Multipath TCP:

Present, future, and its development workflow (CI)

Matthieu Baerts (NGI0) • Netdev 0x19 • 09.03.2025

Plan

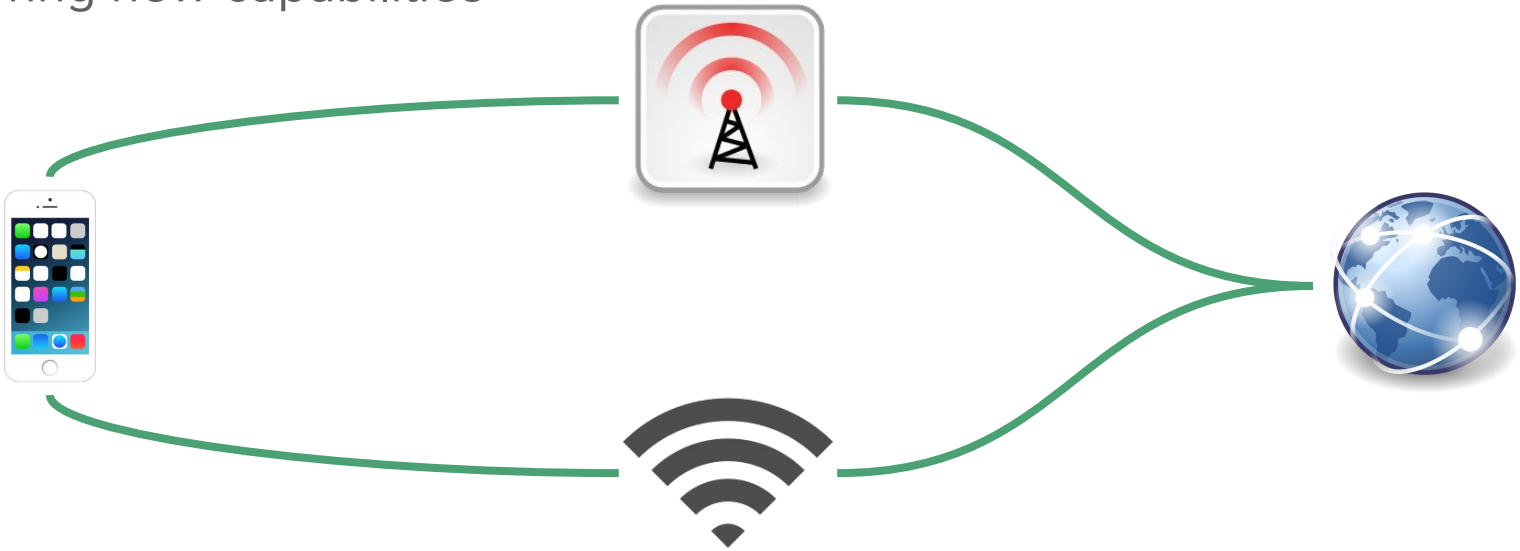
- What is MPTCP?
- How to use MPTCP on Linux?
- Current status & next steps
- Development workflow (CI)

What is MPTCP?

Multipath TCP (MPTCP – RFC-8684)

Exchange data for a single connection over different paths, simultaneously

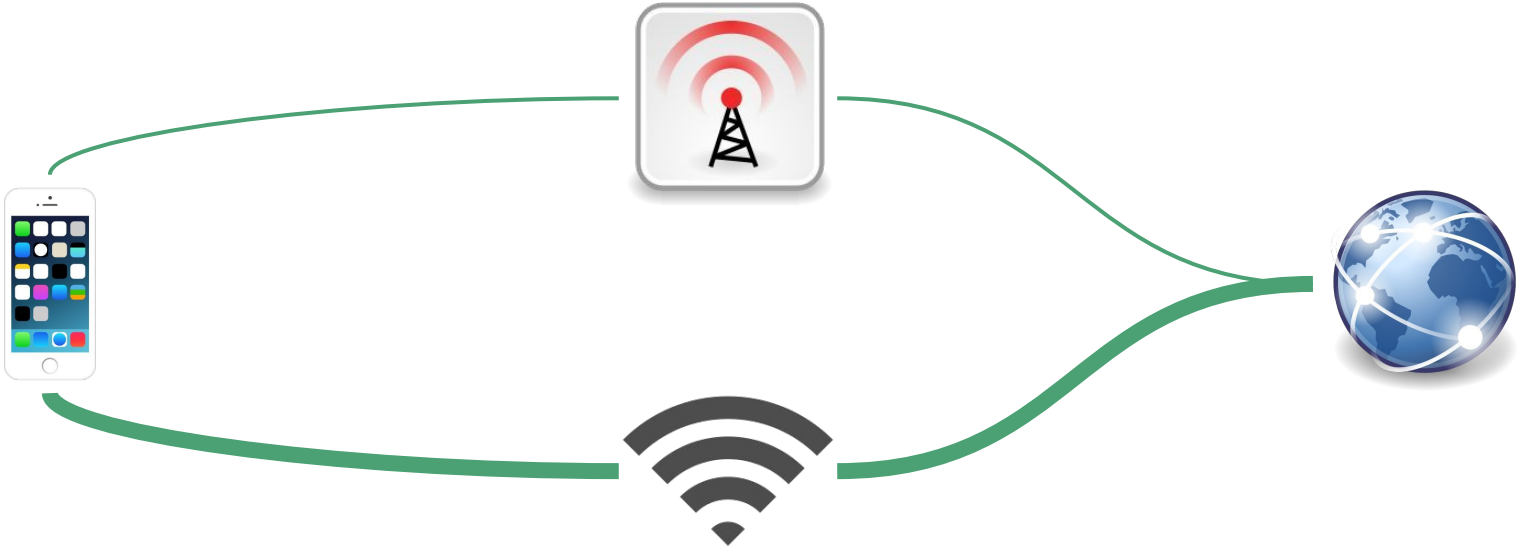
⇒ Bring new capabilities



Multipath TCP (MPTCP – RFC-8684)

Exchange data for a single connection over different paths, simultaneously

⇒ Best network selection

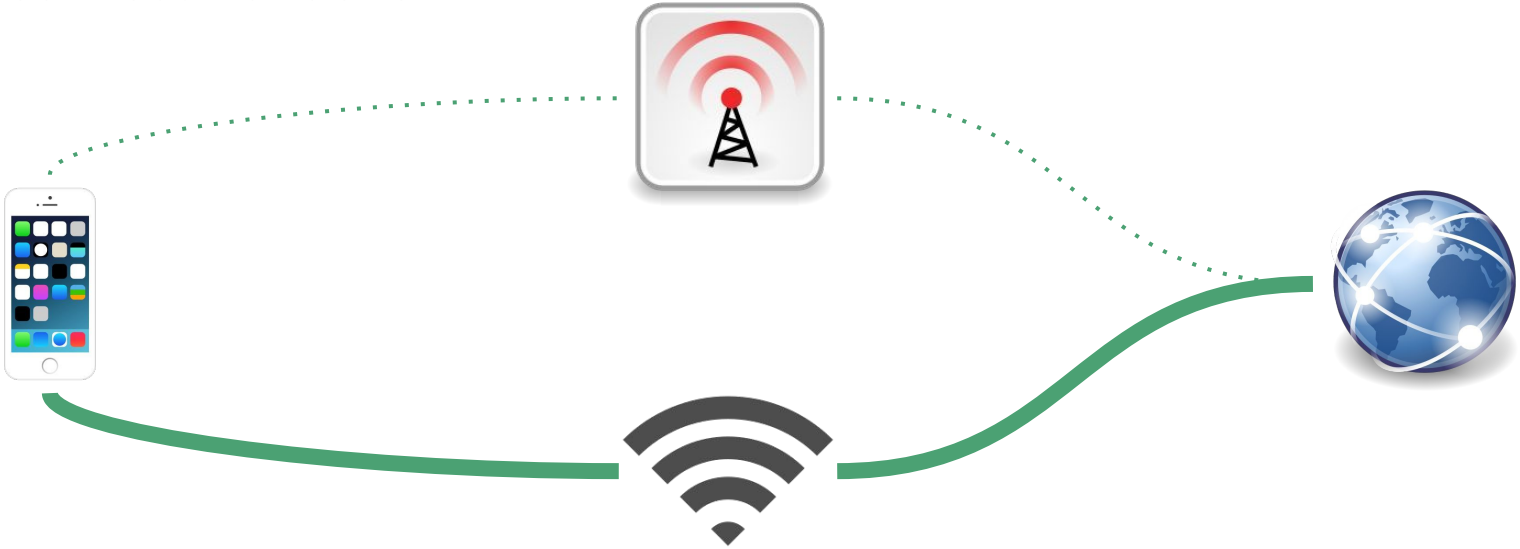


Smartphone and WiFi icons by Blurred203 and Antü Plasma under CC-by-sa, others from Tango project, public domain

Multipath TCP (MPTCP – RFC-8684)

Exchange data for a single connection over different paths, simultaneously

⇒ Seamless handover

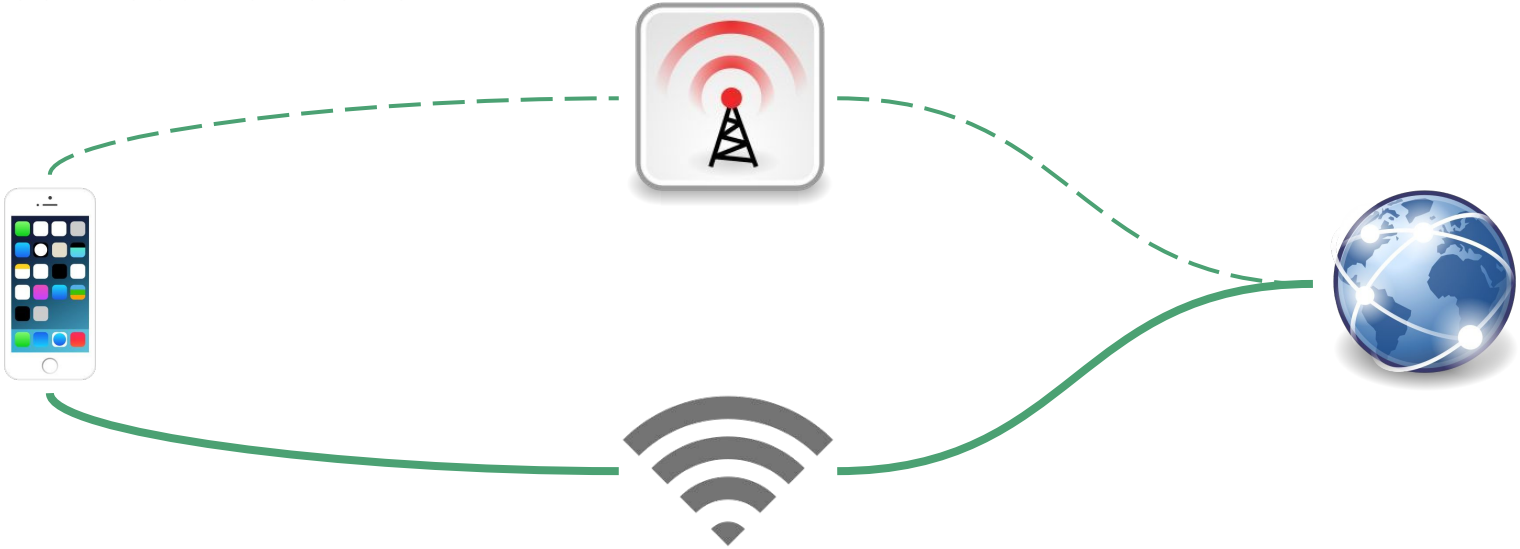


Smartphone and WiFi icons by Blurred203 and Antü Plasma under CC-by-sa, others from Tango project, public domain

Multipath TCP (MPTCP – RFC-8684)

Exchange data for a single connection over different paths, simultaneously

⇒ Seamless handover

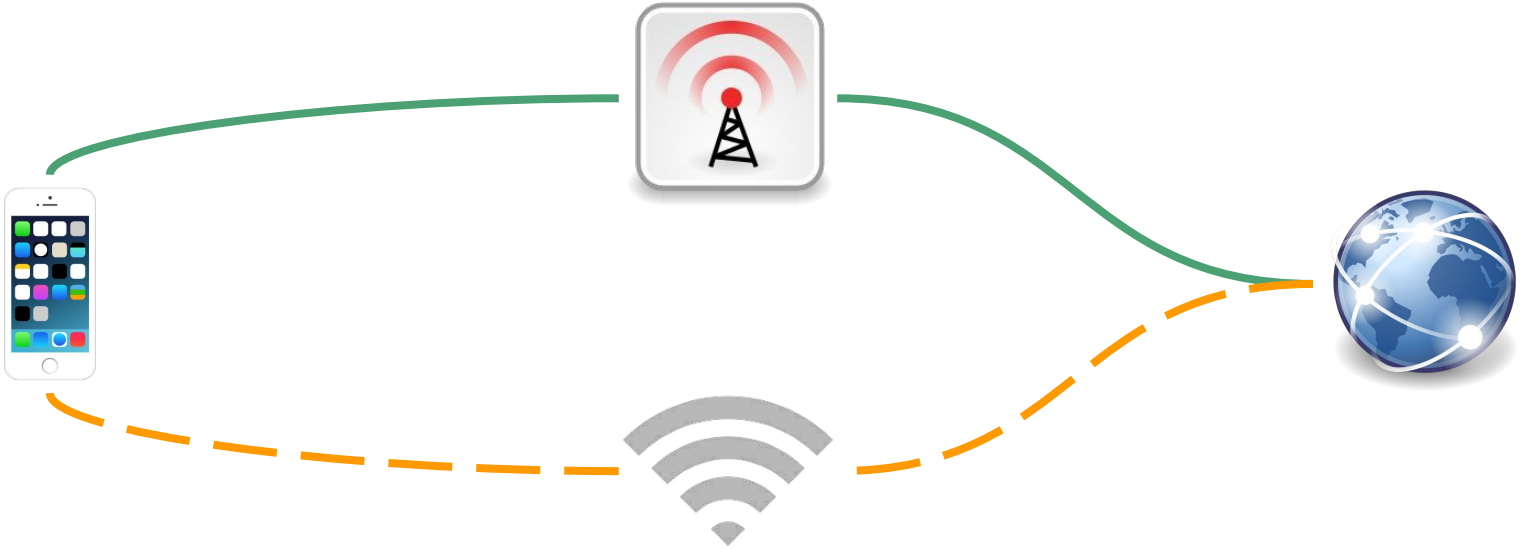


Smartphone and WiFi icons by Blurred203 and Antü Plasma under CC-by-sa, others from Tango project, public domain

Multipath TCP (MPTCP – RFC-8684)

Exchange data for a single connection over different paths, simultaneously

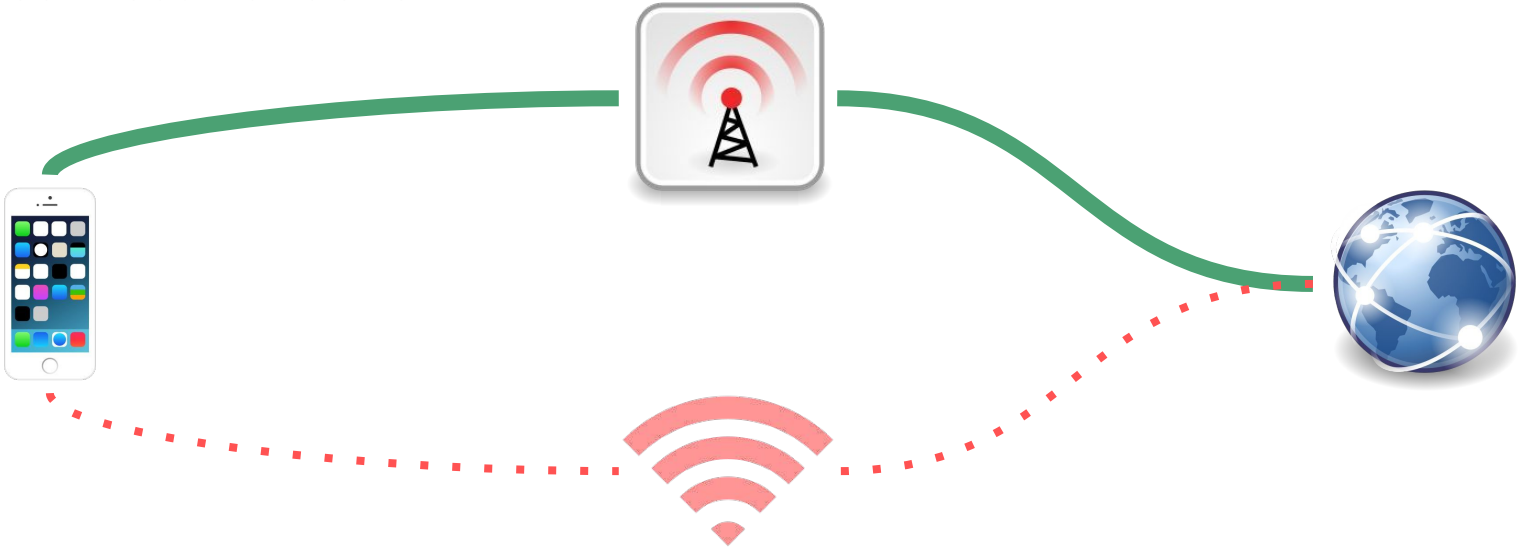
⇒ Seamless handover



Multipath TCP (MPTCP – RFC-8684)

Exchange data for a single connection over different paths, simultaneously

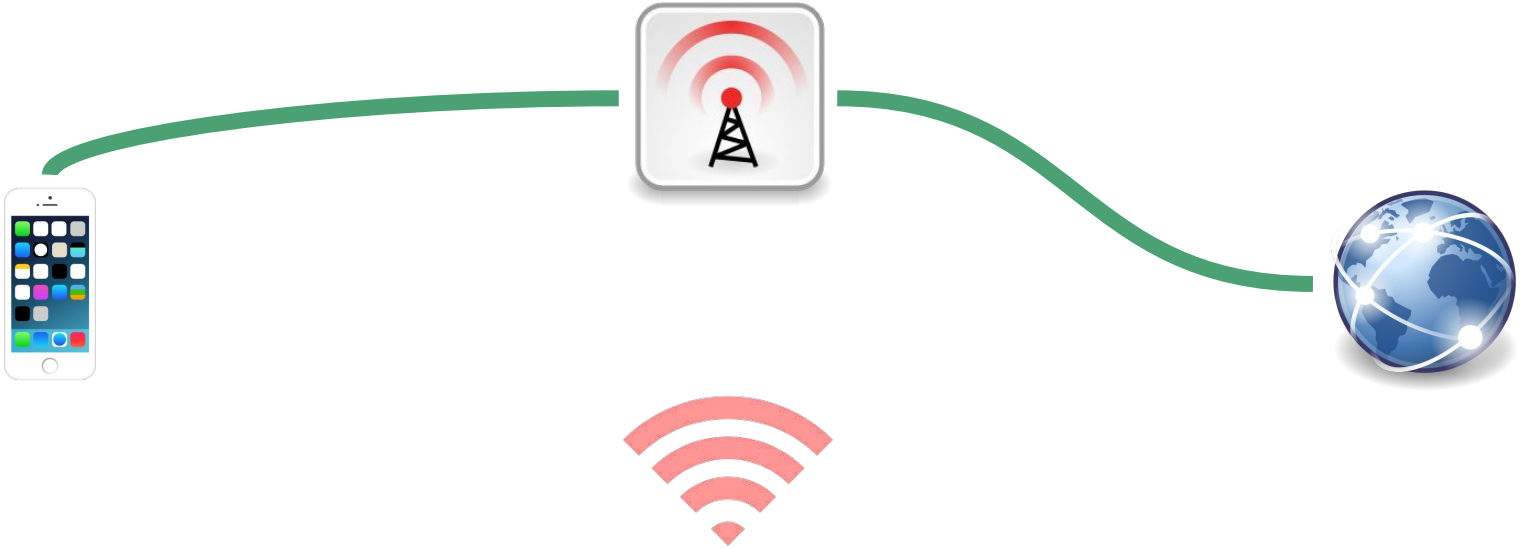
⇒ Seamless handover



Multipath TCP (MPTCP – RFC-8684)

Exchange data for a single connection over different paths, simultaneously

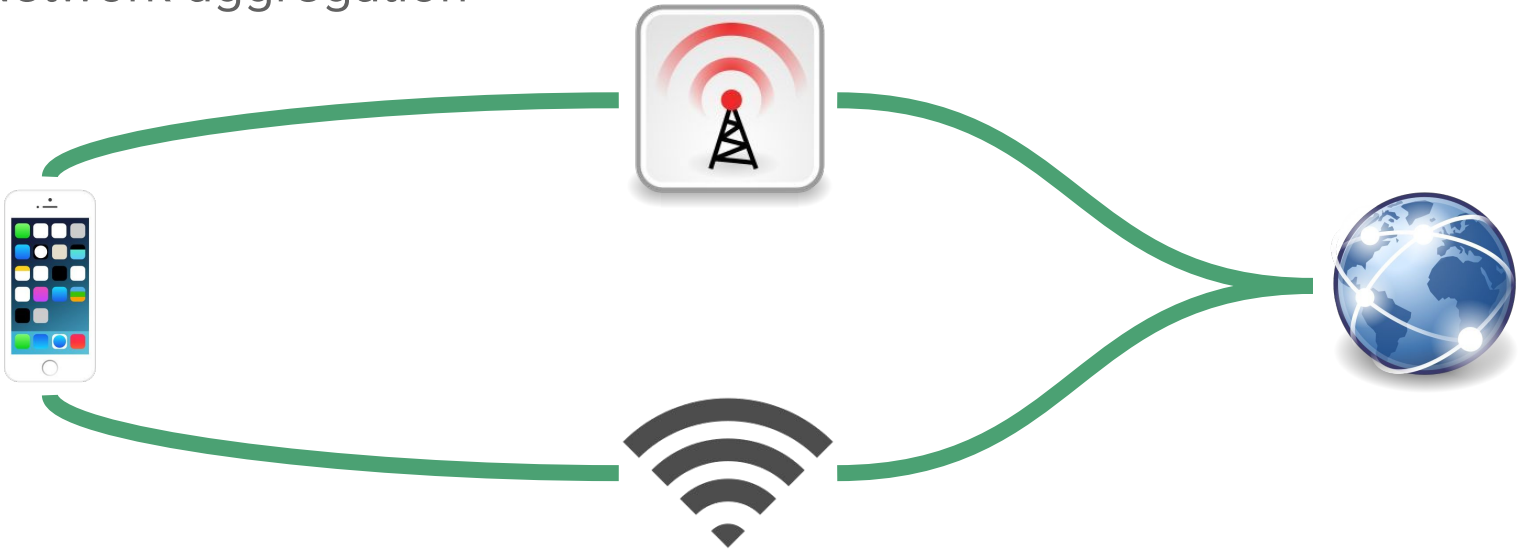
⇒ Seamless handover



Multipath TCP (MPTCP – RFC-8684)

Exchange data for a single connection over different paths, simultaneously

⇒ Network aggregation



Smartphone and WiFi icons by Blurred203 and Antü Plasma under CC-by-sa, others from Tango project, public domain

Multipath TCP (MPTCP – RFC-8684)

Use cases:

- Mobile devices:
  - “walk-out” scenario
- Home Gateways:
  - combine networks, e.g. DSL + cellular or low Earth orbit satellites
- Data centres:
  - fast recoveries, select best paths, aggregation

Concept: Subflow and Fallback

- Subflow: Each path of an MPTCP connection. A subflow is a regular TCP connection carrying extra options in the TCP header.

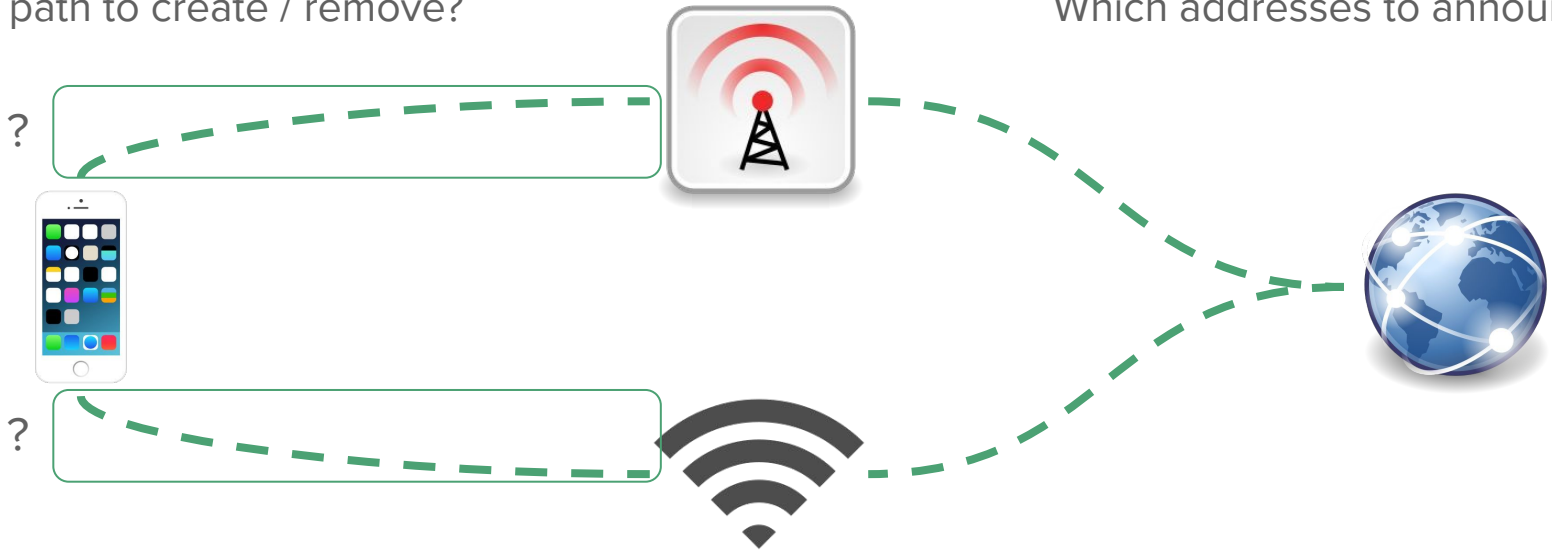
- Fallback: If the other host does not support MPTCP, or in case of a middlebox intercepting TCP connections, there will be fallbacks to TCP. The MPTCP protocol is complex to cope with various middleboxes.

Concept: Path Manager

Typically, different needs for the clients and servers:

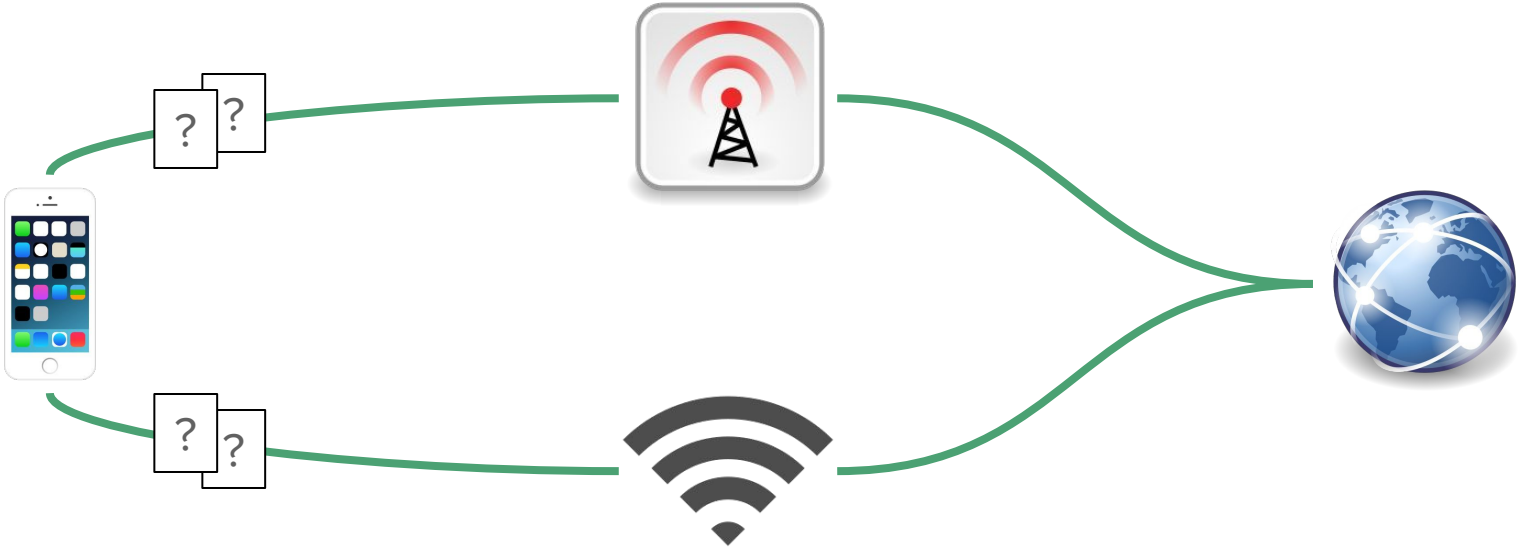
Which path to create / remove?

Which addresses to announce?



Concept: Packet Scheduler

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



Smartphone and WiFi icons by Blurred203 and Antü Plasma under CC-by-sa, others from Tango project, public domain

How to use MPTCP on Linux?

MPTCP on Linux

- The complexity is handled by the kernel
- Opt-in (with possibilities to force apps to use MPTCP):

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

- Minimal behaviour changes for apps compared to TCP
- Path-Manager configured via userspace, e.g. manually:

```
ip mptcp endpoint add <IP address> dev <interface> <type>
```

MPTCP on Linux

- Some tools can automatically set up the MPTCP endpoints, e.g. **`NetworkManager`** and **`mptcpd`**

- Some apps natively support MPTCP, e.g. cURL, HAProxy, Apache Server, Lighttpd, systemd sockets, Go apps (enabled by default on the server side), etc. Check mptcp.dev/apps.html

- Possibilities to force using MPTCP, e.g. `mptcpize` (`LD_PRELOAD`), `GODEBUG=multipathtcp=1`, eBPF, SystemTAP, etc. Check mptcp.dev/setup.html

MPTCP on Linux

- Most Linux distributions have MPTCP support enabled, and **`mptcpd`** packages, including specialised ones like OpenWrt, RPiOS, HAOS.

- Debugging tools supports MPTCP, e.g **`ss -M`**, **`ip mptcp`**, **`nstat`**, **`tcpdump`**, **`ptcpdump`**, WireShark

- Server: an MPTCP listen socket will create a TCP socket if the client didn’t request MPTCP: good to have MPTCP enabled by default!

Demo

Current status & next steps

Current status: general

- 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, reset reasons, etc.

- Info from **`MIB`** counters (`nstat`), **`INET_DIAG`** interface (`ss`) and **`MPTCP_INFO`** / **`MPTCP_FULL_INFO`** socket options

Current & Future: socket options

- Current: Supports most common socket options: `SO`, `IP`, `IPV6`, `TCP`:
  - Imitating TCP's behaviour
  - But adapted to MPTCP case:
    - Inherit the behaviour on all subflows, including future ones? e.g. KeepAlive
    - Or only on the first one? e.g. TCP FastOpen
  - Still possible to change the per-subflow behaviour with eBPF
- Future: Support more uncommon ones, and simplify the maintenance

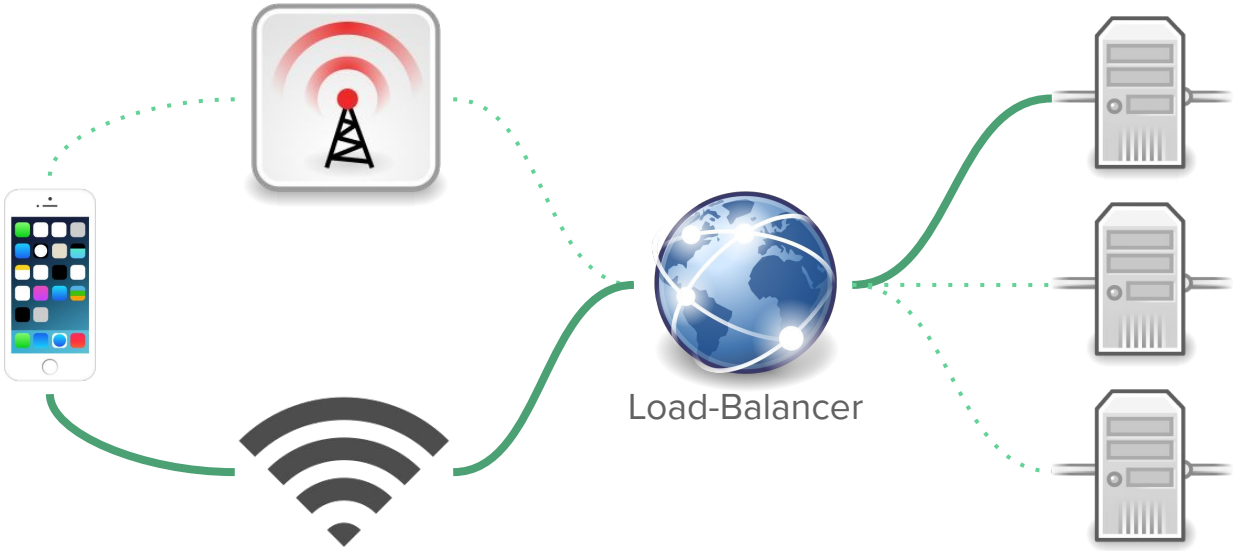
Current status: path managers

- *In-kernel*: Global settings per network namespace: e.g. via **`ip mptcp`**
  - Set endpoints: IP addresses, flags (client-server sides, backup, fullmesh)
  - Set limits: max subflows to establish or accept
  - Monitor connections: created, established, closed, announced, etc.
- *Userspace*: Per connection: e.g. via **`mptcpd`**
  - Reacting to "**`events`**" by sending "**`commands`**"

⇒ Communications: using Netlink

Current status: PM: Deployment behind a Load-Balancer

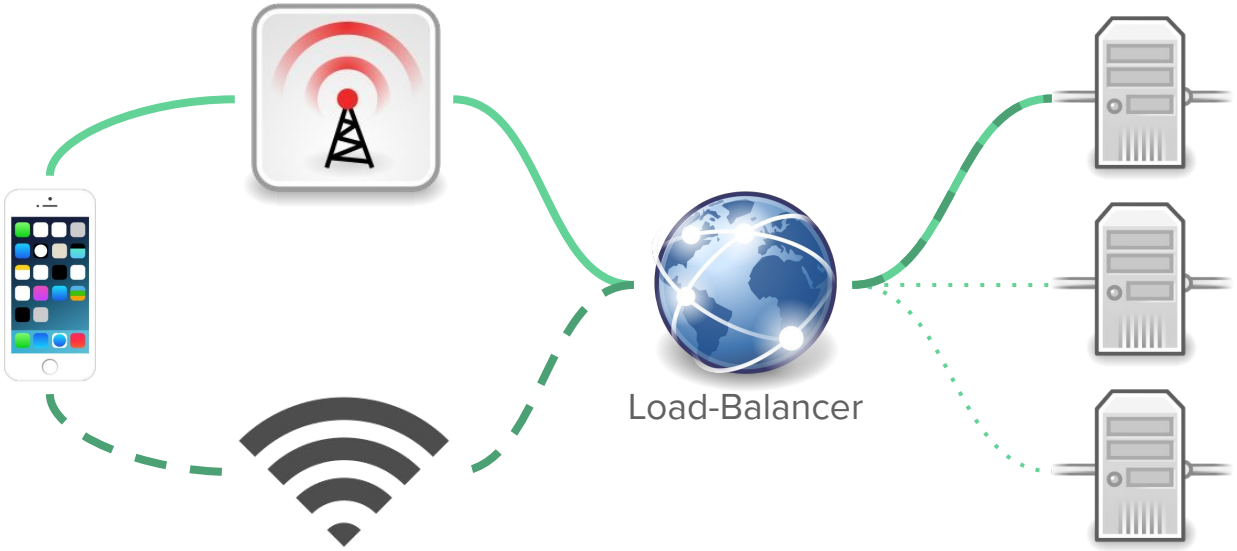
- Initial path: with a random server behind a stateless load-balancer



Anycast: each server is reusing the same public IP

Current status: PM: Deployment behind a Load-Balancer

- Additional paths: how a stateless load-balancer can pick the same server?

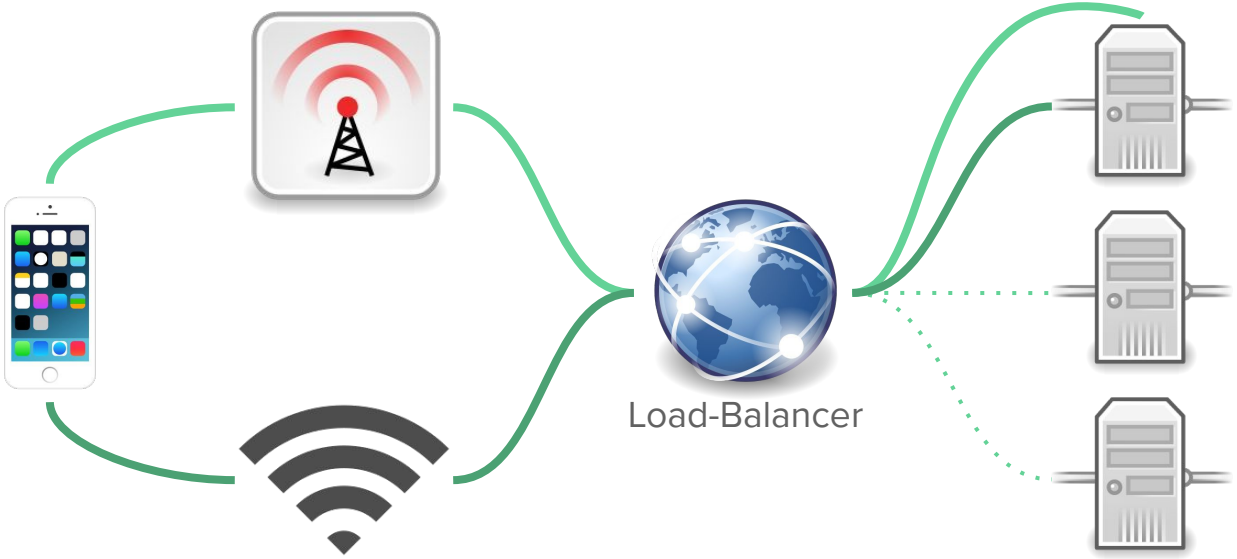


Anycast: each server is reusing the same public IP

Current status: PM: Deployment behind a Load-Balancer

- Additional paths: how a stateless load-balancer can pick the same server?

  ⇒ Servers: tell client not to use initial address and announce a new one.



Anycast: each server is reusing the same public IP

Current & Future: PM: Deployment behind a Load-Balancer

- Current:
  - Server side: fully supported ✅
  - Client side: respect protocol ☑
    - Not creating subflows to the initial address ✅
    - But… the path manager will not create additional paths by default ❌

- Future: better support this use-case on the client side

  And… MPTCP supported by more CDNs? 🤞

Future: Path Manager: More

- In-kernel PM: support less common use-cases, e.g.
  - Re-establishing subflows after network errors
  - Limit to one subflow per network device having multiple IP addresses
  - Force to use specific endpoints when the server announces a new IP

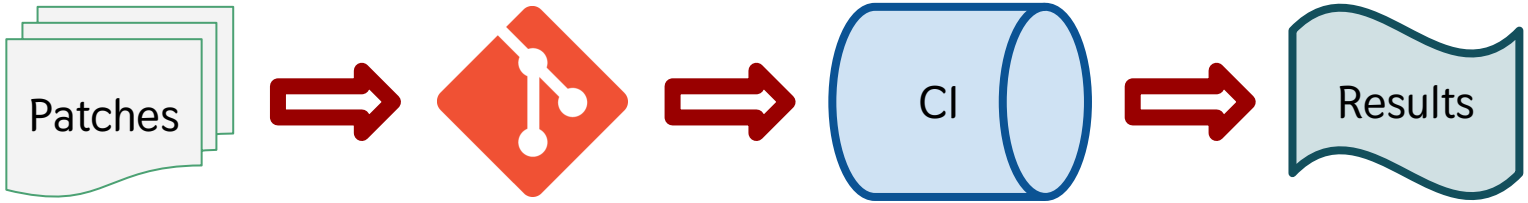
- BPF extension (`struct_ops`):
  - To adapt to specific use-cases, at a lower cost
  - Including quite a bit of clean-up in the current code!

Current & Future: packet scheduler

- Current: only one, generic, limited options, “handover” UC as main focus
- Future:
  - API refactoring to handle more cases
  - Support more corner cases, e.g. paths from “too heterogeneous” environments
  - CI: Better tracking performance regressions
  - BPF extension (Slow progress due to ↑ and various reasons)

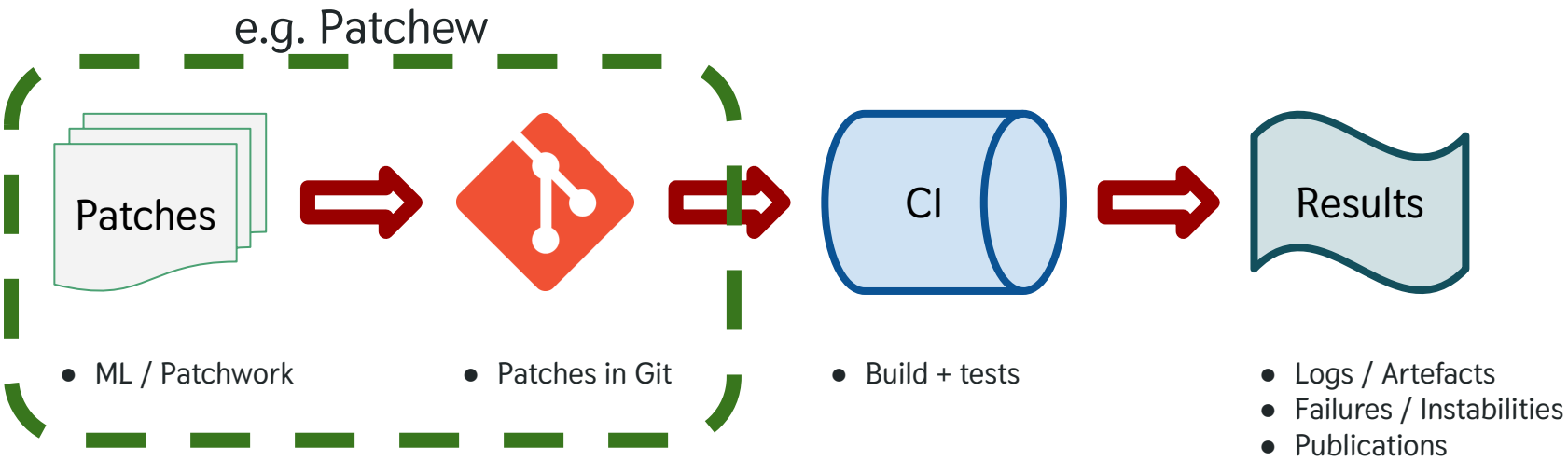
Development workflow (CI)

Workflow



- ML / Patchwork
- Patches in Git
- Build + tests
- Logs / Artefacts
- Failures / Instabilities
- Publications

Workflow


e.g. Patchew
Patches
CI
Results
ML / Patchwork
Patches in Git
Build + tests
Logs / Artefacts
Failures / Instabilities
Publications

Patches ⇒ Git: Patchew can help

[PATCH mptcp-next v2 0/2] tcp: ulp: diag: remove net admin restriction

Matthieu Baerts (NGI0) posted 2 patches **2 days, 16 hours ago** | Diff against v1 Download series mbox

Patches applied successfully (tree, apply log)

```
git fetch https://github.com/multipath-tcp/mptcp_net-next tags/patchew/20250305-mptcp-tcp-ulp-diag-cap-v2-0-d5
```

```
include/net/tcp.h   |  4 ++--
net/ipv4/tcp_diag.c | 21 ++++++++++-----------
net/mptcp/diag.c    | 42 ++++++++++++++++++++++++++++++---------------------------
net/tls/tls_main.c  |  4 ++--
4 files changed, 40 insertions(+), 31 deletions(-)
```

Expand all

Fold all

[PATCH mptcp-next v2 0/2] tcp: ulp: diag: remove net admin restriction

Posted by **Matthieu Baerts (NGI0)** 2 days, 16 hours ago

```
Since its introduction in commit 61723b393292 ("tcp: ulp: add functions
to dump ulp-specific information"), the ULP diag info have been exported
only if the requester had CAP_NET_ADMIN.
```

https://patchew.org

Patches ⇒ Git: Patchew can help

[PATCH mptcp-next v2 0/2] tcp: ulp: diag: remove net admin restriction

Matthieu Baerts (NGI0) posted 2 patches **2 days, 16 hours ago** Diff against v1 Download series mbox

Patches applied successfully (tree, apply log)

```
git fetch https://github.com/multipath-tcp/mptcp_net-next tags/patchew/20250305-mptcp-tcp-ulp-diag-cap-v2-0-d5
```

```
include/net/tcp.h    |  4 ++--
net/ipv4/tcp_diag.c  | 21 +++++++++-----------
net/mptcp/diag.c     | 42 ++++++++++++++++++++++++++--------------
net/tls/tls_main.c   |  4 ++--
4 files changed, 40 insertions(+), 31 deletions(-)
```

Expand all

Fold all

[PATCH mptcp-next v2 0/2] tcp: ulp:

Posted by **Matthieu Baerts (NGI0)** 2 c

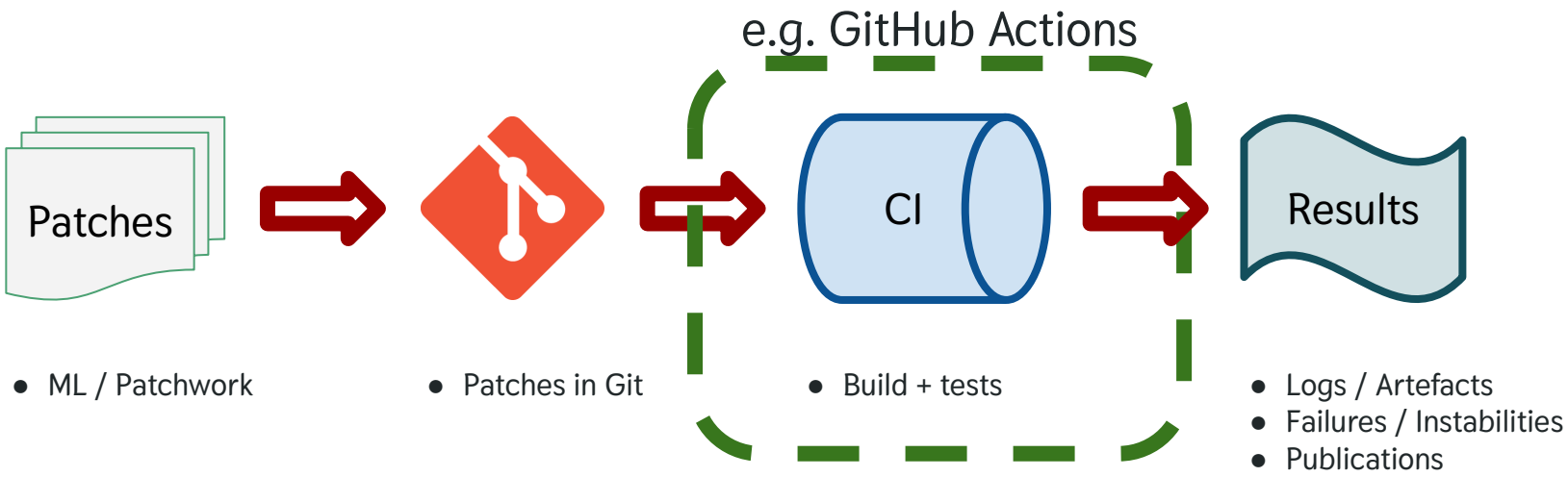
Since its introduction in c
to dump ulp-specific informa
only if the requester had CA

diff view generated by jsdifflib

[PATCH mptcp-next] tcp: ulp: diag: remove net admin restriction	[PATCH mptcp-next v2 1/2] tcp: ulp: diag: always print the name if any
Since its introduction in commit 61723b393292 ("tcp: ulp: add functions to dump ulp-specific information"), the ULP diag info have been exported only if the requester had CAP_NET_ADMIN.	Since its introduction in commit 61723b393292 ("tcp: ulp: add functions to dump ulp-specific information"), the ULP diag info have been exported only if the requester had CAP_NET_ADMIN.
It looks like there is nothing sensitive being exported here by the MPTCP and KTLS layers. So it seems safe to remove this restriction in order to ease the debugging from the userspace side without requiring additional capabilities.	At least the ULP name can be exported without CAP_NET_ADMIN. This will already help identifying which layer is being used, e.g. which TCP connections are in fact MPTCP subflow.
Signed-off-by: Matthieu Baerts (NGI0) <matttbe@kernel.org>	Signed-off-by: Matthieu Baerts (NGI0) <matttbe@kernel.org>
---	---
net/ipv4/tcp_diag.c | 15 +++++++--------	net/ipv4/tcp_diag.c | 21 +++++++++------------
1 file changed, 7 insertions(+), 8 deletions(-)	1 file changed, 10 insertions(+), 11 deletions(-)
diff --git a/net/ipv4/tcp_diag.c b/net/ipv4/tcp_diag.c	diff --git a/net/ipv4/tcp_diag.c b/net/ipv4/tcp_diag.c
index XXXXXXX..XXXXXXX 100644	index XXXXXXX..XXXXXXX 100644
--- a/net/ipv4/tcp_diag.c	--- a/net/ipv4/tcp_diag.c
+++ b/net/ipv4/tcp_diag.c	+++ b/net/ipv4/tcp_diag.c
	@@ -XXX,XX +XXX,XX @@ static int tcp_diag_put_md5sig(struct sk_buff *skb,
	#endif

https://patchew.org

Workflow


e.g. GitHub Actions
Patches
CI
Results
ML / Patchwork
Patches in Git
Build + tests
Logs / Artefacts
Failures / Instabilities
Publications

CI: GitHub Actions can help

Matrix: Tests

Tests (btf-debug)	31m 47s
Tests (btf-normal)	25m 54s
Tests (debug)	1h 5m
Tests (normal)	37m 44s

Publish Tests Results	16s
Notifications	43s
Status	0s

Test Results

66 files ±0	66 suites ±0	0s ±0s	
632 tests ±0	630 -2	0 ±0	2 +2
1 264 runs ±0	1 262 -2	0 ±0	2 +2

For more details on these failures, see this check.

Results for commit dc126c97. ± Comparison against earlier commit c396630c.

Tests (normal)

succeeded yesterday in 37m 44s

- Set up job
- Pull ghcr.io/enricomi/publish-unit-test-result-action:v2.18.0
- Checkout
- Find base branch
- Restore cache for CCache
- Docker image
- Tests

```
1  ▸ Run echo 'KERNEL=="kvm", GROUP="kvm", MODE="0666", OPTIONS+="static
29 KERNEL=="kvm", GROUP="kvm", MODE="0666", OPTIONS+="static_node=kvm"
30 + /usr/bin/docker run --privileged --rm -e INPUT_CCACHE_MAXSIZE=500M
   GITHUB_REF_NAME -e GITHUB_RUN_ID -e GITHUB_ACTIONS=true -e CI=true --
   mptcp_net-next/mptcp_net-next ghcr.io/multipath-tcp/mptcp-upstream-vi
31
40 ▸ Setup environment
43 ▸ Generate kernel config
93 ▸ Build kernel
```

CI: GitHub Actions can help

```
- name: "Tests"
  timeout-minutes: 120
  run: |
    echo 'KERNEL=="kvm", GROUP="kvm", MODE="0666", OPTIONS+="static_node=kvm"' | sudo tee /etc/udev/rules.d/99-kvm4all.rules
    sudo udevadm control --reload-rules
    sudo udevadm trigger --name-match=kvm

    # remove old cache if any
    rm -rvf "${{ github.workspace }}/.virtme/ccache-"* 2>/dev/null

    set -x
    /usr/bin/docker run --privileged --rm \
      -e "INPUT_CCACHE_MAXSIZE=500M" \
      -e "INPUT_CCACHE_DIR=ccache" \
      -e "INPUT_PACKETDRILL_STABLE=${{ steps.branch.outputs.name == 'export-net' && '1' || '0' }}" \
      -e "INPUT_EXTRA_ENV=${{ startsWith(matrix.mode, 'btf-') && 'INPUT_RUN_TESTS_ONLY=bpftest_all' || '' }}" \
      -e "INPUT_TRACE=${RUNNER_DEBUG}" \
      -e "INPUT_GCOV=1" \
      -e "GITHUB_SHA" -e "GITHUB_REF_NAME" -e "GITHUB_RUN_ID" \
      -e GITHUB_ACTIONS=true -e CI=true \
      --workdir "${PWD}" \
      -v "${PWD}:${PWD}" \
      ghcr.io/multipath-tcp/mptcp-upstream-virtme-docker:${{ steps.branch.outputs.name == 'export' && 'latest' || 'net' }} \
        auto-${{ matrix.mode }}
```

CI: Requirements

- Results publicly available, configurable by maintainers
- Many steps to build and run the tests:
  - Setup environment: code, tools, etc.
  - Build kernel with right kconfig, and cache
  - Start a VM with KVM support or dedicated HW
  - Catch errors: call trace, warning messages, kmemleak, etc.

CI: MPTCP case

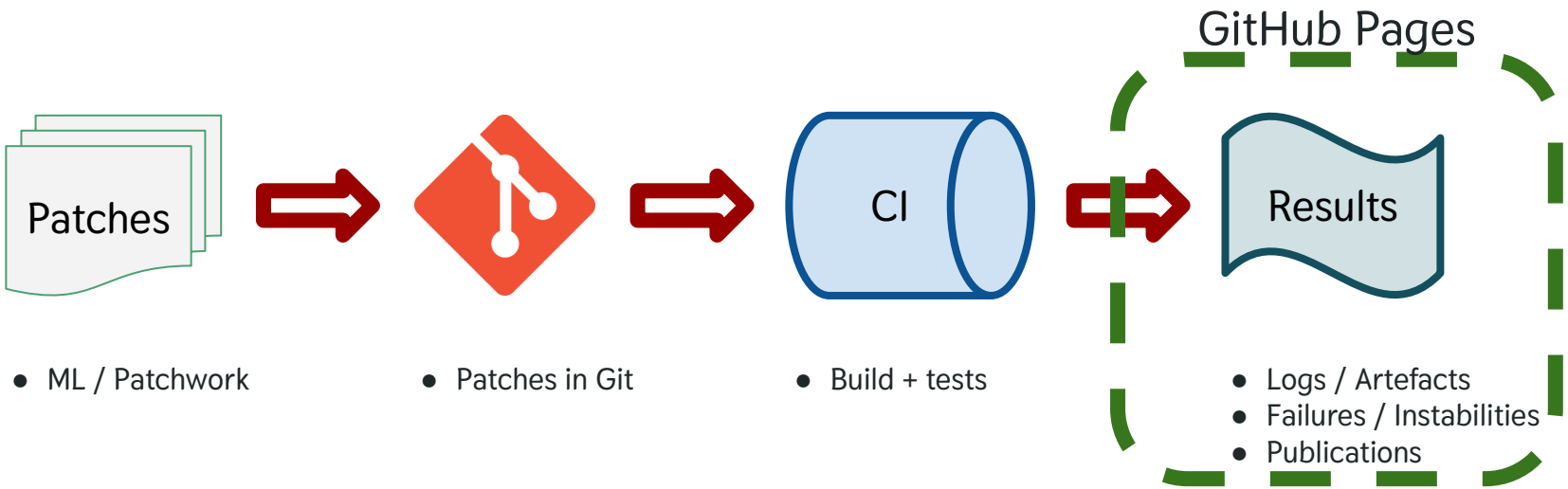
- Environment: containers are helpful to get the same everywhere

```
docker run (...) --privileged mptcp/mptcp-upstream-virtme-docker:latest
```

- VM: **`virtme-ng`** is helpful to build and start a VM
- KVM support: GitHub Actions supports it BUT it is opt-in
- Cache: **`ccache`** is helpful
- Catching errors: not difficult but a few cases to deal with, could be shared

https://github.com/multipath-tcp/mptcp-upstream-virtme-docker

Workflow



- ML / Patchwork
- Patches in Git
- Build + tests
- Logs / Artefacts
- Failures / Instabilities
- Publications

Results: MPTCP case

- Logs / Artefacts: usually easy

```yaml
- name: "Artifacts (always)"
  if: always()
  uses: actions/upload-artifact@v4
  with:
    name: results-${{ matrix.mode }}
    path: |
      conclusion.txt
      summary.txt
      coverage.txt
      *.tap
      config.zstd
      *.tap.xml
      results.json
```

```yaml
- name: "Artifacts (failure)"
  if: failure()
  uses: actions/upload-artifact@v4
  with:
    name: debug-info-${{ matrix.mode }}
    path: |
      vmlinux.zstd
      kmemleak.txt
```

```
6090  Thu, 06 Mar 2025 06:16:07 GMT  ▾ Selftest Test: ./simult_
6091  Thu, 06 Mar 2025 06:16:07 GMT    TAP version 13
6092  Thu, 06 Mar 2025 06:16:07 GMT    1..1
6093  Thu, 06 Mar 2025 06:16:17 GMT    # 01 balanced bwidth
6094  Thu, 06 Mar 2025 06:16:25 GMT    # 02 balanced bwidth - r
6095  Thu, 06 Mar 2025 06:16:33 GMT    # 03 balanced bwidth wit
6096  Thu, 06 Mar 2025 06:16:41 GMT    # 04 balanced bwidth wit
6097  Thu, 06 Mar 2025 06:16:53 GMT    # 05 unbalanced bwidth
```

Results: MPTCP case

- Parse results: TAP parsers or converters to JUnit, etc.

Test Results

66 files ±0	66 suites ±0		0s ⏱ ±0s	
632 tests ±0	630 ✅ -2	0 💤 ±0		2 ❌ +2
1 264 runs ±0	1 262 ✅ -2	0 💤 ±0		2 ❌ +2

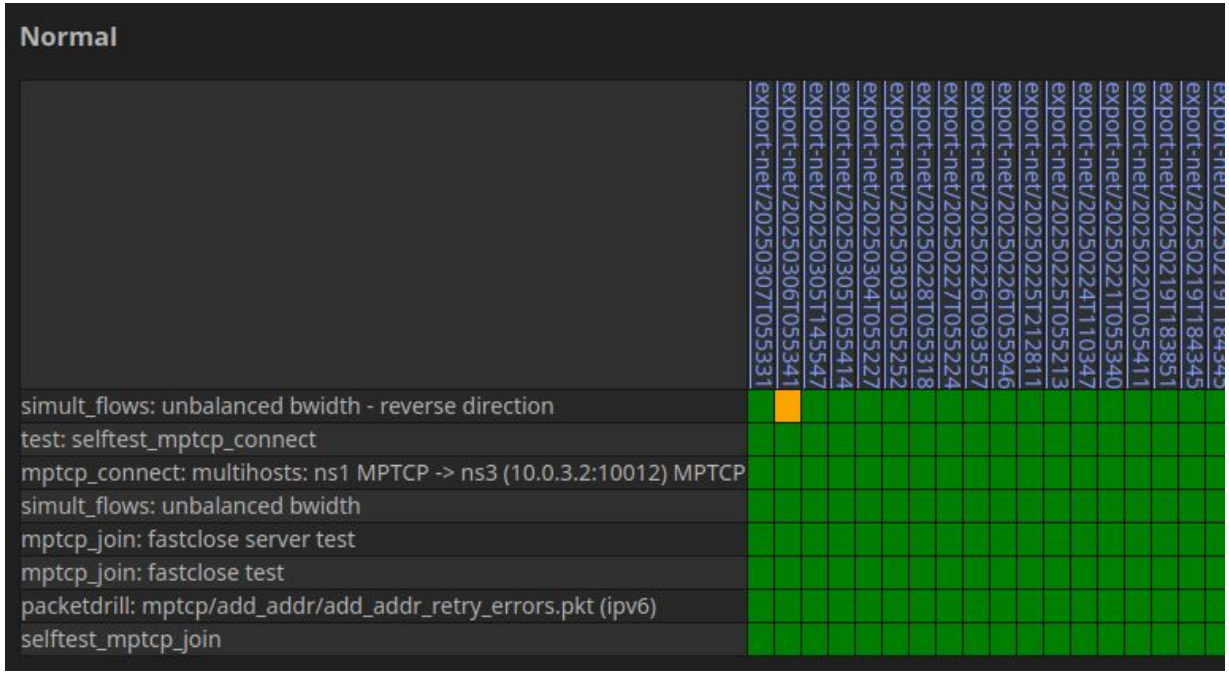
For more details on these failures, see this check.

Results for commit dc126c97. ± Comparison against earlier commit c396630c.

```
All tests:
ok 1 test: kunit
ok 1 test: mptcp_connect_mmap
ok 1 test: selftest_diag
ok 1 test: selftest_mptcp_connect
ok 1 test: selftest_mptcp_join
ok 1 test: selftest_mptcp_sockopt
ok 1 test: selftest_pm_netlink
ok 1 test: selftest_simult_flows
ok 1 test: selftest_userspace_pm
ok 1 test: kunit_mptcp-crypto
ok 1 test: kunit_mptcp-token
ok 1 test: packetdrill_add_addr
ok 1 test: packetdrill_dss
ok 1 test: packetdrill_fastclose
ok 1 test: packetdrill_fastopen
ok 1 test: packetdrill_mp_capable
not ok 1 test: packetdrill_mp_join
ok 1 test: packetdrill_mp_prio
ok 1 test: packetdrill_mp_reset
ok 1 test: packetdrill_regressions
ok 1 test: packetdrill_sockopts
ok 1 test: packetdrill_syscalls
```

Results: MPTCP case

- Check regressions: “homemade” solution publishing “flakes” in HTML.



Results: MPTCP case

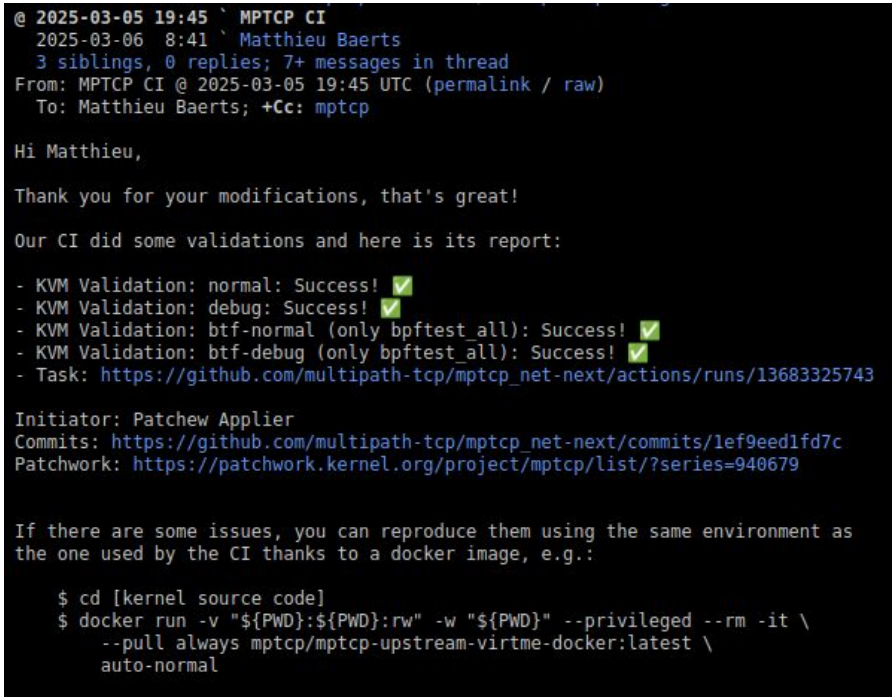
- Publishing results on Patchwork: a bit of plumbing.

Checks

Context	Check	Description
matttbe/checkpatch	warning	total: 0 errors, 40 warnings, 0 checks, 4129 lines checked
matttbe/shellcheck	success	MPTCP selftests files have not been modified
matttbe/build	warning	Build error with: make C=1 net/mptcp/pm_kernel.o
matttbe/KVM_Validation__normal	success	Success! ✅
matttbe/KVM_Validation__debug	success	Success! ✅
matttbe/KVM_Validation__btf-normal__only_bpftest_all_	success	Success! ✅
matttbe/KVM_Validation__btf-debug__only_bpftest_all_	success	Success! ✅

Results: MPTCP case

- Notifications, e.g. IRC and email:



```
@ 2025-03-05 19:45 ` MPTCP CI
  2025-03-06  8:41 ` Matthieu Baerts
  3 siblings, 0 replies; 7+ messages in thread
From: MPTCP CI @ 2025-03-05 19:45 UTC (permalink / raw)
  To: Matthieu Baerts; +Cc: mptcp

Hi Matthieu,

Thank you for your modifications, that's great!

Our CI did some validations and here is its report:

- KVM Validation: normal: Success! ✅
- KVM Validation: debug: Success! ✅
- KVM Validation: btf-normal (only bpftest_all): Success! ✅
- KVM Validation: btf-debug (only bpftest_all): Success! ✅
- Task: https://github.com/multipath-tcp/mptcp_net-next/actions/runs/13683325743

Initiator: Patchew Applier
Commits: https://github.com/multipath-tcp/mptcp_net-next/commits/1ef9eed1fd7c
Patchwork: https://patchwork.kernel.org/project/mptcp/list/?series=940679


If there are some issues, you can reproduce them using the same environment as
the one used by the CI thanks to a docker image, e.g.:

    $ cd [kernel source code]
    $ docker run -v "${PWD}:${PWD}:rw" -w "${PWD}" --privileged --rm -it \
        --pull always mptcp/mptcp-upstream-virtme-docker:latest \
        auto-normal
```

Results: MPTCP case

- Code coverage with GCOV, exported in HTML with LCOV:

  (and tracked on Coveralls.io)

			Coverage	Total	Hit
Current view:	top level				
Test:	export	Lines:	89.1 %	8233	7338
Test Date:	2025-03-07 06:59:24	Functions:	96.2 %	530	510
Legend:	Rating: low: < 75 % medium: >= 75 % high: >= 90 %	Branches:	64.6 %	6533	4222

Filename	Line Coverage				Branch Coverage			Function Co
	Rate		Total	Hit	Rate	Total	Hit	Rate
mptcp/bpf.c		81.7 %	115	94	69.9 %	73	51	80.0 %
mptcp/crypto.c		96.9 %	32	31	92.9 %	14	13	100.0 %
mptcp/crypto_test.c		100.0 %	19	19	75.0 %	8	6	100.0 %

Questions

Discussions

- 🌐 mptcp.dev
- ✉️ mptcp@lists.linux.dev
- 💬 IRC: #mptcp on Libera.chat
- 🎙️ Online Meetings
- 📰 blog.mptcp.dev
- @mptcp@social.kernel.org – @matttbe@fosstodon.org