MPTCP
Extending kernel functionality with eBPF and Netlink

Matthieu Baerts (Tessares)
Mat Martineau (Intel)
https://mptcp.dev
Agenda

MPTCP: Extending kernel functionality with eBPF and Netlink

MultiPath TCP
Netlink extension
BPF extension
Conclusion
MultiPath TCP

What is it?
Exchange data for a single connection over different paths, simultaneously
MultiPath TCP

Part of 5G spec (ATSSS)

- **Steering**
  - 5G OR WiFi
  - best network selection

- **Switching**
  - FROM 5G TO WiFi
  - seamless handover

- **Splitting**
  - 5G AND WiFi
  - network aggregation

**Linux Plumbers Conference | Dublin, Ireland Sept. 12-14, 2022**
MultiPath TCP

What do we have today?

- A dedicated socket: `socket(AF_INET(6), SOCK_STREAM, IPPROTO_MPTCP);`
- Minimal differences in TCP code thanks to TCP ULP (+ skb ext)
- Supports most of the protocol features: multiple subflows, announce addresses and priority, fast close, RST reasons
- Supports many socket options: SO, IP, TCP
- Info from MIB counters, INET_DIAG interface and MPTCP_INFO
- 2 Path Managers and 1 Packet scheduler (see later)
MultiPath TCP

What will we have?

- More control from userspace
- TCP FastOpen (TFO) support for MPTCP
- More socket options
- Maybe TCP CC taking into account multiple paths?
- Hopefully more!
Netlink extension
Netlink extension

Concept: Path Manager

Which path to create/remove? Which address to announce?
Global or per Connection

- Global settings: per netns, e.g. via ip mptcp
  - Set endpoints: addresses, flags
  - Set limits: max subflows to establish or accept
  - Monitor connections: created, established, closed, announced, etc.

- Per connection: via mptcpd
  - Reacting to “events” by sending “commands”
Netlink extension

Pros & Cons

+ Netlink is well known, well tested, clear and stable
+ Only one tweak needed to restrict event access to groups
- Not designed for kernel ⇔ userspace requests (e.g. policy)
- Issues when there are too many events:
  - Losses: tweak the buffer size?
  - Latency: more channels?
BPF extension
BPF extension

Concept: Packet Scheduler

On which available path packets will be sent? Reinject packets to another path?
BPF extension

Packets Scheduler: Similar case

- TCP Congestion Control
  - Initially fully integrated in TCP code
  - With conditions to support multiple algorithms
  - 2005: “pluggable” via kernel modules
  - 2020: BPF STRUCT_OPS with Cubic and DCTCP reimplementation
    - The BPF TCP CCs look like existing TCP CCs kernel modules
BPF extension

BPF to the rescue

- "Pluggable" via a new kernel module
- New BPF STRUCT_OPS: BPF_STRUCT_OPS_TYPE(mptcp_sched_ops)

```c
SEC(".struct_ops")
struct mptcp_sched_ops my_sched = {
  .init = (void *)my_sched_init,
  .release = (void *)my_sched_release,
  .get_subflow = (void *)my_sched_get_subflow,
  .name = "my_bpf_sched",
};
```
BPF extension

Consequences and Questions

- Performances: Indirect calls, flexibility vs optimisation
- API stability: is it considered as “exposed to userspace”? 
- Some structures need to be modified but:
  - How to deal with atomic operations? (BPF helpers?)
  - Security concerns: accessing the connection token?
  - Any security guidelines?
Conclusion

Hopefully MPTCP will be adapted to everyone’s needs!

Questions? Discussions?

Matthieu Baerts (Tessares)
Mat Martineau (Intel)
https://mptcp.dev