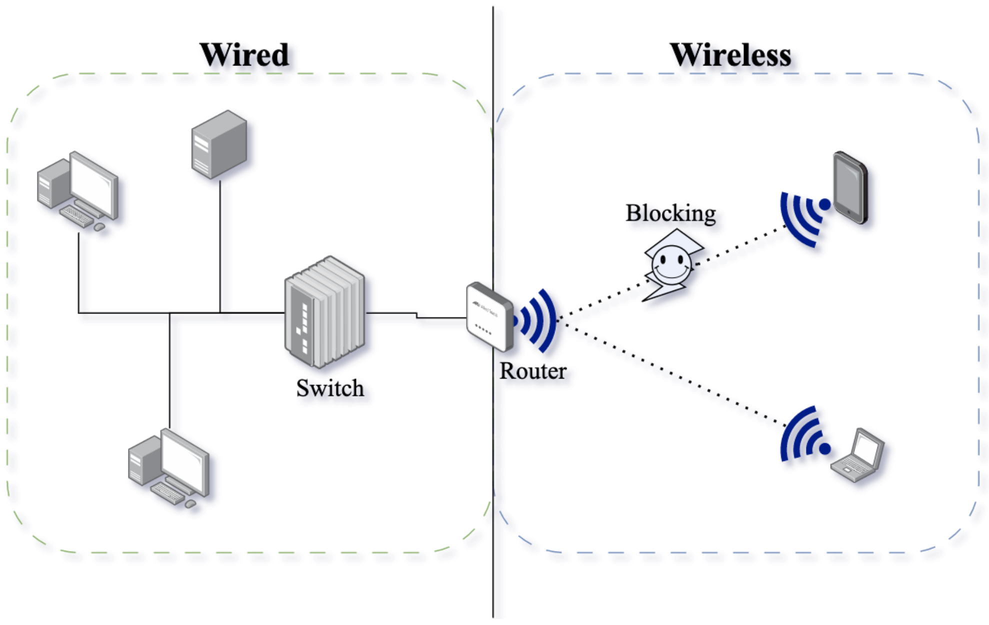

Figure 2.1: 5G bandwidth fluctuations from humans

2.1.1 3 Way handshake (0 RTT)

For TCP to establish a connection it uses a three-way handshake. First it sends a synchronization (SYN) packet to the desired endpoint. The endpoint answers with a acknowledgement and a synchronization packet of its own (SYN/ACK). And 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.

2.1.2 Congestion Control

Different congestion control algorithms

- **Cubic**: ...
- **Vegas**: ...
- **...**: ...

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

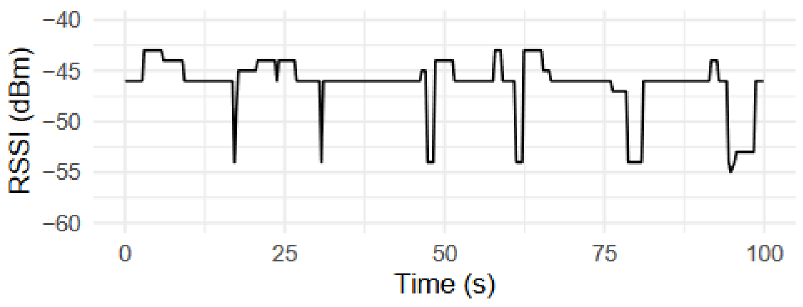

Figure 2.2: 5G bandwidth fluctuations from humans

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" [7]. Its caused by the use of Beamforming 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 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 https://lwn.net/Articles/496509/, 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**: ...
- **Placeholder**: ...

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. More logic inside the networks. Domain splitting and 0RTT.

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 [6]. 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? [6].

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].. (SIDE EFFECTS). 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.

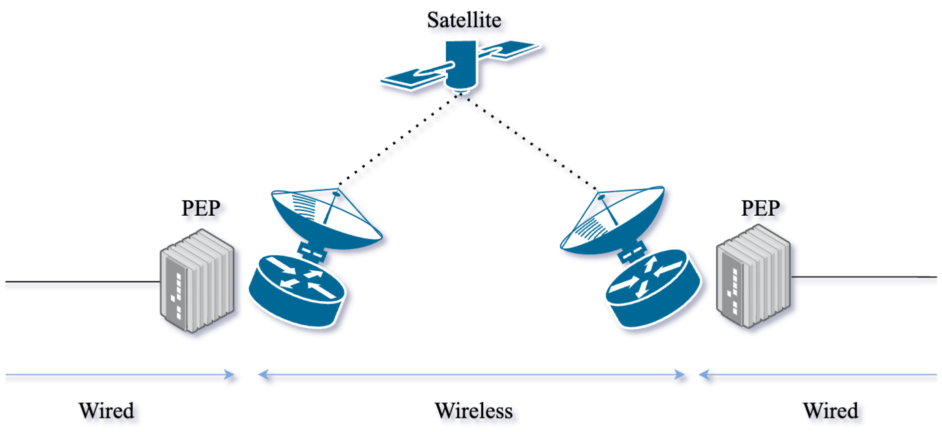

Figure 2.3: PEP installed to support Wireless traffic over satellite.

2.4 Kernel Modules

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

2.4.1 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 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.

```c
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] 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.

[7] 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.