Backpressure for Sockmap based Redirection

LSF/MM/BPF 26, Zagreb | Hemanth Malla

Azure Container Networking

Whats Sockmap being used for ?

- L7 Parsing
- Socket based Policing
- Tetragon
- Meta
- Cilium (A long time ago)

Whats Sockmap being used for ?

- L7 Parsing
- Socket based Policing
- Tetragon
- Meta
- Cilium (A long time ago)

But not many users for redirection in prod !

Motivation - Cilium

Next great local pod

Why Not ?

- TCP Backpressure completely broken
- Misc. Bugs
- Performance issues with small messages

8 Years Later lots of bugs are ironed out..

2026 (through May)

- Jan 24 — `[PATCH bpf-next v9 1-3/3]` Fix incorrect `copied_seq` calculation; Fix `FIONREAD` for sockmap (Jiayuan Chen) — link
- Jan 29 — `bpf, sockmap: Fix af_unix null-ptr-deref` Luczaj) — link
- Apr 1 — `sockmap: Fix use-after-free of sk->sk_soc` `sk_psock_verdict_data_ready()` (Kuniyuki Iwashima)
- Apr 14 — `[PATCH bpf v4 0/5] bpf, sockmap: Fix af_` `update` (annotate af_unix `sk_state` data-races; fix af_u in proto update; take state lock for af_unix iter) — thread
- Apr 20 — `sockmap: Fix sk_psock_drop() race vs` `sock_map_{unhash,close,destroy}()` (Kuniyuki Iwash
- Apr 23 — `bpf, sockmap: Fix wrong rsge offset in b` (Bestswng) — link

2025

- Feb 13 — `[PATCH net 0/4] sockmap, vsock: For connectible sockets allow only connected` (Michal Luczaj; warn on transport-less vsock) — thread
- Feb 19 — `bpf: fix ktls panic with sockmap` (Jiayuan Chen) — link
- Apr 7 — `[PATCH bpf-next v1 0/4] bpf, sockmap: Fix data loss and panic issues` (Jiayuan Chen; data lost during EAGAIN retries; duplicated data transmission; panic in

`ssing uncharge operation` (Jiayuan Chen) — link

`urrency issues between memory charge and`

`sing sk_socket after free when sending` (Jiayuan

`incorrectly pointing to sk` (Jiayuan Chen) —

`refcnt race after locking changes` (Pranav

`ointer dereference in sk_msg_recvmsg()` (Pranav

`ee() when tcp_bpf_send_verdict() fails to`
Iwashima) — link

`tcp: Fix conflicts between MPTCP and sockmap`
ckmap; proto fallback detection with BPF sockmap) —

2024

- Feb 18 — `bpf, sockmap: Fix NULL pointer dereference in sk_psock_verdict_data_ready()` (Shigeru Yoshida) → **CVE-2024-26731** — link
- Apr 2 — `bpf, sockmap: Prevent lock inversion deadlock in map delete elem` (Jakub Sitnicki) *(reverted in May, see below)* — link
- Apr 4 — `bpf, skmsg: fix NULL pointer dereference in sk_psock_skb_ingress_enqueue` (Jason Xing) — link
- May 3 — `net: sockmap, fix missing MSG_MORE causing TCP disruptions` (Fastabend) — link
- May 24 — `sock_map: avoid race between sock_map_close and sk_psock_put` (Thadeu Cascardo, via net) — link
- May 27 — `[PATCH bpf 0/3] Block deletes from sockmap for tracing programs` (allow delete only if update is allowed; revert lock-inversion fix from Apr 2) — thread
- Jun 25 — `bpf: sockmap, fix introduced strparser recursive lock` (Fastabend) — link
- Jul 2 — `skmsg: skip zero length skb in sk_msg_recvmsg` (Geliang Tang, via net) — link
- Jul 17 — `bpf, skmsg: Fix NULL pointer dereference in sk_psock_skb_ingress_enqueue` (Ashwin Kamat) — link
- Sep 6 — `sock_map: add a cond_resched() in sock_hash_free()` (Eric Dumazet) — link
- Oct 13 — `[PATCH bpf v2 0/4] bpf, vsock: Fixes related to sockmap/sockhash redirection` (SK_DROP on unsupported af_vsock; `rx_bytes` / `msg_count` updates on `read_skb`) — thread
- Oct 17 — `[PATCH bpf 0/2] tcp_bpf: Fix the sk_mem_uncharge logic in`

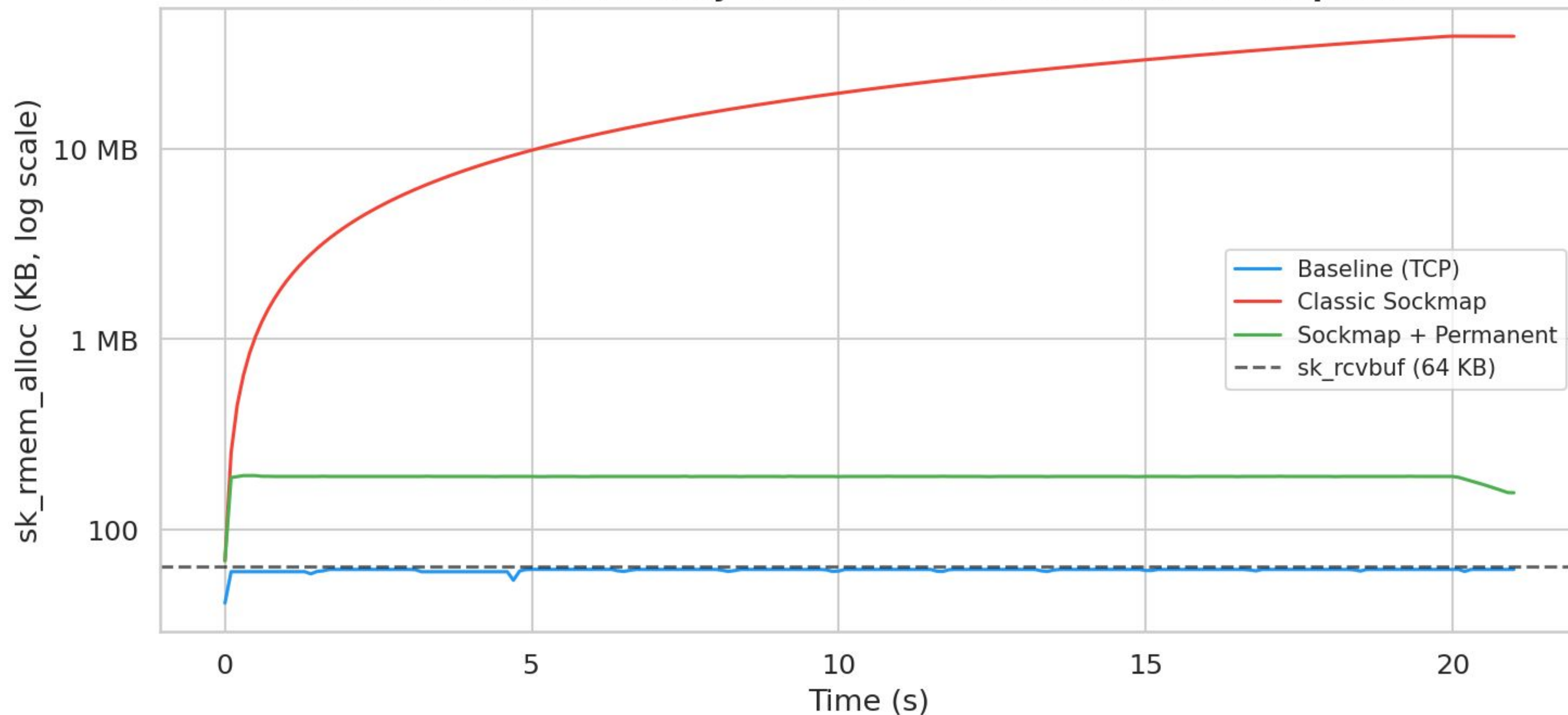
Why Not?

- TCP Backpressure completely broken
- Misc. Bugs
- Performance issues with small messages

Quick Test

- Two Producers
  - Slow - 200 KB/s
  - Fast - 2000 KB/s
- Slow Consumer

Quick Tests - Unbounded Memory growth



Why ?

kernel: sockmap support for blocking redirects #6438

Closed

jrfastab opened on Dec 11, 2018 Member

Sockmap doesn't support blocking the rational is that it could induce head of line blocking. However in the point to point case (or if the BPF author "knows" its ok) its useful to introduce a blocking flag to the redirect API.

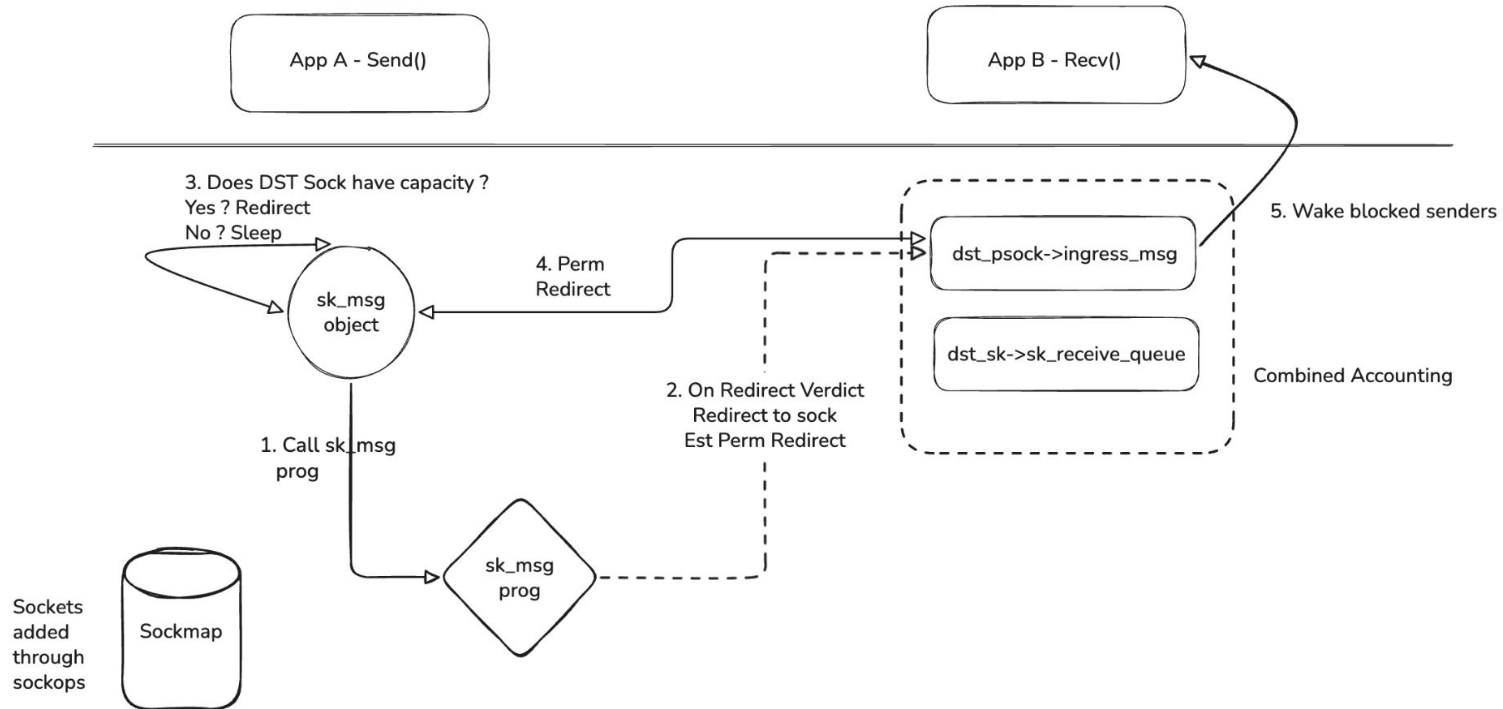
Create sub-issue

Solution - BPF_F_PERMANENT

```
SEC("sk_msg")
int redirect_prog(struct sk_msg_md *msg)
{
    int key = 1;

    return bpf_msg_redirect_hash(msg, &sock_hash, &key,
                                 BPF_F_INGRESS | BPF_F_PERMANENT);
}
```

Solution - BPF_F_PERMANENT



Solution - BPF_F_PERMANENT

- The first SK_MSG verdict runs BPF normally.
- Reserves a source-to-target link if bpf prog returns perm redirect
- The first redirect must queue bytes before the link becomes active.
- Once active, future sends on the source socket bypass bpf prog and go directly to the target.
- If the target receive buffer is full -> sender waits on the target socket.
- When the target application calls recv(), receive memory is uncharged and the blocked sender is woken.

Reserve then commit ?

Should we wait till first successful data transfer ?

- Bpf_msg_redirect_map / bpf_msg_redirect_hash could be called multiple times from bpf prog
- Reserve from sk_psock_msg_verdict
- Commit from tcp_bpf_send_verdict ?

sleep + wake_up_interruptible

- Check space on dst socket in bpf_tcp_ingress
  - On buffer full, register on target socket's waitqueue and sleep
- Wake up blocked sender in __sk_msg_recvmsg()

SK_MSG vs SK_SKB

- sk_msg in sender context
- sk_skb in receiver context
  - Sender cannot sleep
  - Sender may be remote host
  - Rely on backpressure from TCP stack on source socket buffer fill up
  - Check for dst buffer capacity before converting to sk_msg
  - On recv drain from src buffer to dst psock queue
    - Schedule backlog worker / inline ?

kTLS special handling ?

- Do we need to wait for full msg size in target buffer ?

Sockmap as the UX for redirection ?

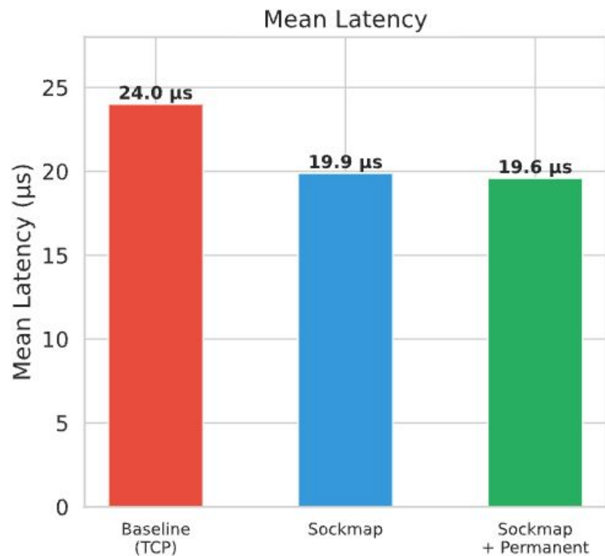
Can we decouple from the map ?

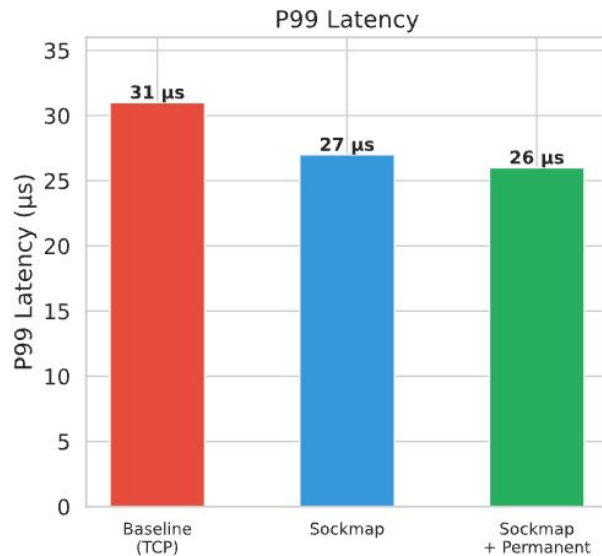
Do we care about one to many ?

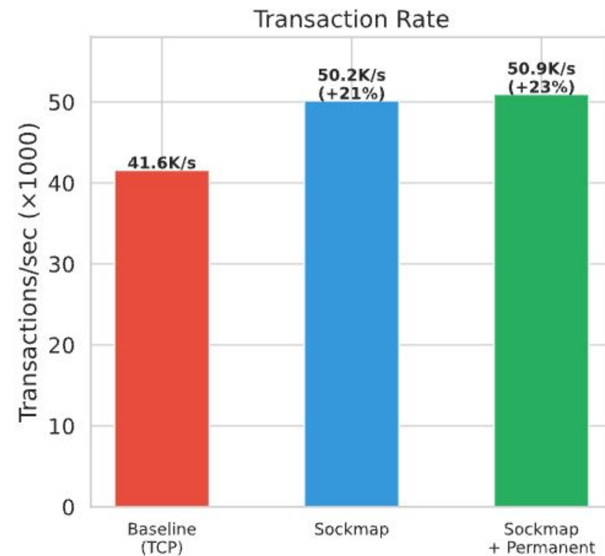
Other Bugs / Blockers for adoption ?

PoC Results - TCP_RR

TCP_RR Summary (1B request/response)

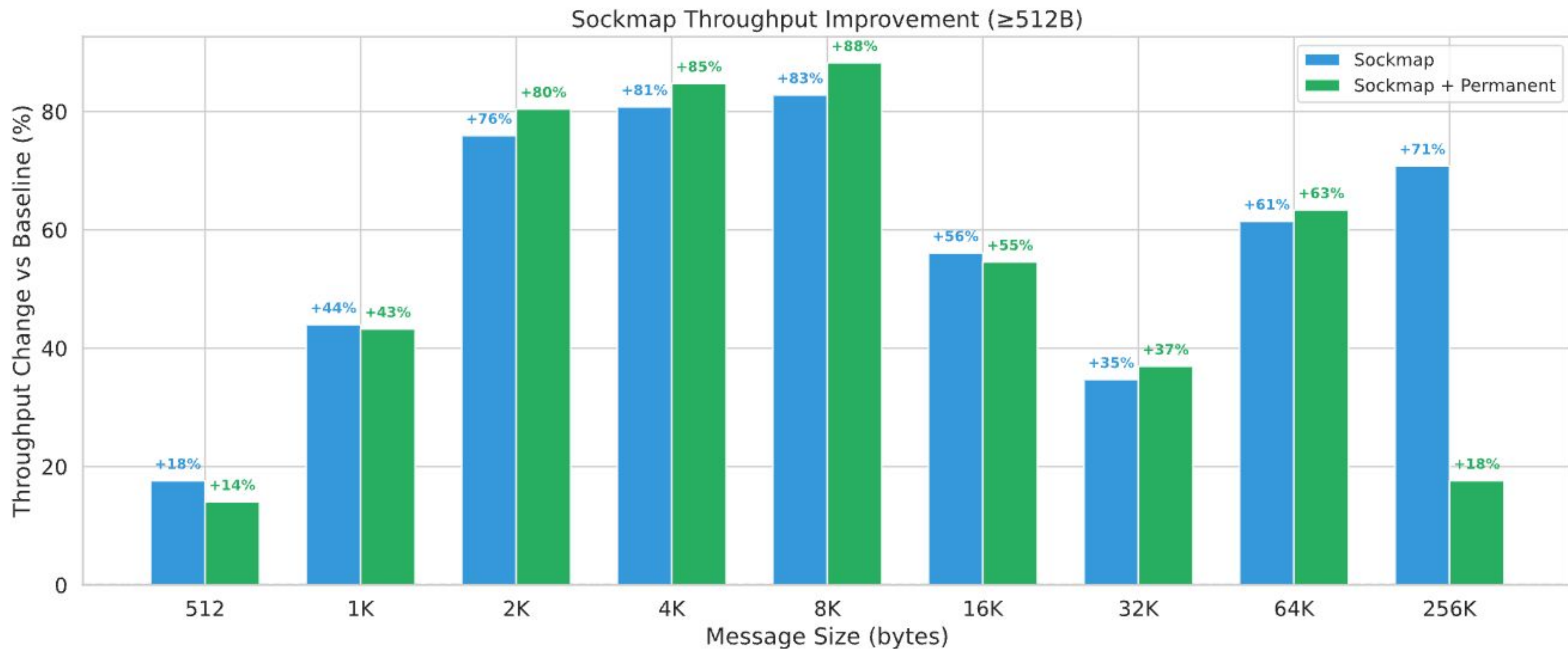




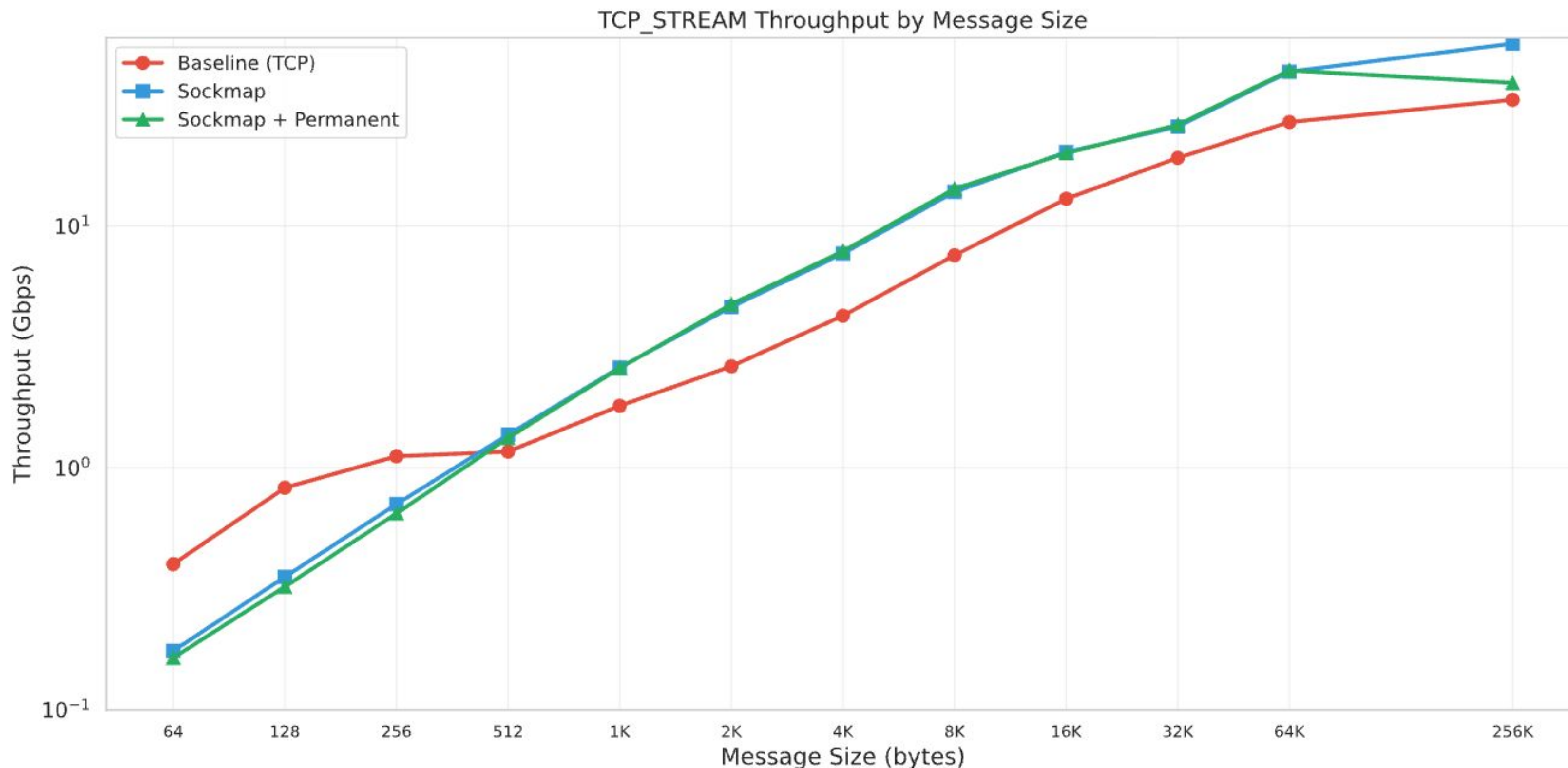


Also see old benchmarks from [Cilium 1.4]

PoC Results - TCP_STREAM



PoC Results - TCP_STREAM



tcp_bpf: improve ingress redirection performance with message corking

From: Cong Wang <xiyou.wangcong-AT-gmail.com>
To: netdev-AT-vger.kernel.org
Subject: [Patch bpf-next v6 0/4] tcp_bpf: improve ingress redirection performance with message corking
Date: Thu, 15 Jan 2026 11:27:33 -0800
Message-ID: <20260115192737.743857-1-xiyou.wangcong@gmail.com>
Cc: hemanthmalla-AT-gmail.com, john.fastabend-AT-gmail.com, jakub-AT-cloudflare.com, zijianzhang-AT-bytedance.com, bpf-AT-vger.kernel.org, Cong Wang <xiyou.wangcong-AT-gmail.com>
Archive-link: Article

This patchset improves skmsg ingress redirection performance by a) sophisticated batching with kworker; b) skmsg allocation caching with kmem cache.

As a result, our patches significantly outperforms the vanilla kernel in terms of throughput for almost all packet sizes. The percentage improvement in throughput ranges from 3.13% to 160.92%, with smaller packets showing the highest improvements.

For latency, it induces slightly higher latency across most packet sizes compared to the vanilla, which is also expected since this is a natural side effect of batching.

Here are the detailed benchmarks:

Throughput	64	128	256	512	1k	4k	16k	32k	64k	128k	256k
Vanilla	0.17±0.02	0.36±0.01	0.72±0.02	1.37±0.05	2.60±0.12	8.24±0.44	22.38±2.02	25.49±1.28	43.07±1.36	66.87±4.14	73.70±7.15
Patched	0.41±0.01	0.82±0.02	1.62±0.05	3.33±0.01	6.45±0.02	21.50±0.08	46.22±0.31	50.20±1.12	45.39±1.29	68.96±1.12	78.35±1.49
Percentage	141.18%	127.78%	125.00%	143.07%	148.08%	160.92%	106.52%	97.00%	5.38%	3.13%	6.32%

Latency	64	128	256	512	1k	4k	16k	32k	63k
Vanilla	5.80±4.02	5.83±3.61	5.86±4.10	5.91±4.19	5.98±4.14	6.61±4.47	8.60±2.59	10.96±5.50	15.02±6.78
Patched	6.18±3.03	6.23±4.38	6.25±4.44	6.13±4.35	6.32±4.23	6.94±4.61	8.90±5.49	11.12±6.10	14.88±6.55
Percentage	6.55%	6.87%	6.66%	3.72%	5.68%	4.99%	3.49%	1.46%	-0.93%

Thank You