DDOS MITIGATION WITH NETMAP AND RUST

45 days later

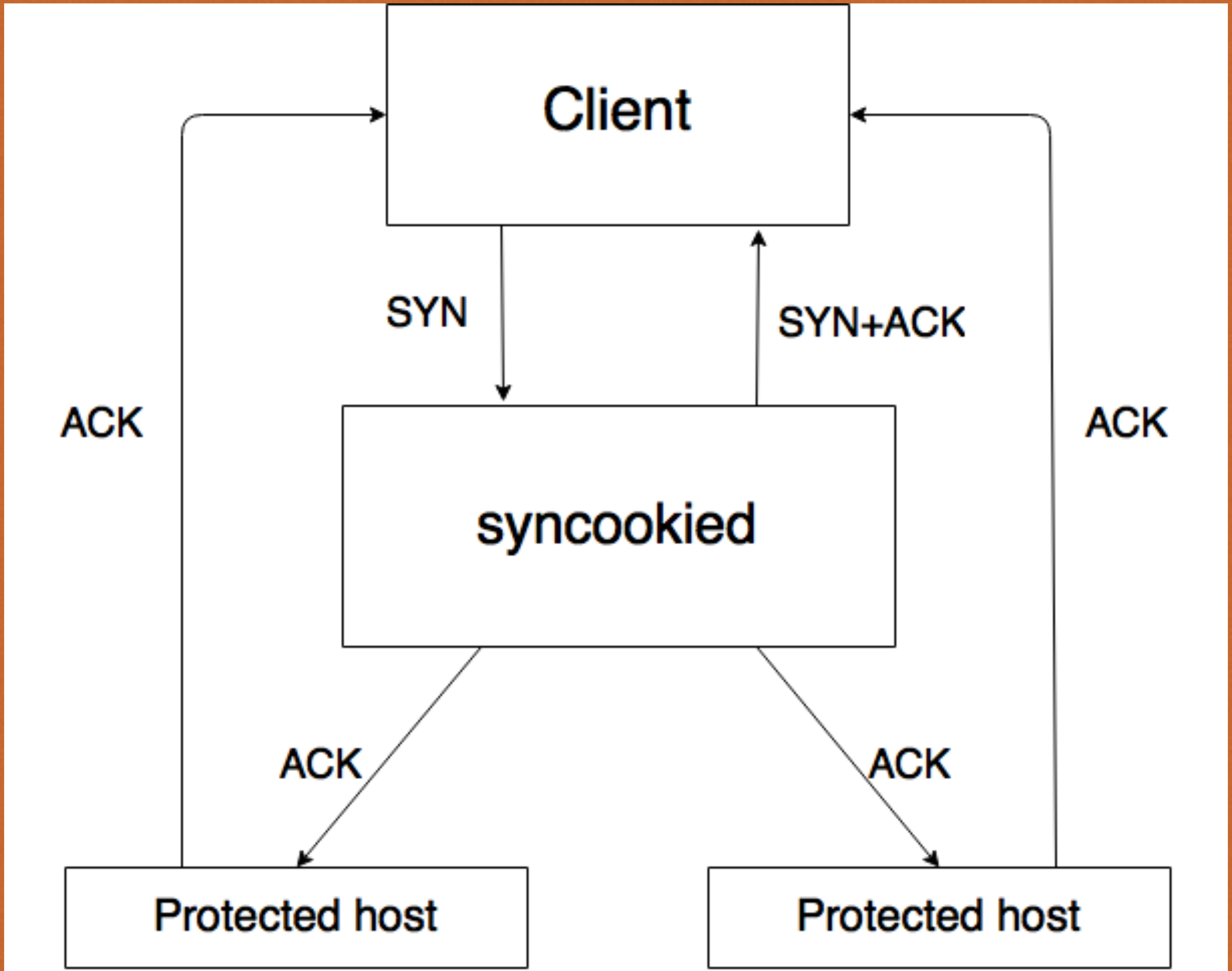
WHAT'S THE PROBLEM AGAIN?

- TCP 3-way handshake
- kernel can't handle the load

HOW DO WE SOLVE IT?

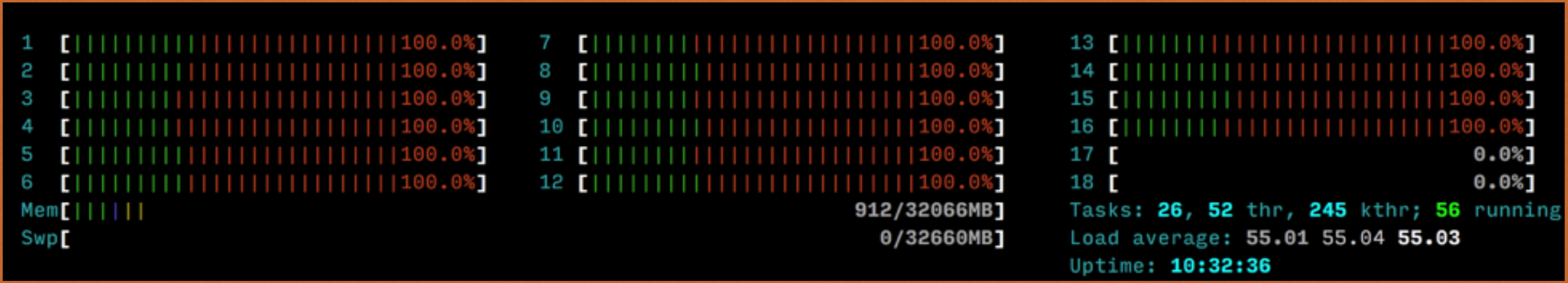
- developing our own solution from scratch using userland networking

HOW DO WE SOLVE IT?



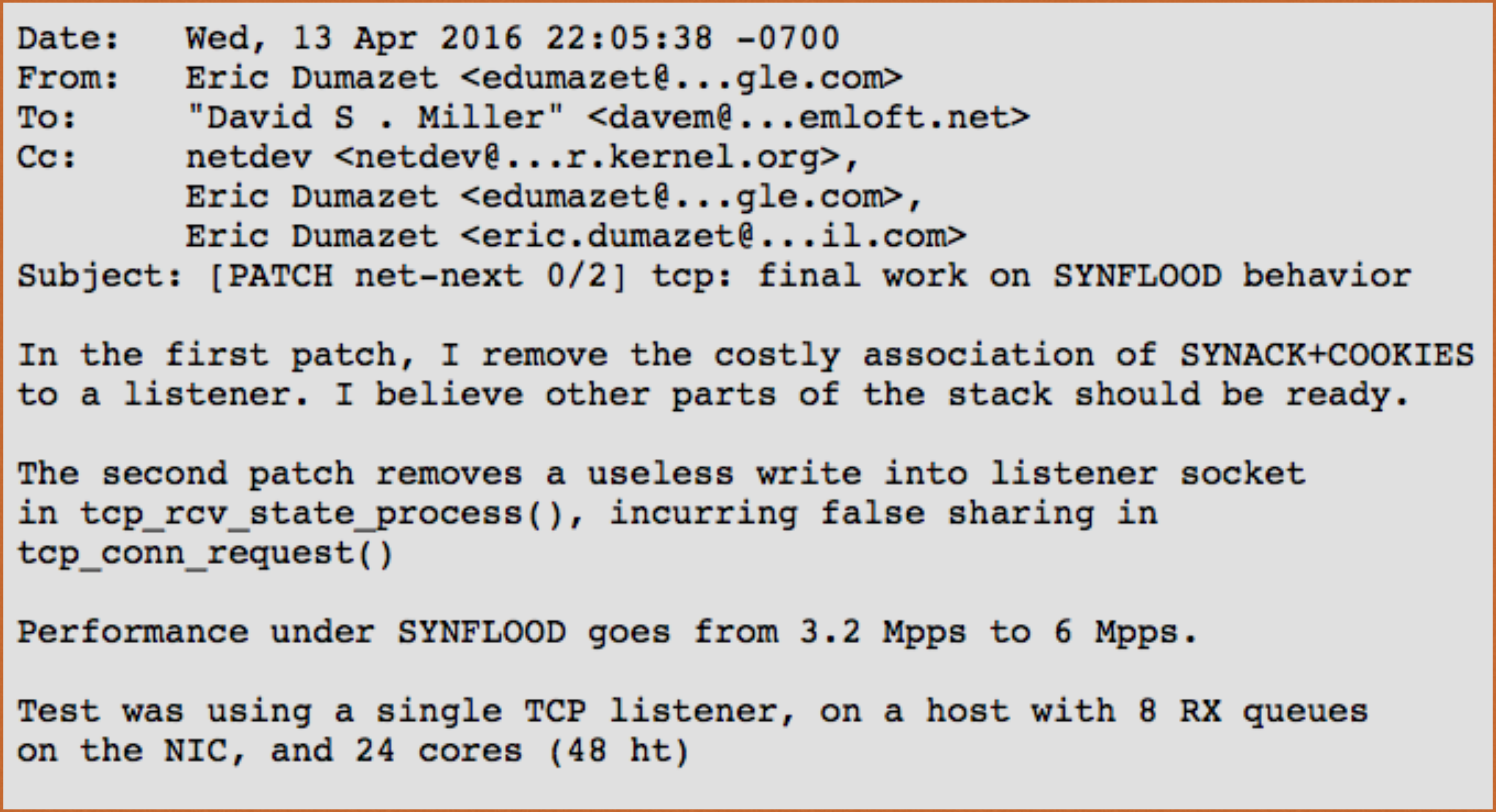
WHERE WE LEFT LAST TIME

- 5M SYN packets, 16 cores utilised



```
1  [||||||||||||||||||||||||||100.0%]    7  [||||||||||||||||||||||||||100.0%]    13 [||||||||||||||||||||||||||100.0%]
2  [||||||||||||||||||||||||||100.0%]    8  [||||||||||||||||||||||||||100.0%]    14 [||||||||||||||||||||||||||100.0%]
3  [||||||||||||||||||||||||||100.0%]    9  [||||||||||||||||||||||||||100.0%]    15 [||||||||||||||||||||||||||100.0%]
4  [||||||||||||||||||||||||||100.0%]    10 [||||||||||||||||||||||||||100.0%]    16 [||||||||||||||||||||||||||100.0%]
5  [||||||||||||||||||||||||||100.0%]    11 [||||||||||||||||||||||||||100.0%]    17 [                            0.0%]
6  [||||||||||||||||||||||||||100.0%]    12 [||||||||||||||||||||||||||100.0%]    18 [                            0.0%]
Mem[||||||                                                        912/32066MB]    Tasks: 26, 52 thr, 245 kthr; 56 running
Swp[                                                                0/32660MB]    Load average: 55.01 55.04 55.03
                                                                                  Uptime: 10:32:36
```

A CHALLENGER APPEARS



```
Date:    Wed, 13 Apr 2016 22:05:38 -0700
From:    Eric Dumazet <edumazet@...gle.com>
To:      "David S . Miller" <davem@...emloft.net>
Cc:      netdev <netdev@...r.kernel.org>,
         Eric Dumazet <edumazet@...gle.com>,
         Eric Dumazet <eric.dumazet@...il.com>
Subject: [PATCH net-next 0/2] tcp: final work on SYNFLOOD behavior

In the first patch, I remove the costly association of SYNACK+COOKIES
to a listener. I believe other parts of the stack should be ready.

The second patch removes a useless write into listener socket
in tcp_rcv_state_process(), incurring false sharing in
tcp_conn_request()

Performance under SYNFLOOD goes from 3.2 Mpps to 6 Mpps.

Test was using a single TCP listener, on a host with 8 RX queues
on the NIC, and 24 cores (48 ht)
```

CHALLENGE
ACCEPTED
memegenerator.net

WHY SO SLOW?

- we're so slow, kernel developers are catching up
- that's no good

WHY SO SLOW?

- netmap generic driver
- using host ring
- locks
- channels
- rebuilding packets each time

NETMAP GENERIC DRIVER

- why is it slow?
- packets still go through linux network stack
- solution: use native driver
- problem: doesn't work at first
- solution: read the source, drop host ring

HOST RING

- netmap has a concept of "host ring"
- used to inject packets back into linux network stack
- problem: can only have one per interface, with 12 queues this is gonna be a contention point
- solution: get rid of it, use a dedicated host and L2-forwarding

LOCKS

- want multiple hosts behind our protection
- want hot config reload
- use a global "config" structure and protect it with locks
- it's mostly readonly anyway, that would be no problem, right?

LOCKS

- it's mostly readonly anyway, that would be no problem, right?
- wrong
- when you're dealing with 10M packets per second and have to look up things for every packet (twice)
- rust std uses wrappers around pthread locks
- these are not known for good performance
- no good for Chrome, definitely no good for us

LOCKS

- problem: no good for Chrome, definitely no good for us
- solution: replace all locks with parking_lot locks
- invented by Chrome people
- smaller in size, threads are kept in a separate structure from the lock
- adaptive: tries spinlock first

LOCKS

- much better
- still slow?
- thread local storage to the rescue
- copy necessary information to TLS
- sync with global configuration every 10 seconds ± some microseconds
- almost zero contention

CHANNELS

- rust channels, used to push packets from RX to TX thread

- basically a vector behind a mutex

- MUTEX?!

- yeah, we just removed these, and this is their comeback

CHANNELS

- problem: mutex strikes back
- solution: port rust std channels to parking_lot
- https://github.com/polachok/mpsc

CHANNELS

- https://github.com/polachok/mpsc
- better, but not still there yet
- we're not doing any "m" in mpsc
- let's go with fully lockless channels

CHANNELS

- let’s go with fully lockless channels
- there’s a crate for that: https://crates.io/crates/bounded-spsc-queue
- pretty good!
- still does copying (RX → Channel, Channel → TX)
- get rid of second copy, use a reference

REBUILDING PACKET

- we have to build a reply (SYN + ACK) packet for each SYN packet

- using libpnet for packet parsing and building

- great library overall

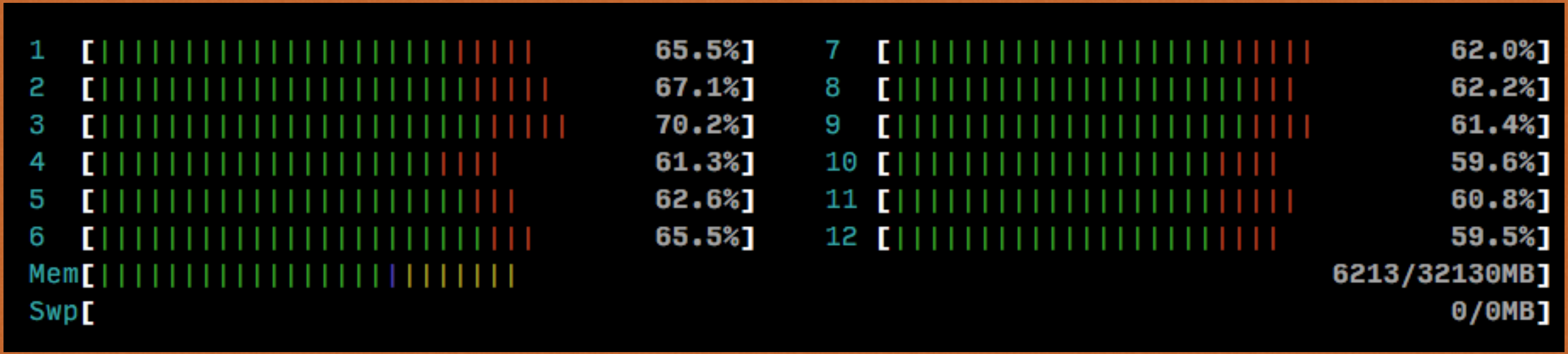
- uses byte level operations and shifts

REBUILDING PACKET

- note that the reply is 80% identical for each SYN
- let's build a template, copy it over and replace stuff we need to replace (ip, port, ack, seq)
- 30% faster

WHERE ARE WE NOW?

- 12.65M packets, 7.5 cores utilised (12 cores at ~60%)



```
1  [||||||||||||||||||||||||||        65.5%]    7  [|||||||||||||||||||||||||         62.0%]
2  [|||||||||||||||||||||||||||       67.1%]    8  [||||||||||||||||||||||||          62.2%]
3  [||||||||||||||||||||||||||||      70.2%]    9  [|||||||||||||||||||||||||         61.4%]
4  [||||||||||||||||||||||||          61.3%]    10 [|||||||||||||||||||||||           59.6%]
5  [|||||||||||||||||||||||||         62.6%]    11 [||||||||||||||||||||||||          60.8%]
6  [||||||||||||||||||||||||||        65.5%]    12 [|||||||||||||||||||||||           59.5%]
Mem[|||||||||||||||||||||||||                                                  6213/32130MB]
Swp[                                                                                  0/0MB]
```

WHERE ARE WE NOW?

- 12.65M packets, 7.5 cores utilised (12 cores at ~60%)
- multiple hosts
- live config reload
- pcap-style packet filtering ("tcp and dst port 80 or 22")
- cookie replies validated, states kept for each valid connection
- per-thread metrics in influx

Q&A