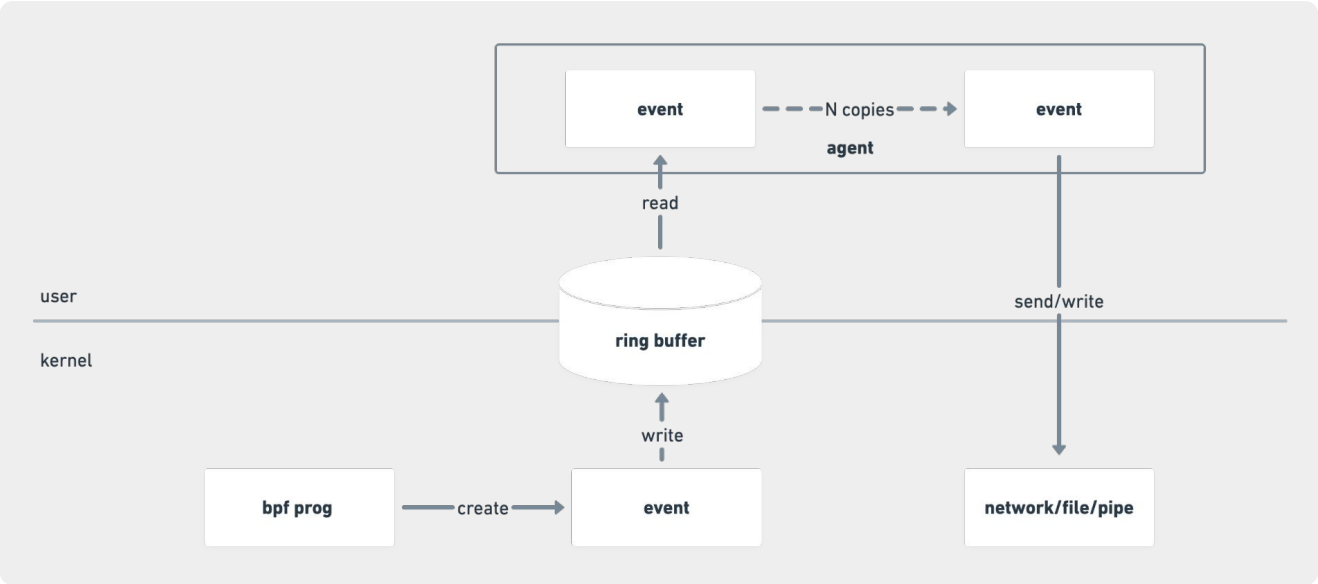
UDP send kfunc via BPF netpoll

Song Liu, Mahe Tardy, Liam Wisehart

Motivation

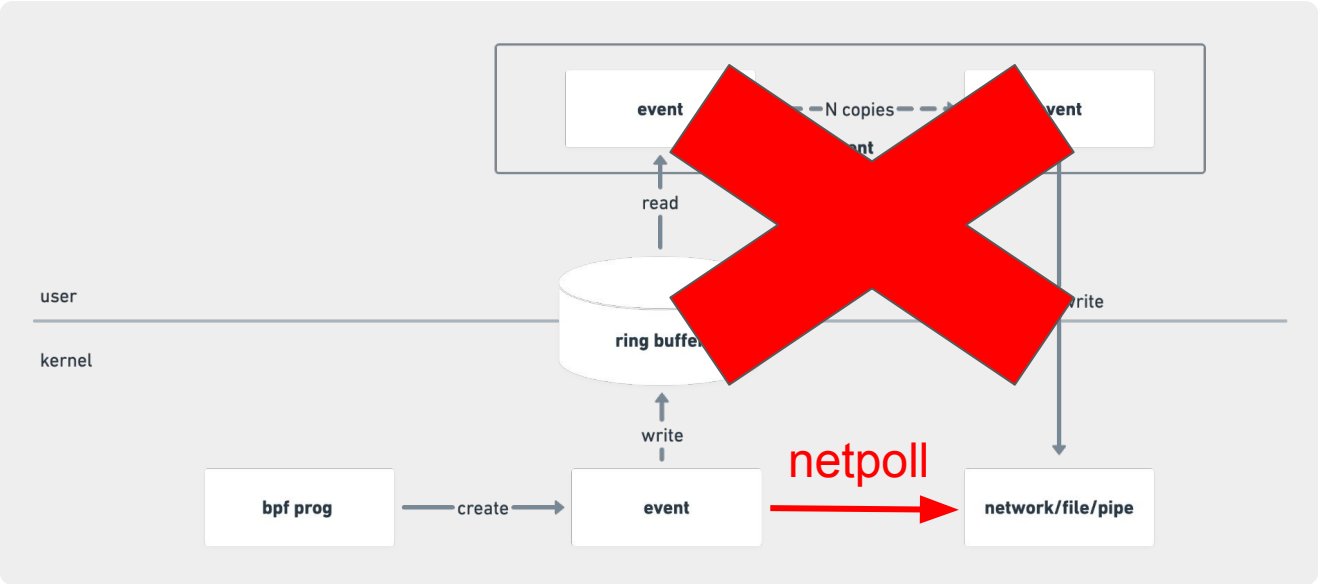
- Make Tetragon “**agentless**”: enforcement & observability can continue without the user space agent.
- Great from a cpu/mem perspective that only BPF progs runs.
- If not observability event, send **heartbeat messages** through network.
- **Replace logging libraries** via ring buffers

Tetragon **bpf_netpoll** use case



From **Splicing BPF maps to the NIC** at LSF/MM/BPF 2025 in Montreal (last year)

Tetragon **bpf_netpoll** use case



From **Splicing BPF maps to the NIC** at LSF/MM/BPF 2025 in Montreal (last year)

A bit of context on doing this

- At LSF/MM/BPF 25 we presented the motivation.
- We initially thought about using `vmsplice` & `splice` (or iouring) to map memory region to pipe/file/network but it appeared to be quite unpopular. Presented some PoC hacks.
- Audience pointed using the netconsole infra as a first step.

What is **netpoll**?

- Netpoll is the kernel infrastructure behind netconsole, it allows to send packet from any context, bypassing the normal network stack.
- Looks ideal from a BPF context & no recursion event from network progs.
- First version of patch set sent available on the mailing list: https://lore.kernel.org/bpf/20260309131635.302424-1-mahe.tardy@gmail.com/

How does **bpf_netpoll** works?

- Similar implementation to the bpf_crypto kfuncs: `bpf_netpoll_create()` & `bpf_netpoll_[acquire|release]()`
- Typically call `bpf_netpoll_create` with `struct bpf_netpoll_opts opts` which contains **dev**, **local port/IP**, **remote port/IP** and **mac**.
- The `bpf_netpoll_create` returns a `struct bpf_netpoll *` to reuse with `bpf_netpoll_send_udp` when sending data.

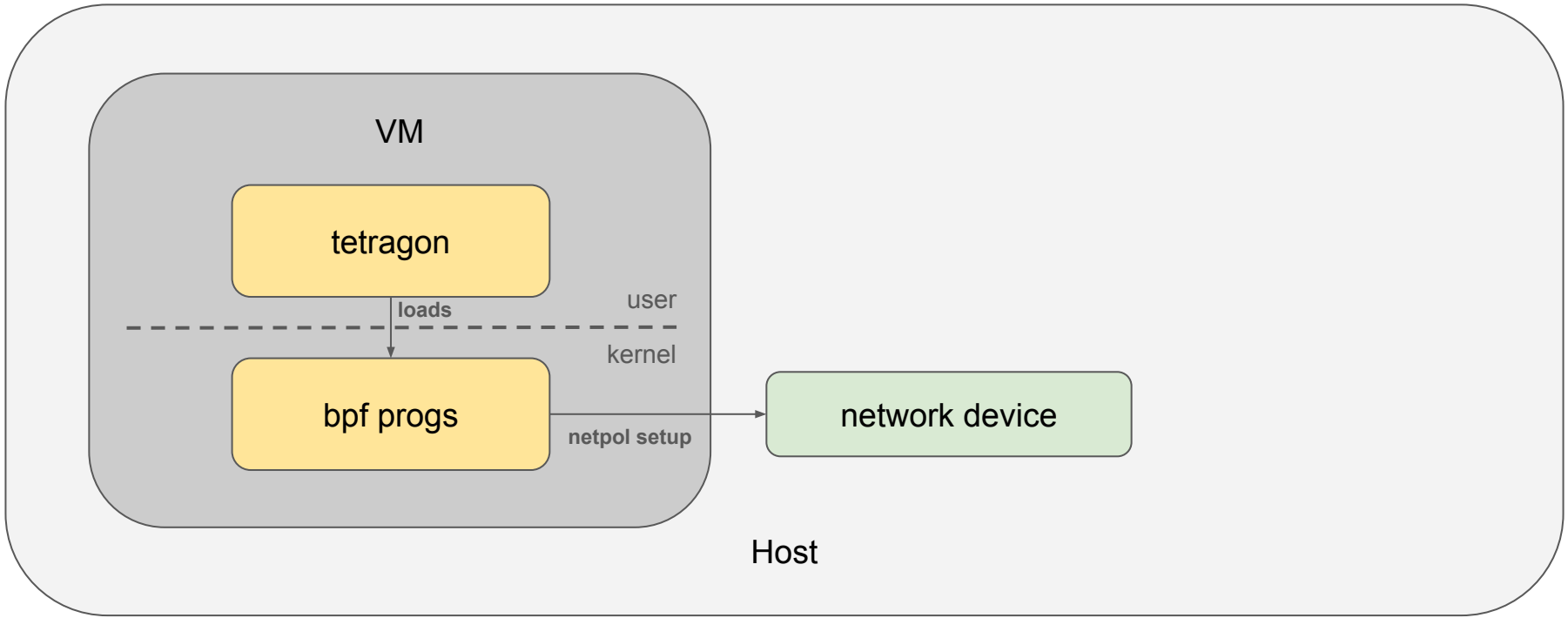
```
From: Mahe Tardy <mahe.tardy@gmail.com>
To: bpf@vger.kernel.org
Cc: andrew+netdev@lunn.ch, davem@davemloft.net, edumazet@google.com,
        kuba@kernel.org, pabeni@redhat.com, martin.lau@linux.dev,
        daniel@iogearbox.net, john.fastabend@gmail.com, ast@kernel.org,
        andrii@kernel.org, eddyz87@gmail.com, song@kernel.org,
        Mahe Tardy <mahe.tardy@gmail.com>
Subject: [RFC PATCH bpf-next 1/4] bpf: Add netpoll kfuncs for sending UDP packets
Date: Mon,  9 Mar 2026 13:16:32 +0000   [thread overview]
Message-ID: <20260309131635.302424-2-mahe.tardy@gmail.com> (raw)
In-Reply-To: <20260309131635.302424-1-mahe.tardy@gmail.com>

From: Song Liu <song@kernel.org>

Add BPF kfuncs that allow BPF programs to send UDP packets via the
netpoll infrastructure. This provides a mechanism for BPF programs
(e.g., LSM hooks) to emit telemetry over UDP without depending on
the regular networking stack.

The API consists of four kfuncs:

  bpf_netpoll_create()   - Allocate and set up a netpoll context
                           (sleepable, SYSCALL prog type only)
  bpf_netpoll_acquire()  - Acquire a reference to a netpoll context
  bpf_netpoll_release()  - Release a reference (cleanup via
                           queue_rcu_work since netpoll_cleanup sleeps)
  bpf_netpoll_send_udp() - Send a UDP packet (any context, LSM prog
                           type only for now)

The implementation follows the established kfunc lifecycle pattern
(create/acquire/release with refcounting, kptr map storage, dtor
registration). The netpoll context is wrapped in a refcounted
bpf_netpoll struct. Cleanup is deferred via queue_rcu_work() because
netpoll_cleanup() takes rtnl_lock.

AI was used to generate the code and each line was manually reviewed.

Reviewed-by: Mahe Tardy <mahe.tardy@gmail.com>
Signed-off-by: Song Liu <song@kernel.org>
---
 drivers/net/Kconfig         |  11 ++
 include/linux/bpf_netpoll.h |  38 +++++++
 kernel/bpf/verifier.c       |   3 +
 net/core/Makefile           |   1 +
 net/core/bpf_netpoll.c      | 209 ++++++++++++++++++++++++++++++++++++++++++
 5 files changed, 262 insertions(+)
 create mode 100644 include/linux/bpf_netpoll.h
 create mode 100644 net/core/bpf_netpoll.c
```

How does **bpf_netpoll** works?

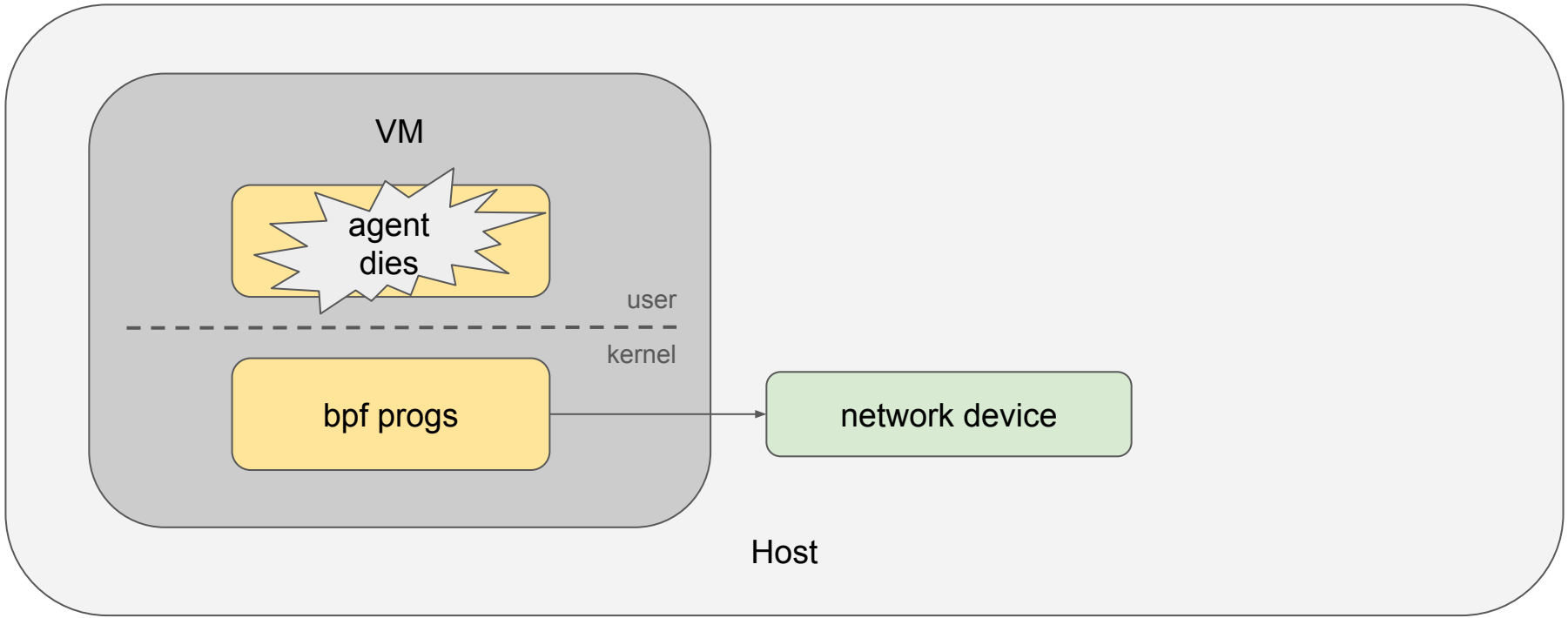
```c
SEC("syscall")
int netpoll_setup(void *ctx)
{
    struct bpf_netpoll_opts opts = {};
    struct bpf_netpoll *bnp;

    /* ... fill opts ... */

    bnp = bpf_netpoll_create(&opts, sizeof(opts), &err);
    if (!bnp)
        return 1;

    err = netpoll_ctx_insert(bnp);
    if (err && err != -EEXIST)
        return 1

    return 0;
}
```

```c
SEC("lsm/file_open")
int BPF_PROG(netpoll_send, struct file *file)
{
    struct __netpoll_ctx_value *v;
    struct bpf_netpoll *bnp;

    /* ... */

    v = netpoll_ctx_value_lookup();
    if (!v)
        return 0;

    bpf_rcu_read_lock();
    bnp = v->ctx;
    if (!bnp) {
        bpf_rcu_read_unlock();
        return 0;
    }

    send_status = bpf_netpoll_send_udp(bnp, send_data, sizeof(send_data));
    bpf_rcu_read_unlock();
    return 0;
}
```

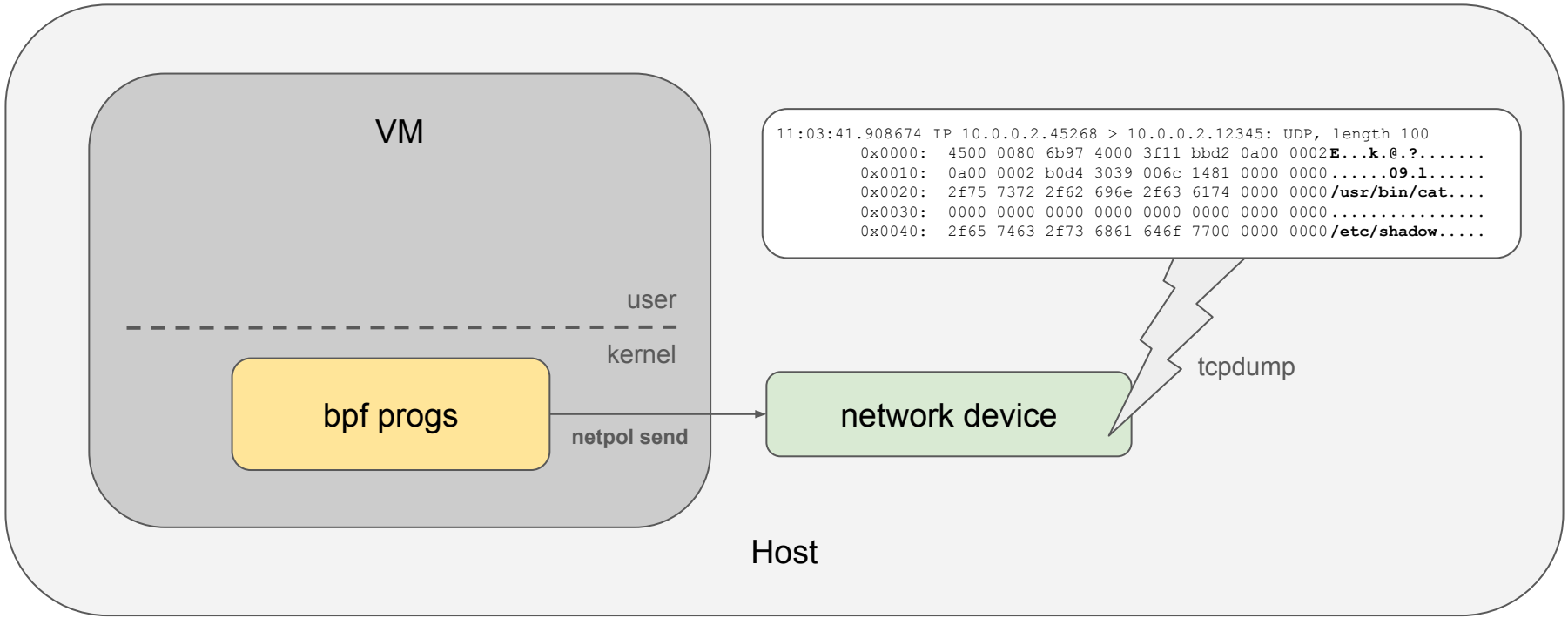
Tetragon **bpf_netpoll** use case



Tetragon **bpf_netpoll** use case



Tetragon **bpf_netpoll** use case



Tetragon **bpf_netpoll** use case

- See the demo!

This can be combined with **bpf_crypto**

- Combined with `bpf_crypto` kfuncs we can send encrypted UDP packet from the BPF progs without any agents.
- See the demo!

Fresh patch

```
commit 7808a7d77aebb7a729daa8aba3e8a749f9e59496
Author: Song Liu <song@kernel.org>
Date:   5 minutes ago
    bpf: Add TCP socket kfuncs for one-way client communication

    Add BPF kfuncs for one-way client TCP sockets:
      - bpf_sock_create: create a kernel TCP socket
      - bpf_sock_connect: connect to a remote address
      - bpf_sock_sendmsg: send data (sleepable)
      - bpf_sock_setsockopt: set socket options (e.g. kTLS)
      - bpf_sock_release: release socket (sleepable, synchronous)

    Use struct socket directly as the kptr type — no wrapper struct
    needed.  bpf_sock_release is synchronous; the map dtor is safe
    because map destruction runs from a workqueue.  BPF programs
    that need deferred release can use bpf_wq instead.
```