MPTCP in the upstream kernel

A long road that started almost 15 years ago

5th of February 2023
Agenda

1. **MultiPath TCP**
   Introduction and use cases

2. **What can we do today?**
   And what will we be able to do tomorrow?

3. **The long road to have MPTCP upstream**
   15 years: protocol definition, experimentations, rewriting from scratch
MultiPath TCP
What is it MPTCP?

• Extension to TCP, defined in RFC 8684
• One TCP session is no longer tight to a fixed pair of IP/ports
• Exchange data for a single connection over different paths, simultaneously
Smartphone use-case (Apple iOS - Android in South Korea)
Smartphone use-case (Apple iOS - Android in South Korea)

Without MPTCP

Smartphone and WiFi icons by Blurred203 and Antü Plasma under CC-by-sa, others from Tango project, public domain
MultiPath TCP
Typical use cases: Smartphone use-case

Smartphone use-case (Apple iOS - Android in South Korea)

Without MPTCP

Smartphone and WiFi icons by Blurred203 and Antü Plasma under CC-by-sa, others from Tango project, public domain
Smartphone use-case (Apple iOS - Android in South Korea)

*Without MPTCP*

Smartphone and WiFi icons by Blurred203 and Antü Plasma under CC-by-sa, others from Tango project, public domain
MultiPath TCP
Typical use cases: Smartphone use-case

Smartphone use-case (Apple iOS - Android in South Korea)

Without MPTCP

Smartphone and WiFi icons by Blurred203 and Antü Plasma under CC-by-sa, others from Tango project, public domain
Smartphone use-case (Apple iOS - Android in South Korea)

With MPTCP
MultiPath TCP
Typical use cases: Smartphone use-case

Smartphone use-case (Apple iOS - Android in South Korea)

With MPTCP
MultiPath TCP
Typical use cases: Smartphone use-case

Smartphone use-case (Apple iOS - Android in South Korea)

With MPTCP

Smartphone and WiFi icons by Blurred203 and Antü Plasma under CC-by-sa, others from Tango project, public domain
MultiPath TCP
Typical use cases: Hybrid access network
MultiPath TCP
Typical use cases: 5G (ATSSS)

Steering

5G OR WiFi

best network selection

Switching

FROM 5G TO WiFi

seamless handover

and vice versa

Splitting

5G AND WiFi

network aggregation

Defined in 3GPP Release 16, ATSSS is a core network function in 5G networks, playing a key role in managing data traffic between 3GPP (5G, 4G) networks and non-3GPP (Wi-Fi) networks.

improved end-user experience

Logos from 3GPP and Wi-Fi Alliance
1. **MultiPath TCP**
   - Introduction and use cases

2. **What can we do today?**
   - And what will we be able to do tomorrow?

3. **The long road to have MPTCP upstream**
   - 15 years: protocol definition, experimentations, rewriting from scratch
What can we do today?

How to use it?

• Get a recent enough kernel:

>= 5.6 but last stable version is recommended, see ChangeLog
What can we do today?
How to use it?

• Get a recent enough kernel:

>= 5.6 but last stable version is recommended, see [Changelog](#)

• Create an MPTCP socket:

```c
socket(AF_INET(6), SOCK_STREAM, IPPROTO_MPTCP);
```

Or use `mptcpize`: LD_PRELOAD to force creating MPTCP socket
What can we do today?

How to use it?

• Get a recent enough kernel:

  >= 5.6 but last stable version is recommended, see ChangeLog

• Create an MPTCP socket:

  `socket(AF_INET(6), SOCK_STREAM, IPPROTO_MPTCP);`

  Or use `mptcpcize`: LD_PRELOAD to force creating MPTCP socket

• Configure the network:

  With: NetworkManager 1.40+ or mptcpd or ip mptcp + ip route
What can we do today?
How to use it? Example

• Get a recent GNU/Linux distribution

• Manual network configuration for additional IPs (or use NM)
  - `sudo ip mptcp endpoint add <IP> dev <iface>`
  - `sudo ip rule add from <IP> table 42`
  - `sudo ip route add default via <next hop> table 42`

• Run your app:
  - `mptcpize iperf3 --client|--server`
What can we do today?

Features

• Most protocol features are supported: multiple subflows, announce addresses and priority, fast close, etc.

• Many socket options are supported: **SO, IP, TCP**

• Info from MIB counters, **INET_DIAG** interface and **MPTCP_INFO**

• 2 Path Managers and 1 Packet scheduler
MultiPath TCP
Concept: Path Manager: global vs per connection

Which path to create/remove? Which address to announce?

Smartphone and WiFi icons by Blurred203 and Antü Plasma under CC-by-sa, others from Tango project, public domain
On which available path packets will be sent? Reinject packets to another path?
What will we have?
Extracted from the wish list

- eBPF Packet scheduler:
  - Ending up changing the scheduler and its API
What will we have?

Extracted from the wish list

- eBPF Packet scheduler:
  - **Ending up** changing the scheduler and its API
- More socket options:
  - `[gs]etsockopt(...)`
What will we have?
Extracted from the wish list

- eBPF Packet scheduler:
  - **Ending up** changing the scheduler and its API
- More socket options:
  - `[gs]etsockopt(...)`
- Golang support:
  - No compatible with mptcpize (LD_PRELOAD)
  - net package doesn’t allow selecting another protocol
What will we have?

Extracted from the wish list

- eBPF Packet scheduler:
  - Ending up changing the scheduler and its API

- More socket options:
  - [gs]etsockopt(...)  

- Golang support:
  - Eventually used MPTCP by default instead of TCP?
  - Accepted proposition 🥂: github.com/golang/go/issues/56539
1. **MultiPath TCP**
   - Introduction and use cases

2. **What can we do today?**
   - And what will we be able to do tomorrow?

3. **The long road to have MPTCP upstream**
   - 15 years: protocol definition, experimentations, rewriting from scratch
Road to have MPTCP upstream
The beginning

- Project started ~15 years ago at UCLouvain 🇧🇪🍟🍻
• Project started ~15 years ago at UCLouvain
Road to have MPTCP upstream

The beginning

- Project started ~15 years ago at UCLouvain
- As fork:
  - For experimentations, validating the concept
  - Initial author: Sébastien Barré
  - Moving to “production ready”: Christoph Paasch, Gregory Detal
• Project started ~15 years ago at UCLouvain
• As fork:
  — For experimentations, validating the concept
  — Initial author: Sébastien Barré
  — Moving to “production ready”: Christoph Paasch, Gregory Detal
• MPTCPv0 RFC published in January 2013
• Used in production on servers having millions of clients
Road to have MPTCP upstream
Maintaining a fork

- It is easy to fork …
Road to have MPTCP upstream
Maintaining a fork

• It is easy to fork …
  – but you will pay for it!
Road to have MPTCP upstream

Maintaining a fork

- It is easy to fork …
  - but you will pay for it!
Road to have MPTCP upstream
Maintaining a fork: different levels

• It is easy to fork …
  – but you will pay for it!

• The Linux kernel is big, complex, very active
Road to have MPTCP upstream
Maintaining a fork: different levels

• It is easy to fork …
  – but you will pay for it!

• The Linux kernel is big, complex, very active

• The fork is quite invasive:
  – 21k lines in total
  – 2.5k in TCP / IP / … with many “if (mptcp)”
  – With duplicated functions adapted for MPTCP case
Road to have MPTCP upstream
Maintaining a fork: nightmare mode

• Now imagine you have to deploy it on various embedded systems, with different LTS kernels, from very old versions (v3.4)
• Backports and conflicts 🎉
• Git’s rerere and TopGit to the rescue:
  – Cherry-Pick once, propagate
  – Resolve conflicts once
Road to have MPTCP upstream
Still used today

- Most MPTCP deployments today are still using this fork:
  - Millions of devices in different types of deployments
  - New releases done 2 days ago (kernels v4.14, v4.19, v5.4)
  - Probably (one of) the last releases!
Road to have MPTCP upstream
Still used today

• Most MPTCP deployments today are still using this fork:
  – Millions of devices in different types of deployments
  – New releases done 2 days ago (kernels v4.14, v4.19, v5.4)
  – Probably (one of) the last releases!

• MPTCP support in the upstream kernel started in 2020 (v5.6)
  – Why a so long delay?
  – Not a new idea: discussions and attempts in 2010 & 2015
Road to have MPTCP upstream
Upstreaming: requirements

• Linux TCP is highly optimized
Road to have MPTCP upstream
Upstreaming: requirements

• Linux TCP is highly optimized

• New implementation cannot affect existing TCP stack:
  – No performance regressions
  – Maintainable and possibility to disable it
  – Can be extended via the userspace
Road to have MPTCP upstream

Upstreaming: requirements

• Linux TCP is highly optimized

• New implementation cannot affect existing TCP stack:
  – No performance regressions
  – Maintainable and possibility to disable it
  – Can be extended via the userspace

• Cannot take the initial fork:
  – Built to support experiments and rapid changes but not generic enough
  – Special purpose implementation of MPTCP
Road to have MPTCP upstream
Upstreaming: solutions

- Rewriting (almost) from scratch
- A different concept: introduction of MPTCP socket
- Minimal differences in TCP code thanks to TCP ULP (+ SKB ext)
- Carefully review and detail modifications in TCP stack
- APIs to extend the path-manager and the scheduler
- And …
Road to have MPTCP upstream
Upstreaming: solutions

• A lot of work!
  – Special thanks to Mat Martineau and other fellows at Intel (Peter, Ossama, Kishen, Todd)
  – RedHat (Paolo, Florian, Davide, etc.), SUSE (Geliang), Apple (Christoph), Tessares (Benjamin, myself), and more (Dmytro, Menglong, Poorva, Yonglong, Nicolas, Netdev maintainers, etc.)
Conclusion

A long road… and it is not over!

Questions? Discussions?

Mailist list is open!
One public conf call per week!

mptcp.dev
tessares.net

@mattbe@fosstodon.org
@mptcp@social.kernel.org
matthieu.baerts@tessares.net
Backup slides
Road to have MPTCP upstream
Development: contributors are welcome

• Virtme is great to start working in the kernel

• Build, run, test with a one-line command:

```bash
cd <kernel source code>
docker run -v "${PWD}:${PWD}:rw" -w "${PWD}" --privileged --rm -it \   --pull always mptcp/mptcp-upstream-virtme-docker:latest \<entrypoint options>
```
MPTCP fork
Relations between structures

A special TCP socket (meta) is used to interact with the apps and the subflows