Scanning the Internet in < 5'

on a budget

Daniel Roethlisberger, Swisscom Cyber Defense
https://infosec.exchange/@droe, daniel@roe.ch
2024-09-27, BSides Ljubljana

swisscom

> «scanning entire Internet in under 5 minutes»
>
> — masscan

> «can scan the IPv4 address space in 5 minutes»
>
> — ZMap

443/tcp on 0.0.0.0/0

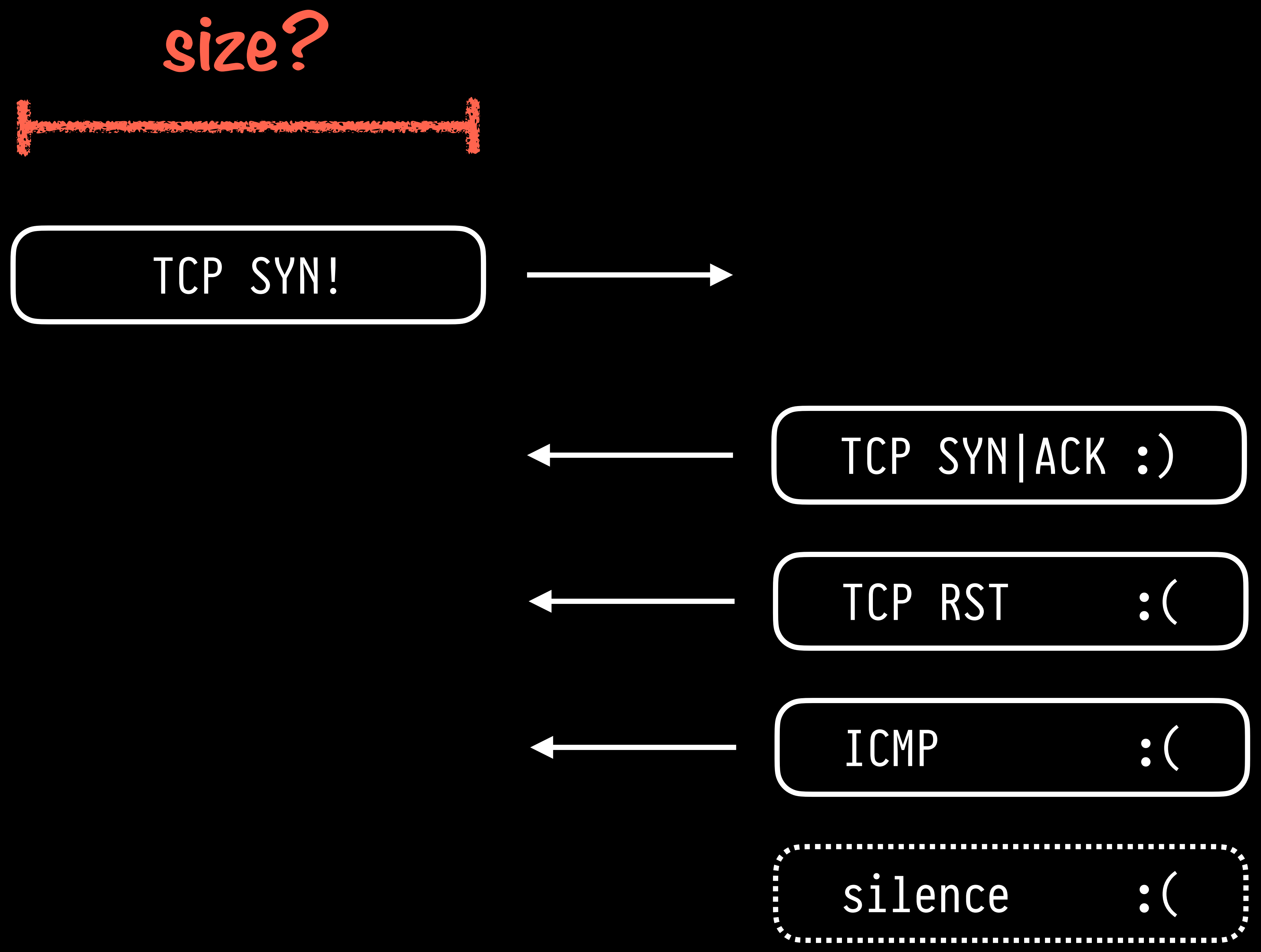
4'294'967'296 hosts

```
0.0.0.0/8           # RFC1122: "This host on this network"
10.0.0.0/8          # RFC1918: Private-Use
100.64.0.0/10       # RFC6598: Shared Address Space
127.0.0.0/8         # RFC1122: Loopback
169.254.0.0/16      # RFC3927: Link Local
172.16.0.0/12       # RFC1918: Private-Use
192.0.0.0/24        # RFC6890: IETF Protocol Assignments
192.0.2.0/24        # RFC5737: Documentation (TEST-NET-1)
192.88.99.0/24      # RFC3068: 6to4 Relay Anycast
192.168.0.0/16      # RFC1918: Private-Use
198.18.0.0/15       # RFC2544: Benchmarking
198.51.100.0/24     # RFC5737: Documentation (TEST-NET-2)
203.0.113.0/24      # RFC5737: Documentation (TEST-NET-3)
224.0.0.0/4         # RFC5771: Multicast/Reserved
240.0.0.0/4         # RFC1112: Reserved
255.255.255.255/32  # RFC0919: Limited Broadcast
```

3'702'258'431

~~4'294'967'296~~ hosts

at 10 Gbps



size?

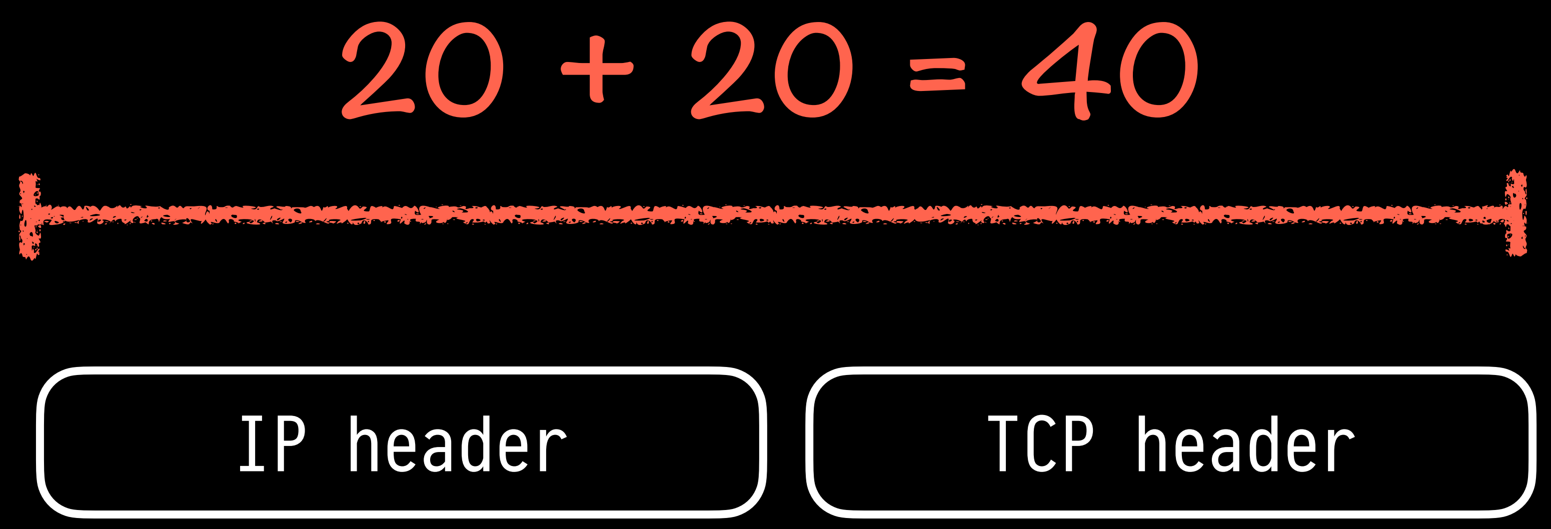
TCP SYN!

TCP SYN|ACK :)

TCP RST :(

ICMP :(

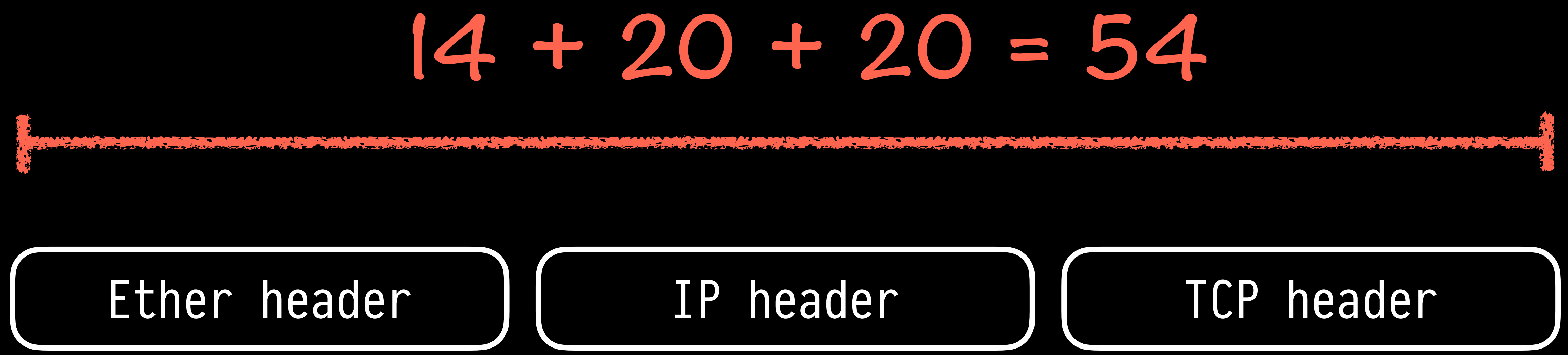
silence :(



20 + 20 = 40

IP header

TCP header

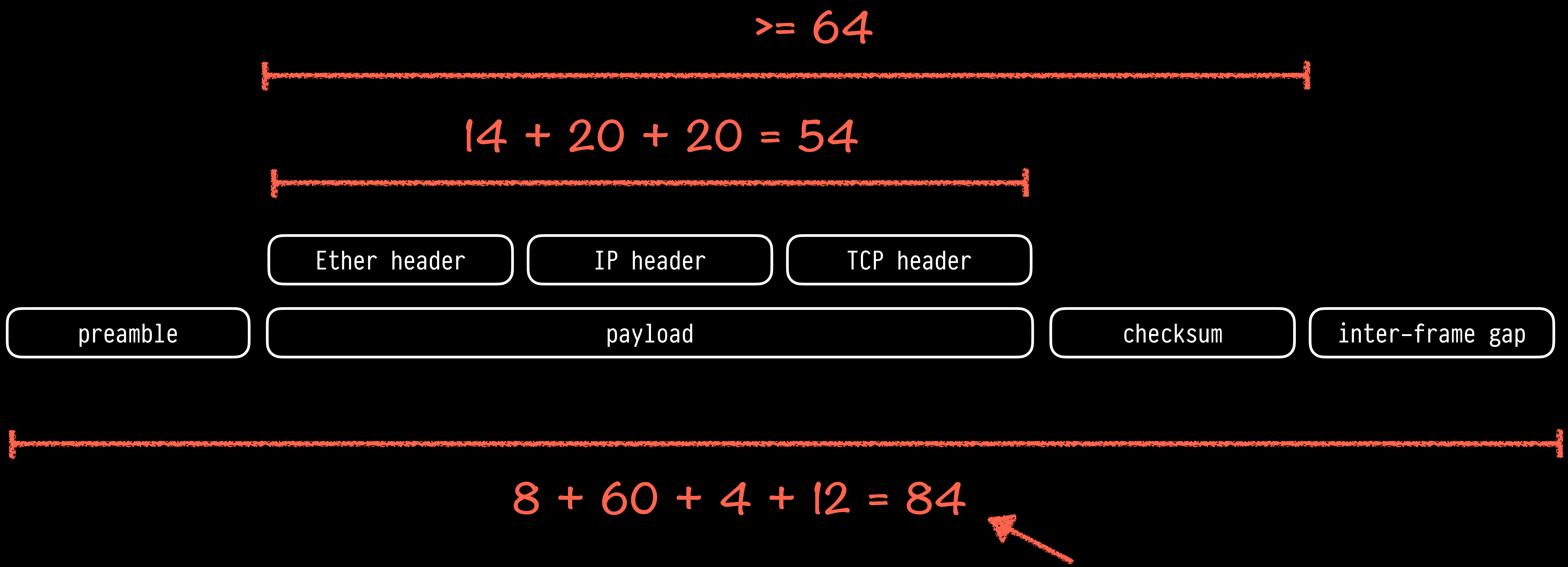


14 + 20 + 20 = 54

Ether header

IP header

TCP header



>= 64

14 + 20 + 20 = 54

Ether header

IP header

TCP header

preamble

payload

checksum

inter-frame gap

8 + 60 + 4 + 12 = 84

$$packetrate = \frac{bitrate}{packetsize} = \frac{10 \cdot 10^9}{84 \cdot 8} \approx 14.88 \cdot 10^6 \quad [\text{packets/s}]$$

$$packetrate = \frac{bitrate}{packetsize} = \frac{10 \cdot 10^9}{84 \cdot 8} \approx 14.88 \cdot 10^6 \quad \text{[packets/s]}$$

$$time = \frac{\#targets}{packetrate} = \frac{3'702'258'431}{14.88 \cdot 10^6} \approx 249 \quad \text{[s]}$$

14.88 Mp/s

Image source: https://www.youtube.com/theslowmoguys

Image source: https://www.youtube.com/theslowmoguys

Let's try this at home

CHF254.63 ~~CHF268.03~~ -5%

CHF4.58 off over CHF229.25 >

Qotom Q20331G9 Mini PC 5*2.5G I226-V Lan 4 SFP+ Atom Fileserver C3338R C3558R C3758 C3758R Firewall Router Mini PC Server

★★★½☆ 3.5 2 Reviews | 23 sold

Color: NO RAM NO M.2

NO RAM NO M.2 | 8G RAM 128G M.2 | 8G RAM 256G M.2 | 8G RAM 512G M.2 | 16G RAM 128G M.2 | 16G RAM 256G M.2 | 16G RAM 512G M.2 | 32G RAM 512G M.2

Bundle: Q20332G9 C3758

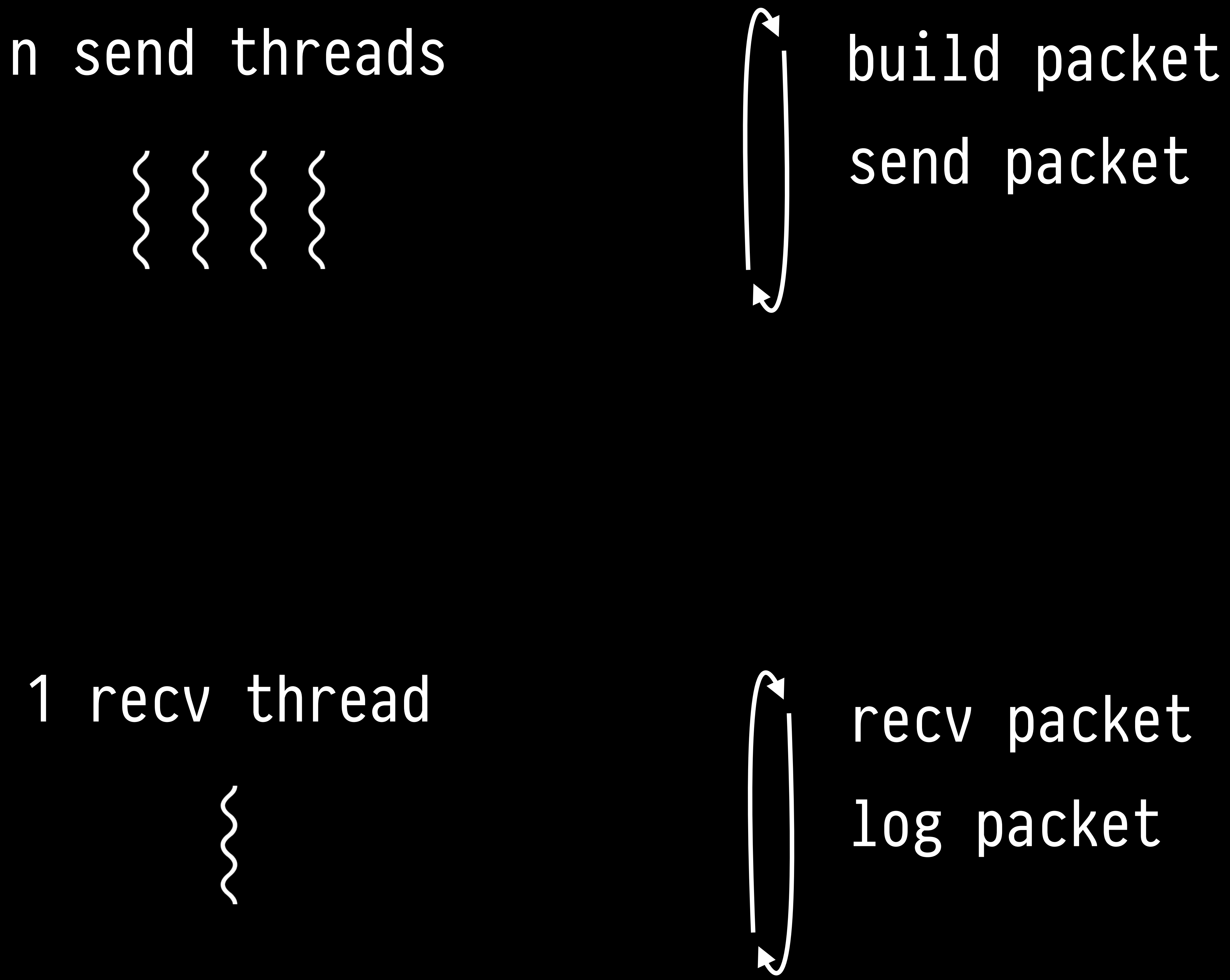
Q20311G9 C3338R | Q20321G9 C3558R | Q20332G9 C3758 | Q20331G9 C3758R

Image source: https://www.youtube.com/@ServeTheHomeVideo

```
% git clone https://github.com/zmap/zmap.git

% cd zmap && cmake . && make

% sudo src/zmap -p 443 --blocklist conf/blocklist.conf -i ix0 \
                -T 7 --cores 7,0,2,4,6,1,3,5 -r 1000000000
```

1.78 Mp/s

12% of 14.88 Mp/s



n send threads

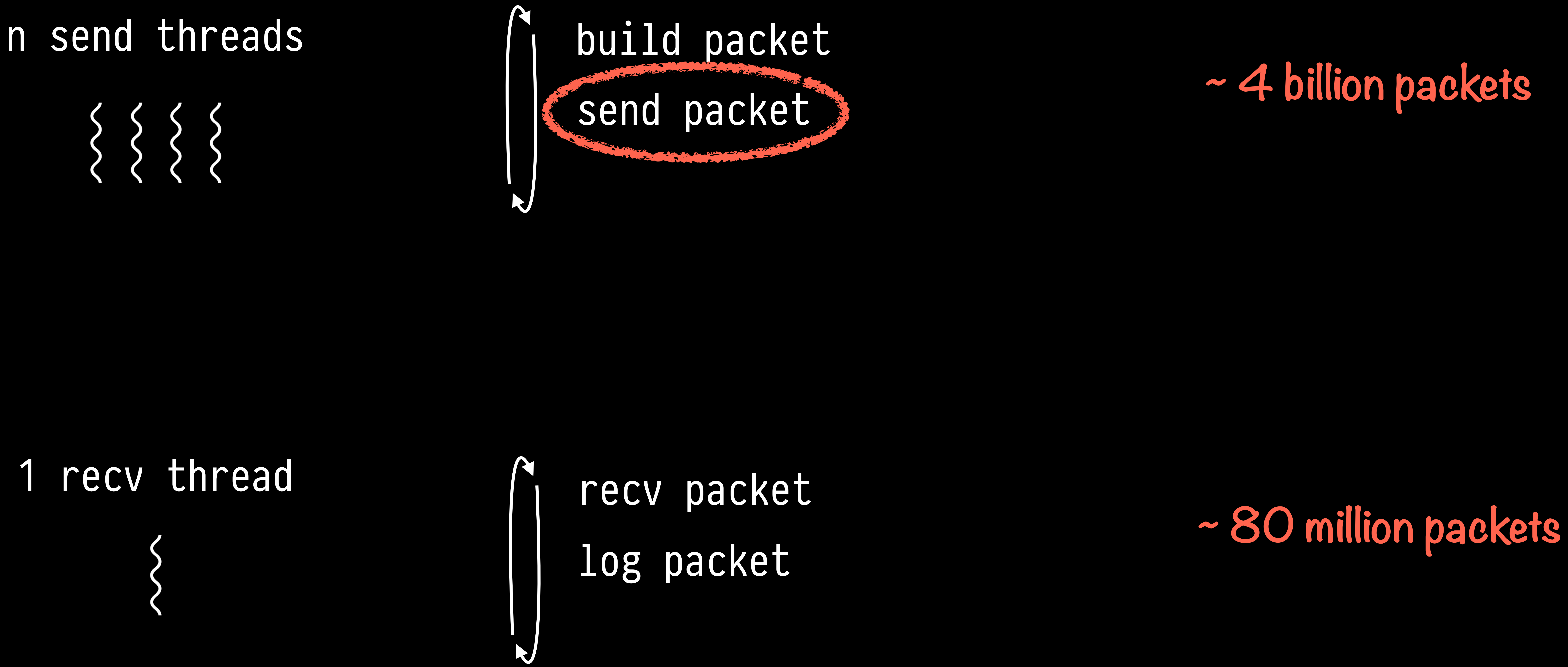
build packet

send packet

1 recv thread

recv packet

log packet



n send threads

build packet

send packet

~ 4 billion packets

1 recv thread

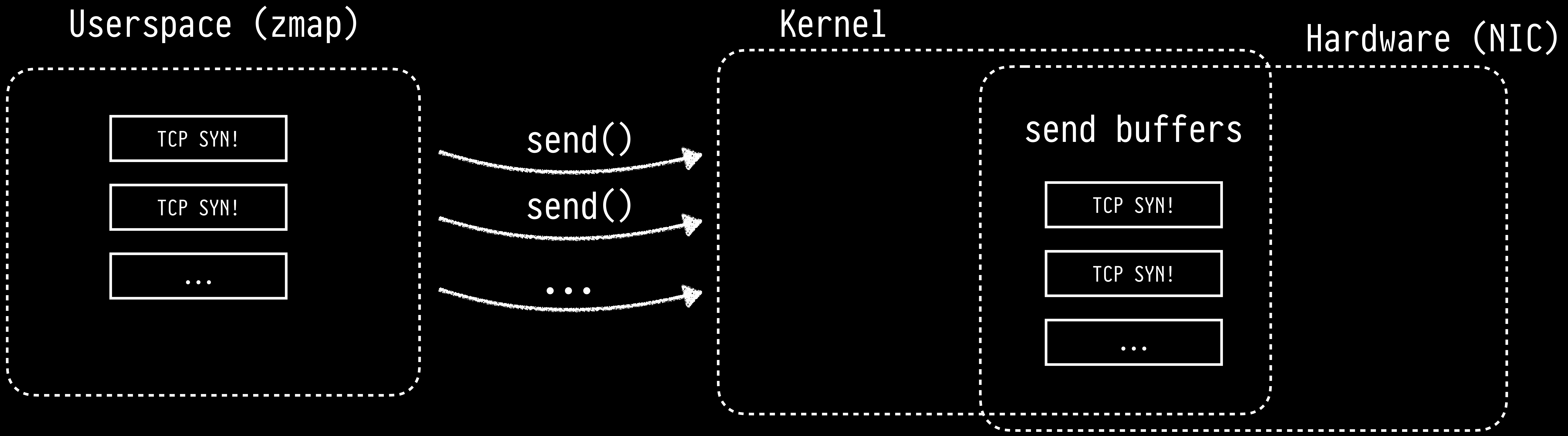
recv packet

log packet

~ 80 million packets

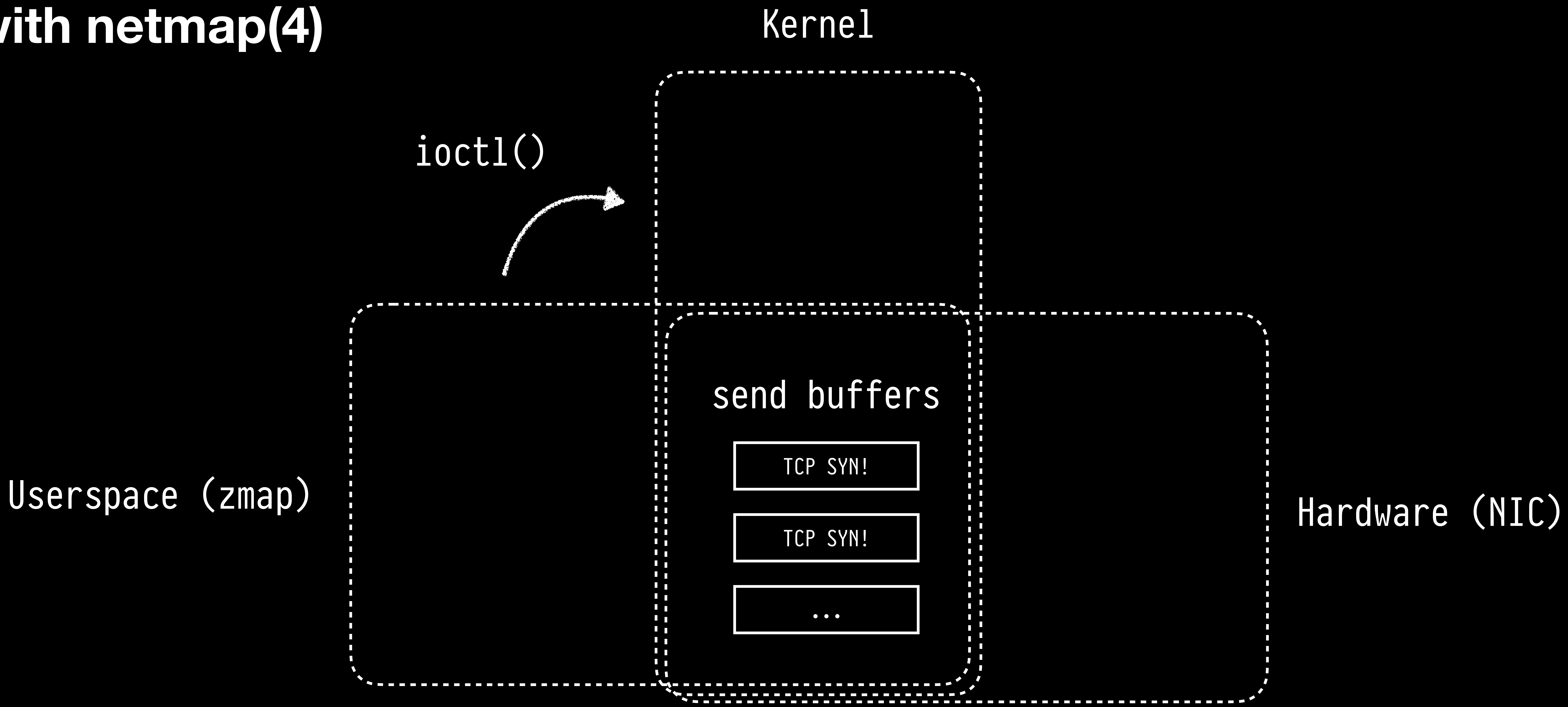
sending packets

syscall per packet



sending packets

with netmap(4)



10.28 Mp/s

69% of 14.88 Mp/s

how to optimize performance

- measure
- identify potential
- improve code
- measure again

how to optimize performance

- measure
- identify potential
- improve code
- measure again

```
Thread 0x18bc8:
10000  ??? (in libthr.so.3) (0x8268e6a75)
 10000  start_send + 70 (in zmap) (0x21ef16)
  3387  send_run + 1757 (in zmap) (0x21ad7d)
   3240  validate_gen + 68 (in zmap) (0x21cd34)
    113  rijndaelEncrypt + 163 (in zmap) (0x236993)
    112  rijndaelEncrypt + 523 (in zmap) (0x236afb)
    108  rijndaelEncrypt + 254 (in zmap) (0x2369ee)
    ...
   18  rijndaelEncrypt + 891 (in zmap) (0x236c6b)
   14  validate_gen + 25 (in zmap) (0x21cd09)
   ...
  2575  send_run + 2161 (in zmap) (0x21af11)
   1733  shard_get_next_target + 203 (in zmap) (0x21b8eb)
    1646  blocklist_lookup_index + 40 (in zmap) (0x232608)
     1553  constraint_lookup_index + 90 (in zmap) (0x233bca)
     10  constraint_lookup_index + 61 (in zmap) (0x233bad)
     ...
    24  blocklist_lookup_index + 55 (in zmap) (0x232617)
    8  blocklist_lookup_index + 51 (in zmap) (0x232613)
    ...
   219  shard_get_next_target + 32 (in zmap) (0x21b840)
   162  shard_get_next_target + 157 (in zmap) (0x21b8bd)
   ...
  2044  send_run + 2042 (in zmap) (0x21ae9a)
   1394  ioctl + 10 (in libc.so.7) (0x827266e0a)
   213  send_batch_internal + 231 (in zmap) (0x2312e7)
   49  memcpy + 137 (in libc.so.7) (0x8272888f9)
   40  memcpy + 54 (in libc.so.7) (0x8272888a6)
   36  send_batch_internal + 195 (in zmap) (0x2312c3)
   33  memcpy + 51 (in libc.so.7) (0x8272888a3)
   18  memcpy + 141 (in libc.so.7) (0x8272888fd)
   ...
  717  send_run + 1835 (in zmap) (0x21adcb)
   141  synscan_make_packet + 83 (in zmap) (0x228893)
   43  synscan_make_packet + 199 (in zmap) (0x228907)
   41  synscan_make_packet + 257 (in zmap) (0x228941)
   ...
  370  send_run + 1629 (in zmap) (0x21acfd)
  206  send_run + 1453 (in zmap) (0x21ac4d)
  53  send_run + 1639 (in zmap) (0x21ad07)
  ...
```

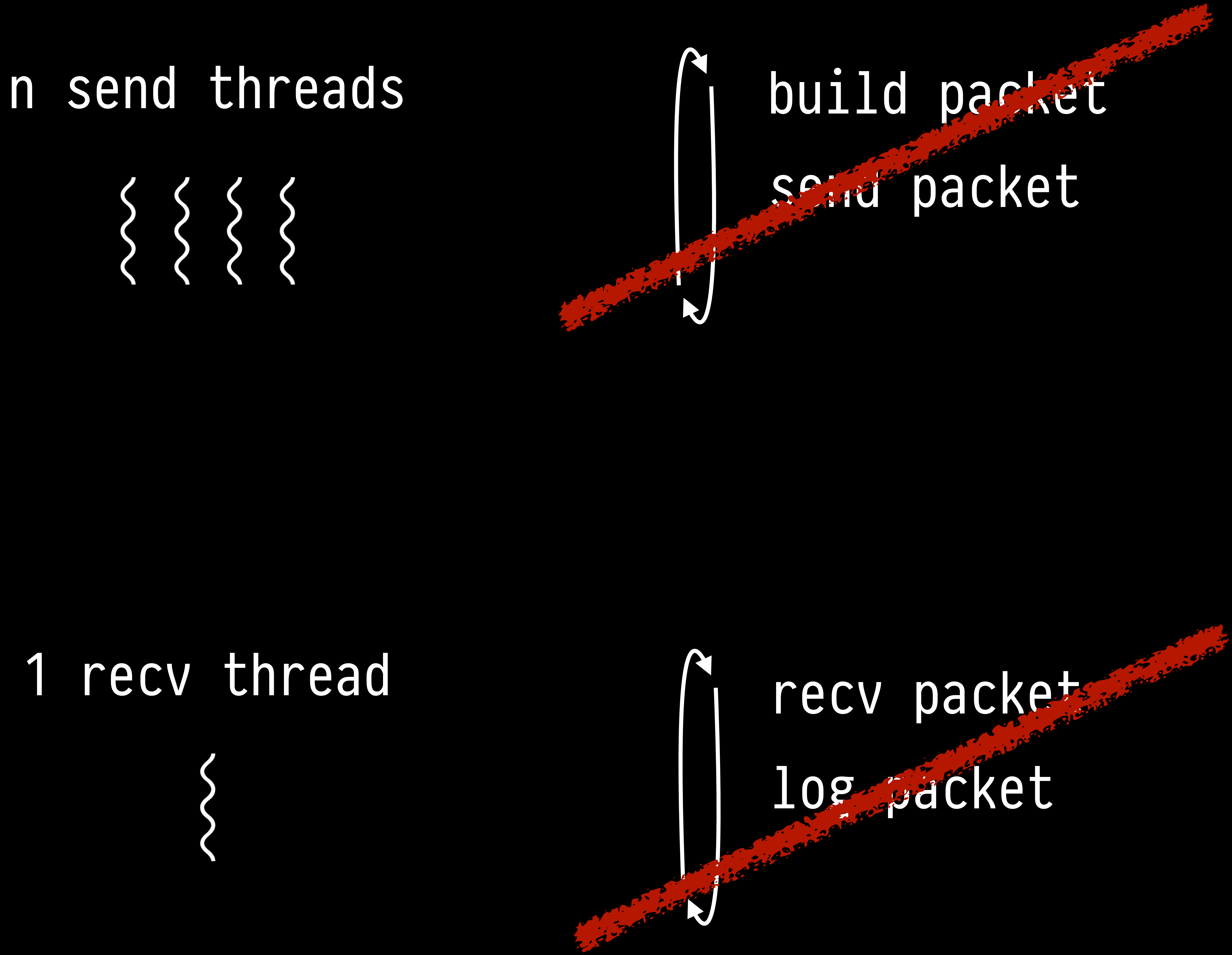
33.9 %	rijndael encrypt
25.8 %	target selection
13.9 %	sending
4.2 %	copying
7.2 %	packet building
15.0 %	other

where ~~how~~ to optimize performance

- measure
- identify potential
- improve code
- measure again

```
Thread 0x18bc8:
10000  ??? (in libthr.so.3) (0x8268e6a75)
  10000  start_send + 70 (in zmap) (0x21ef16)
    3387  send_run + 1757 (in zmap) (0x21ad7d)
      3240  validate_gen + 68 (in zmap) (0x21cd34)
        113  rijndaelEncrypt + 163 (in zmap) (0x236993)
        112  rijndaelEncrypt + 523 (in zmap) (0x236afb)
        108  rijndaelEncrypt + 254 (in zmap) (0x2369ee)
        ...
      18  rijndaelEncrypt + 891 (in zmap) (0x236c6b)
      14  validate_gen + 25 (in zmap) (0x21cd09)
      ...
    2575  send_run + 2161 (in zmap) (0x21af11)
      1733  shard_get_next_target + 203 (in zmap) (0x21b8eb)
        1646  blocklist_lookup_index + 40 (in zmap) (0x232608)
          1553  constraint_lookup_index + 90 (in zmap) (0x233bca)
          10  constraint_lookup_index + 61 (in zmap) (0x233bad)
          ...
        24  blocklist_lookup_index + 55 (in zmap) (0x232617)
        8  blocklist_lookup_index + 51 (in zmap) (0x232613)
        ...
      219  shard_get_next_target + 32 (in zmap) (0x21b840)
      162  shard_get_next_target + 157 (in zmap) (0x21b8bd)
      ...
    2044  send_run + 2042 (in zmap) (0x21ae9a)
      1394  ioctl + 10 (in libc.so.7) (0x827266e0a)
      213  send_batch_internal + 231 (in zmap) (0x2312e7)
      49  memcpy + 137 (in libc.so.7) (0x8272888f9)
      40  memcpy + 54 (in libc.so.7) (0x8272888a6)
      36  send_batch_internal + 195 (in zmap) (0x2312c3)
      33  memcpy + 51 (in libc.so.7) (0x8272888a3)
      18  memcpy + 141 (in libc.so.7) (0x8272888fd)
      ...
    717  send_run + 1835 (in zmap) (0x21adcb)
      141  synscan_make_packet + 83 (in zmap) (0x228893)
      43  synscan_make_packet + 199 (in zmap) (0x228907)
      41  synscan_make_packet + 257 (in zmap) (0x228941)
      ...
    370  send_run + 1629 (in zmap) (0x21acfd)
    206  send_run + 1453 (in zmap) (0x21ac4d)
    53  send_run + 1639 (in zmap) (0x21ad07)
    ...
```

33.9 %	rijndael encrypt
25.8 %	target selection
13.9 %	sending
4.2 %	copying
7.2 %	packet building
15.0 %	other



n send threads

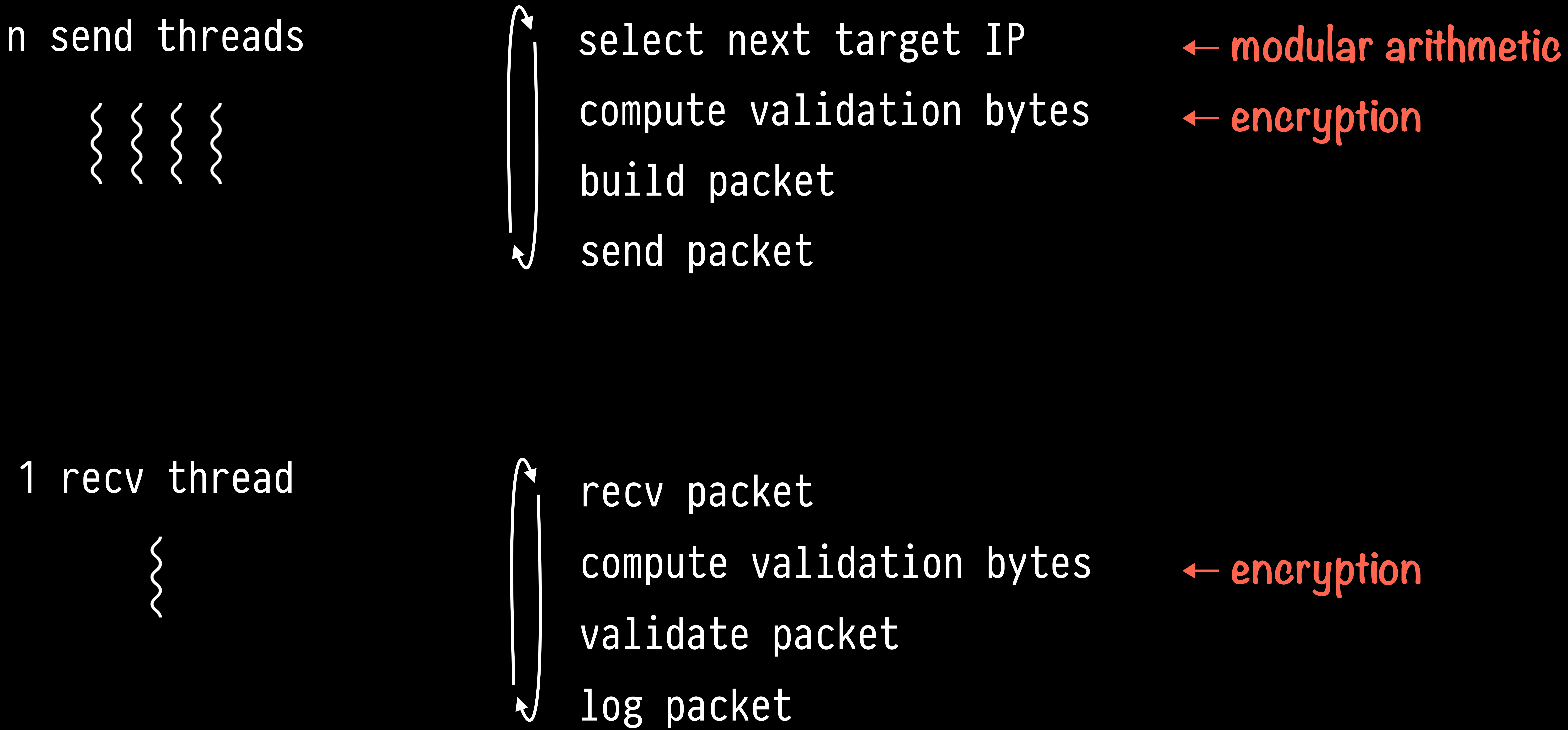
~~build packet~~

~~send packet~~

1 recv thread

~~recv packet~~

~~log packet~~



n send threads

select next target IP ← modular arithmetic

compute validation bytes ← encryption

build packet

send packet

1 recv thread

recv packet

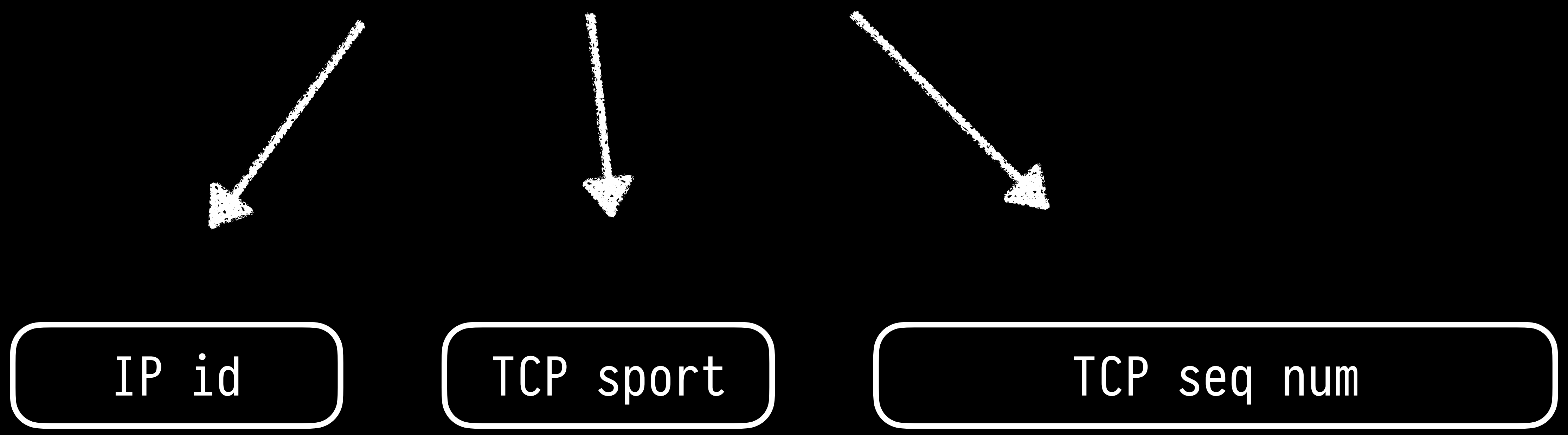
compute validation bytes ← encryption

validate packet

log packet

AES encryption

$$validation\ bytes = Enc_k(src\ ip||dst\ ip||dst\ port||0)$$

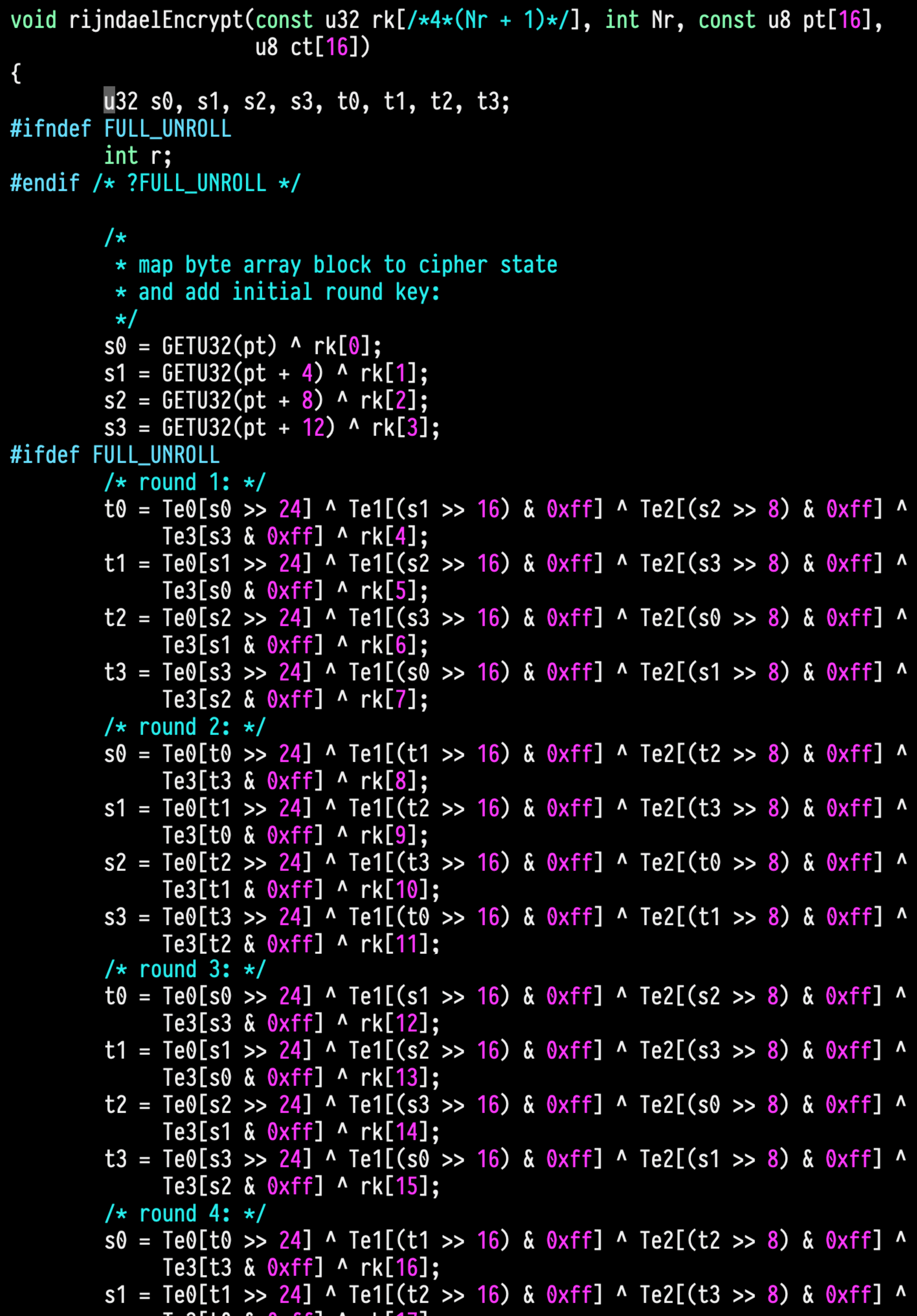


IP id

TCP sport

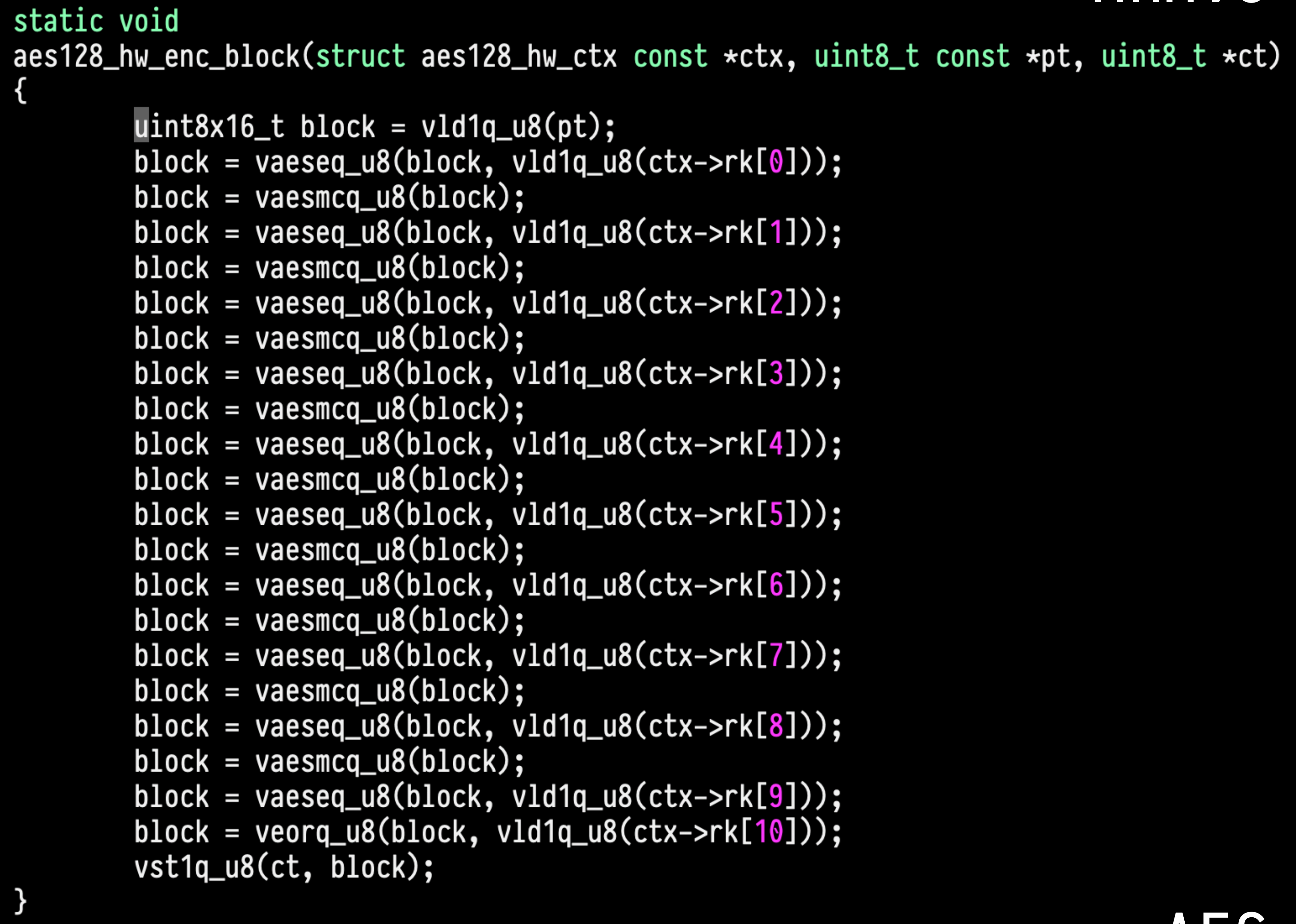
TCP seq num

Software AES



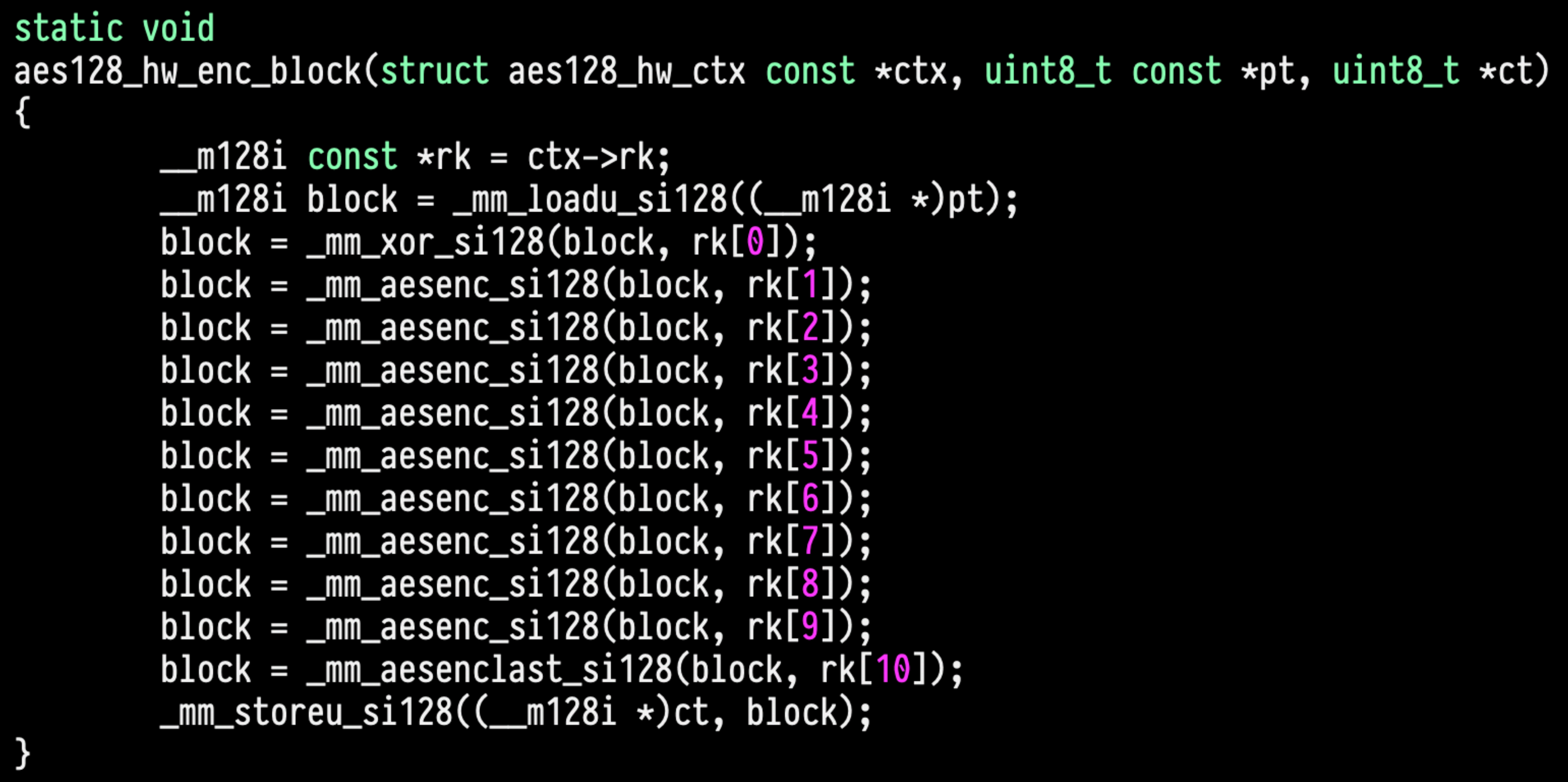
```
void rijndaelEncrypt(const u32 rk[/*4*(Nr + 1)*/], int Nr, const u8 pt[16],
                     u8 ct[16])
{
        u32 s0, s1, s2, s3, t0, t1, t2, t3;
#ifndef FULL_UNROLL
        int r;
#endif /* ?FULL_UNROLL */

        /*
         * map byte array block to cipher state
         * and add initial round key:
         */
        s0 = GETU32(pt) ^ rk[0];
        s1 = GETU32(pt + 4) ^ rk[1];
        s2 = GETU32(pt + 8) ^ rk[2];
        s3 = GETU32(pt + 12) ^ rk[3];
#ifdef FULL_UNROLL
        /* round 1: */
        t0 = Te0[s0 >> 24] ^ Te1[(s1 >> 16) & 0xff] ^ Te2[(s2 >> 8) & 0xff] ^
             Te3[s3 & 0xff] ^ rk[4];
        t1 = Te0[s1 >> 24] ^ Te1[(s2 >> 16) & 0xff] ^ Te2[(s3 >> 8) & 0xff] ^
             Te3[s0 & 0xff] ^ rk[5];
        t2 = Te0[s2 >> 24] ^ Te1[(s3 >> 16) & 0xff] ^ Te2[(s0 >> 8) & 0xff] ^
             Te3[s1 & 0xff] ^ rk[6];
        t3 = Te0[s3 >> 24] ^ Te1[(s0 >> 16) & 0xff] ^ Te2[(s1 >> 8) & 0xff] ^
             Te3[s2 & 0xff] ^ rk[7];
        /* round 2: */
        s0 = Te0[t0 >> 24] ^ Te1[(t1 >> 16) & 0xff] ^ Te2[(t2 >> 8) & 0xff] ^
             Te3[t3 & 0xff] ^ rk[8];
        s1 = Te0[t1 >> 24] ^ Te1[(t2 >> 16) & 0xff] ^ Te2[(t3 >> 8) & 0xff] ^
             Te3[t0 & 0xff] ^ rk[9];
        s2 = Te0[t2 >> 24] ^ Te1[(t3 >> 16) & 0xff] ^ Te2[(t0 >> 8) & 0xff] ^
             Te3[t1 & 0xff] ^ rk[10];
        s3 = Te0[t3 >> 24] ^ Te1[(t0 >> 16) & 0xff] ^ Te2[(t1 >> 8) & 0xff] ^
             Te3[t2 & 0xff] ^ rk[11];
        /* round 3: */
        t0 = Te0[s0 >> 24] ^ Te1[(s1 >> 16) & 0xff] ^ Te2[(s2 >> 8) & 0xff] ^
             Te3[s3 & 0xff] ^ rk[12];
        t1 = Te0[s1 >> 24] ^ Te1[(s2 >> 16) & 0xff] ^ Te2[(s3 >> 8) & 0xff] ^
             Te3[s0 & 0xff] ^ rk[13];
        t2 = Te0[s2 >> 24] ^ Te1[(s3 >> 16) & 0xff] ^ Te2[(s0 >> 8) & 0xff] ^
             Te3[s1 & 0xff] ^ rk[14];
        t3 = Te0[s3 >> 24] ^ Te1[(s0 >> 16) & 0xff] ^ Te2[(s1 >> 8) & 0xff] ^
             Te3[s2 & 0xff] ^ rk[15];
        /* round 4: */
        s0 = Te0[t0 >> 24] ^ Te1[(t1 >> 16) & 0xff] ^ Te2[(t2 >> 8) & 0xff] ^
             Te3[t3 & 0xff] ^ rk[16];
        s1 = Te0[t1 >> 24] ^ Te1[(t2 >> 16) & 0xff] ^ Te2[(t3 >> 8) & 0xff] ^
```

ARMv8 CE



```
static void
aes128_hw_enc_block(struct aes128_hw_ctx const *ctx, uint8_t const *pt, uint8_t *ct)
{
        uint8x16_t block = vld1q_u8(pt);
        block = vaeseq_u8(block, vld1q_u8(ctx->rk[0]));
        block = vaesmcq_u8(block);
        block = vaeseq_u8(block, vld1q_u8(ctx->rk[1]));
        block = vaesmcq_u8(block);
        block = vaeseq_u8(block, vld1q_u8(ctx->rk[2]));
        block = vaesmcq_u8(block);
        block = vaeseq_u8(block, vld1q_u8(ctx->rk[3]));
        block = vaesmcq_u8(block);
        block = vaeseq_u8(block, vld1q_u8(ctx->rk[4]));
        block = vaesmcq_u8(block);
        block = vaeseq_u8(block, vld1q_u8(ctx->rk[5]));
        block = vaesmcq_u8(block);
        block = vaeseq_u8(block, vld1q_u8(ctx->rk[6]));
        block = vaesmcq_u8(block);
        block = vaeseq_u8(block, vld1q_u8(ctx->rk[7]));
        block = vaesmcq_u8(block);
        block = vaeseq_u8(block, vld1q_u8(ctx->rk[8]));
        block = vaesmcq_u8(block);
        block = vaeseq_u8(block, vld1q_u8(ctx->rk[9]));
        block = veorq_u8(block, vld1q_u8(ctx->rk[10]));
        vst1q_u8(ct, block);
}
```

AES-NI



```
static void
aes128_hw_enc_block(struct aes128_hw_ctx const *ctx, uint8_t const *pt, uint8_t *ct)
{
        __m128i const *rk = ctx->rk;
        __m128i block = _mm_loadu_si128((__m128i *)pt);
        block = _mm_xor_si128(block, rk[0]);
        block = _mm_aesenc_si128(block, rk[1]);
        block = _mm_aesenc_si128(block, rk[2]);
        block = _mm_aesenc_si128(block, rk[3]);
        block = _mm_aesenc_si128(block, rk[4]);
        block = _mm_aesenc_si128(block, rk[5]);
        block = _mm_aesenc_si128(block, rk[6]);
        block = _mm_aesenc_si128(block, rk[7]);
        block = _mm_aesenc_si128(block, rk[8]);
        block = _mm_aesenc_si128(block, rk[9]);
        block = _mm_aesenclast_si128(block, rk[10]);
        _mm_storeu_si128((__m128i *)ct, block);
}
```

12.13 Mp/s

82% of 14.88 Mp/s

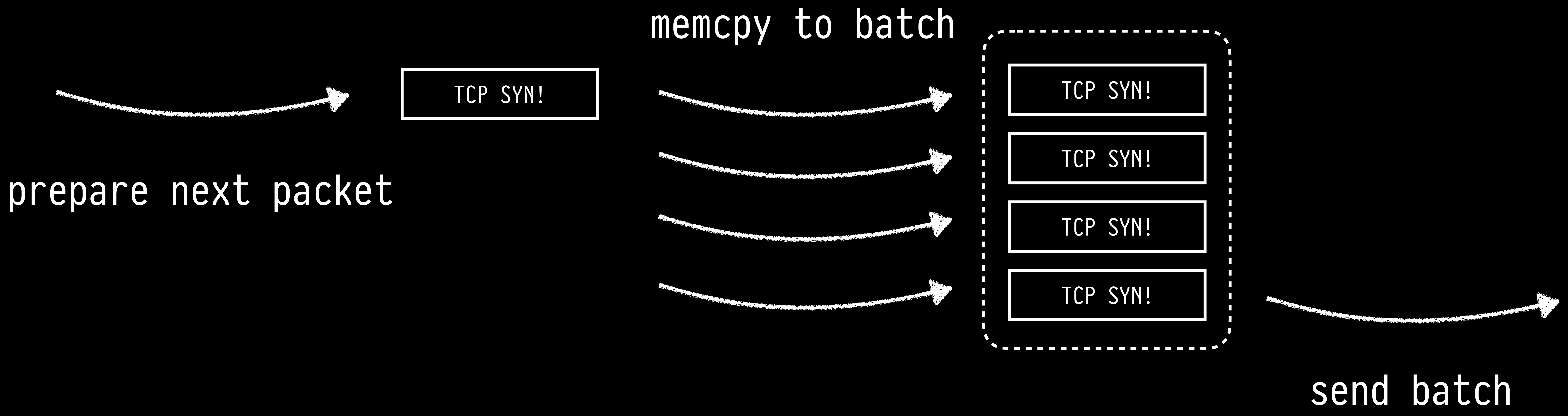
~~how~~ to optimize performance
where

- measure
- identify potential
- improve code
- measure again

```
Thread 0x18bc8:
10000  ??? (in libthr.so.3) (0x8268e6a75)
  10000  start_send + 70 (in zmap) (0x21ef16)
    3387  send_run + 1757 (in zmap) (0x21ad7d)
      3240  validate_gen + 68 (in zmap) (0x21cd34)
        113  rijndaelEncrypt + 163 (in zmap) (0x236993)
        112  rijndaelEncrypt + 523 (in zmap) (0x236afb)
        108  rijndaelEncrypt + 254 (in zmap) (0x2369ee)
        ...
      18  rijndaelEncrypt + 891 (in zmap) (0x236c6b)
      14  validate_gen + 25 (in zmap) (0x21cd09)
      ...
    2575  send_run + 2161 (in zmap) (0x21af11)
      1733  shard_get_next_target + 203 (in zmap) (0x21b8eb)
        1646  blocklist_lookup_index + 40 (in zmap) (0x232608)
          1553  constraint_lookup_index + 90 (in zmap) (0x233bca)
          10  constraint_lookup_index + 61 (in zmap) (0x233bad)
          ...
        24  blocklist_lookup_index + 55 (in zmap) (0x232617)
        8  blocklist_lookup_index + 51 (in zmap) (0x232613)
        ...
      219  shard_get_next_target + 32 (in zmap) (0x21b840)
      162  shard_get_next_target + 157 (in zmap) (0x21b8bd)
      ...
    2044  send_run + 2042 (in zmap) (0x21ae9a)
      1394  ioctl + 10 (in libc.so.7) (0x827266e0a)
      213  send_batch_internal + 231 (in zmap) (0x2312e7)
      49  memcpy + 137 (in libc.so.7) (0x8272888f9)
      40  memcpy + 54 (in libc.so.7) (0x8272888a6)
      36  send_batch_internal + 195 (in zmap) (0x2312c3)
      33  memcpy + 51 (in libc.so.7) (0x8272888a3)
      18  memcpy + 141 (in libc.so.7) (0x8272888fd)
      ...
    717  send_run + 1835 (in zmap) (0x21adcb)
      141  synscan_make_packet + 83 (in zmap) (0x228893)
      43  synscan_make_packet + 199 (in zmap) (0x228907)
      41  synscan_make_packet + 257 (in zmap) (0x228941)
      ...
    370  send_run + 1629 (in zmap) (0x21acfd)
    206  send_run + 1453 (in zmap) (0x21ac4d)
    53  send_run + 1639 (in zmap) (0x21ad07)
    ...
```

✅	33.9 %	rijndael encrypt
	25.8 %	target selection
	13.9 %	sending
	4.2 %	copying
	7.2 %	packet building
	15.0 %	other

Less copying

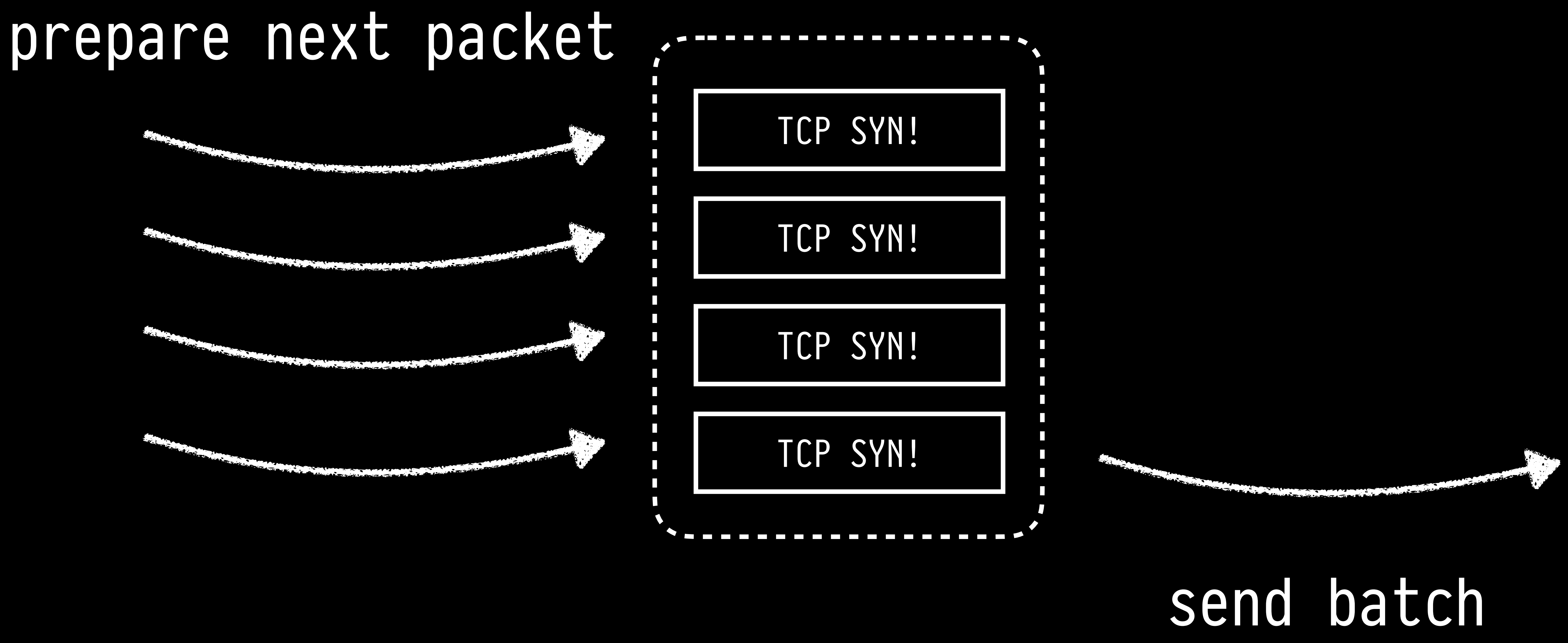


memcpy to batch

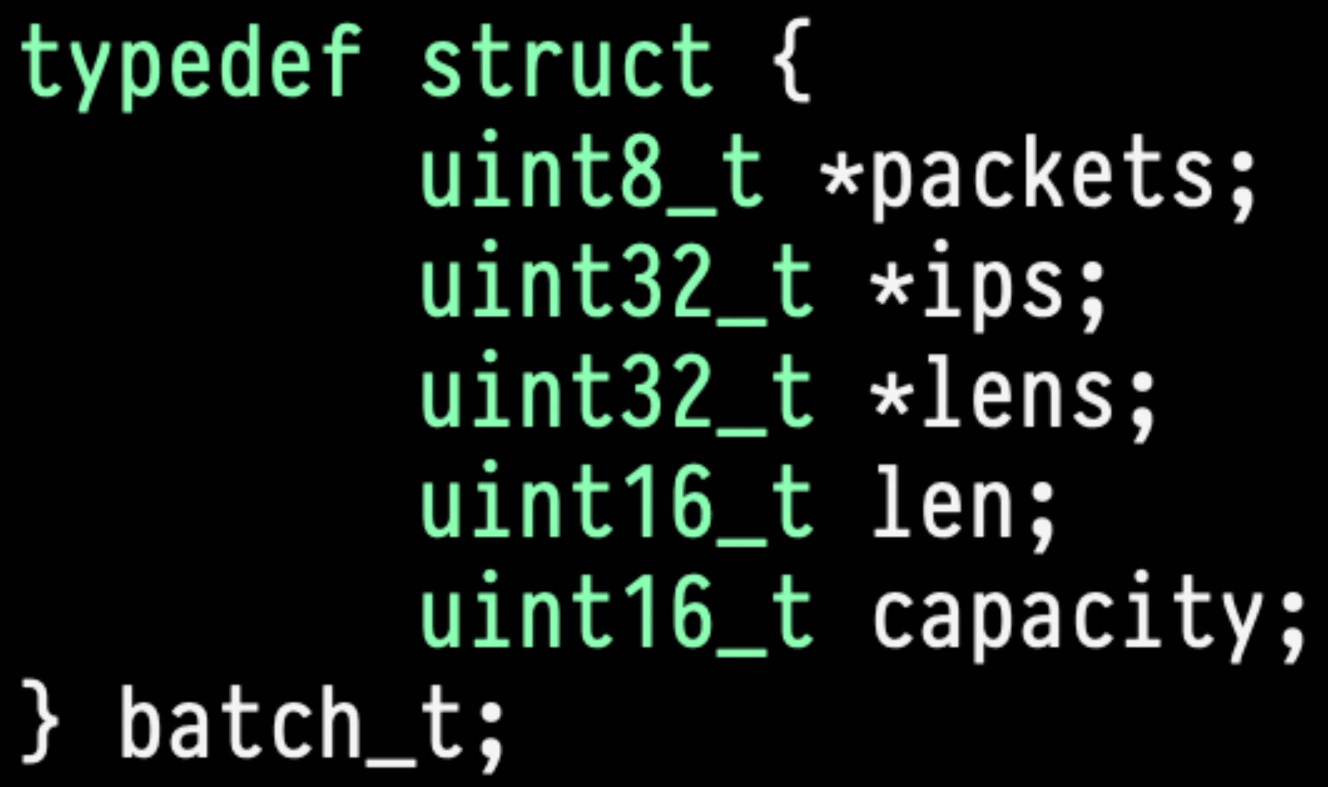
prepare next packet

send batch

Less copying

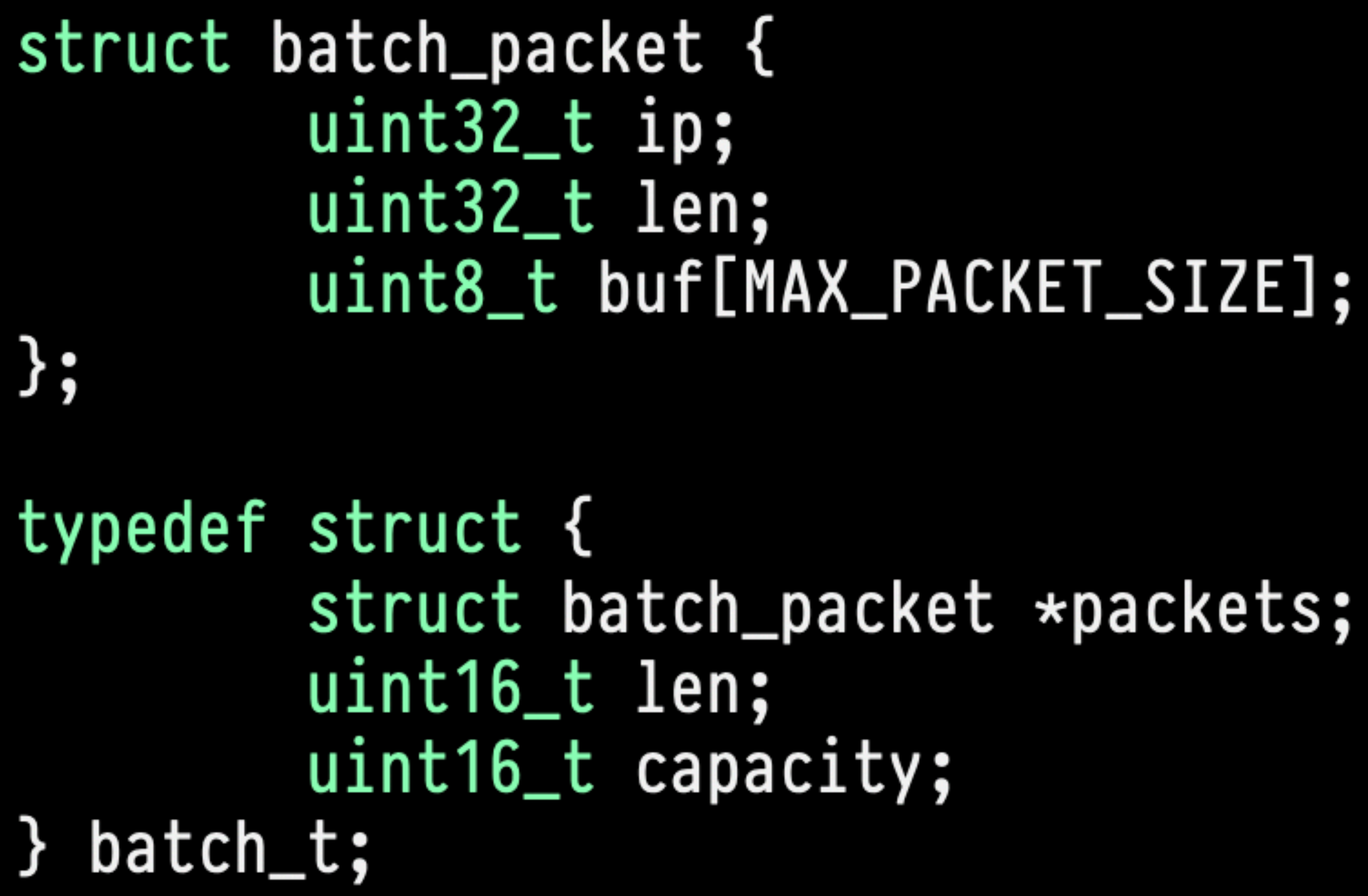


Cache locality



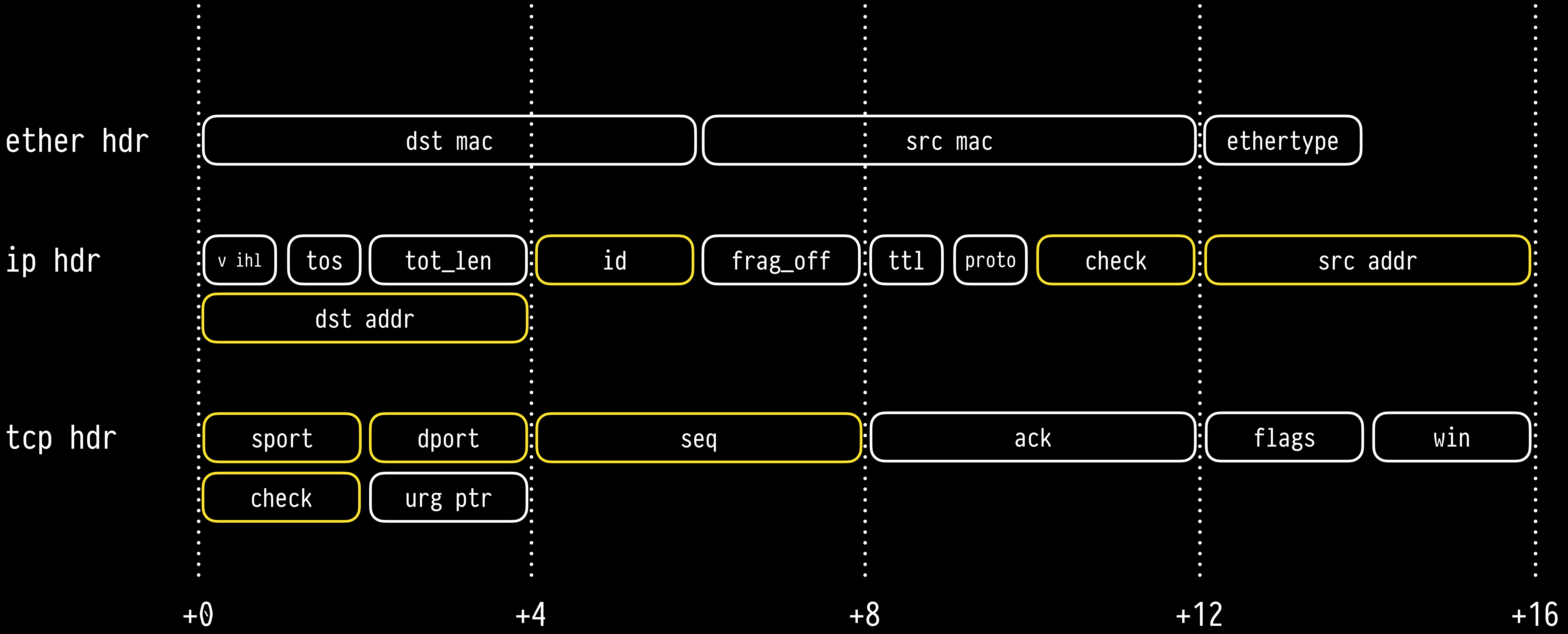
```
typedef struct {
        uint8_t *packets;
        uint32_t *ips;
        uint32_t *lens;
        uint16_t len;
        uint16_t capacity;
} batch_t;
```

→

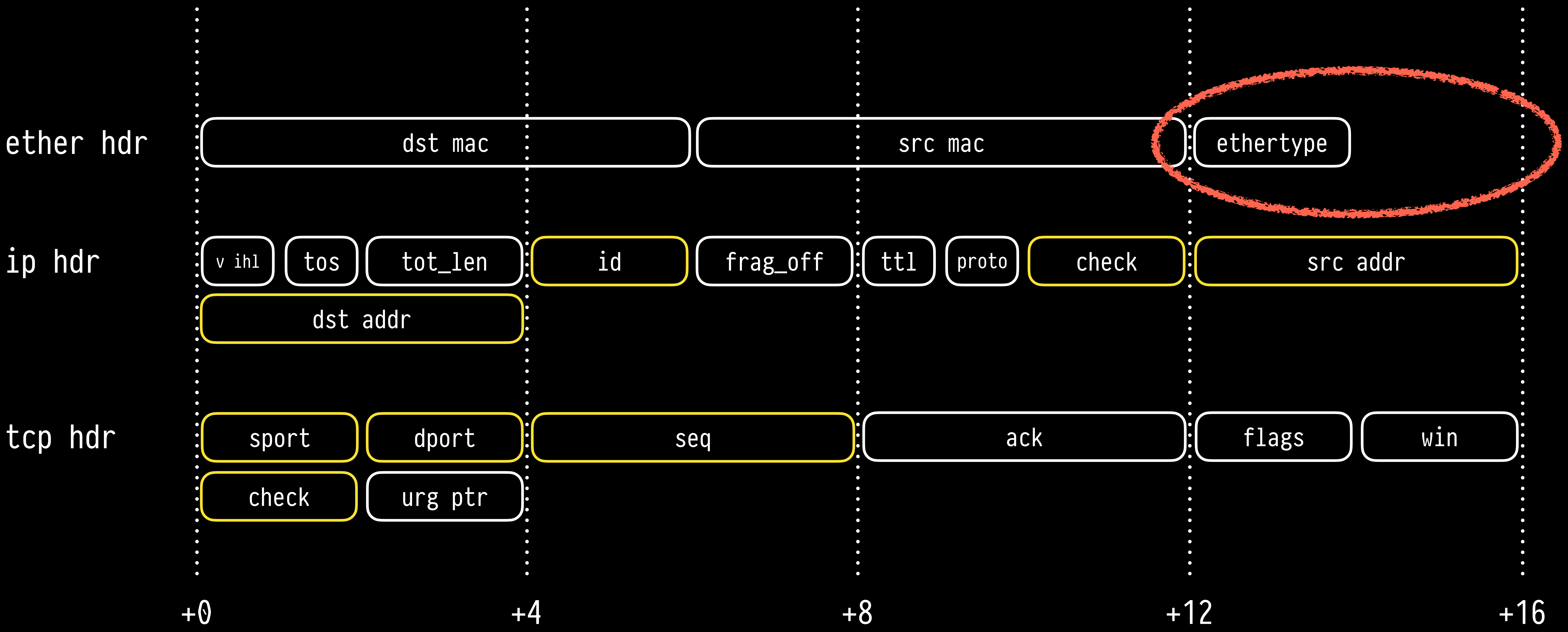


```
struct batch_packet {
        uint32_t ip;
        uint32_t len;
        uint8_t buf[MAX_PACKET_SIZE];
};

typedef struct {
        struct batch_packet *packets;
        uint16_t len;
        uint16_t capacity;
} batch_t;
```

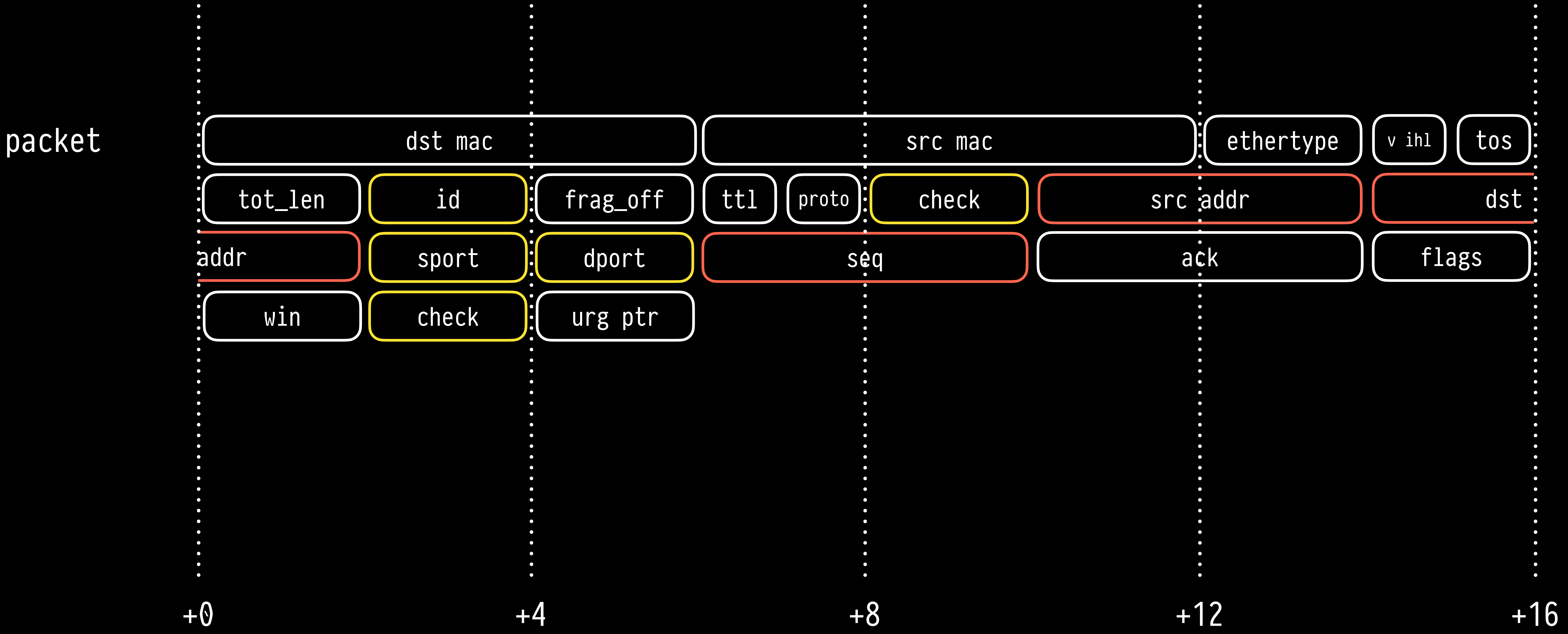
Buffer alignment



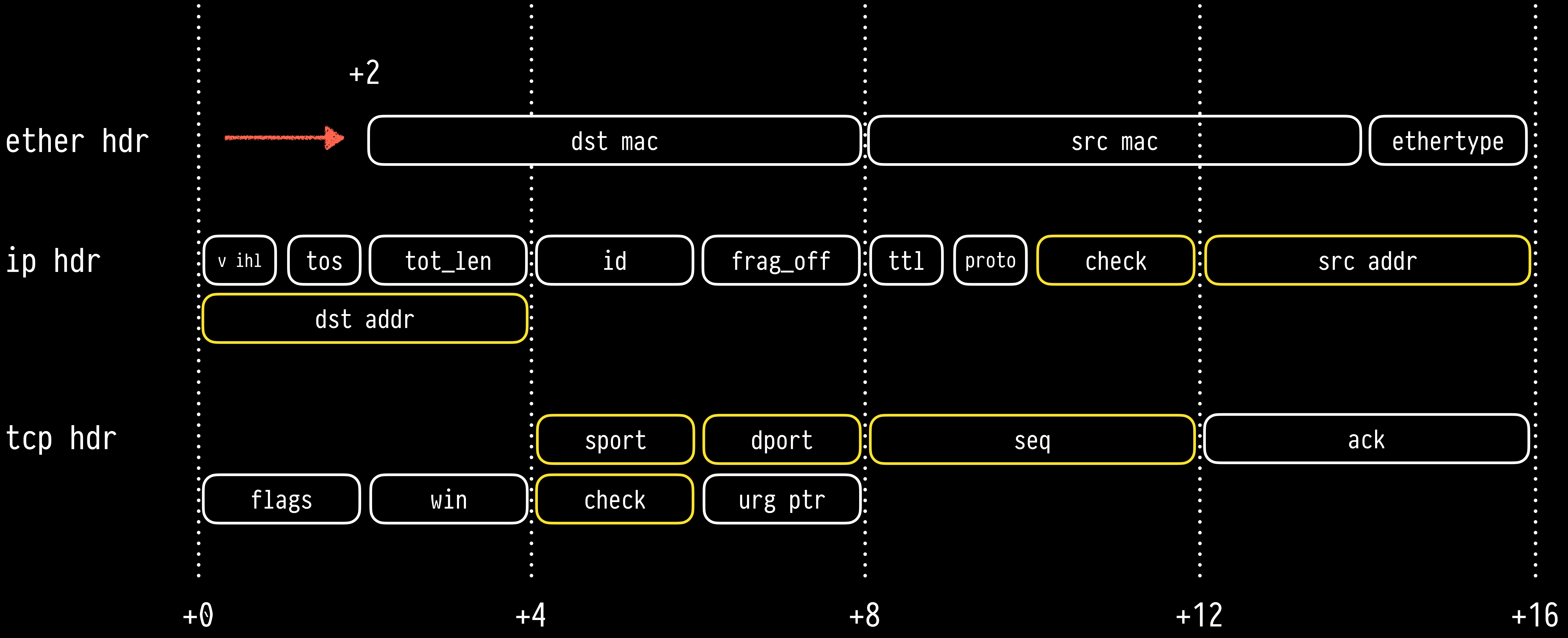
Buffer alignment



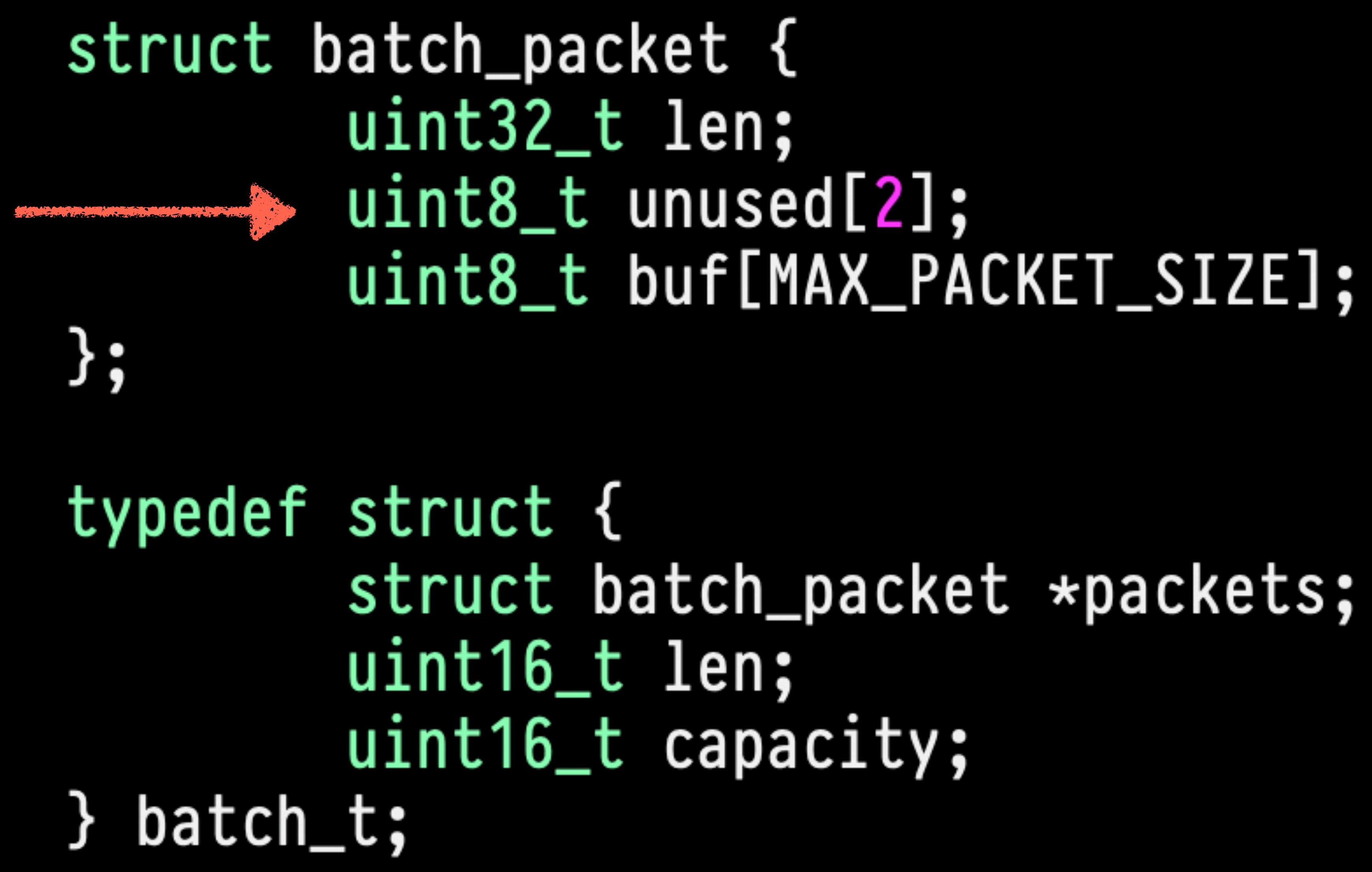
Buffer alignment



Buffer alignment



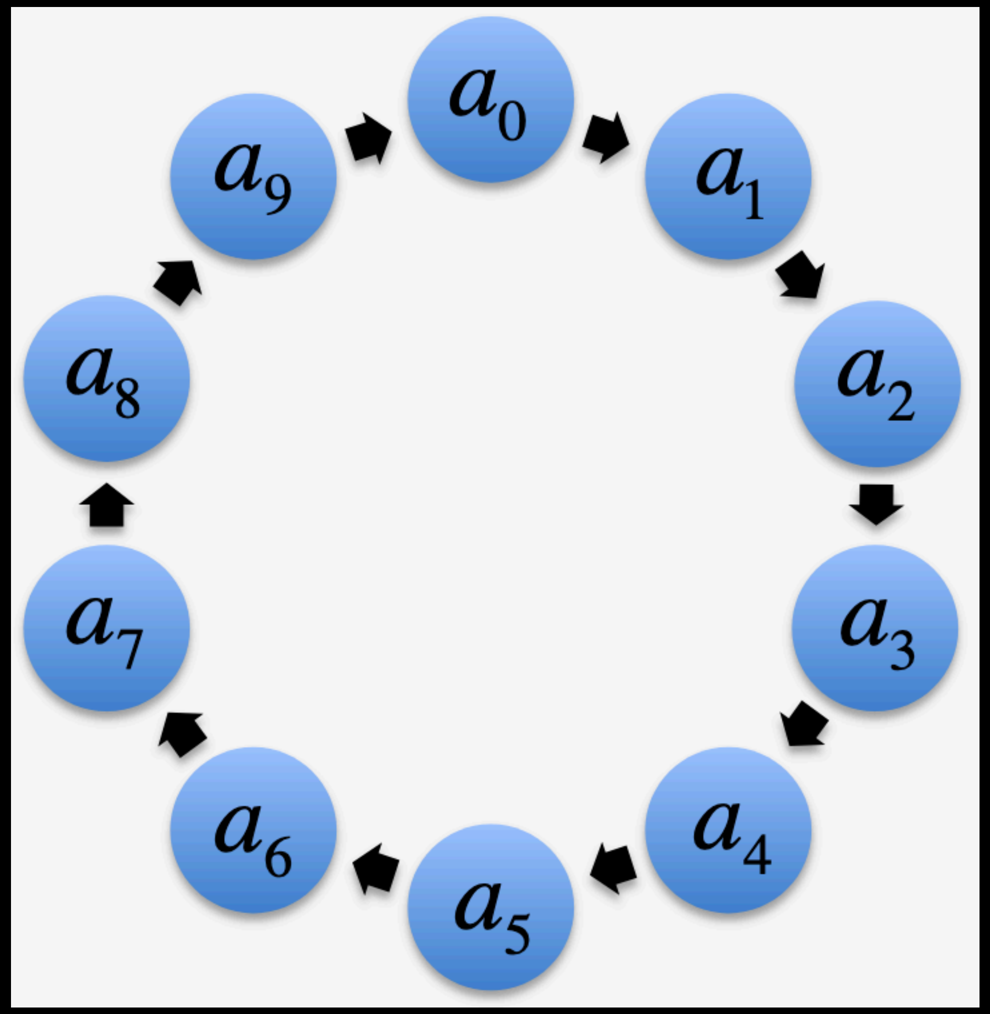
Buffer alignment



```
struct batch_packet {
        uint32_t len;
        uint8_t unused[2];
        uint8_t buf[MAX_PACKET_SIZE];
};

typedef struct {
        struct batch_packet *packets;
        uint16_t len;
        uint16_t capacity;
} batch_t;
```

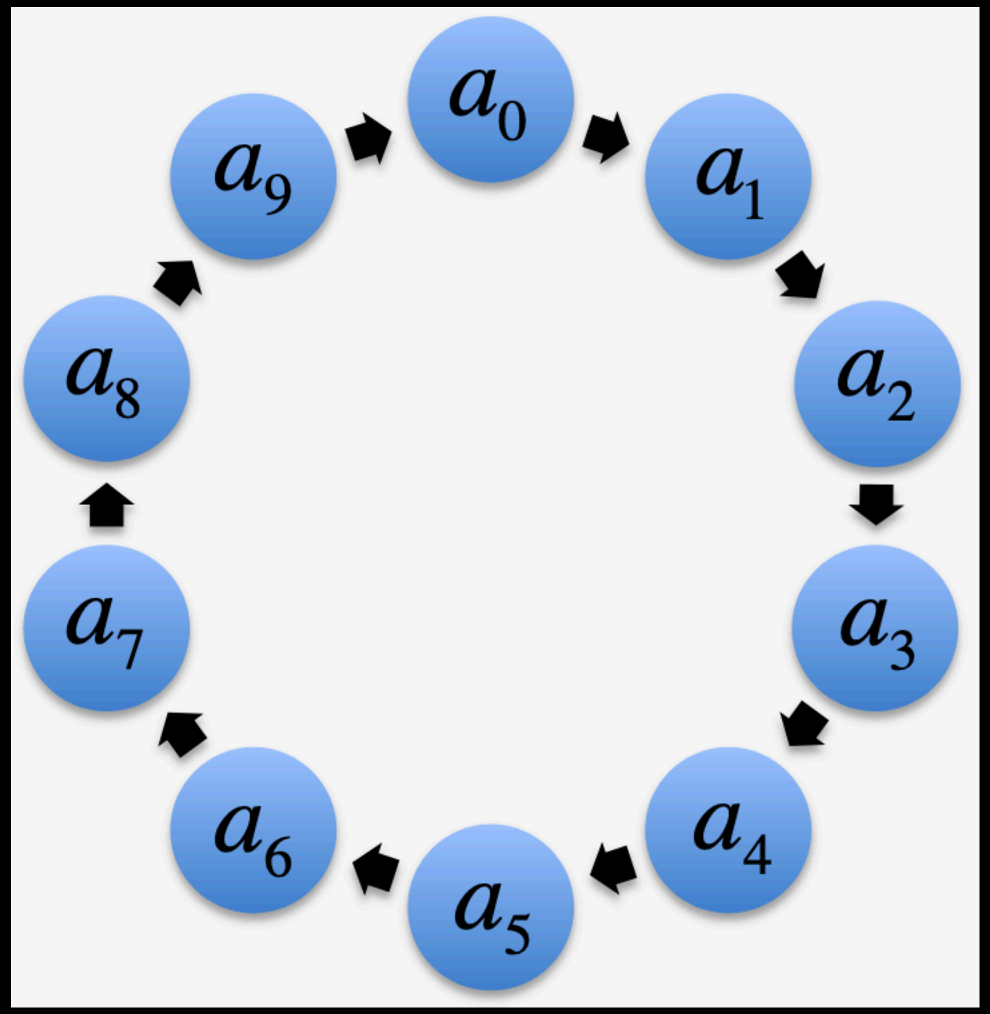
12.57 Mp/s

84% of 14.88 Mp/s

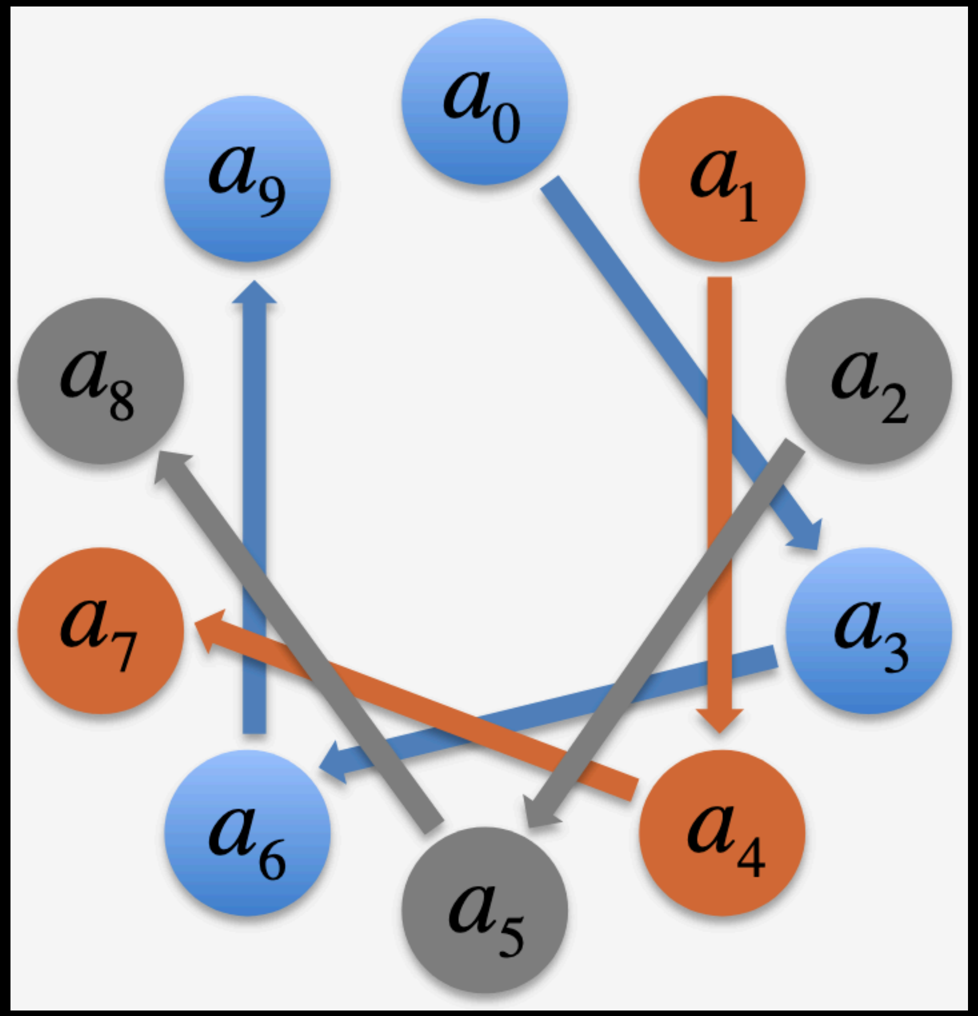


$$addr_{i+1} = g \cdot addr_i \mod p$$

Image source: Zippier ZMap; David Adrian, Zakir Durumeric, Guishan Singh and J Alex Halderman, University of Michigan



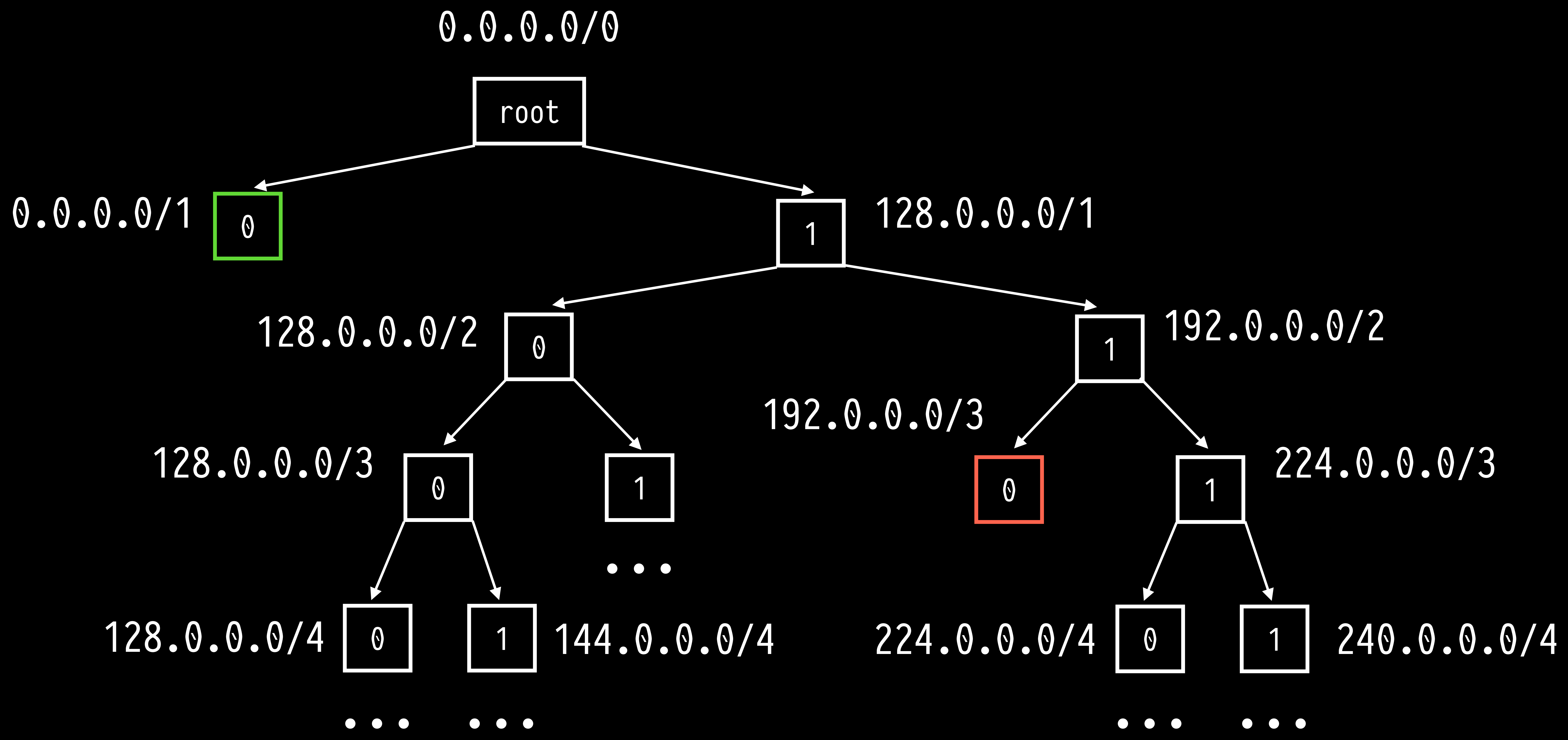
$$addr_{i+1} = g \cdot addr_i \mod p$$



$$addr_{i+n} = g^n \cdot addr_i \mod p$$

Image source: Zippier ZMap; David Adrian, Zakir Durumeric, Guishan Singh and J Alex Halderman, University of Michigan

```
0.0.0.0/8           # RFC1122: "This host on this network"
10.0.0.0/8          # RFC1918: Private-Use
100.64.0.0/10       # RFC6598: Shared Address Space
127.0.0.0/8         # RFC1122: Loopback
169.254.0.0/16      # RFC3927: Link Local
172.16.0.0/12       # RFC1918: Private-Use
192.0.0.0/24        # RFC6890: IETF Protocol Assignments
192.0.2.0/24        # RFC5737: Documentation (TEST-NET-1)
192.88.99.0/24      # RFC3068: 6to4 Relay Anycast
192.168.0.0/16      # RFC1918: Private-Use
198.18.0.0/15       # RFC2544: Benchmarking
198.51.100.0/24     # RFC5737: Documentation (TEST-NET-2)
203.0.113.0/24      # RFC5737: Documentation (TEST-NET-3)
224.0.0.0/4         # RFC5771: Multicast/Reserved
240.0.0.0/4         # RFC1112: Reserved
255.255.255.255/32  # RFC0919: Limited Broadcast
```


0.0.0.0/0
root
0.0.0.0/1
0
1
128.0.0.0/1
128.0.0.0/2
0
1
192.0.0.0/2
192.0.0.0/3
128.0.0.0/3
0
1
...
0
1
224.0.0.0/3
128.0.0.0/4
0
1
144.0.0.0/4
224.0.0.0/4
0
1
240.0.0.0/4
...
...
...
...

0.0.0.0/20

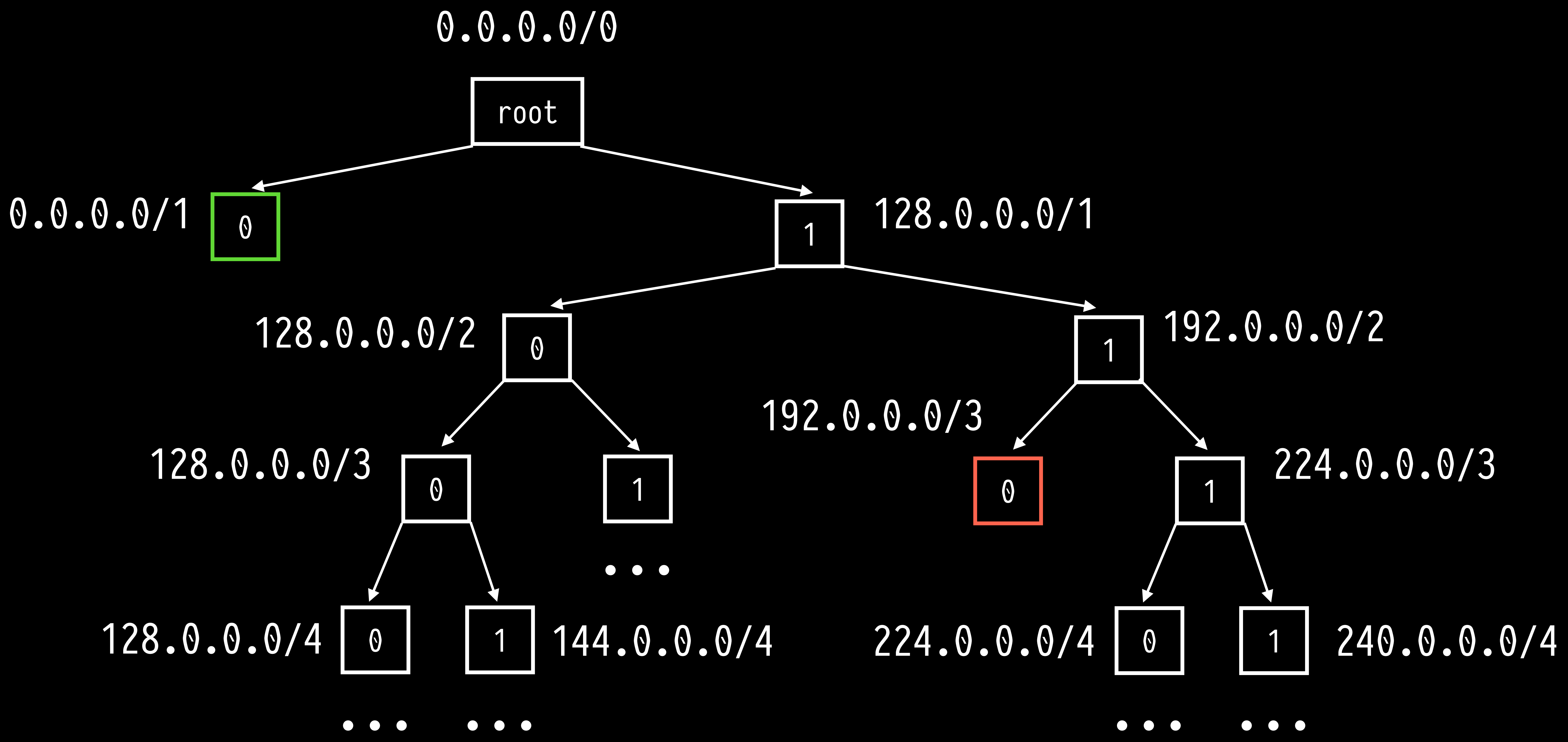
0.0.16.0/20

0.0.32.0/20

0.0.48.0/20

0.0.64.0/20

...



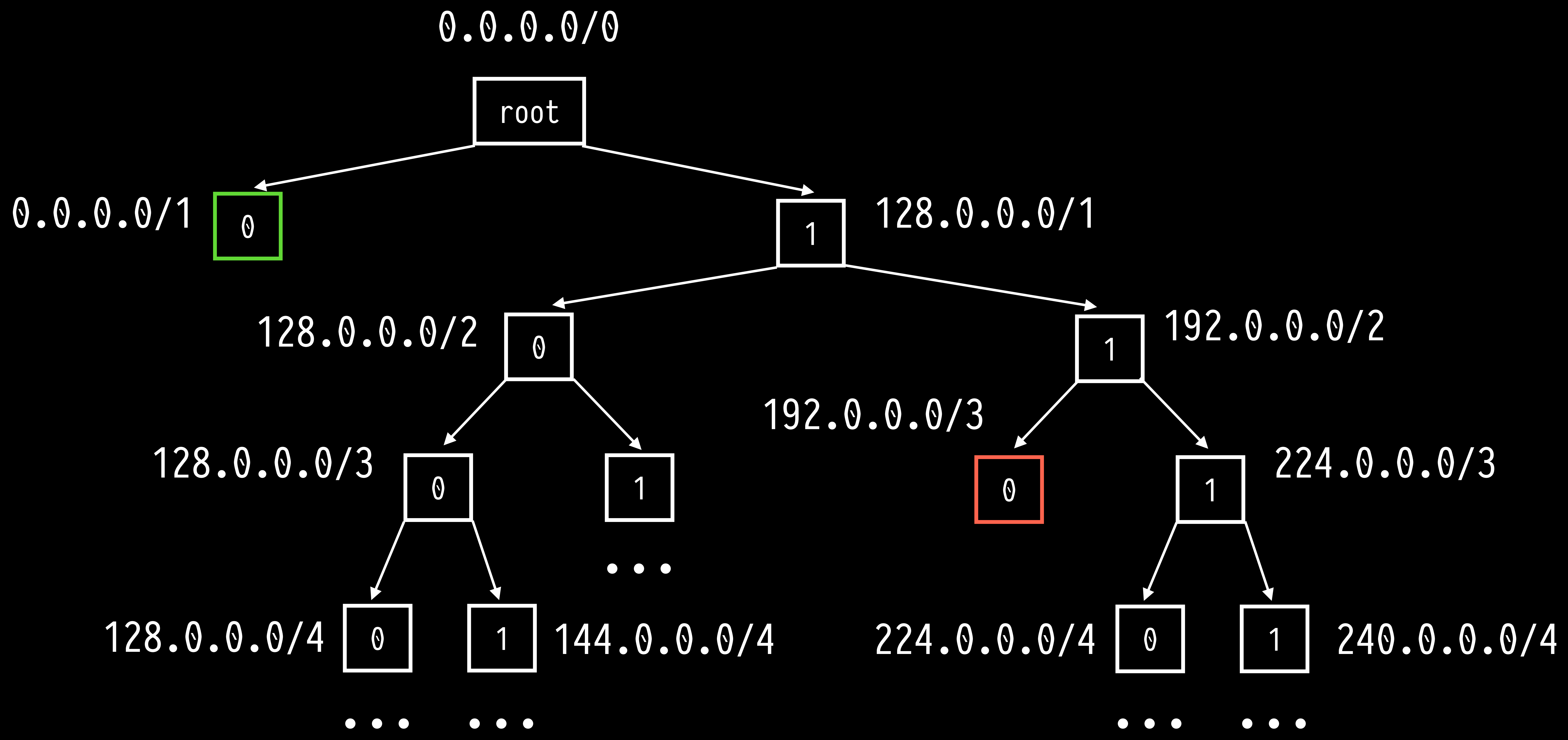
3'702'243'328 addrs

15'104 addrs

0.0.0.0/20
0.0.16.0/20
0.0.32.0/20
0.0.48.0/20
0.0.64.0/20
...

4 MB

3'702'243'328 addrs

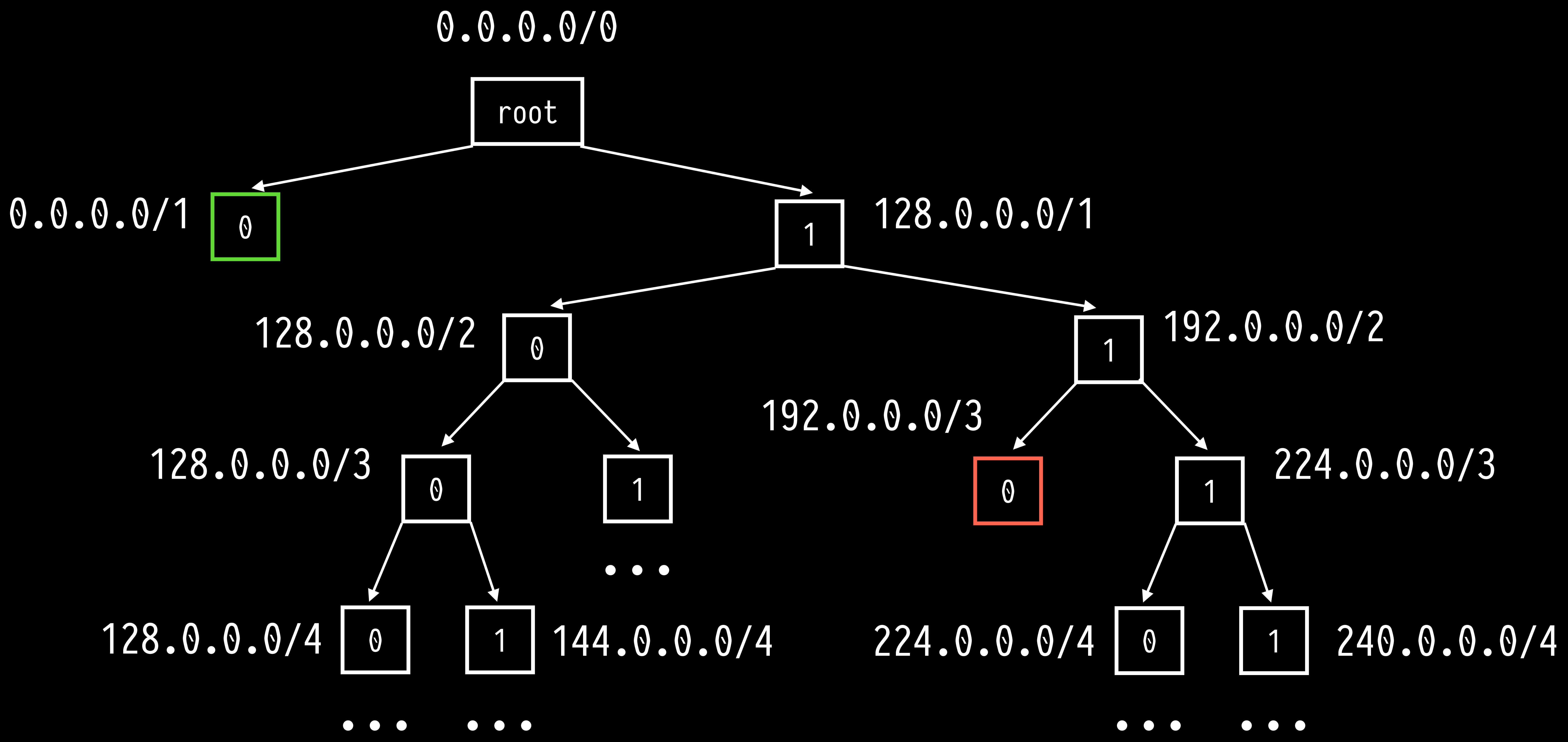


15'104 addrs

Optimize the optimization

0.0.0.0/18
0.0.64.0/18
0.0.128.0/18
0.0.196.0/18
0.0.240.0/18
• • •

1 MB



3'702'194'176 addrs

64'256 addrs

13.81 Mp/s

93% of 14.88 Mp/s

(╯°□°)╯︵ ┻━┻

13.81 Mp/s

4 minutes 28 seconds

┬─┬ノ(゜-゜ノ)

swisscom