

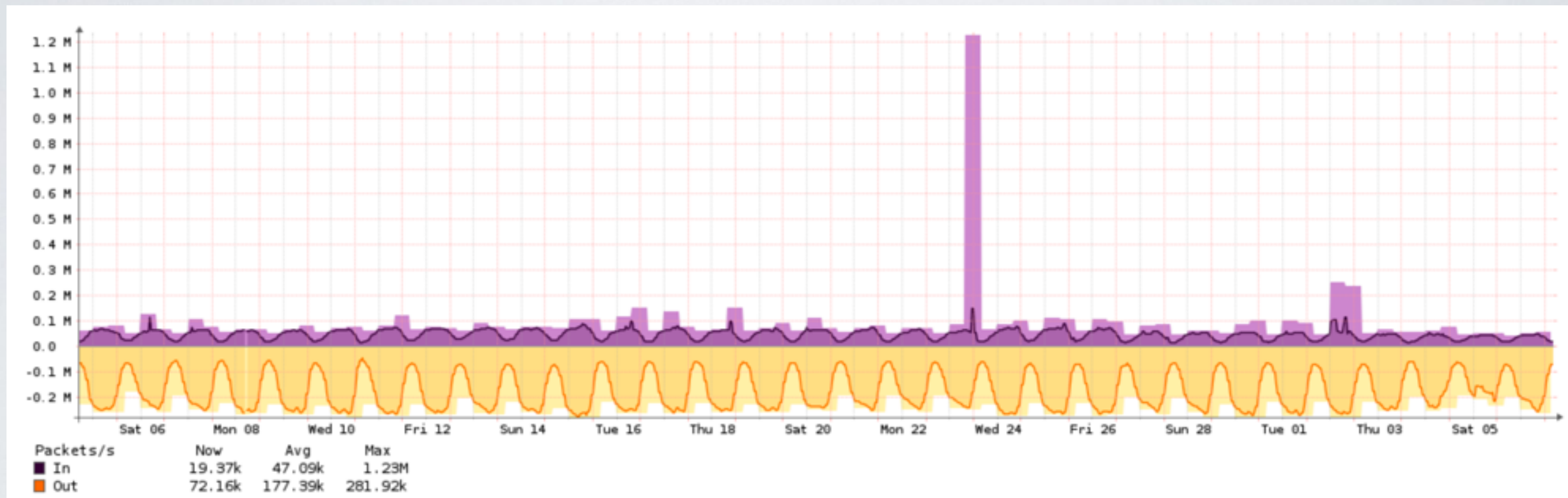


DDOS MITIGATION WITH NETMAP & RUST

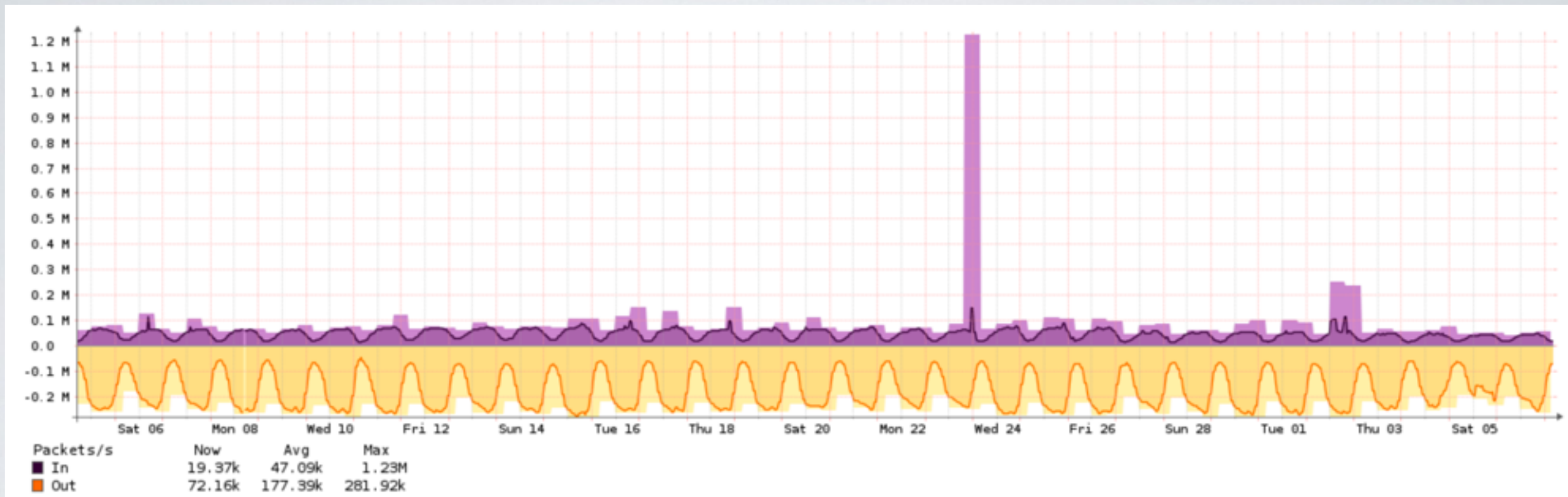
Alexander Polakov

WHAT'S THE PROBLEM?

WHAT'S THE PROBLEM?

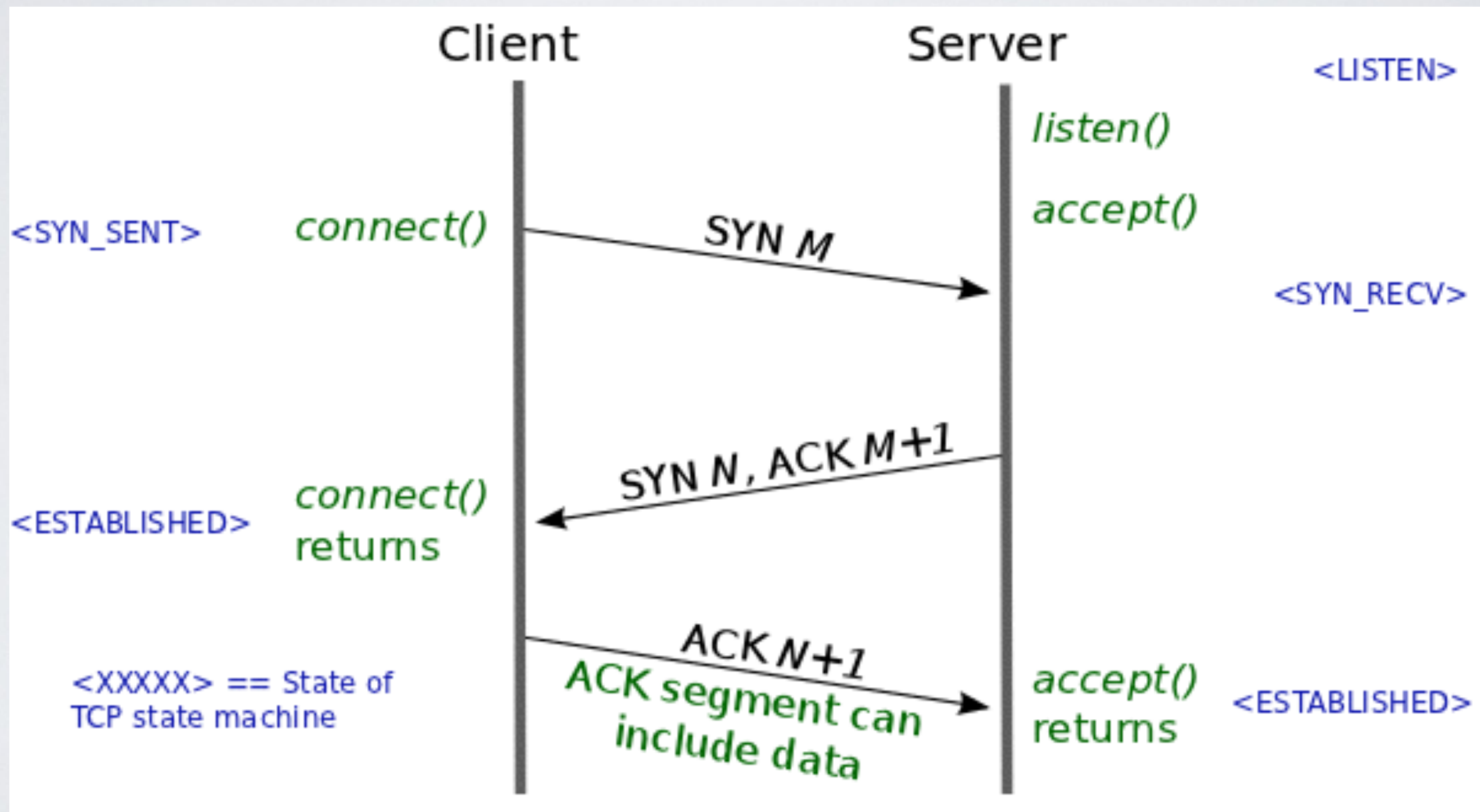


WHAT'S THE PROBLEM?



```
Request timeout for icmp_seq 0
Request timeout for icmp_seq 1
Request timeout for icmp_seq 2
Request timeout for icmp_seq 3
Request timeout for icmp_seq 4
Request timeout for icmp_seq 5
Request timeout for icmp_seq 6
Request timeout for icmp_seq 7
Request timeout for icmp_seq 8
Request timeout for icmp_seq 9
Request timeout for icmp_seq 10
Request timeout for icmp_seq 11
Request timeout for icmp_seq 12
Request timeout for icmp_seq 13
Request timeout for icmp_seq 14
```

TCP 3-WAY HANDSHAKE



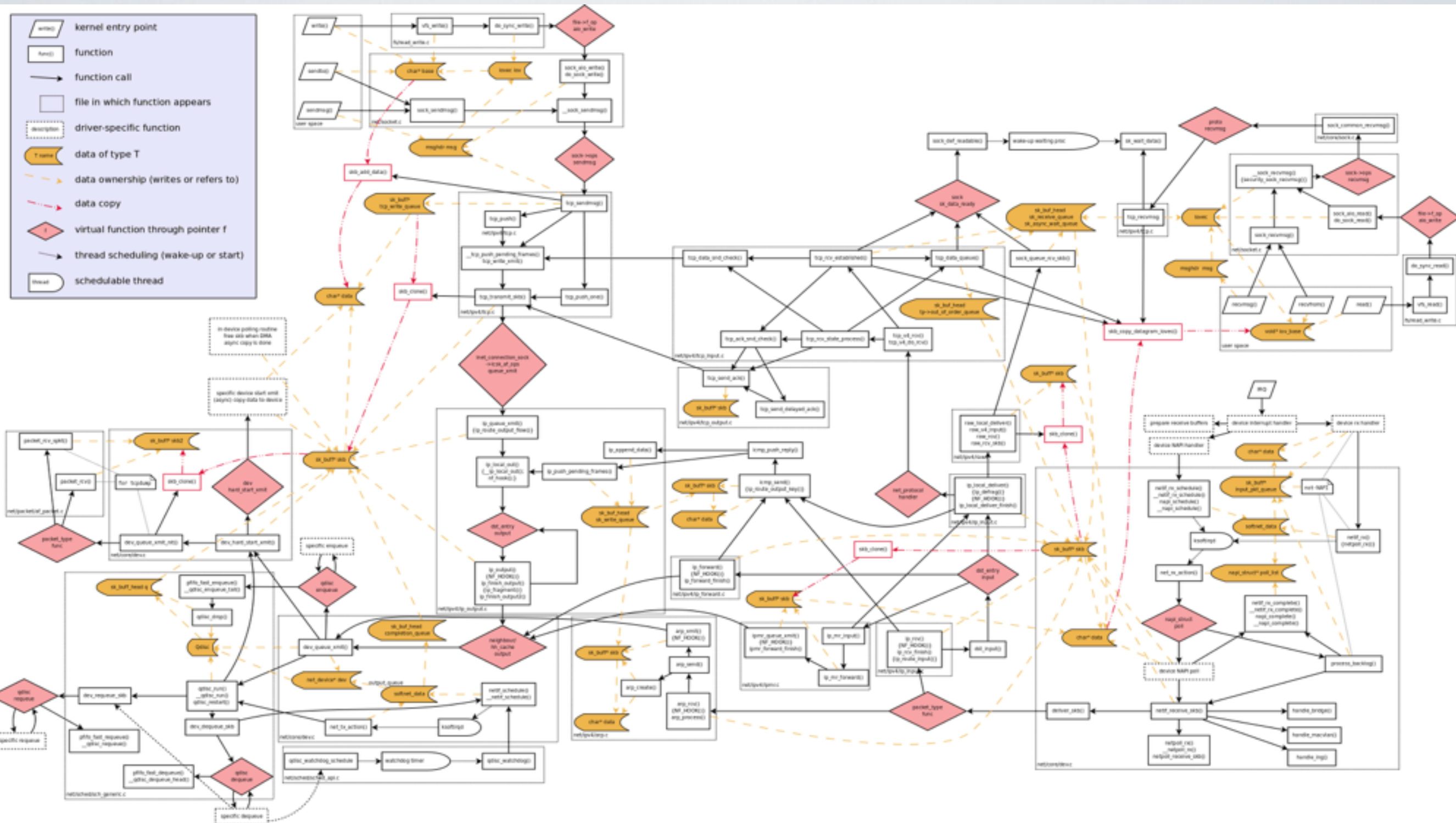
SYN COOKIES

- D.J. Bernstein, 1996

```
#[inline]
fn secure_tcp_syn_cookie(source_addr: u32, dest_addr: u32, source_port: u16,
                        dest_port: u16, sseq: u32, data: u32, tcp_cookie_time: u32) -> u32 {
    /*
     * Compute the secure sequence number.
     * The output should be:
     *   HASH(sec1,saddr,sport,daddr,dport,sec1) + sseq + (count * 2^24)
     *   + (HASH(sec2,saddr,sport,daddr,dport,count,sec2) % 2^24).
     * Where sseq is their sequence number and count increases every
     * minute by 1.
     * As an extra hack, we add a small "data" value that encodes the
     * MSS into the second hash value.
     */
    (Wrapping(cookie_hash(source_addr, dest_addr, source_port, dest_port, 0, 0))
     + Wrapping(sseq) + Wrapping(tcp_cookie_time << COOKIEBITS)
     + ((Wrapping(cookie_hash(source_addr, dest_addr, source_port, dest_port, tcp_cookie_time, 1))
        + Wrapping(data)) & Wrapping(COOKIEMASK))).0
}
```

STILL A PROBLEM?

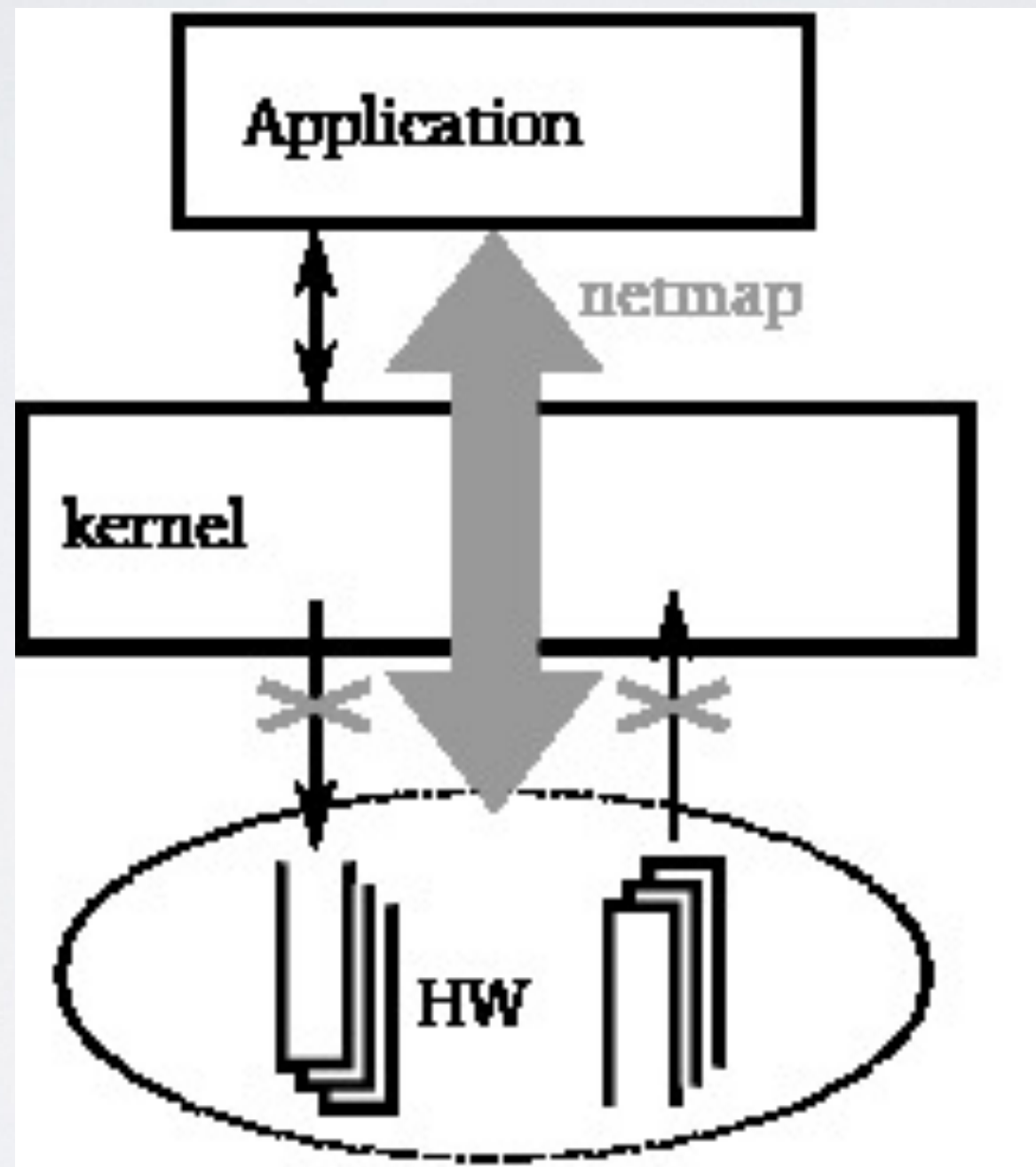
HOW LINUX NETWORK STACK WORKS



IDEA: KERNEL BYPASS

- multiple solutions
- intel dpdk
- netmap
- solarflare

HOW NETMAP WORKS

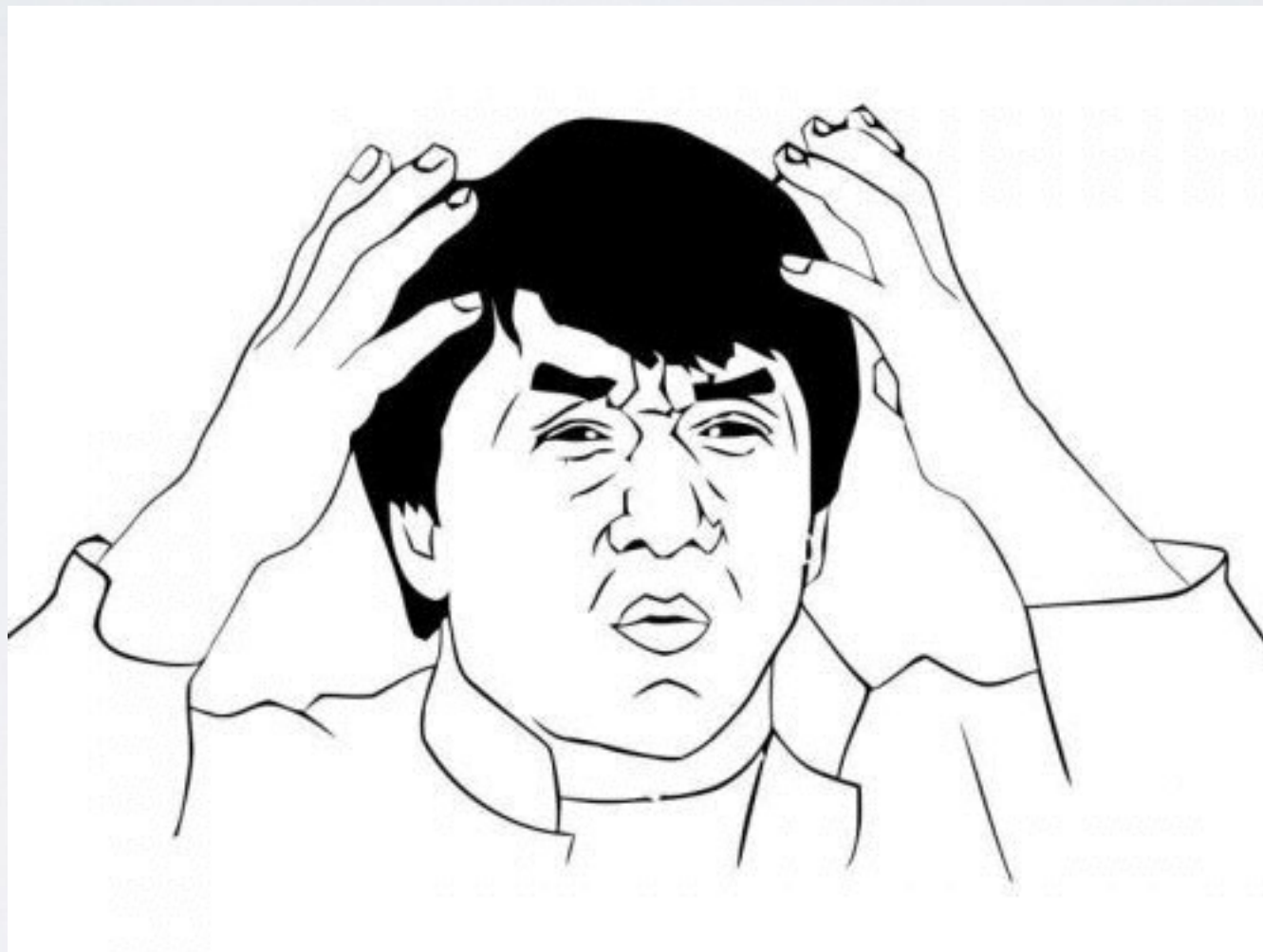


FIRST PROTOTYPE IN C++

- Alexey Manikin

```
[[~/Documents/src/syncookie_netmap]% find . -name '*.cpp' -o -name '*.h' |xargs -L -N1 cat|wc -l  
11019
```


FIRST PROTOTYPE IN C++



WHAT IS RUST?

- zero-cost abstractions
- move semantics
- guaranteed memory safety
- threads without data races
- trait-based generics
- pattern matching
- type inference
- minimal runtime
- efficient C bindings



WHAT IS RUST?

- stuff you know and love from functional languages (map/filter/fold) with performance of C for loops
- generics (partying like it's 2004)
- no null pointers or use after free
- thread-safe channels (bounded and unbounded)
- atomics, mutexes in standard library
- more cool stuff in crossbeam library
- borrow checker:
 - one or more references (**&T**) to a resource,
 - exactly one mutable reference (**&mut T**).
- package manager

















SECOND PROTOTYPE

```
2701 total  
[[~/Documents/src/syncookied]% wc -l src/*.rs  
  109 src/cookie.rs  
   33 src/csum.rs  
  271 src/main.rs  
  494 src/netmap.rs  
  229 src/packet.rs  
    5 src/sha1.rs  
    8 src/util.rs  
 1149 total
```

SECOND PROTOTYPE (CONT.)

- Thanks Robert Clipsham

 libpnet/libpnet Provide iterator interface for Vec fields ✓	3
#183 opened 5 days ago by polachok	
 libpnet/netmap_sys Fix types for req, ring and slot types ✓	1
#10 opened 5 days ago by polachok	
 libpnet/libpnet IPv4 flags ✓	6
#182 opened 9 days ago by polachok	
 libpnet/libpnet Use slices where possible as we are copying anyway ✓	3
#181 opened 9 days ago by polachok	
 alexcrichon/ctest Allow user-specified flags override default ✗	1
#7 opened 12 days ago by polachok	
 alexcrichon/ctest Make it possible to disable -Werror ✓	6
#6 opened 14 days ago by polachok	
 libpnet/netmap_sys Various fixes ✓	6
#9 opened 14 days ago by polachok	
 libpnet/netmap_sys Add ctest tests ✓	7
#8 opened 14 days ago by polachok	
 AlexeyManikin/syncookie_netmap Beget uptime module	0
#1 opened 16 days ago by polachok	
 libpnet/netmap_sys Add travis	3
#5 opened 16 days ago by polachok	
 polachok/netmap_sys Add travis ✓	0
#2 opened 16 days ago by polachok	
 polachok/netmap_sys Travis ✓	0
#1 opened 16 days ago by polachok	
 libpnet/netmap_sys Use gcc crate instead of homegrown hack ✓	5
#4 opened 17 days ago by polachok	
 libpnet/netmap_sys Use libc from crates.io	1
#3 opened 19 days ago by polachok	

ARCHITECTURE

- one thread per RX ring
- one thread per TX ring
- connected with bounded queue
(`sync::mpsc::sync_channel`)
- no locks
- no dynamic memory allocation

ARCHITECTURE: RX

- RX thread reads packet off the ring

```
loop {
  if let Some(_) = netmap.poll(netmap::Direction::Input) {
    for ring in netmap.rx_iter() {
      let mut fw = false;
      for (slot, buf) in ring.iter() {
        stats.received += 1;
        match packet::handle_input(buf) {
          Action::Drop => {
            stats.dropped += 1;
          },
          Action::Forward => {
            stats.forwarded += 1;
            slot.set_flags(netmap::NS_FORWARD as u16);
            fw = true;
          },
          Action::Reply(packet) => {
            stats.queued += 1;
            chan.send(packet);
          }
        }
      }
    }
    if fw {
      ring.set_flags(netmap::NR_FORWARD as u32);
    }
  }
}
```

ARCHITECTURE: RX

- RX thread parses packet into a struct and passes it to TX thread over a channel

```
pub struct IngressPacket {  
    pub ether_source: MacAddr,  
    pub ether_dest: MacAddr,  
    pub ipv4_source: Ipv4Addr,  
    pub ipv4_destination: Ipv4Addr,  
    pub tcp_source: u16,  
    pub tcp_destination: u16,  
    pub tcp_timestamp: [u8;4],  
    pub tcp_sequence: u32,  
    pub tcp_mss: u32  
}
```

ARCHITECTURE:TX

- TX thread receives packet from channel and builds a reply

```
loop {
  if let Some(_) = netmap.poll(netmap::Direction::Output) {
    for ring in netmap.tx_iter() {
      for (slot, buf) in ring.iter() {
        let pkt = chan.recv().expect("Expected RX not to die on us");
        let len = packet::handle_reply(pkt, buf);
        slot.set_flags(netmap::NS_BUF_CHANGED as u16 /* | netmap::NS_REPORT as u16
        slot.set_len(len as u16);
        stats.sent += 1;
      }
    }
  }
}
```

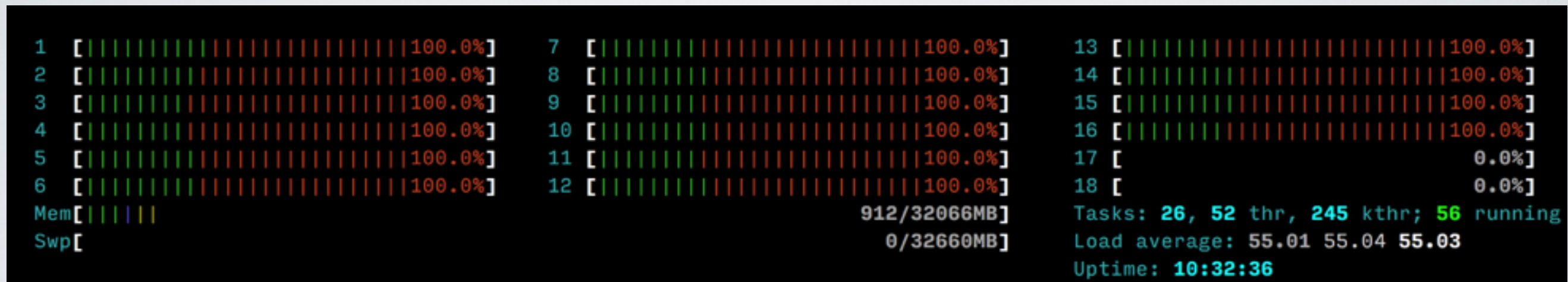

ARCHITECTURE:TX

```
let cookie_time = ::TCP_COOKIE_TIME.load(Ordering::Relaxed);
let (seq_num, mss_val) = cookie::generate_cookie_init_sequence(
    pkt.ipv4_source, pkt.ipv4_destination,
    pkt.tcp_source, pkt.tcp_destination, pkt.tcp_sequence,
    1460 /* FIXME */, cookie_time as u32);
let mut tcp = MutableTcpPacket::new(&mut ip.payload_mut()[0..20 + 24]).unwrap();
tcp.set_source(pkt.tcp_destination);
tcp.set_destination(pkt.tcp_source);
tcp.set_sequence(seq_num);
tcp.set_acknowledgement(pkt.tcp_sequence + 1);
tcp.set_window(65535);
tcp.set_syn(1);
tcp.set_ack(1);
tcp.set_data_offset(11);
tcp.set_checksum(0);
{
    let options = tcp.get_options_raw_mut();
    {
        let mut mss = MutableTcpOptionPacket::new(&mut options[0..4]).unwrap();
        mss.set_number(TcpOptionNumbers::MSS);
        mss.get_length_raw_mut()[0] = 4;
        let mss_payload = mss.payload_mut();
        mss_payload[0] = (mss_val >> 8) as u8;
        mss_payload[1] = (mss_val & 0xff) as u8;
    }
}
```

PERFORMANCE

- linux kernel 1.5M SYN packets per second
- 10M SYN packets per second with 30% loss
- 5M SYN packets per second with 0% loss
- ixgbe, 24 cores, 16 cores utilised
- needs improvement

PERFORMANCE



- 70% time in kernel

NETMAP DRIVERS

- 2 modes of operation
- generic: works with any driver/card, utilises kernel processing routines at cost of performance loss
- native: driver patched with netmap support, best performance

NETMAP DRIVERS

- use native driver for best performance, right?
- right
- the problem is...

IT DOESN'T WORK

IT DOESN'T WORK

with our card model

NETMAP DRIVERS

- there's an issue on netmap github
- investigating

Q&A