Early Experiences Proving the Correctness of a Network Stack Implementation

Alain Kägi,[*] Linus Brogan,[*] Aubrey Birdwell,[+] Caitlyn Wilde,[*] Michael Harper,[*] Levi Overcast,[+] Richard Weiss,[+] Jens Mache[*]

[*]Lewis & Clark [+]Evergreen College

[These * Are * My * Photons, CC BY-NC 2.0]

Everything in its right place

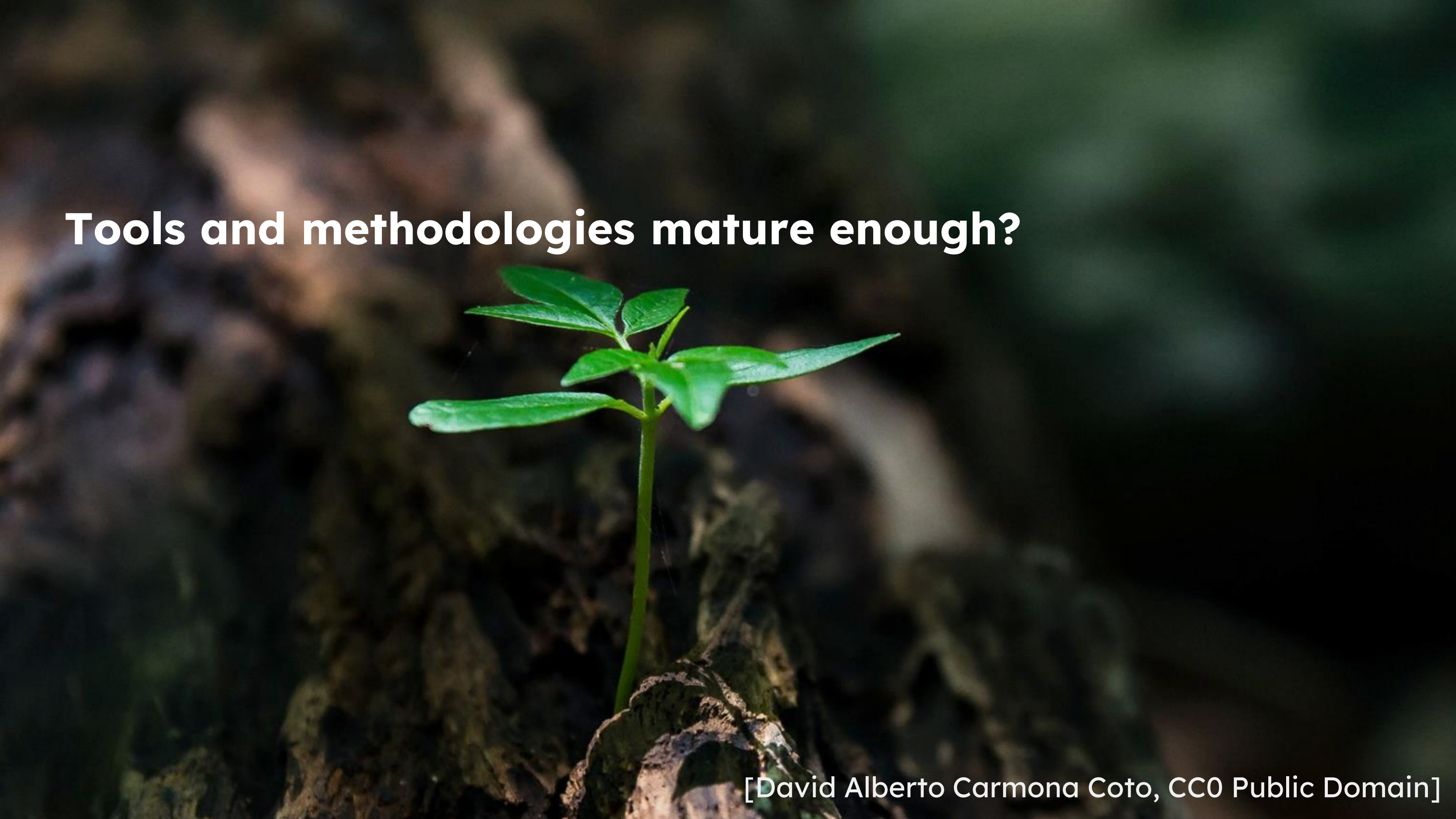
[These * Are * My * Photons, CC BY-NC 2.0]

Everything in its right place

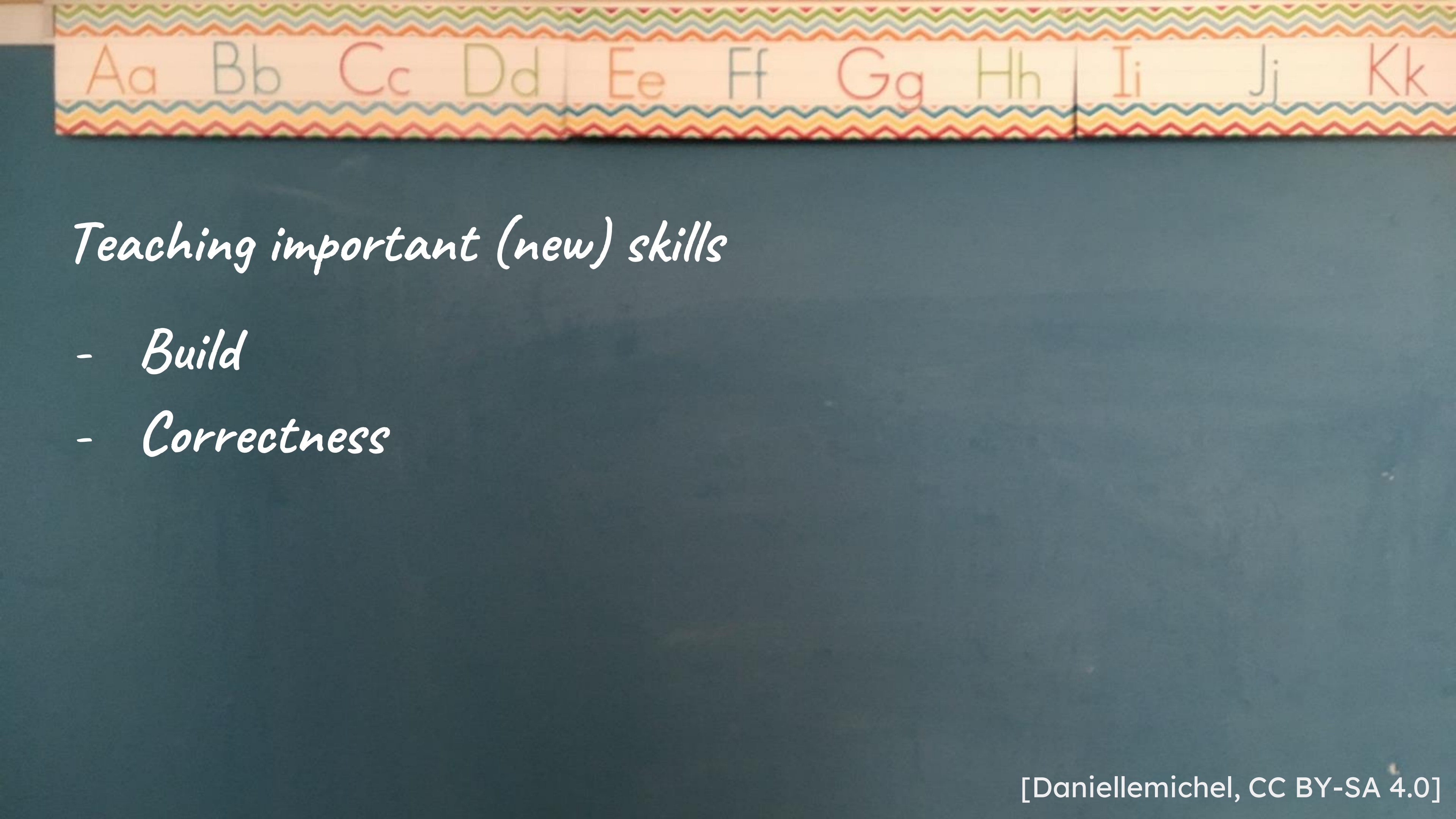
- Correct
- Reliable
- Secure
- Confidential
- Efficient

[These * Are * My * Photons, CC BY-NC 2.0]

Tools and methodologies mature enough?



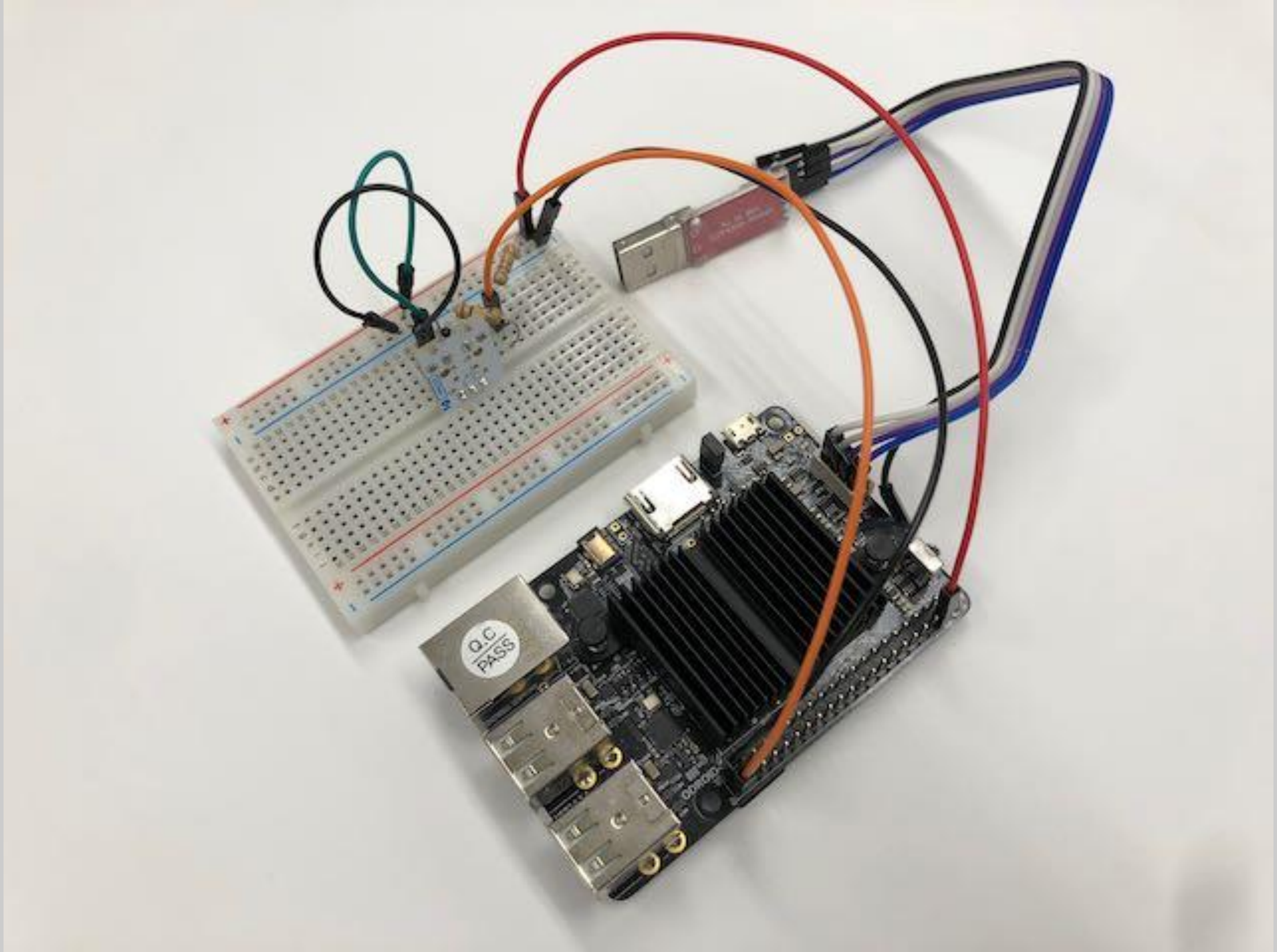
[David Alberto Carmona Coto, CC0 Public Domain]

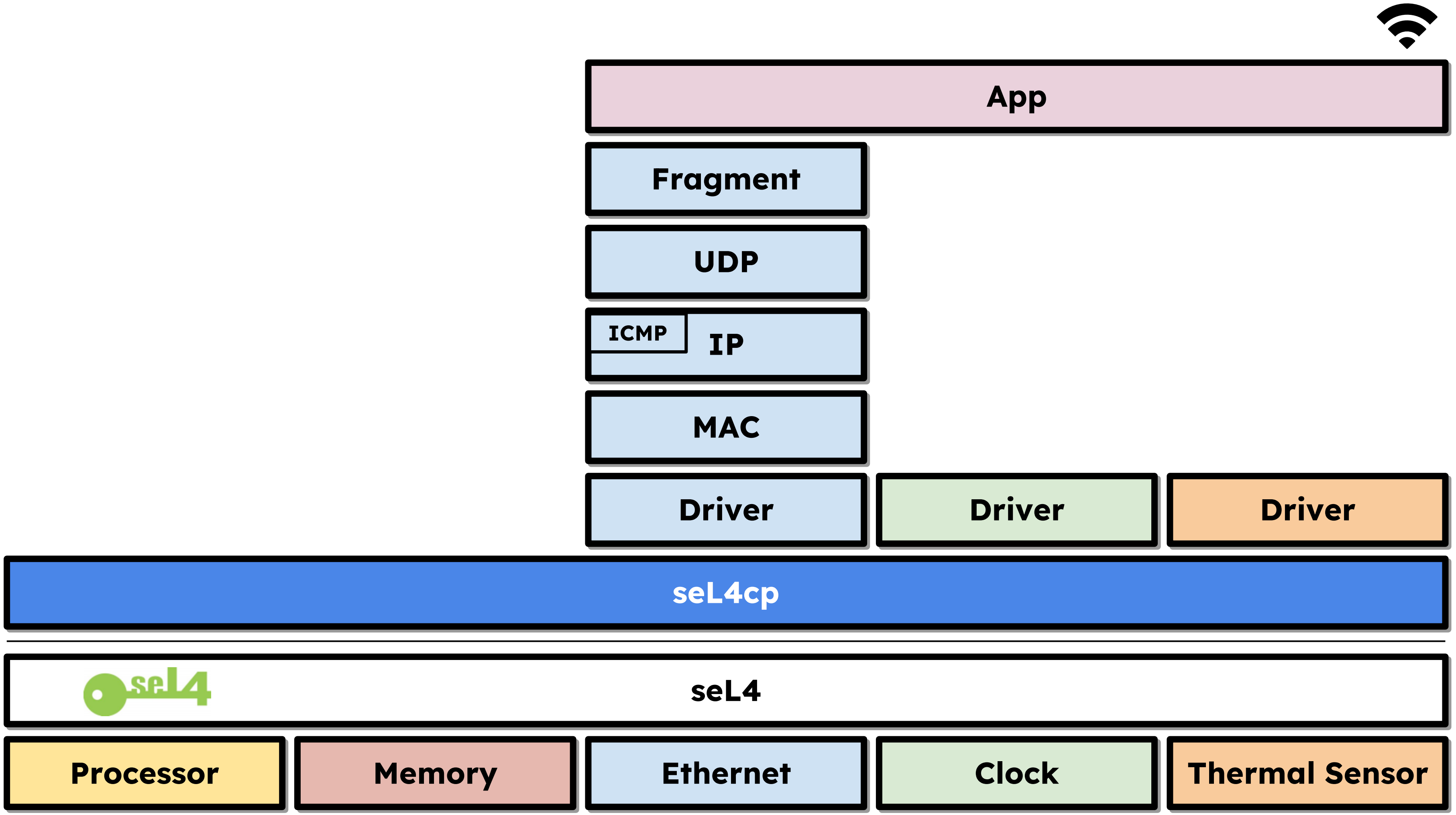


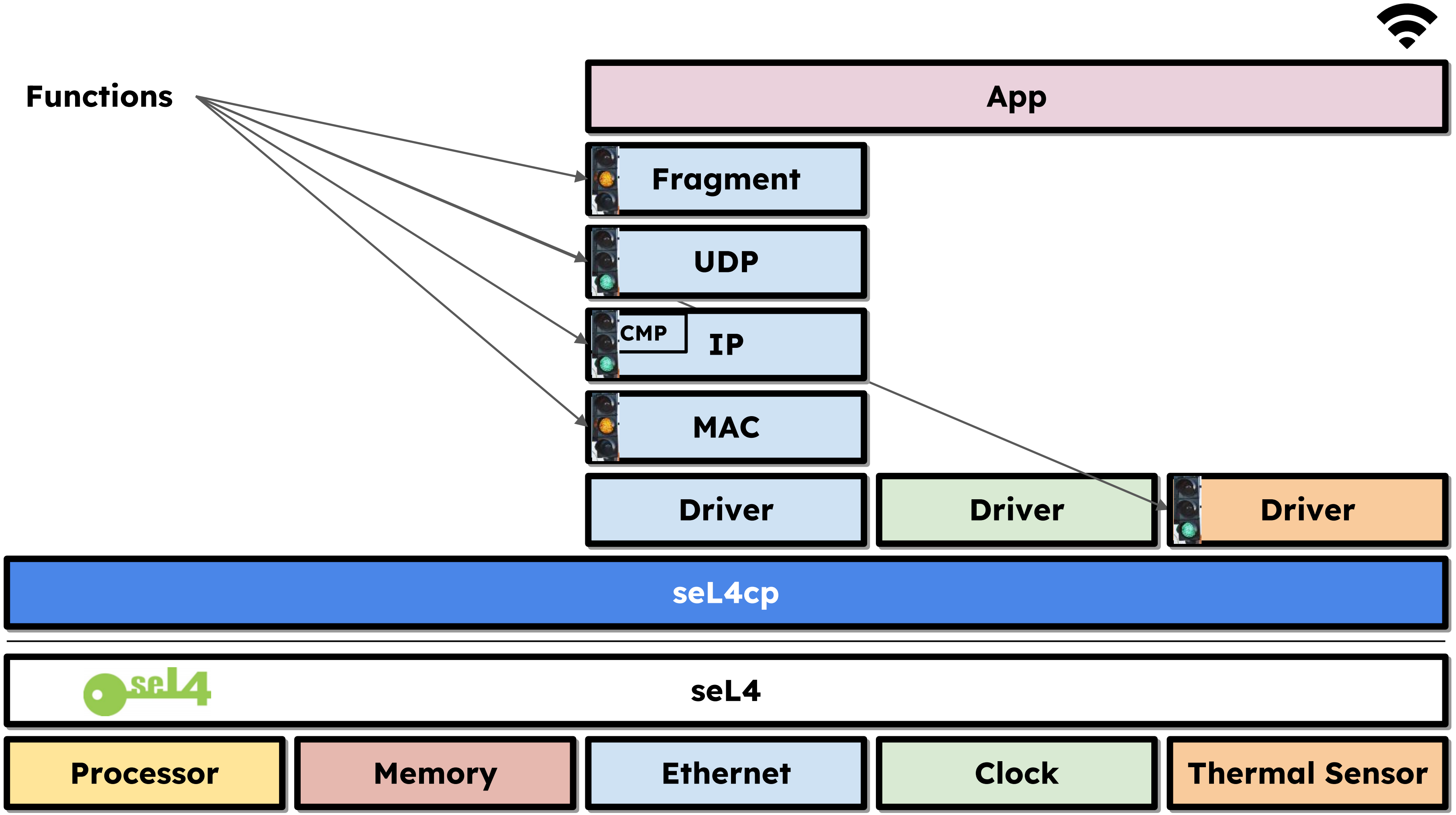
Teaching important (new) skills

- Build
- Correctness

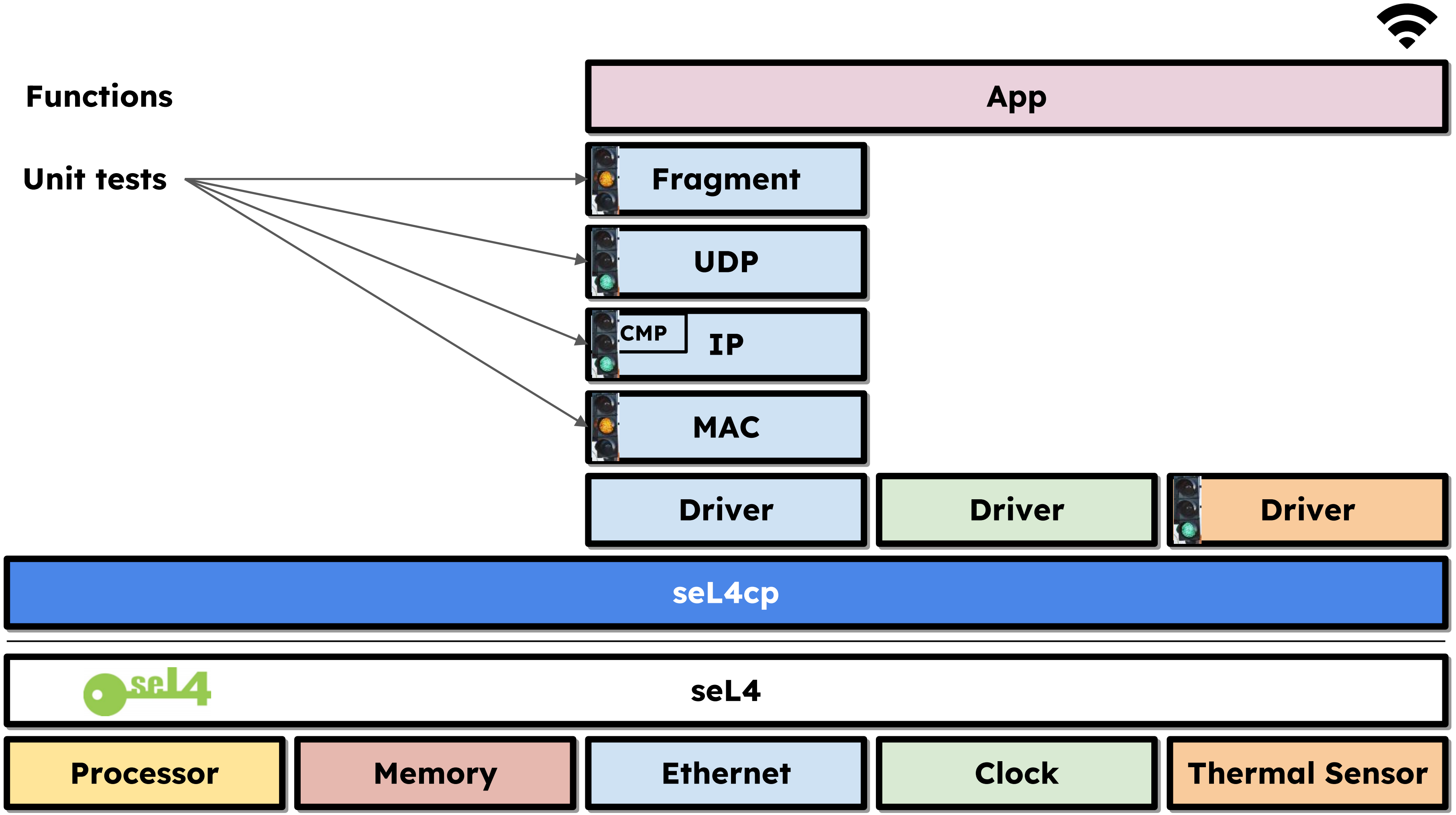
[FOTO:FORTEPAN / Lencse Zoltán, CC BY-SA 3.0]


Q.C
PASS
ODROID

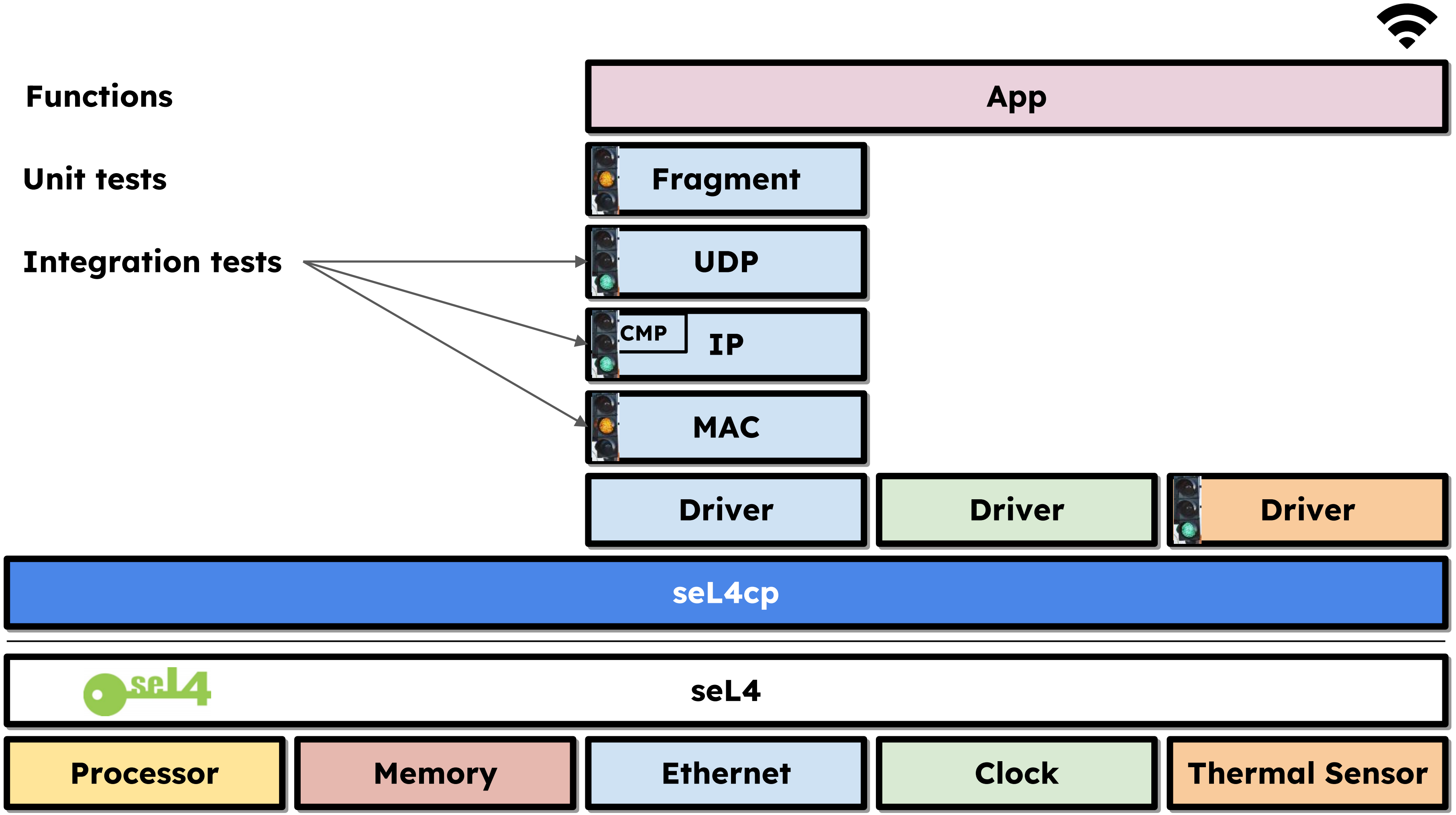

App
Fragment
UDP
ICMP
IP
MAC
Driver
Driver
Driver
seL4cp
seL4
seL4
Processor
Memory
Ethernet
Clock
Thermal Sensor



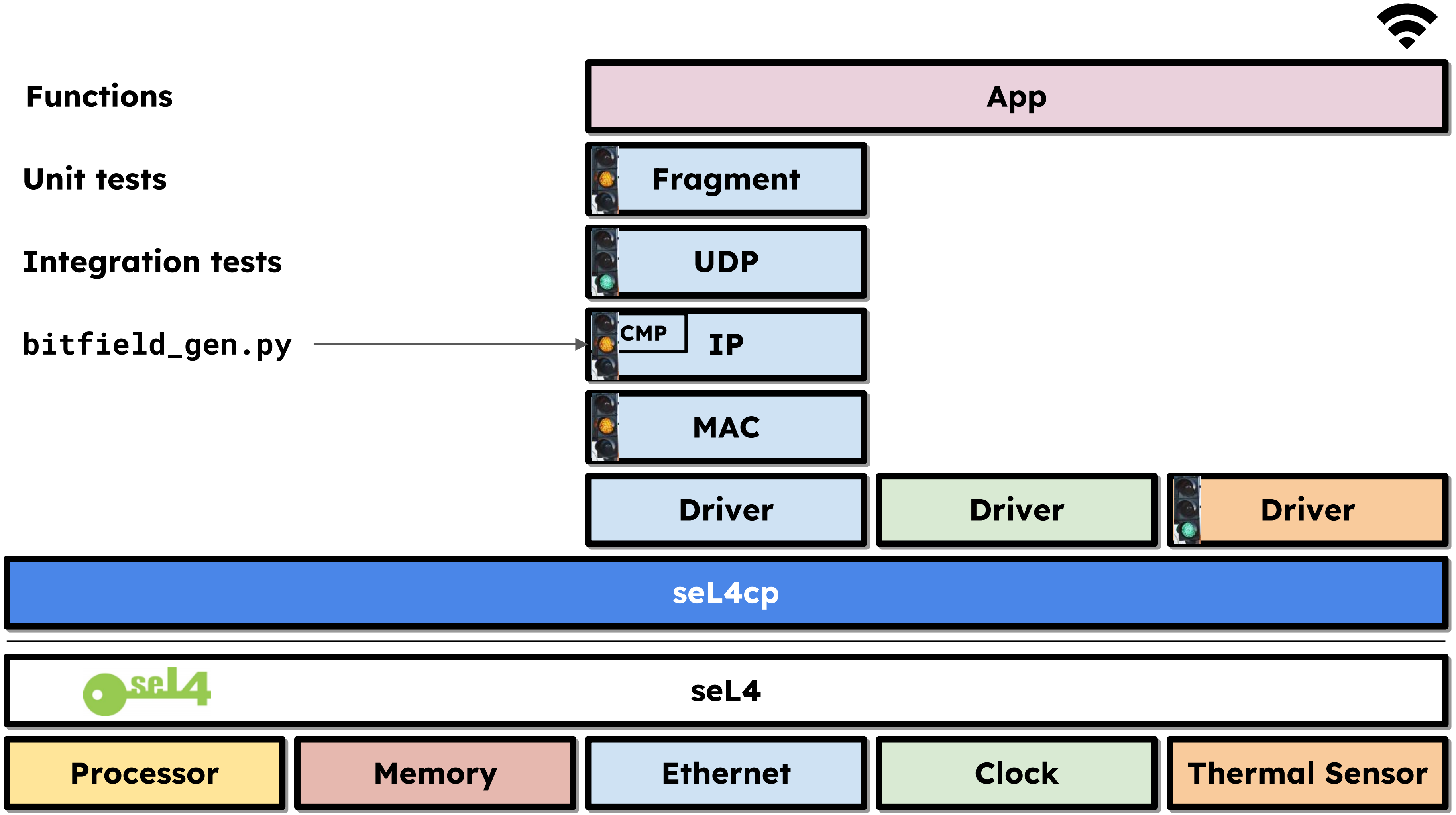
[pxhere.com, CC0 Public Domain]



[pxhere.com, CC0 Public Domain]



[pxhere.com, CC0 Public Domain]



[pxhere.com, CC0 Public Domain]

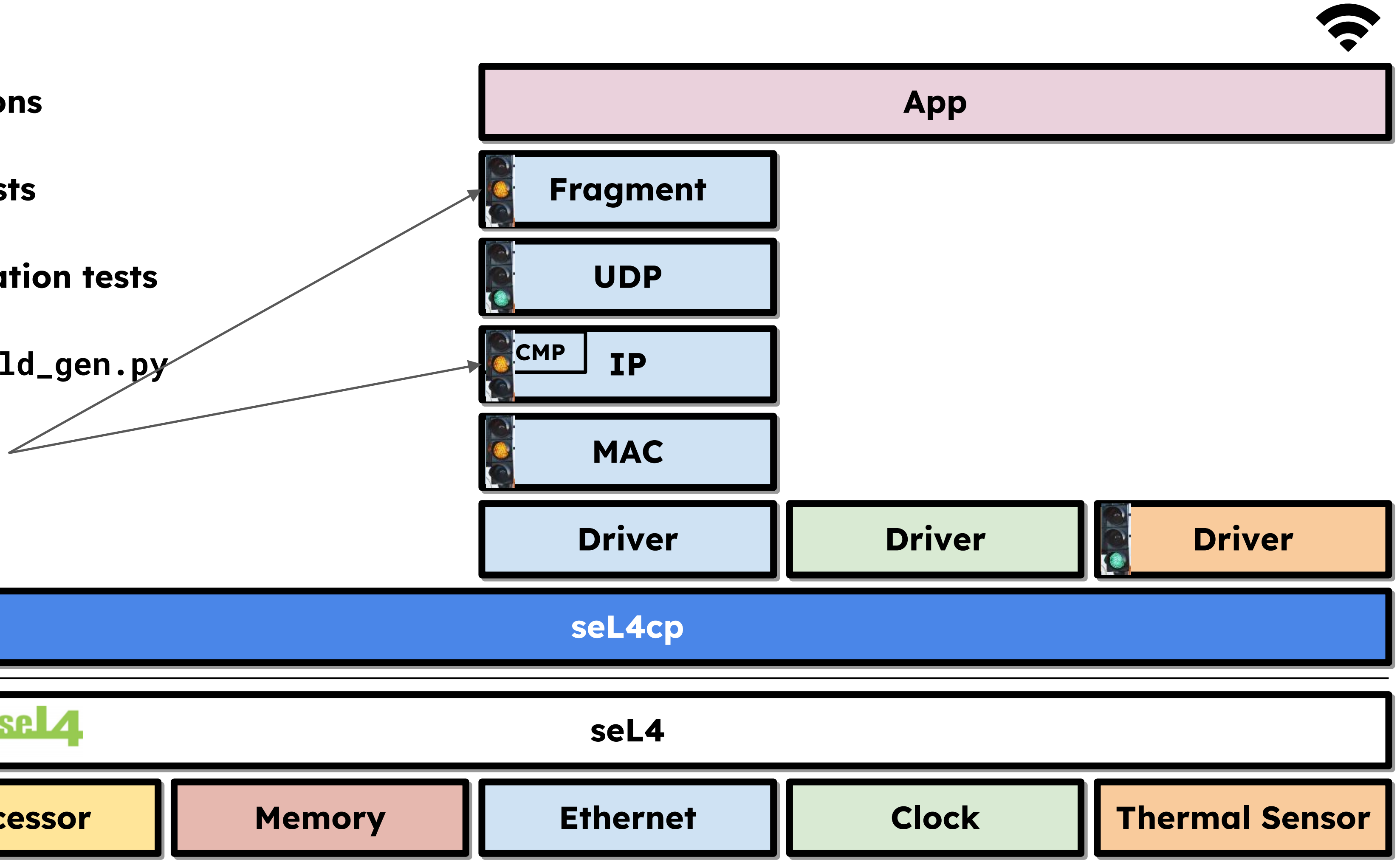
Functions

Unit tests

Integration tests

bitfield_gen.py

Proofs



[pxhere.com, CC0 Public Domain]

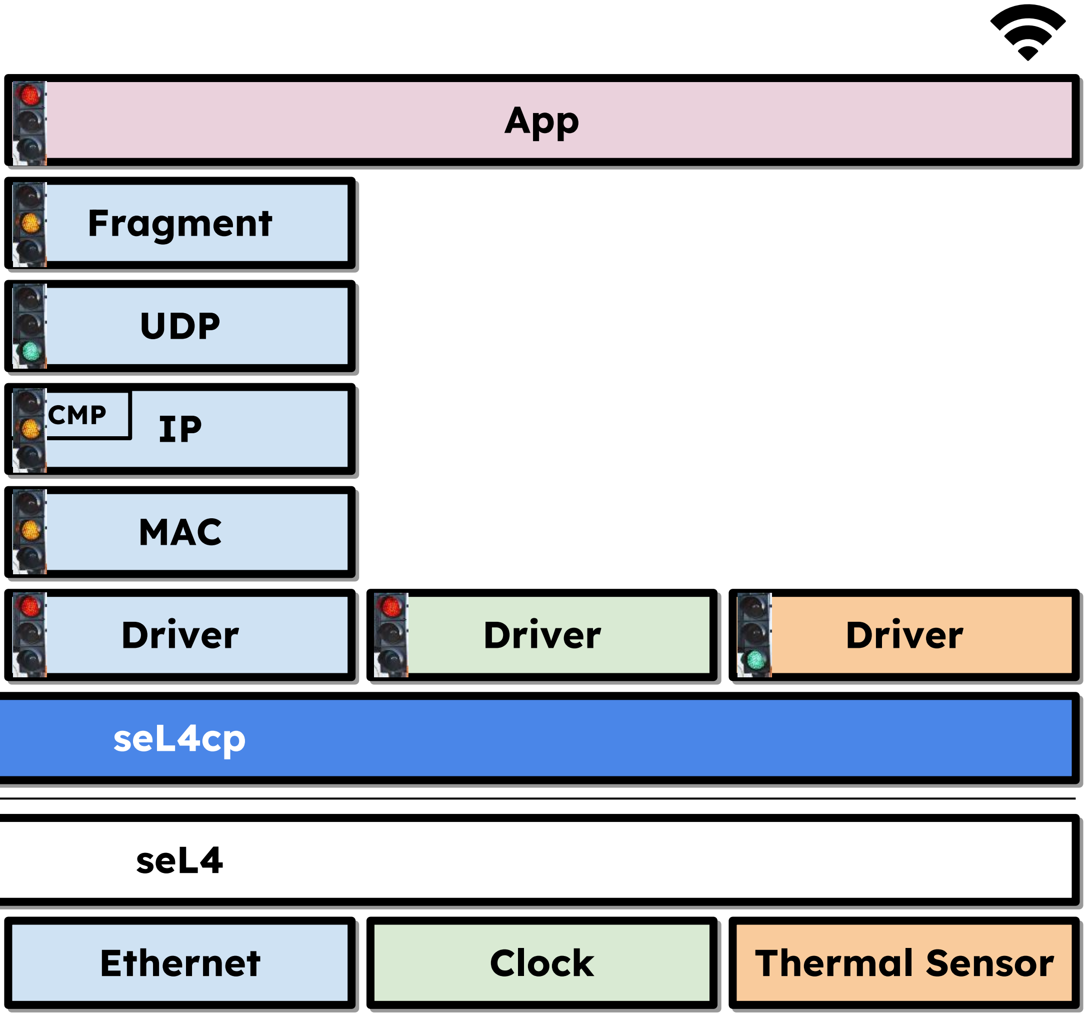
Functions

Unit tests

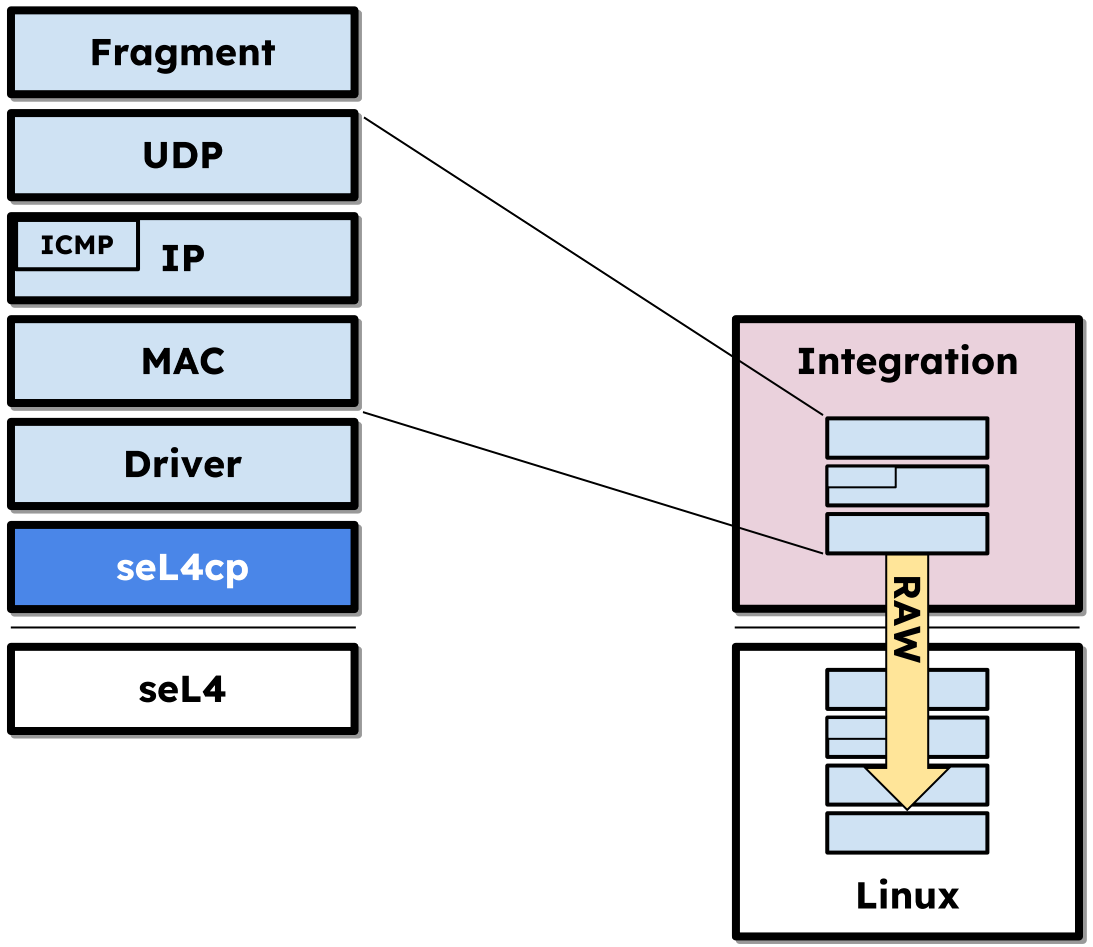
Integration tests

`bitfield_gen.py`

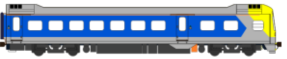
Proofs



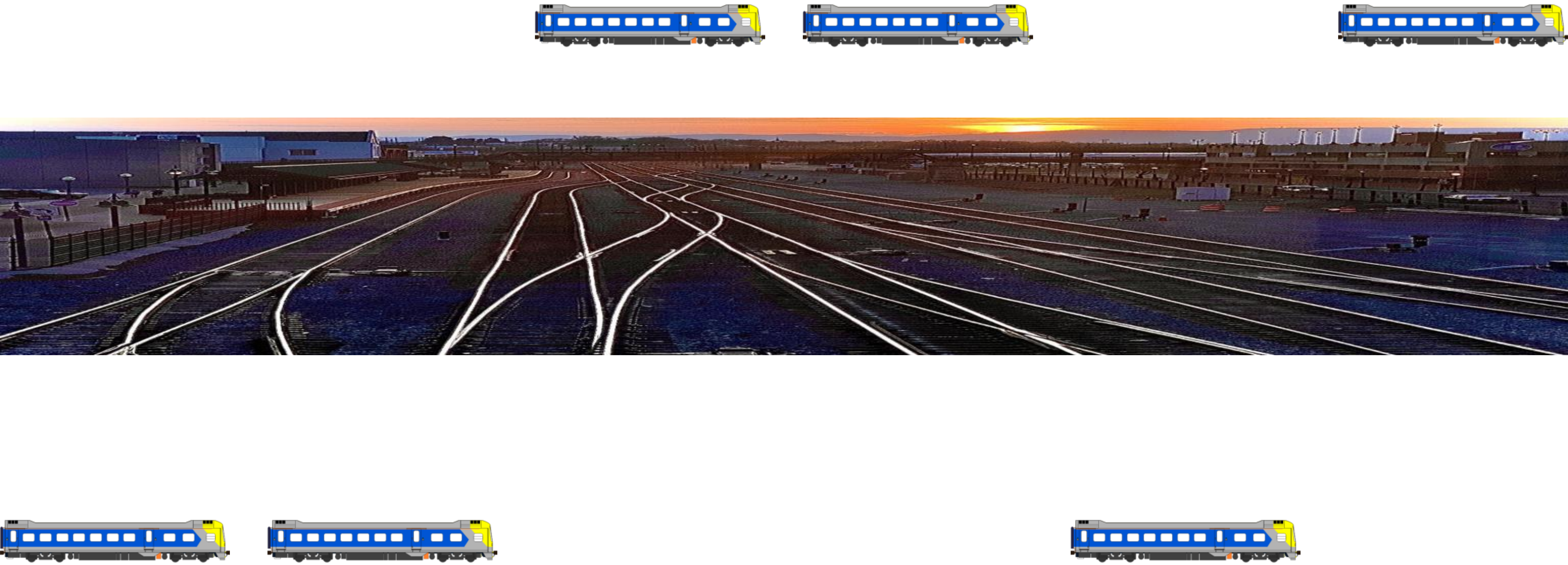
[pxhere.com, CC0 Public Domain]


Fragment
UDP
ICMP
IP
MAC
Driver
seL4cp
seL4
Integration
RAW
Linux

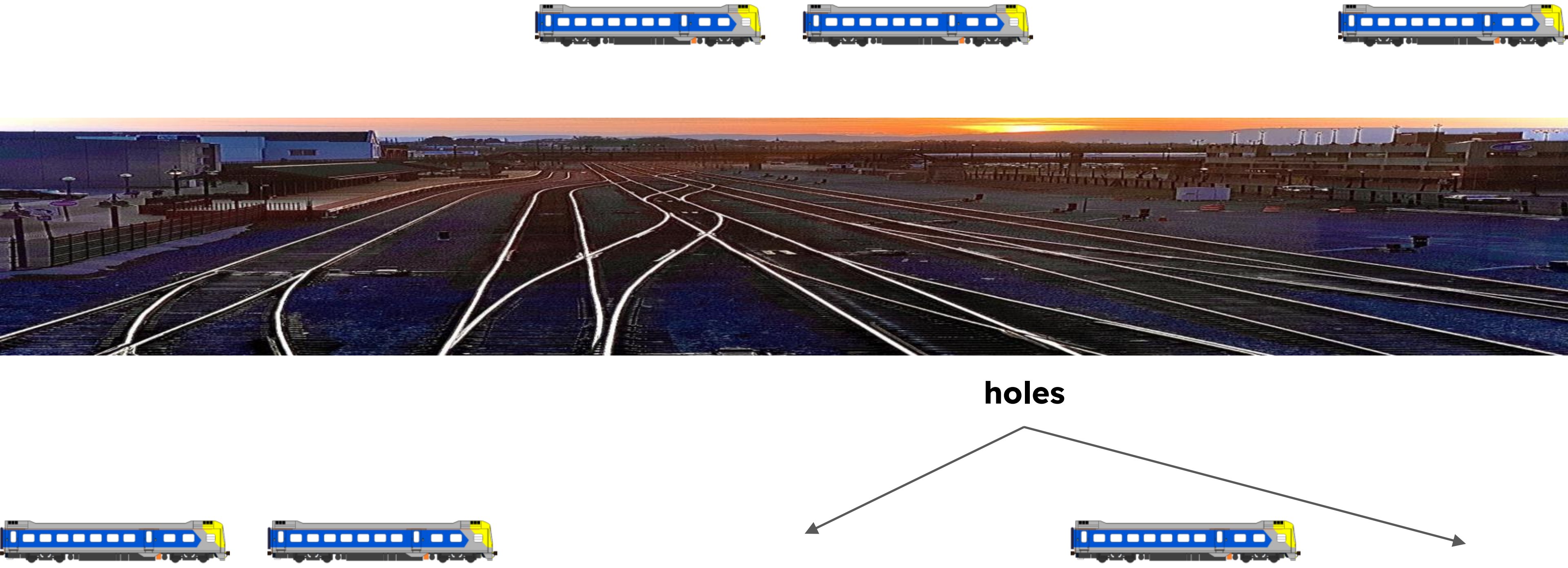
[freesvg.org, CC0 Public Domain & M F Flaherty, CC BY-NC-ND 2.0]



[freesvg.org, CC0 Public Domain & M F Flaherty, CC BY-NC-ND 2.0]

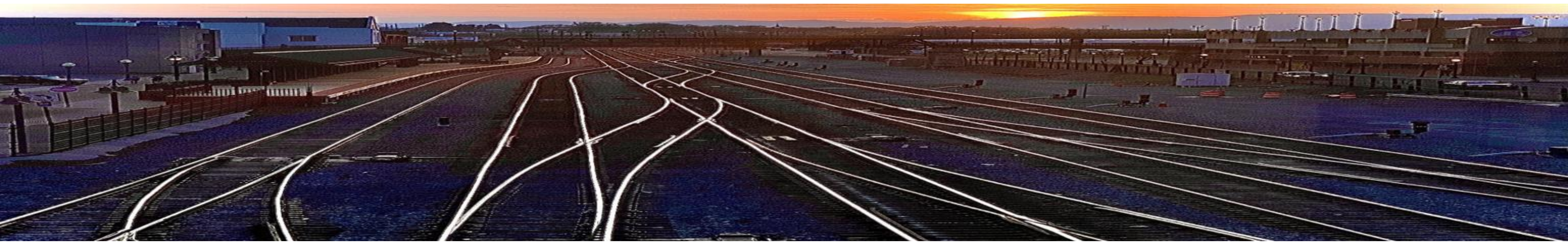


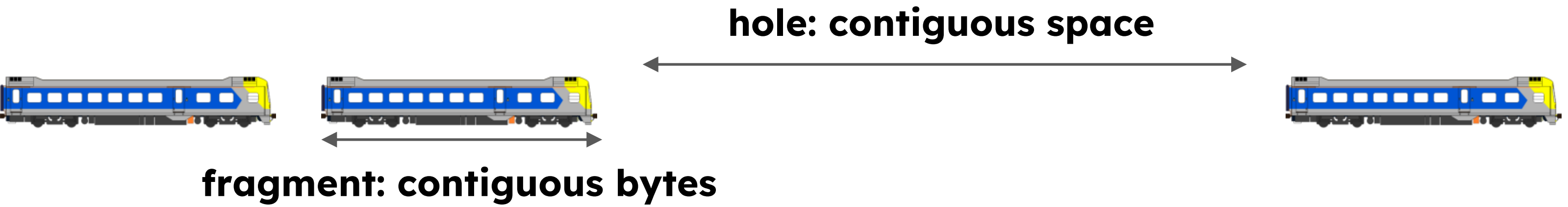
[freesvg.org, CC0 Public Domain & M F Flaherty, CC BY-NC-ND 2.0]



[freesvg.org, CC0 Public Domain & M F Flaherty, CC BY-NC-ND 2.0]

fragment :: interval
hole :: interval
interval ≡ { position, length }





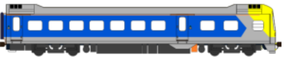
[freesvg.org, CC0 Public Domain & M F Flaherty, CC BY-NC-ND 2.0]

initial state: a single big hole

$$\underbrace{\hspace{20em}}$$

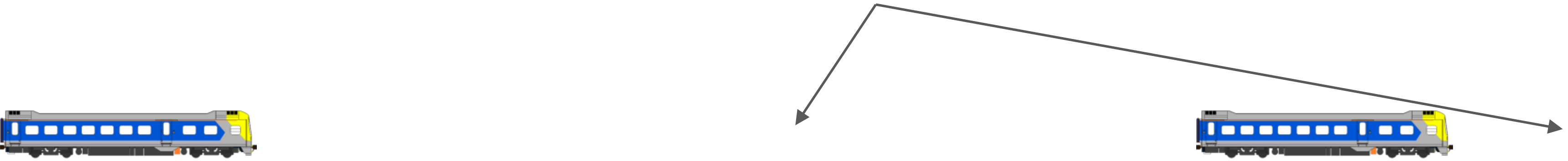
[freesvg.org, CC0 Public Domain & M F Flaherty, CC BY-NC-ND 2.0]

a single, smaller hole



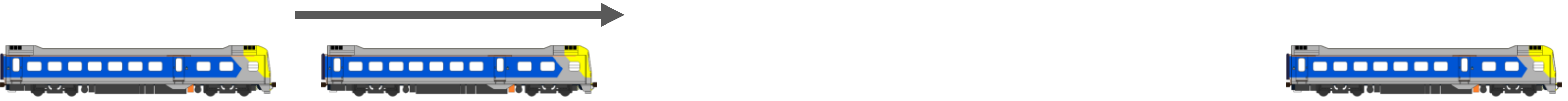
[freesvg.org, CC0 Public Domain & M F Flaherty, CC BY-NC-ND 2.0]

***n* holes ⇒ (*n* + 1) (smaller) holes**



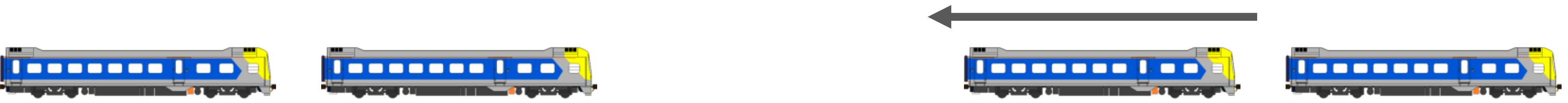
[freesvg.org, CC0 Public Domain & M F Flaherty, CC BY-NC-ND 2.0]

existing hole shrinks from the left



[freesvg.org, CC0 Public Domain & M F Flaherty, CC BY-NC-ND 2.0]

existing hole shrinks from the right



existing hole is filled perfectly

```
int reassemble(struct fragment *f) {
    struct hole *h = container_of(hole_list.next, struct hole, node);
    while (h != container_of(&hole_list, struct hole, node)) {

        if (f->first > h->last) goto next;                  // step 2
        if (f->last < h->first) goto next;                  // step 3
        list_del(&h->node);                                 // step 4
        if (f->first > h->first)                            // step 5
            list_add_tail(&hole_list, &hole_new(h->first, f->first - 1)->node);
        if (f->last < h->last && f->more)                   // step 6
            list_add_tail(&hole_list, &hole_new(f->last + 1, h->last)->node);
        free(h);
        break;
      next:
        h = container_of(h->node.next, struct hole, node);  // step 7 and 1
    }
    if (list_is_empty(&hole_list)) return 1;                // step 8

    return 0;
}
```

```
fun process1 :: "interval ⇒ interval ⇒ interval list" where
  "process1 h f = (     if h = f                                        then []
                   else if beg h < beg f ∧ beg f + len f < beg h + len h then split_hole_c h f
                   else if beg h = beg f                                 then split_hole_l h f
                   else                                                       split_hole_r h f)"
```

```
 0      7 8     15 16    23 24    31
+--------+--------+--------+--------+
|     Source      |   Destination   |
|      Port       |      Port       |
+--------+--------+--------+--------+
|                 |                 |
|     Length      |    Checksum     |
+--------+--------+--------+--------+
|
|          data octets ...
+---------------- ...

     User Datagram Header Format
```

Checksum is the 16-bit one's complement of the one's complement sum of a pseudo header of information from the IP header, the UDP header, and the data, padded with zero octets at the end (if necessary) to make a multiple of two octets.

[RFC 768, IETF, Jon Poster, Ed.]

```
    /* Compute Internet Checksum for "count" bytes
     *         beginning at location "addr".
     */
register long sum = 0;

 while( count > 1 )  {
    /*  This is the inner loop */
        sum += * (unsigned short) addr++;
        count -= 2;
}

    /*  Add left-over byte, if any */
if( count > 0 )
        sum += * (unsigned char *) addr;

    /*  Fold 32-bit sum to 16 bits */
while (sum>>16)
    sum = (sum & 0xffff) + (sum >> 16);

checksum = ~sum;
```

```
    /* Compute Internet Checksum for "count" bytes
     *         beginning at location "addr".
     */
register long sum = 0;

 while( count > 1 )  {
    /*  This is the inner loop */
        sum += * (unsigned short) addr~~++~~;
        count -= 2;
}

    /*  Add left-over byte, if any */
if( count > 0 )
        sum += * (unsigned char *) addr;

    /*  Fold 32-bit sum to 16 bits */
while (sum>>16)
    sum = (sum & 0xffff) + (sum >> 16);

checksum = ~sum;
```

addr += 2

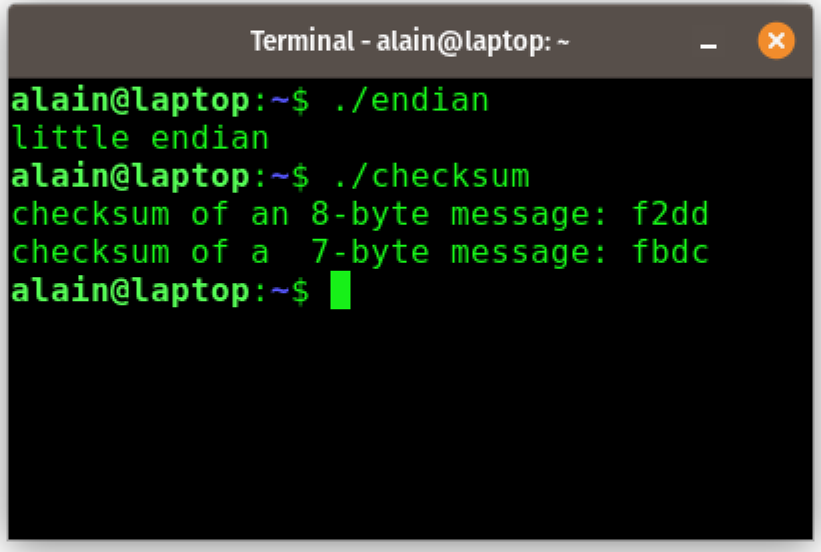
(unsigned short *)

```
    /* Compute Internet Checksum for "count" bytes
     *         beginning at location "addr".
     */
register long sum = 0;

 while( count > 1 )  {
    /*  This is the inner loop */
        sum += * (unsigned short) addr~~++~~;
        count -= 2;
}

    /*  Add left-over byte, if any */
if( count > 0 )
        sum += * (unsigned char *) addr;

    /*  Fold 32-bit sum to 16 bits */
while (sum>>16)
    sum = (sum & 0xffff) + (sum >> 16);

checksum = ~sum;
```

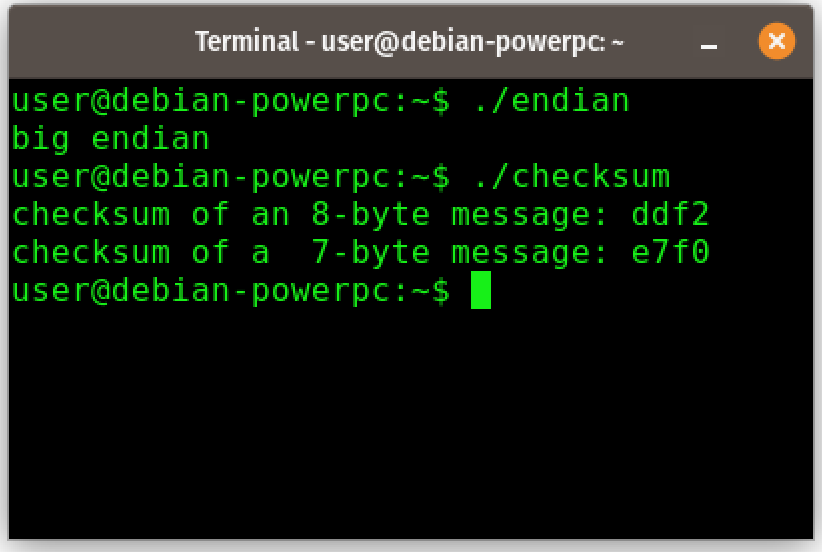
addr += 2

(unsigned short *)

wrong in big endian



```
alain@laptop:~$ ./endian
little endian
alain@laptop:~$ ./checksum
checksum of an 8-byte message: f2dd
checksum of a  7-byte message: fbdc
alain@laptop:~$
```



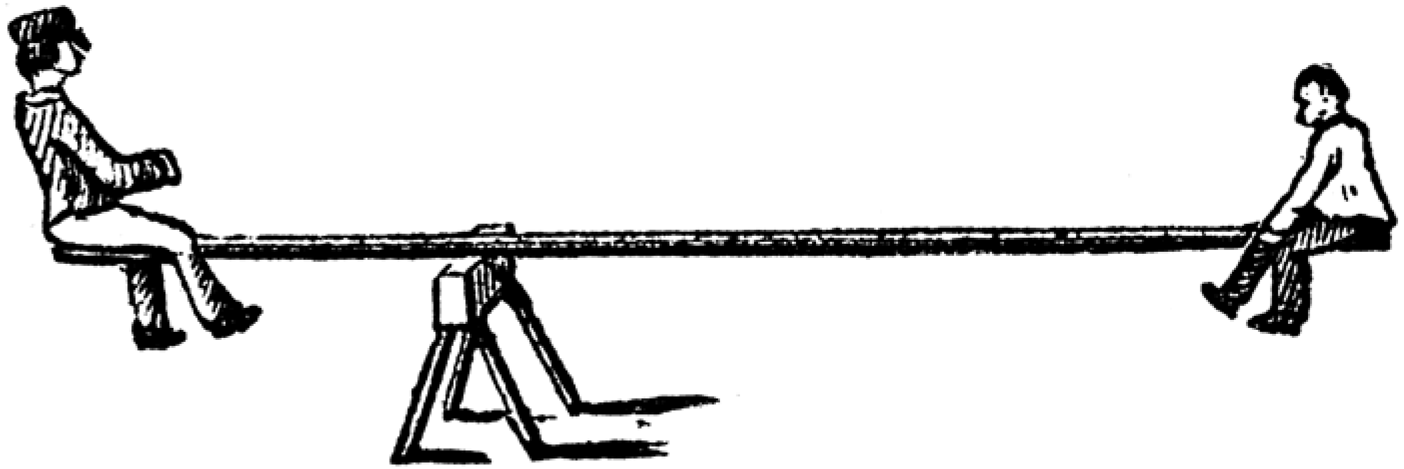
```
user@debian-powerpc:~$ ./endian
big endian
user@debian-powerpc:~$ ./checksum
checksum of an 8-byte message: ddf2
checksum of a  7-byte message: e7f0
user@debian-powerpc:~$
```

```
theory Word_16
imports
  More_Word
  Signed_Words
begin

lemma len16: "len_
simp

lemma word16_and_m
  ‹x AND 0xFFFF =
```

```
theory More_Word
imports
  "HOL-Library.Word"
  More_Arithmetic
  More_Divides
begin

lemma unat_power_lower [simp]:
```

```
theory Word
imports
  "HOL-Library.Type_Length"
begin

...

quotient_type (overloaded) 'a word = int / ‹λk l. take_bit LENGTH('a) k = take_bit LENGTH('a::len) l›
  morphisms rep Word by (auto intro!: equivpI reflpI sympI transpI)
...
```

```
theory Type_Length
imports
  Numeral_Type"
begin

class len0 =
  fixes len_of :: "'a itself ⇒ nat"
...
```



[Image taken from G. P. Quackenbos A.M. A Natural Philosophy: *Embracing the Most Recent Discoveries in the Various Branches of Physics, and Exhibiting the Application of Scientific Principles in Every-day Life* (New York: D. Appleton and Company, 1859)]

Tobias Nipkow, Gerwin Klein

Concrete Semantics

with Isabelle/HOL

October 16, 2017

Springer-Verlag

“These endless software updates are killing your joie de vivre.”

[Image taken from The New Yorker, David Sipress]

FoxNet — Network Protocol Stack in Standard ML

Hello — An Operating System in Standard ML

Kestrel Institute's APT Toolkit — Correct-by-construction TCP/IP stack

Mirage — An OCaml TCP/IP Networking Stack

Netsem — Rigorous Test-Oracle Specification & Validation for TCP/IP

Reimplementing TCP/IP — A User-space TCP Implementation in OCaml

Secure MANET — Security Verification of MANET Routing Protocols

Stenning's Protocol — Implemented in UDP and Verified in Isabelle

TRENTOS — A Secure Operating System over seL4

VerifiedSCION — Internet Architecture for Secure Routing & Forwarding