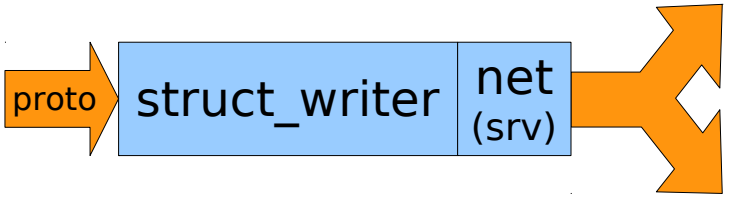
hbook/root/struct_writer
interface & protocol

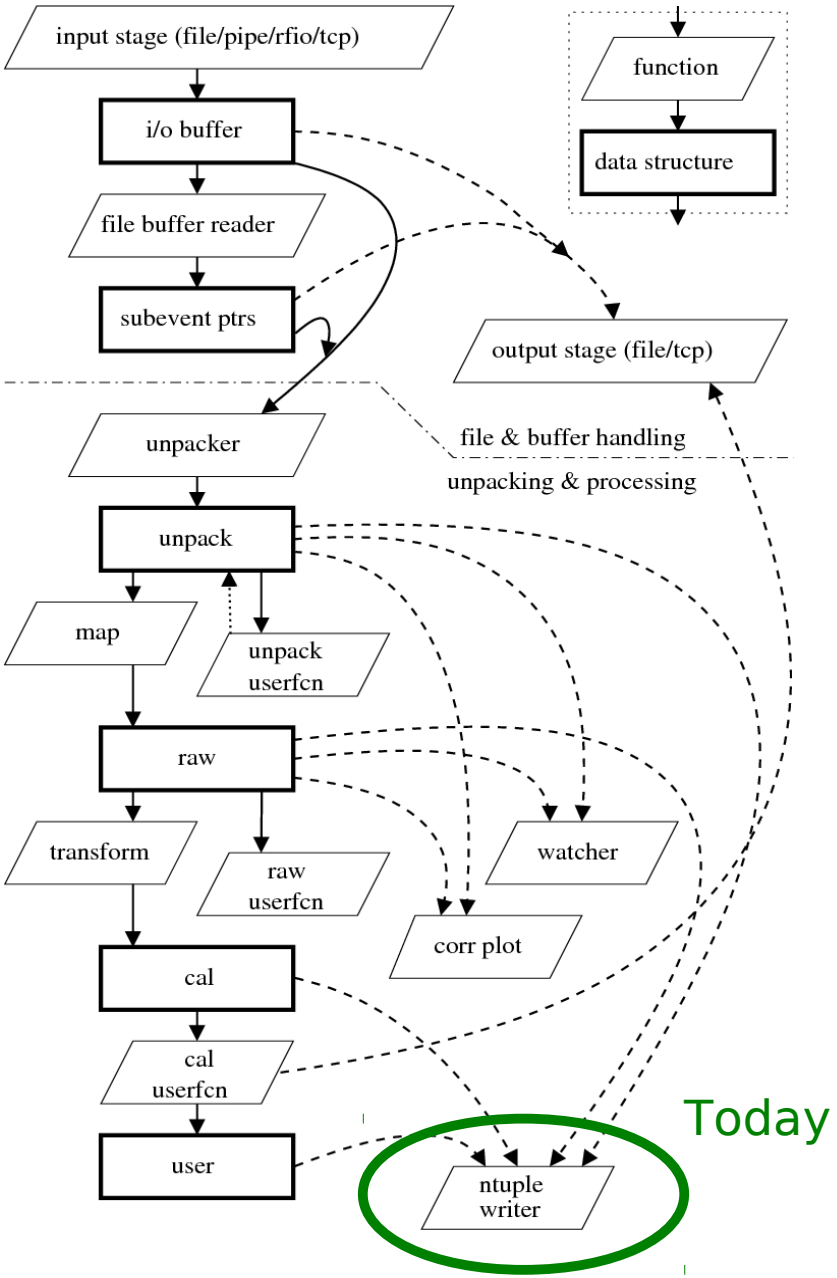


A simple tool to pass data along

Håkan T. Johansson, Chalmers, Göteborg

Lichtenberghaus, Darmstadt, May 2015

Primary scope of UCESB



'Quick-n-dirty' generic unpacking and data 'quality' monitor

Unpack code generation from C structure-like specification:

```
SUPER_TDC(slot)
{
  UINT32 value;
}
SUBEVENT(ONE_CRATE)
{
  tdc1 = SUPER_TDC(slot=5);
  tdc2 = SUPER_TDC(slot=6);
}
EVENT
{
  crate1 = ONE_CRATE(type=5);
}
```

http://fy.chalmers.se/~f96hajo/ucesb/

Module .spec structure

```
VME_CAEN_V775(geom,crate)
{
  MEMBER(DATA12_OVERFLOW data[32] ZERO_SUPPRESS);

  UINT32 header NOENCODE {
      8_13:  count;
      16_23: crate = MATCH(crate);
      24_26: 0b010;
      27_31: geom = MATCH(geom);
  }
  list(0<=index<header.count) {
      UINT32 ch_data NOENCODE {
          0_11:  value;

          12:    overflow;
          13:    underflow;
          14:    valid;

          16_20: channel;

          24_26: 0b000;
          27_31: geom = CHECK(geom);

          ENCODE(data[channel],(value=value,overflow=overflow));
      }
  }
  UINT32 eob {
      0_23:  event_number;
      24_26: 0b100;
      27_31: geom = CHECK(geom);
  }
}
```

CAEN V775 (TDC) data format:

Header

31 - 27	26-24	23 - 16		13 - 8	
geom	0 1 0	crate		count	

Data word

31 - 27	26-24		20 - 16		14	13	12	11 - 0
geom	0 0 0		channel		V	U	O	value

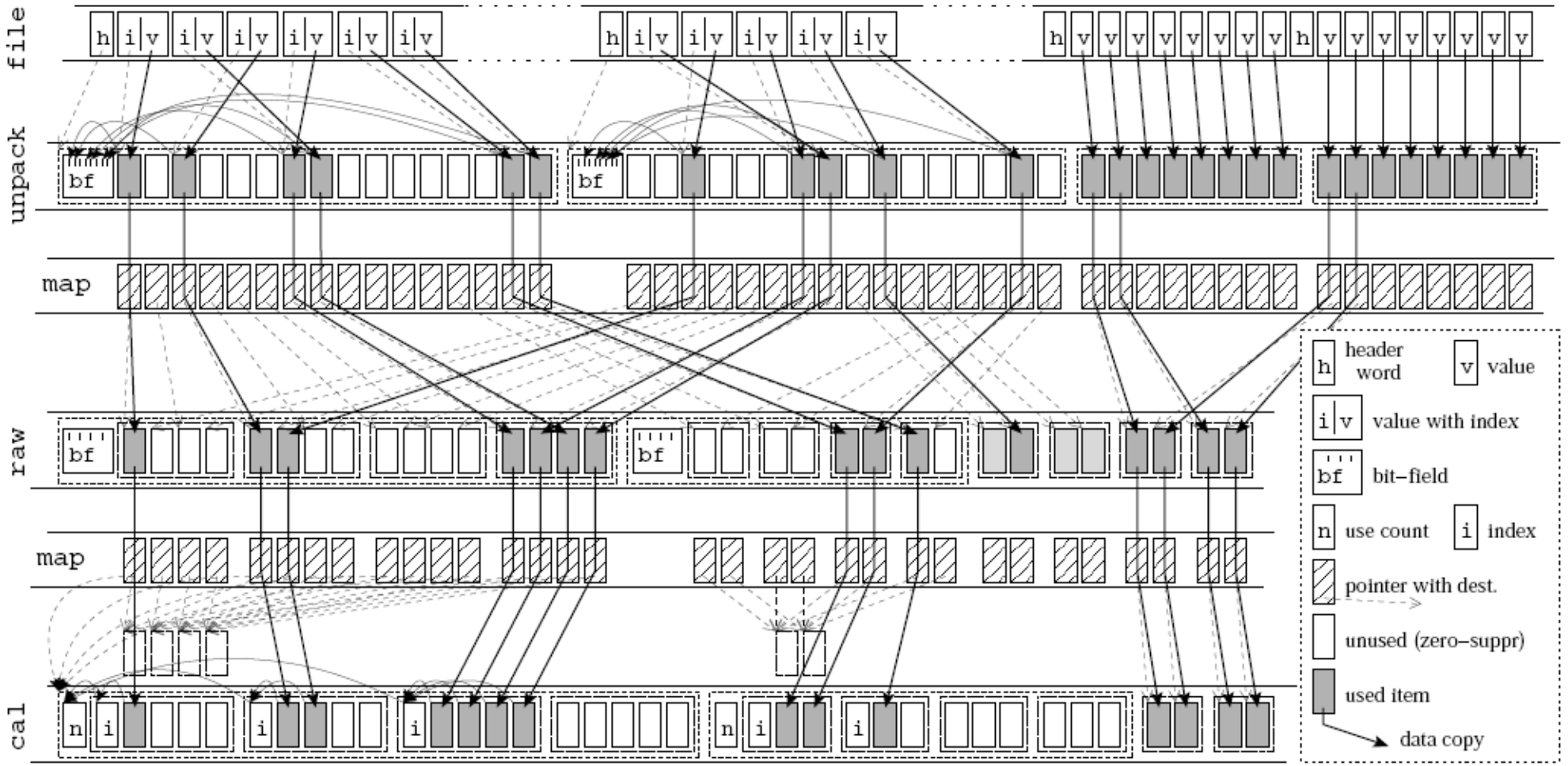
End marker

31 - 27	26-24	23 - 0
geom	1 0 0	event_number

Data structures for unpacking

“Show me your code and conceal your data structures, and I shall continue to be mystified. Show me your data structures, and I won't usually need your code; it'll be obvious.”

Eric Raymond



ntuple & ROOT tree generation

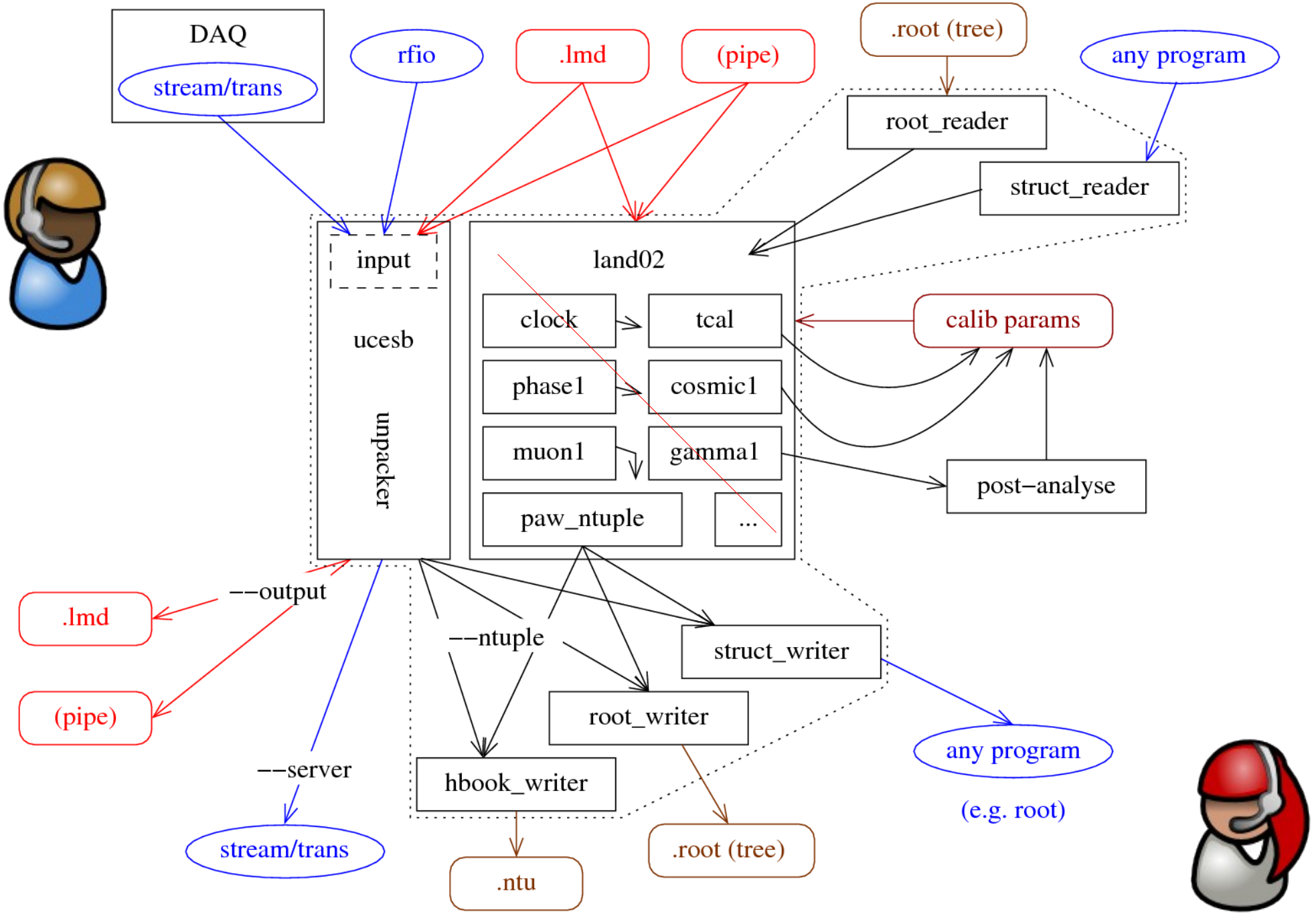
Select which data-levels to include.

Optionally limit which detectors / channels are included (also with indices).

Output file type selected by extension.

```
--ntuple=UNPACK,file.ntu

--ntuple=RAW,file.ntu

--ntuple=UNPACK,RAW,file.root

--ntuple=RAW,POS2,N,TFW,TOF,file.root

--ntuple=UNPACK,fastbus,camac,file.root
```

ucesb/(land02/ggland) interaction



'Any' program processing

```
#include "ext_data_client.h"
#include "ext_h101.h"
#include <stdlib.h>
#include <stdio.h>

int main(int argc,char *argv[])
{
  struct ext_data_client *client;

  EXT_STR_h101 event;
  EXT_STR_h101_layout event_layout = EXT_STR_h101_LAYOUT_INIT;

  if (argc < 2)
    {
      fprintf (stderr,"No server name given, usage: %s SERVER\n",argv[0]);
      exit(1);
    }

  client = ext_data_connect_stderr(argv[1]);

  if (client == NULL)
    exit(1);

  if (ext_data_setup_stderr(client,
                  &event_layout,sizeof(event_layout),
                  sizeof(event)))
    {
      for ( ; ; )
        {
          if (!ext_data_fetch_event_stderr(client,&event,sizeof(event)))
            break;

          /* Do whatever is wanted with the data. */

          printf ("%10d: %2d\n",event.EVENTNO,event.TRIGGER);
        }
    }
  ext_data_close_stderr(client);
  return 0;
}
```

Using the same structures as generated (virtually) for ntuples and ROOT trees

'Any' program processing

```
empty/empty /dev/null \
  --ntuple=UNPACK,STRUCT_HH,ext_h101.h
```

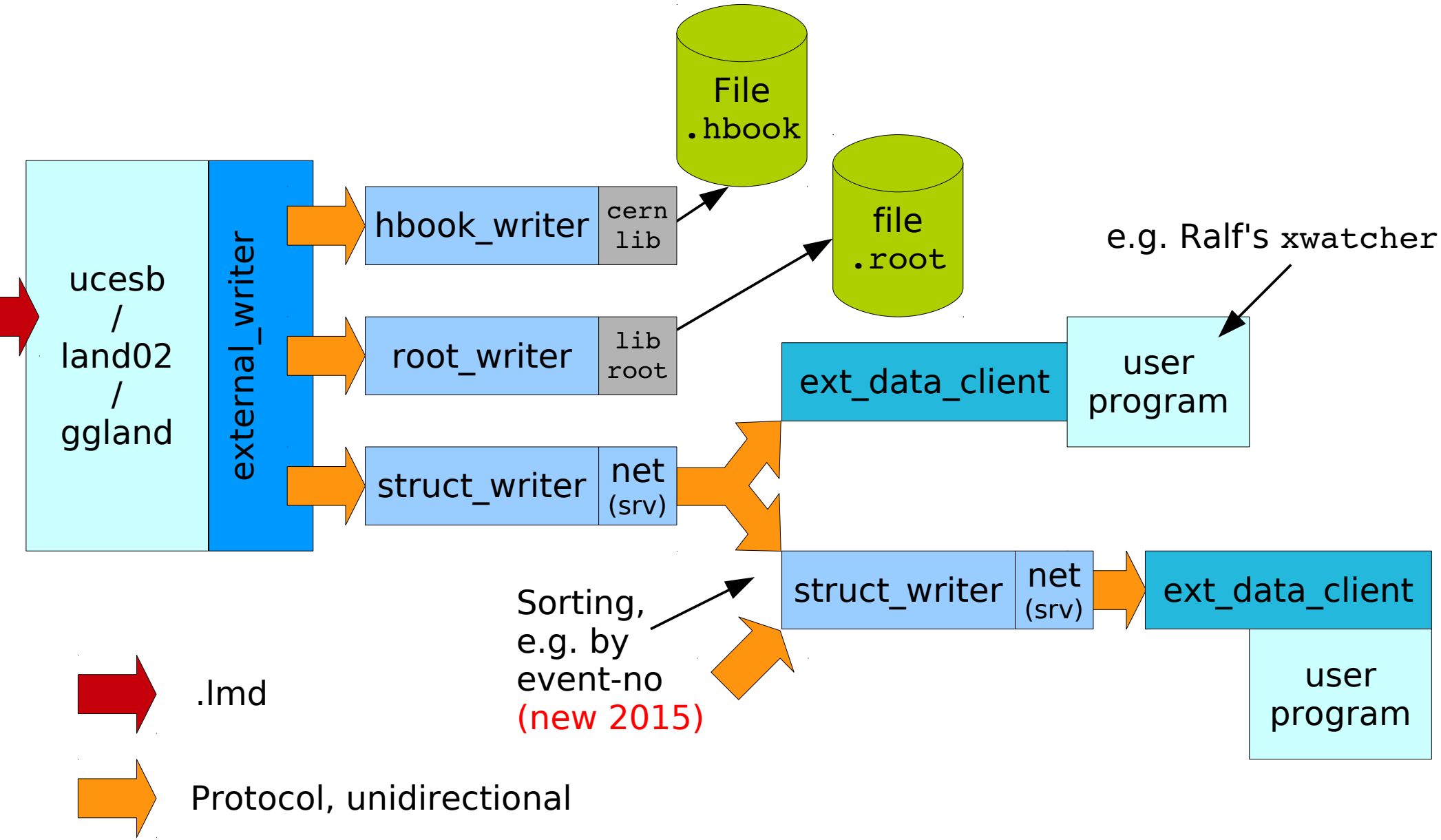
```
typedef struct EXT_STR_h101_t
{
  // UNPACK
  uint32_t TRIGGER;
  int32_t  EVENTNO;

} EXT_STR_h101;
```

Also as network server (any # clients)

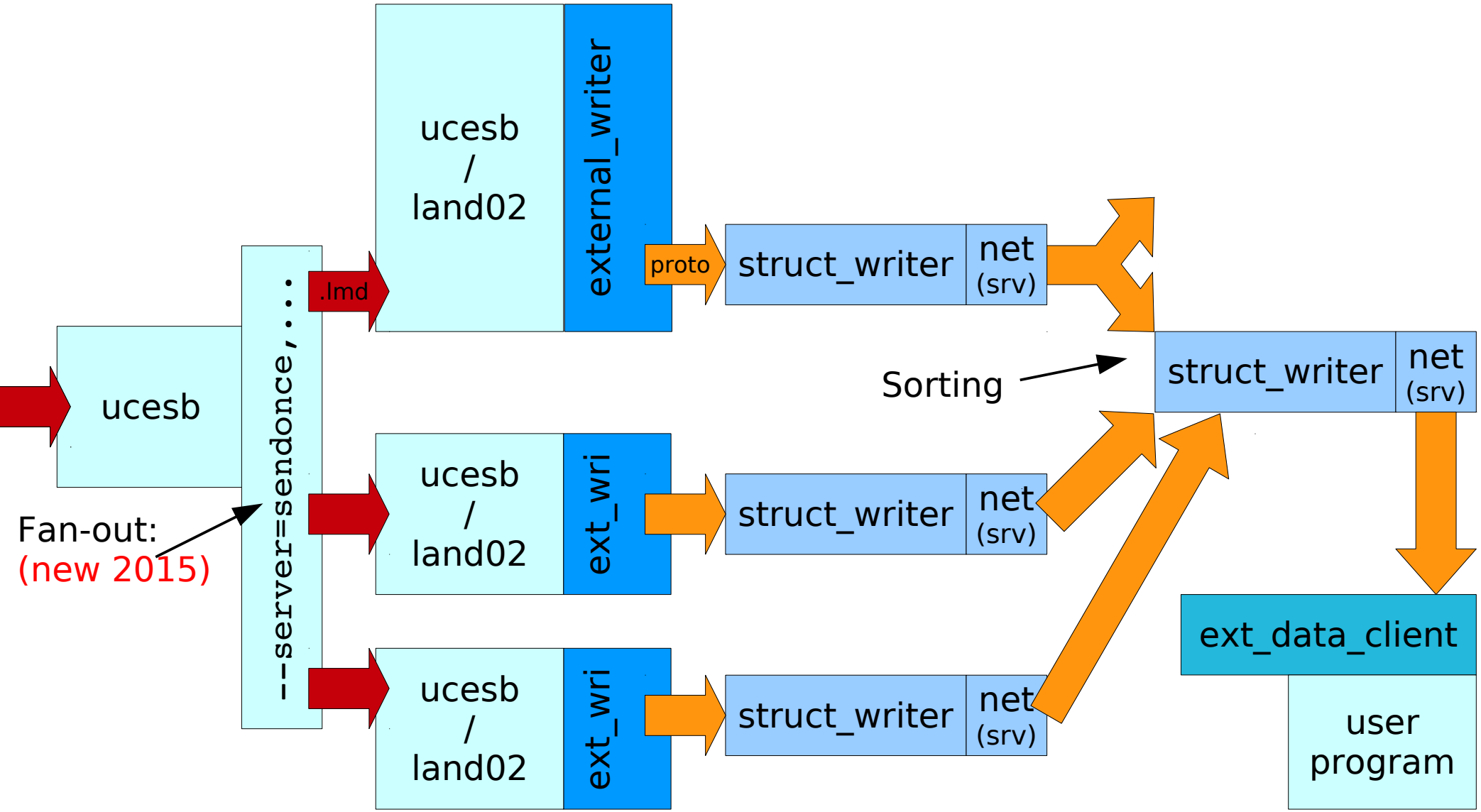
```
cc -g -O3 -o ext_reader_h101_stderr -I. ext_data_reader_stderr.c hbook/ext_data_client.o
```

```
empty/empty INFILE.lmd --ntuple=UNPACK,STRUCT,- | \
  ./ext_reader_h101_stderr -
```

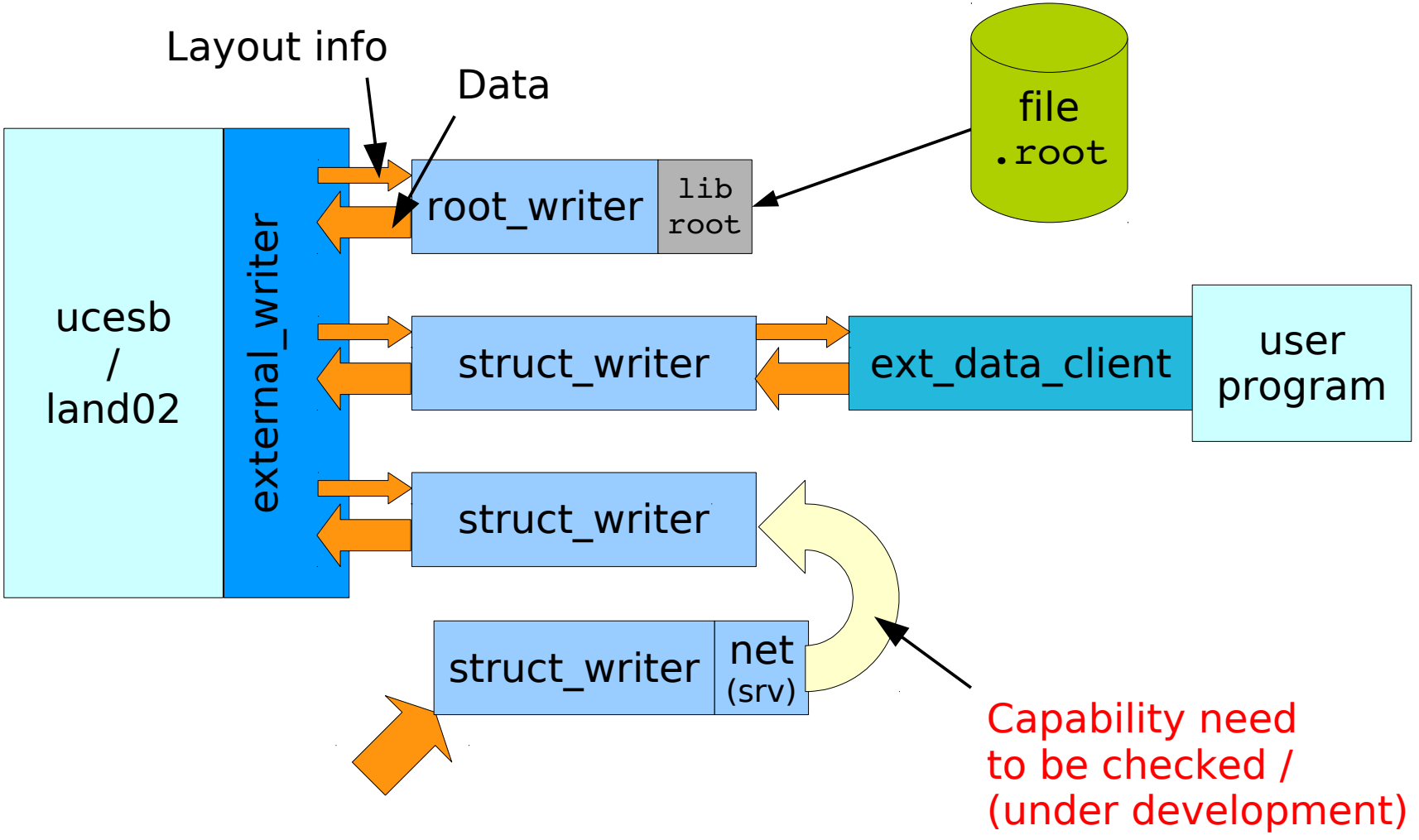
Data flow possibilities



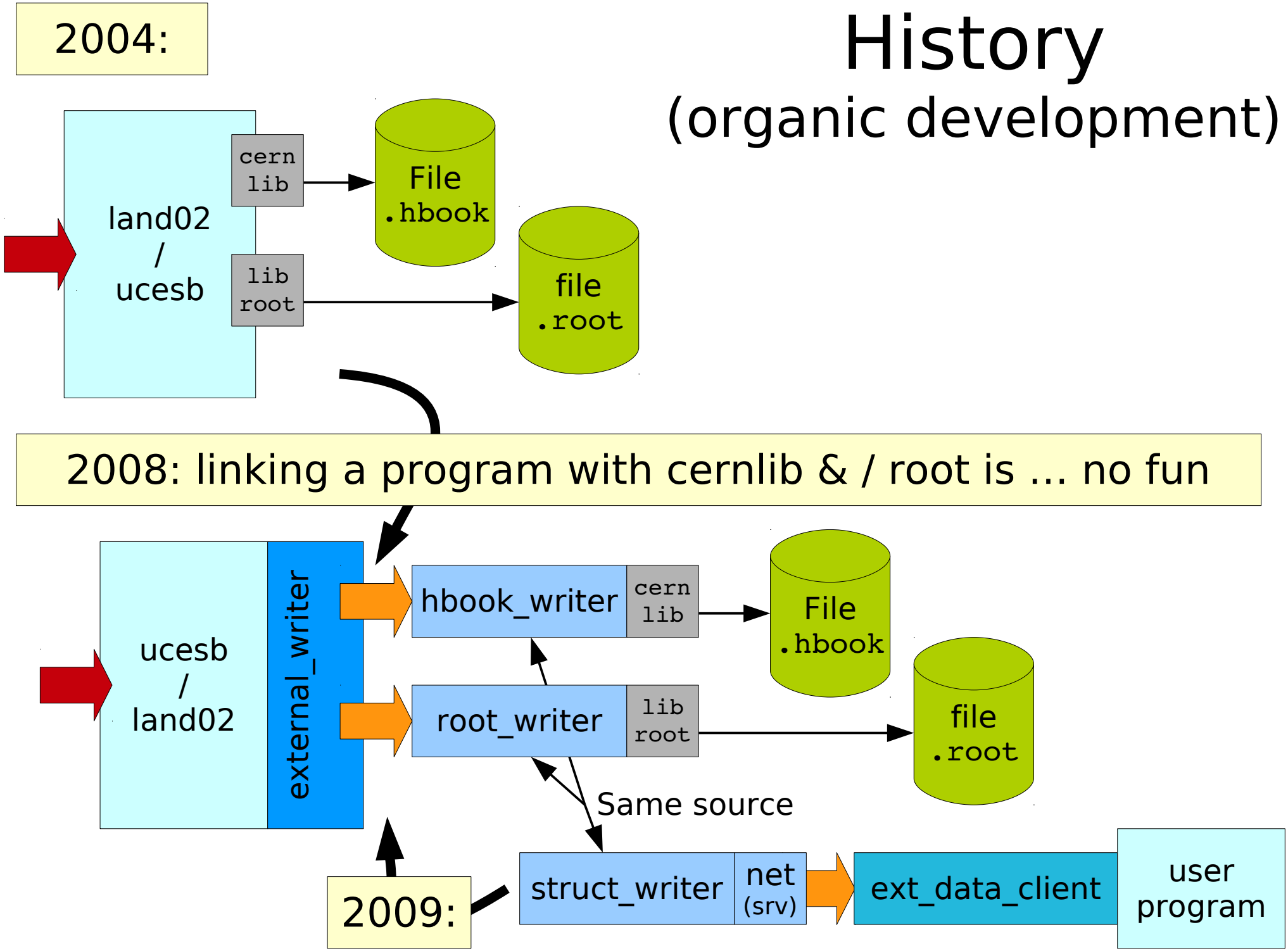
Easy parallel processing



Reverse flow ('reading')

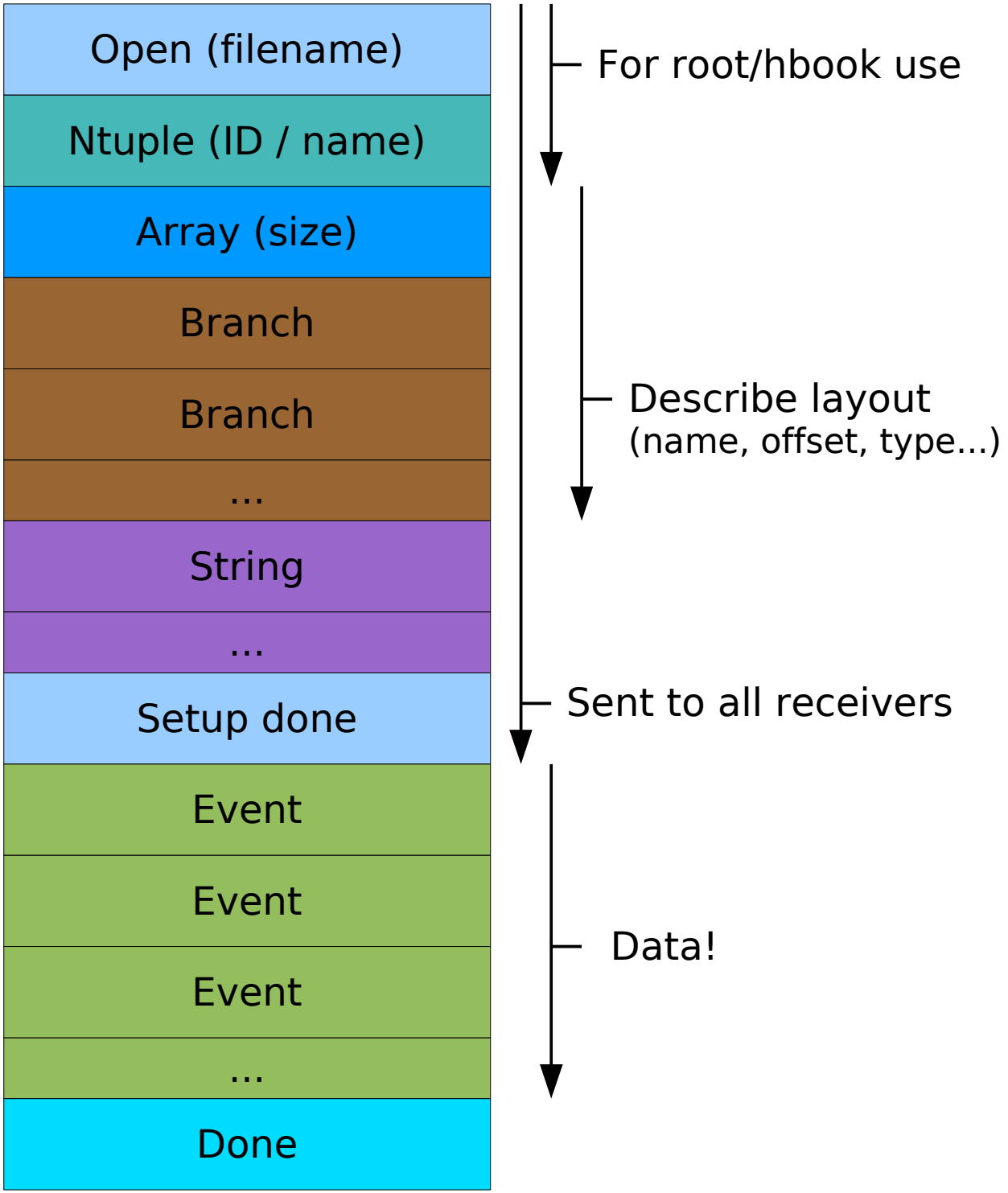


History
(organic development)



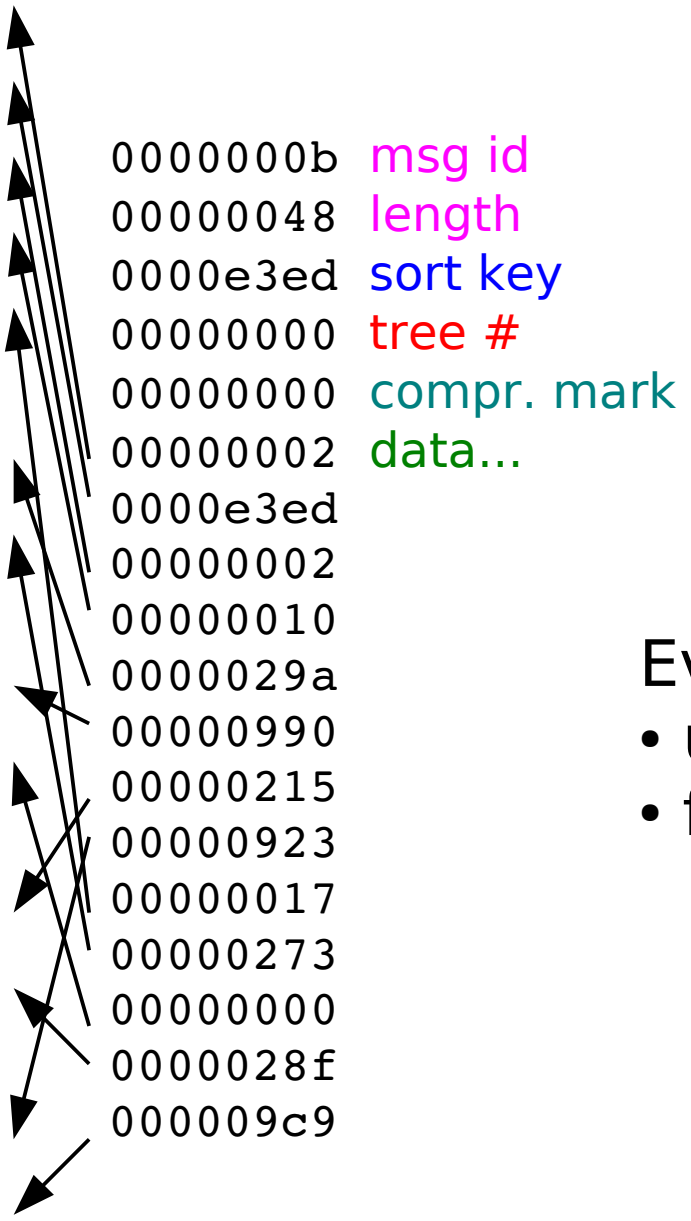
Protocol

- Unidirectional, streaming
  (fits pipeline)
- 'Network order'
  (byteswap: `htonl`/`ntohl`)
- For live/online only:
  (protocol not stable, defined by current code)
  For longterm: .lmd
  Medium: .root



Structure & protocol

TRIGGER	2
EVENTNO	58349
TFWn	2
TFWi[32]	0x10=16 0x17=23
TFWe1[32]	0x29a=666 0x273=627
TFWt1[32]	0x990=2448 0
TFWe1[32]	0x215=553 0x28f=655
TFWt1[32]	0x923=2339 0x9c9=2505

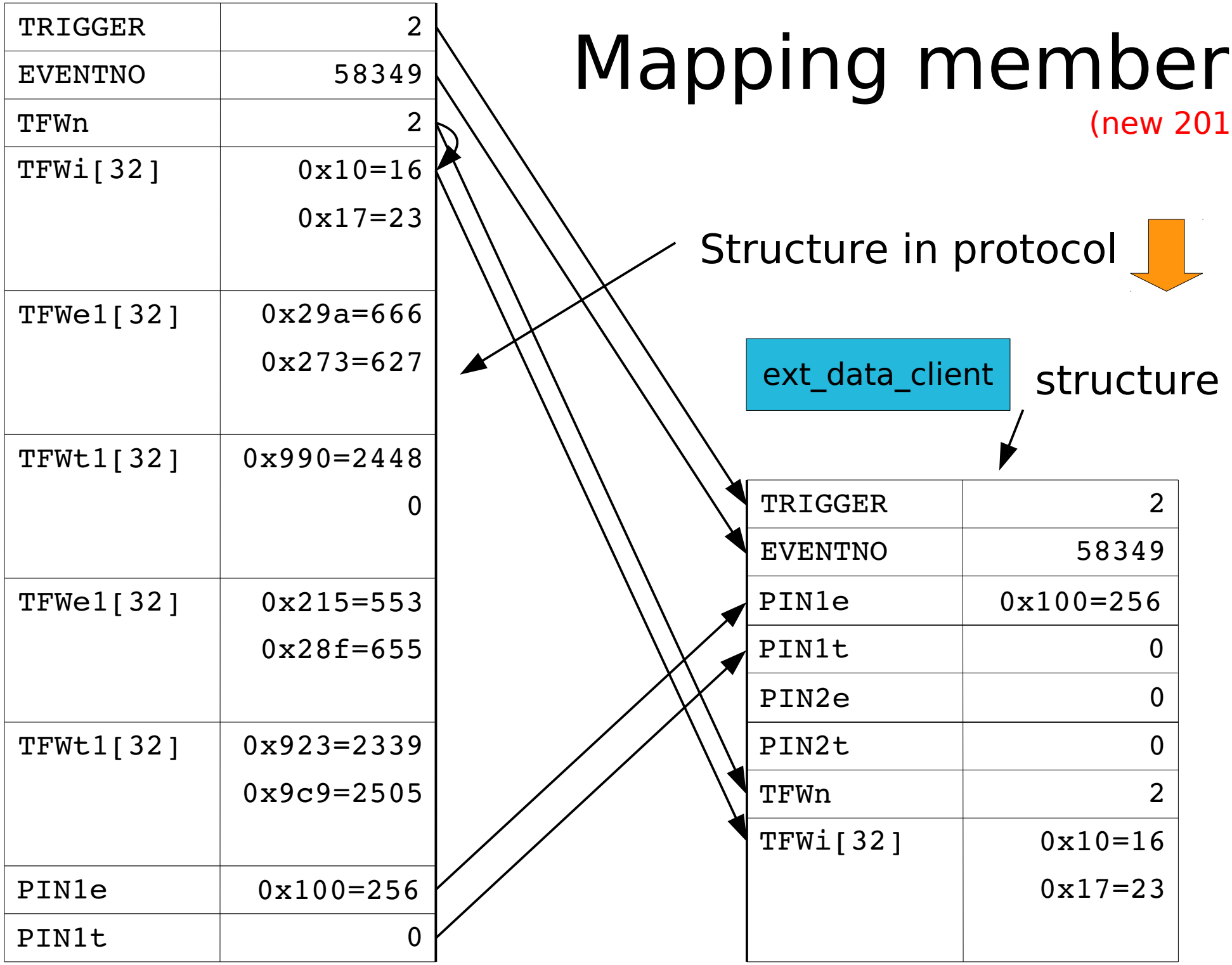


Every item is 32 bits:

- uint32_t
- float

Mapping members

(new 2015)



Finale!

http://fy.chalmers.se/~f96hajo/ucesb/

More with less is FUN!

Thank you!