Using Upstream MPTCP in Linux Systems

Mat Martineau and Ossama Othman

August 21, 2020

Netdev 0x14

Disclaimer

Linux® is a registered trademark of Linux Torvalds in the U.S. and other countries.

Wi-Fi® is a registered trademark of the Wi-Fi Alliance.

Other trademarks and trade names are those of their respective owners.

Introduction

Multipath TCP is Now Upstream!

Over 150 commits as of July 2020

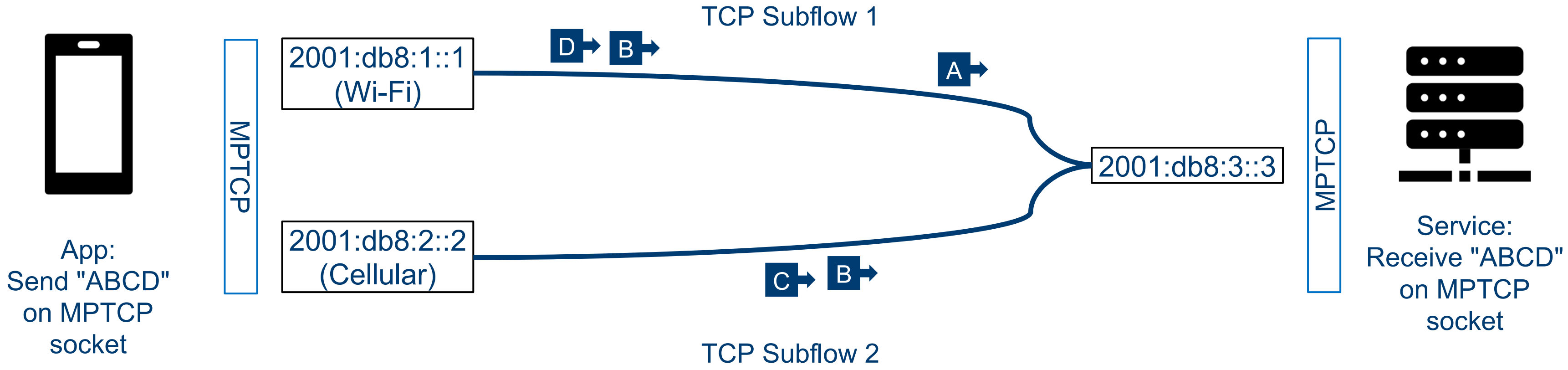
Linux v5.8 MPTCP Features

- Establish MPTCPv1 connections
- Create multiple subflows
- In-kernel path management
- inet_diag support

Collaborative Effort

- Contributors from Red Hat, Tessares, Apple, and Intel.
- Community is growing
- Shortcut to GitHub project
  - https://is.gd/mptcp_upstream

What is Multipath TCP?



What is Multipath TCP?

- A protocol layer above (and intermingled with) TCP
- Starts out looking similar to a normal TCP connection
  - Extra TCP options during the handshake allow multiple paths later
  - This is the initial "subflow"
- Peers share information about additional available IP addresses using TCP options
- Additional TCP subflows can be "joined" to the logical connection

What is Multipath TCP?

- Sender chooses to send data on one or more subflows
  - The subflow streams may carry out-of-order or redundant data
- TCP options carry an additional layer of sequence numbers for data packets
  - Maps a range of TCP sequence numbers to the MPTCP sequence space
  - Includes a MPTCP-level ACK
- Receiver reassembles the MPTCP-level data stream and acknowledges

MPTCP Use Cases

Steering

- Use the best network
- Move data flow between a low latency or high bandwidth link

Switching

- Seamless handover between mobile and Wi-Fi networks

Splitting

- Use combined bandwidth of multiple interfaces
- Hybrid Access: DSL+LTE

MPTCP is used in the 5G Access Traffic Steering, Switching, and Splitting (ATSSS) standard for all these purposes

Versions and Specifications

- Experimental RFC 6824: MPTCP v0
  - Initial draft May 2009, published January 2013
  - In use by most current deployments
- RFC 8684: MPTCP v1
  - Initial draft October 2013, published March 2020
  - Upstream Linux only supports MPTCPv1
  - Addresses issues seen in v0 deployments
  - Not reverse-compatible

Key changes between MPTCP v0 and v1

- Different (and incompatible) connection handshake
  - Change made to better support TCP Fast Open
  - A connection will proceed as regular TCP if the listener does not support the requested MPTCP version
- Moves to SHA256
- Reliable exchange of additional addresses
- MPTCP-level fast close using TCP RST

MPTCP Support in Linux Releases

- 5.6
  - Single subflow
- 5.7
  - Multiple subflow
  - In-kernel path management and related generic netlink interface
  - inet_diag support
- 5.8
  - Improved performance and reliability
  - Better receive window handling

Upstreaming Lessons

MPTCP's Upstreaming Journey

What can similar projects learn from our experience?

- Project characteristics:
  - Significant new functionality that doesn't fit in the drivers/staging tree
  - Close coupling with or modifications to critical existing kernel code
  - Multiple organizations involved
- Upstreaming paradox
  - Maintainers need patch sets of reviewable size
  - Hard to propose an initial patch set without investing a lot of work up front

Cautionary Tales

- Trying to upstream a new framework first did not work out
  - Extensible TCP options framework and TCP-MD5 refactor was rejected
- Don't spend a lot of time guessing what maintainers do and don't want
  - RFC patch sets can help a lot
- Avoid spending a lot of time churning with prototype code
  - Make sure you're getting value out of prototype work

What We Recommend

Building your upstream community

- Make your project known early! Reach out on mailing lists and propose a conference talk. You may be surprised at who gets involved or cheers you on.
- Build a team that includes experts for the new feature, experts on the existing code, communicators, and automation builders.
- Weekly meetings helped strengthen community and accountability
- Face-to-face meetings, even once or twice a year, are valuable

What We Recommend

Various tips

- 'topgit' is handy for revising patch sets and rebasing on the upstream tree
- Have a variety of ways to coordinate: Mailing list, IRC, issue tracker
- CI running kselftests, syzkaller, and other checks has been extremely valuable

What We Recommend

Patch set partitioning

1. Upstream any independent building blocks
   - See: skb_ext functionality in skbuff.h
2. Send any prerequisite changes to existing code
   - Changes to TCP and the networking core
3. Foundational code
   - Single subflow MPTCP
4. Meaningful baseline functionality
   - Multiple subflows!

Note: Keep each patch set to a maximum of 12-20 patches of reasonable size

Using MPTCP

Kernel Build-time Configuration

- Main options
  - CONFIG_MPTCP
  - CONFIG_MPTCP_IPV6 is optional
    - Note: not compatible with CONFIG_IPV6=m
- MPTCP self test support
  - CONFIG_VETH
  - CONFIG_NET_SCH_NETEM

Using an MPTCP Socket

MPTCP is selected when creating the socket

```
    socket(AF_INET6, SOCK_STREAM, IPPROTO_MPTCP)
```

- After the socket is created, use connect/bind/listen/accept and send/recv functions as you would for TCP.
- Differences from TCP
  - Advanced features like zerocopy are not supported (yet?)
  - Socket options may require attention

Socket Options

- Supporting TCP options on an MPTCP connection is complex
  - Option settings for TCP subflows might interfere with MPTCP operation
  - Subflows may be added or removed over life of a MPTCP socket
- MPTCP sockets do not currently support TCP socket options
  - Exception: Connections in "TCP fallback" do have TCP socket option support
- Linux v5.9 will handle SO_REUSEPORT and SO_REUSEADDR
- Planning for advanced MPTCP control via socket options in future kernels

System-level Runtime Configuration

- Per-network-namespace sysctl: net.mptcp.enabled (on by default)
- Default behavior: Additional subflows are not initiated or accepted
- Using multiple subflows requires configuration from userspace
- Long-term: Userspace path management with mptcpd or similar
- Today: 'ip mptcp' command
  - Version iproute2-ss200602 or later
  - Commands set systemwide MPTCP behavior

System-level Runtime Configuration

Using the 'ip mptcp' command

- Allow peers to add new subflows

  -
    ```
    sudo ip mptcp limits set subflow 4
    ```
  - This would allow four additional subflows to join each MPTCP connection if requested by the peer
- Example use case: MPTCP-capable server communicating with mobile device peers. The mobile devices initiate connections and subflows over Wi-Fi and cellular, with NAT on one or both interfaces.

System-level Runtime Configuration

Using the 'ip mptcp' command

- Initiate new subflows
  - `sudo ip mptcp limits set subflow 2`
  - `sudo ip mptcp endpoint add 192.0.2.10 subflow`
  - Each existing MPTCP connection will try to create an additional subflow with 192.0.2.10 as the source address. The destination address will be the one used for the initial connection.
- Example use case: MPTCP-capable device connecting to a server with a public IP address.

Userspace Path Management

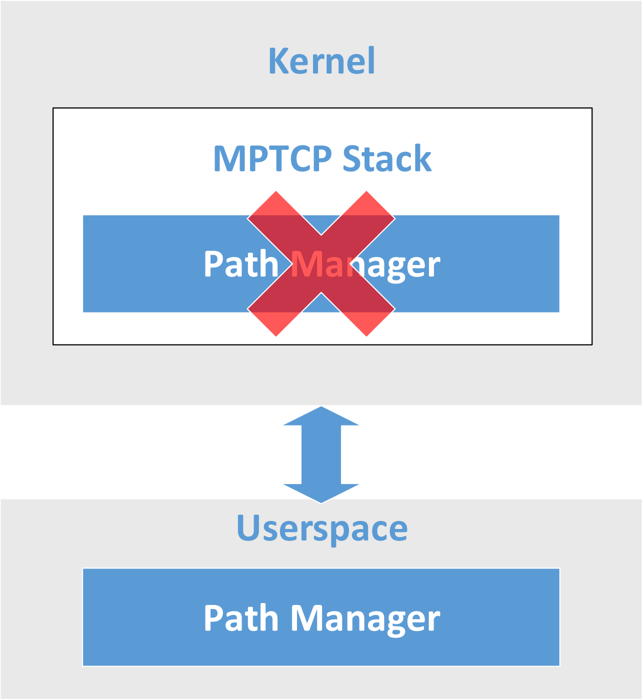
MPTCP Path Management In Userspace

Advantages

- Platform integration
  - Mobile platforms
  - Carrier integration
  - Persistent database of good and bad endpoints
- Simpler kernel-side code
  - Network interface and address tracking in userspace
- Per-application policy integration
- Per-connection path management

Disadvantages (server side)

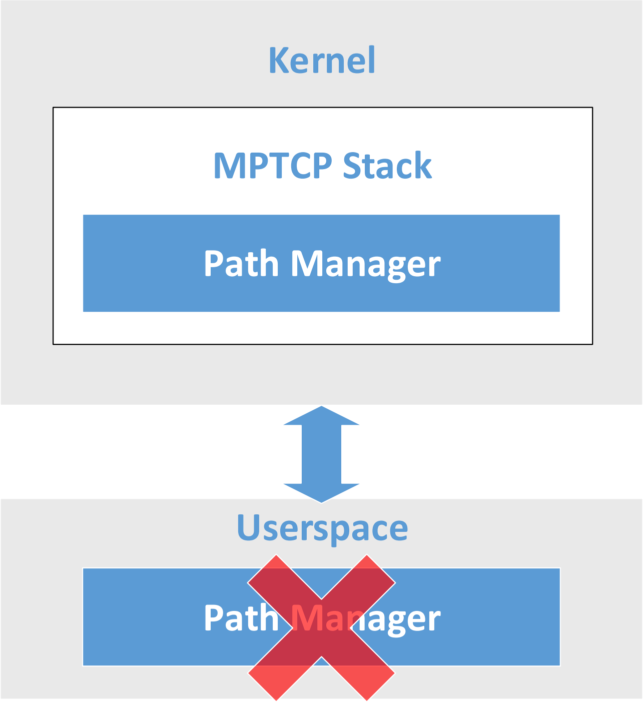
- Bottleneck under heavy connection load



MPTCP Path Management In-Kernel

In-kernel path management may be more suitable for server side

- Advantages
  - Generally improved performance due to less overhead
- Disadvantages
  - Kernel module based path management is more complex
    - Increased maintenance burden, e.g. tied to a specific kernel version
    - Bugs generally have a greater impact on system stability
  - Global path management configuration



MPTCP Generic Netlink API

Events triggered during specific MPTCP connection operations

- New connection, connection closed, new subflow, etc
- Userspace handles path management events as needed

Commands may be issued from userspace to alter connection

- Announce new addresses, create subflows, change priority, etc

API found in include/uapi/linux/mptcp.h

- Could be used by NetworkManager, wicd, ConnMan, or others

Multipath TCP Daemon – mptcpd

Reference implementation for userspace MPTCP path management

- Extensible MPTCP path management framework
- Network interface and address monitoring
- Not intended to replace existing network managers like NetworkManager
- Leverages MPTCP path management generic netlink API
  - Dispatches events to path management plugins
  - Exposes an API that plugins may use to send commands to the kernel
- https://github.com/intel/mptcpd/

Multipath TCP Daemon – Plugins

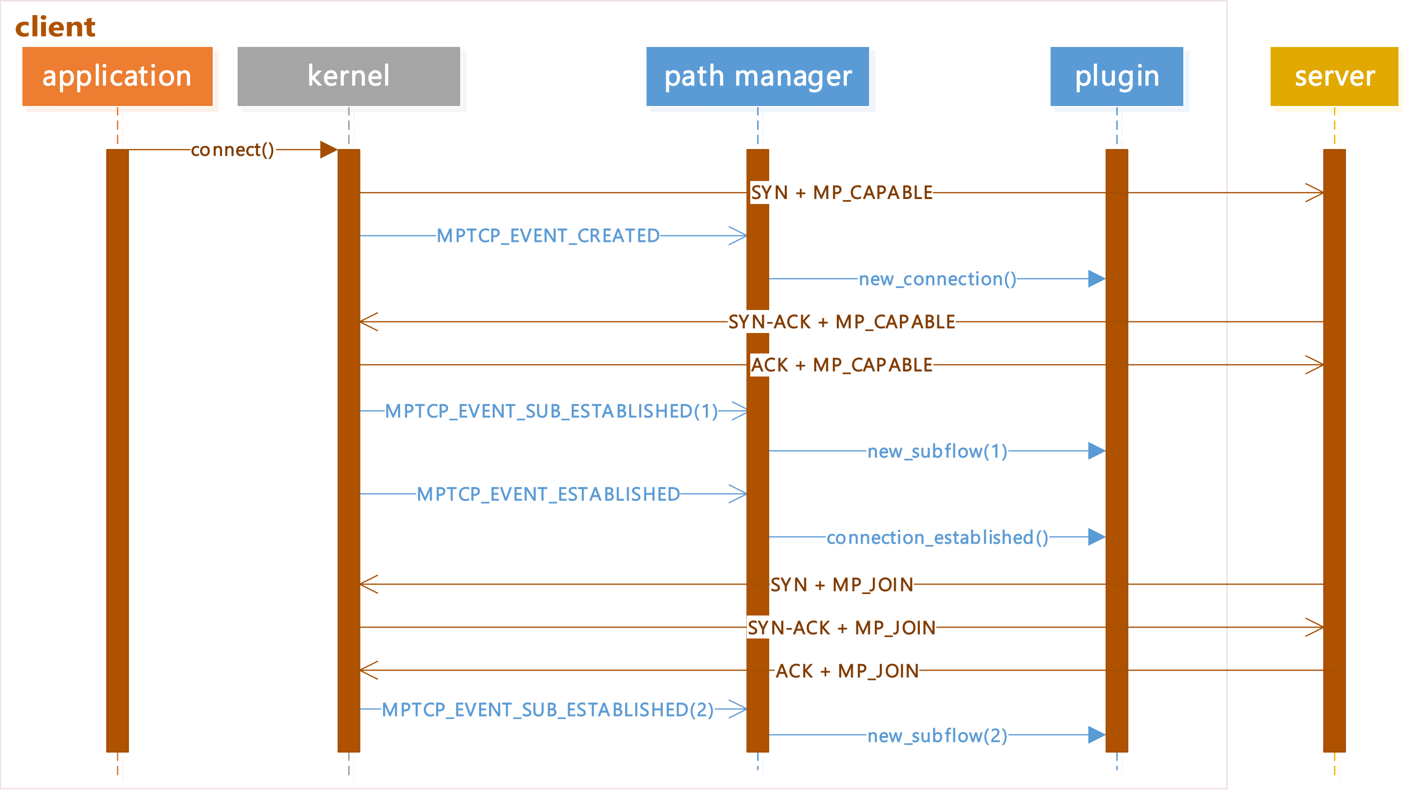
Path management implemented through plugins

- Plugins implement callbacks that correspond to network monitoring and MPTCP generic netlink events
- Plugin API header: <mptcpd/plugin.h>
- Network monitoring API header: <mptcpd/network_monitor.h>

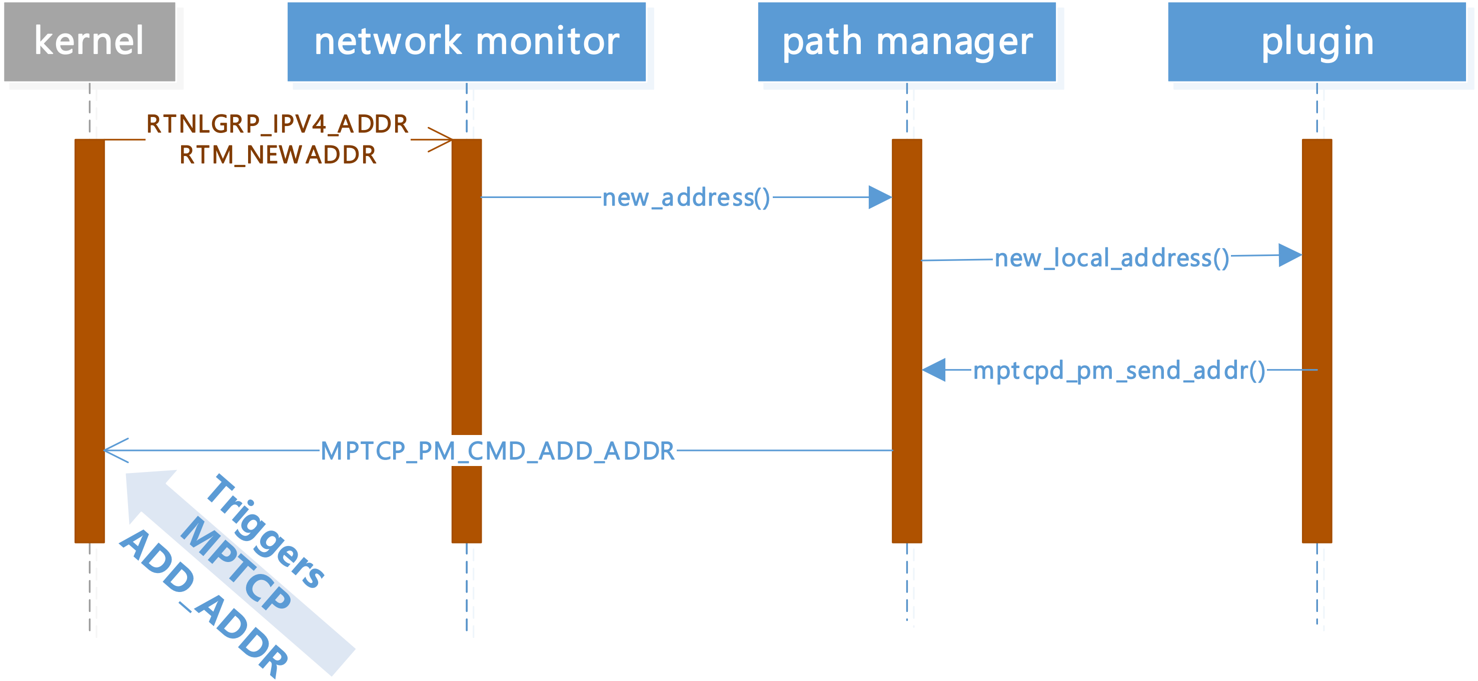
mptcpd library (libmptcpd)

- Plugins send MPTCP generic netlink commands to the kernel through functions found in libmptcpd
- Command API header: <mptcpd/path_manager.h>

Multipath TCP Daemon – Event Handling



Multipath TCP Daemon – Network Monitor



Multipath TCP Daemon – Deployment

Required capabilities

- CAP_NET_ADMIN

Systemd integration

- mptcpd installs a systemd unit file if detected at build time
- Dynamic user support for improved security

Configuration file installed in system configuration directory

- For example, “/etc/mptcpd”

Plugins installed in package library directory, e.g. “/usr/lib/mptcpd”

- mptcpd ships with single-subflow-per-interface “sspi” reference plugin

Closing

Summary

- Programs can begin using IPPROTO_MPTCP with Linux v5.7 and later
- MPTCP is ready for some handover-based server use cases in Linux v5.8 and later
- Userspace path managers will add MPTCP functionality to Linux PCs and mobile devices

Ongoing work

- Continue adding features from RFC 8684
- Netlink interface for userspace path management
- Better utilization of multiple subflows
- SYN cookie support
- Support more TCP socket options
- TCP Fast Open
- Configurable packet scheduler (for choosing subflows to send data)
- Improve performance

Contacts

- Github project: https://github.com/multipath-tcp/mptcp_net-next
- Mailing List: mptcp@lists.01.org
  - Subscribe at https://lists.01.org/postorius/lists/mptcp.lists.01.org/
- IRC: #MPTCPUpstream on freenode.net

- Mat Martineau: mathew.j.martineau@linux.intel.com
- Ossama Othman: ossama.othman@intel.com

Resources

- Previous talks
  - Netdev 0x12: https://netdevconf.info/0x12/session.html?how-hard-can-it-be-adding-multipath-tcp-to-the-upstream-kernel
  - Netdev 0x13: https://netdevconf.info/0x13/session.html?skb-meta-data-extensions
  - Linux Plumbers 2019: https://www.linuxplumbersconf.org/event/4/contributions/435/
  - DevConf.CZ 2020: https://devconfcz2020a.sched.com/event/YOx8/how-to-not-implement-a-not-so-new-net-protocol
- RFC 8684 / MPTCP v1: https://www.rfc-editor.org/rfc/rfc8684.html