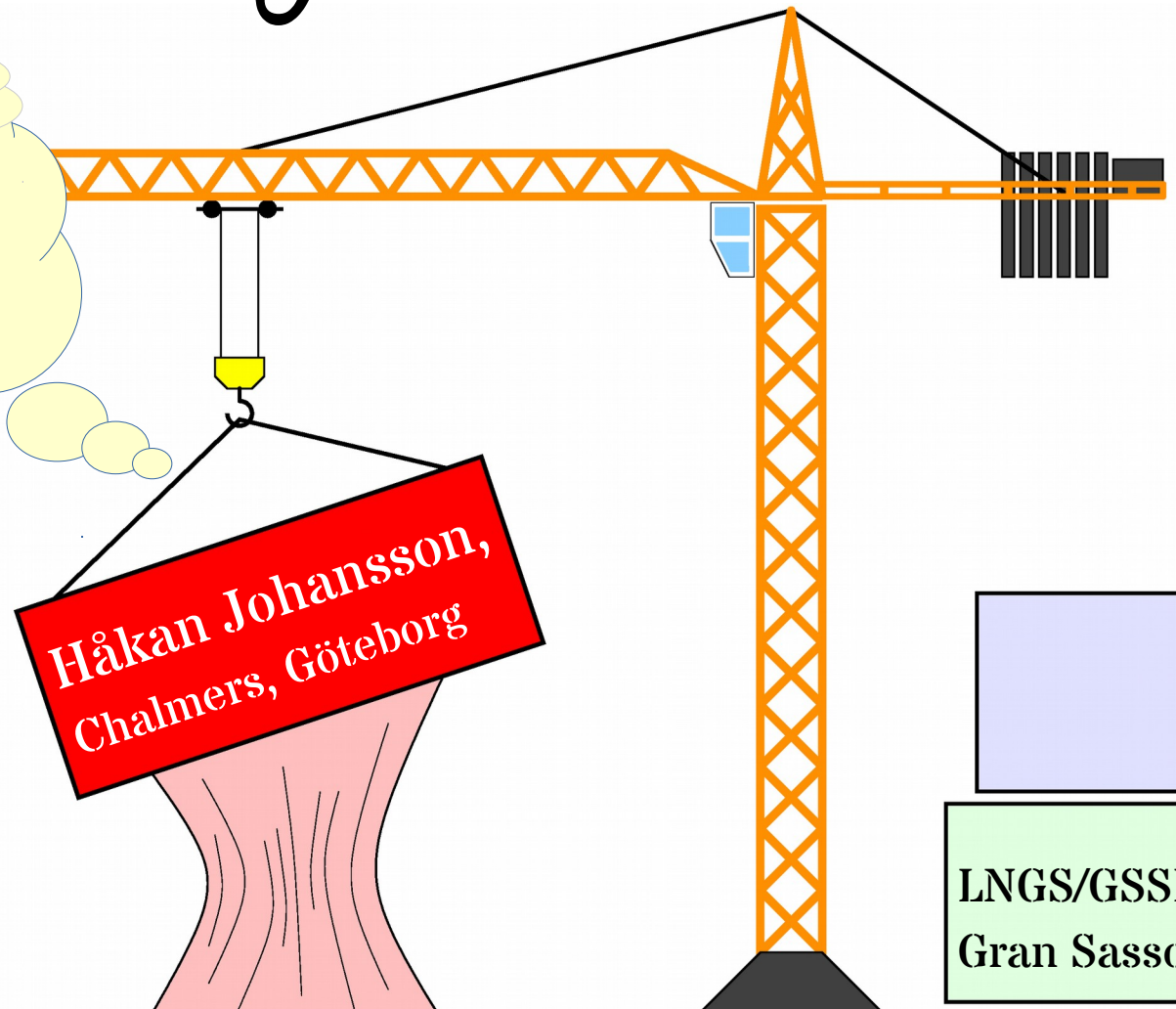
Mechanics of sticky events

with Giovanni Bruni:

Trace

compression

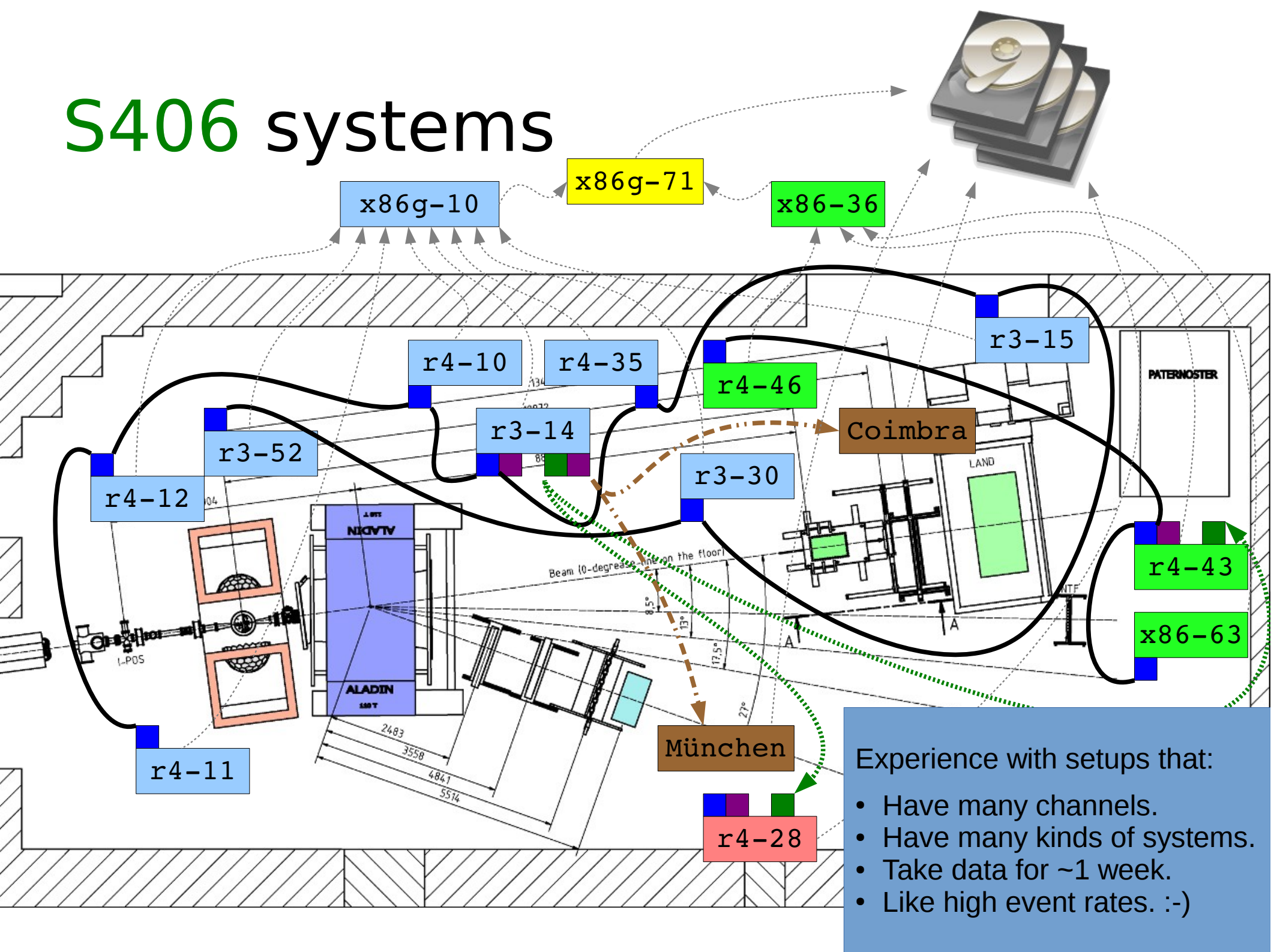
→ page 14



Håkan Johansson,
Chalmers, Göteborg

LNGS/GSSI,
Gran Sasso, May, 2017

S406 systems



Experience with setups that:

- Have many channels.
- Have many kinds of systems.
- Take data for ~1 week.
- Like high event rates. :-)

Introduction

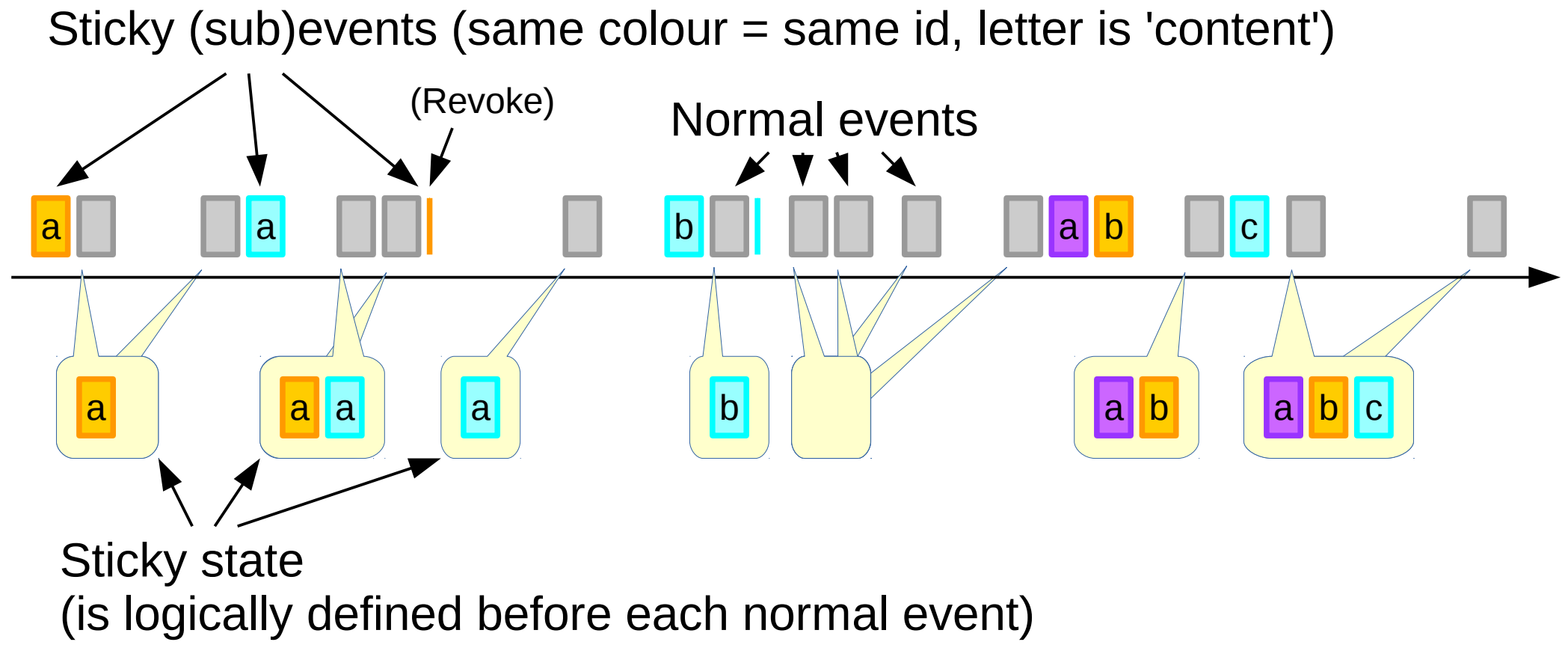
- Store slow-control information (HV settings, magnet currents...) in the data stream
  - Integrate with a distributed DAQ (NuSTAR).
  - Follow the DAQ topology.
- 'Normal' events not suitable:
  - Just flow through the DAQ / analysis.
  - Late connected clients / files would not get earlier set values.
- New concept: sticky events.
  - Delivered however late the connection is.

Sticky subevents

- Packaged in sticky events.
- The sticky thing is the subevents.
  - Sticky = held active until replaced.
- Sticky subevents identified (as usual) by
  - `type/subtype/ctrl/crate/procid`
- Removed as active with `length = -1`.

Sticky events: simple semantics

- Sticky subevents are valid until replaced
- … or revoked (replace by nothing)

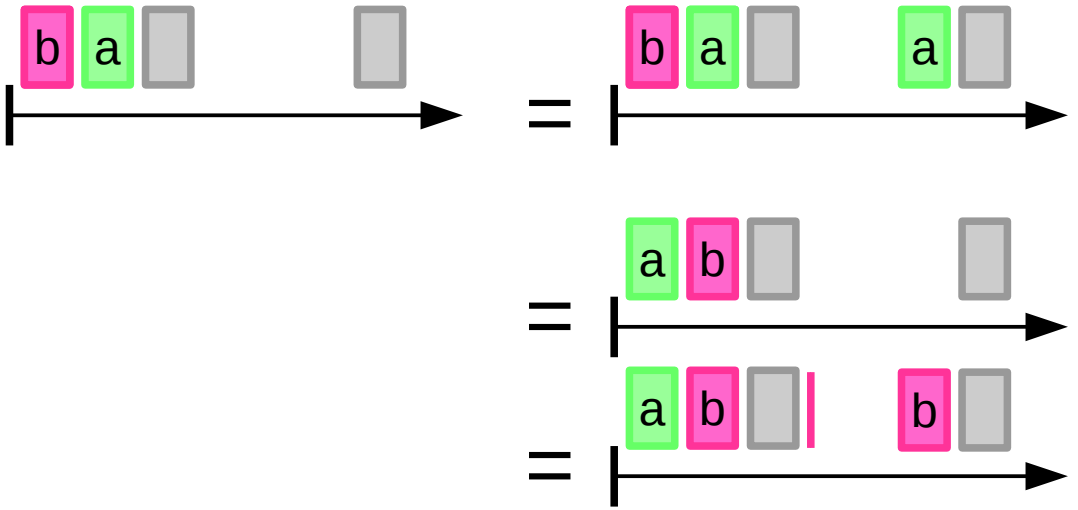


Guaranteed delivery

- An receiver (either file or network client) will **before** each normal event have received *exactly* the (at that point) active set of sticky subevents.

Sticky subevents may be delivered:

- Multiple times.
- In any order.

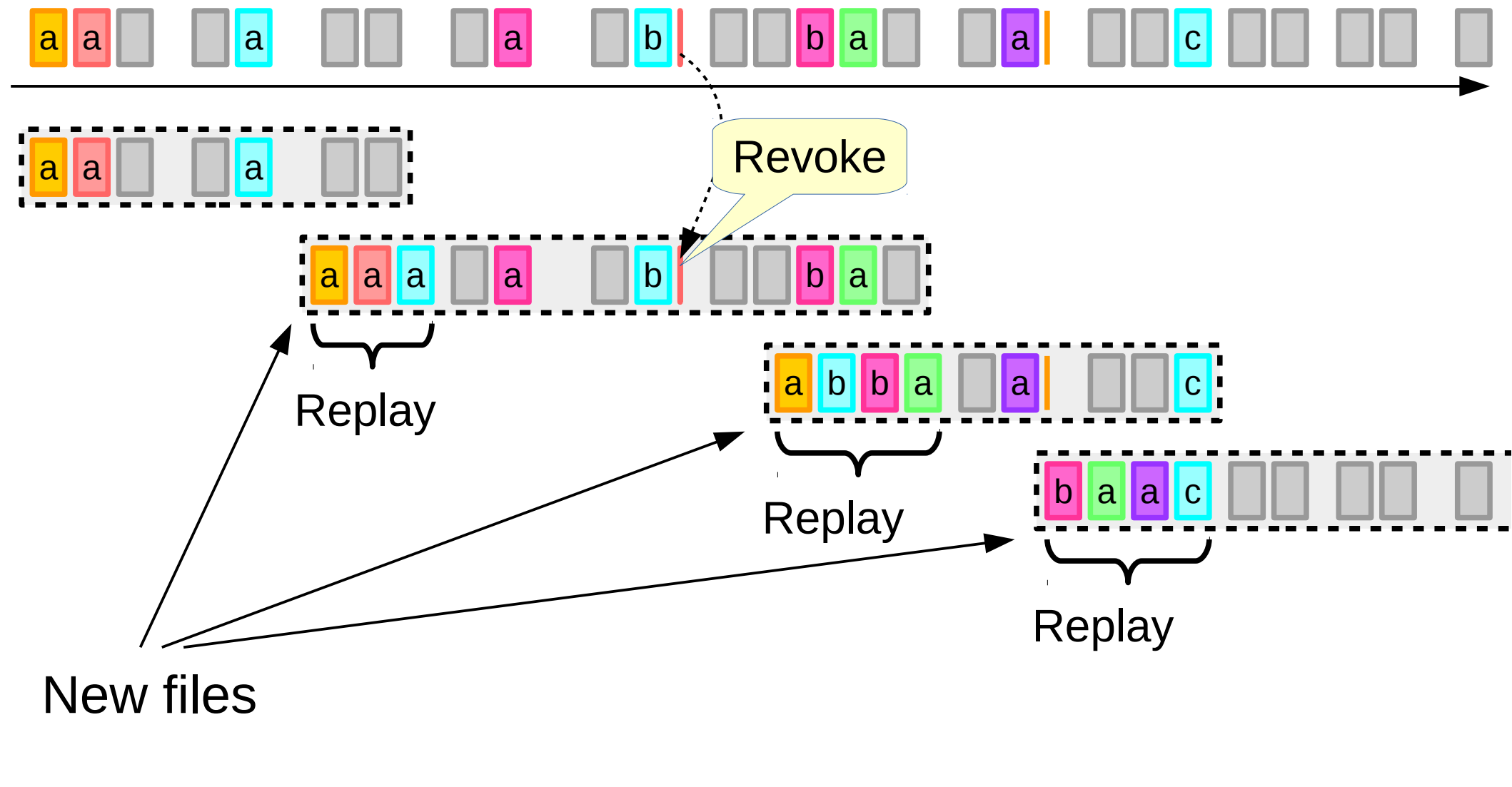


DAQ / proxy servers

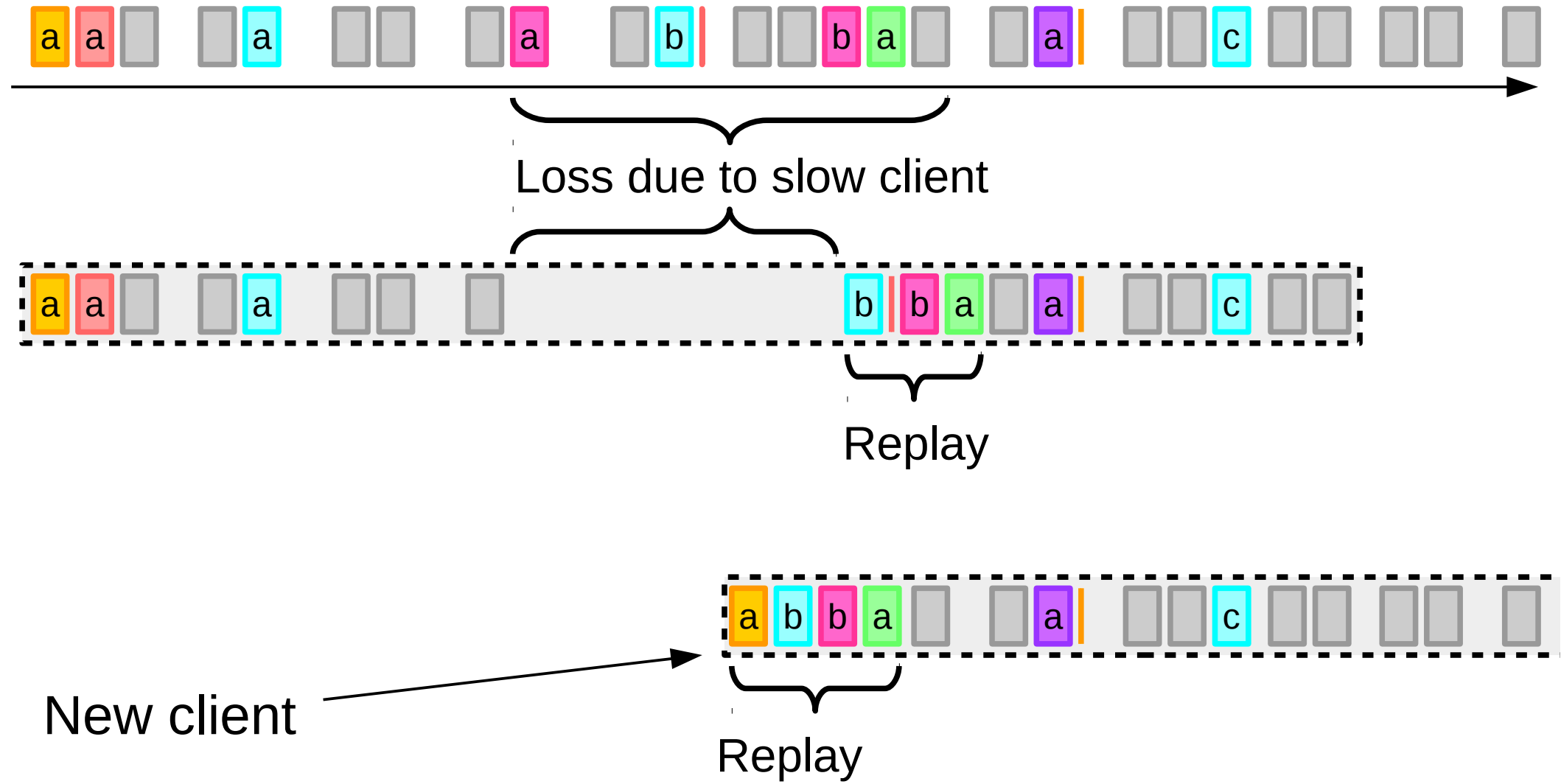
- Absorb the complications in standard programs.
- Keeps analysis clients simple.

Not so much a design choice,
rather a lucky side-effect.

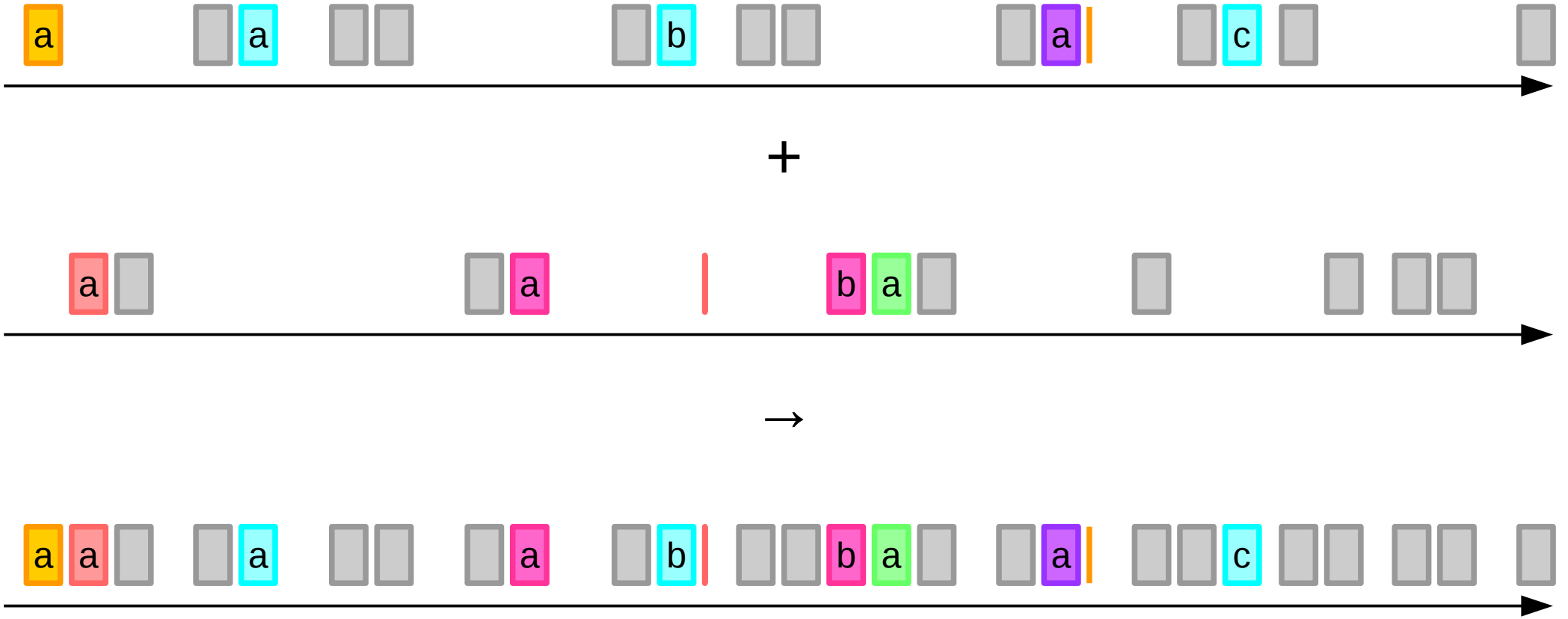
Output stages keep track



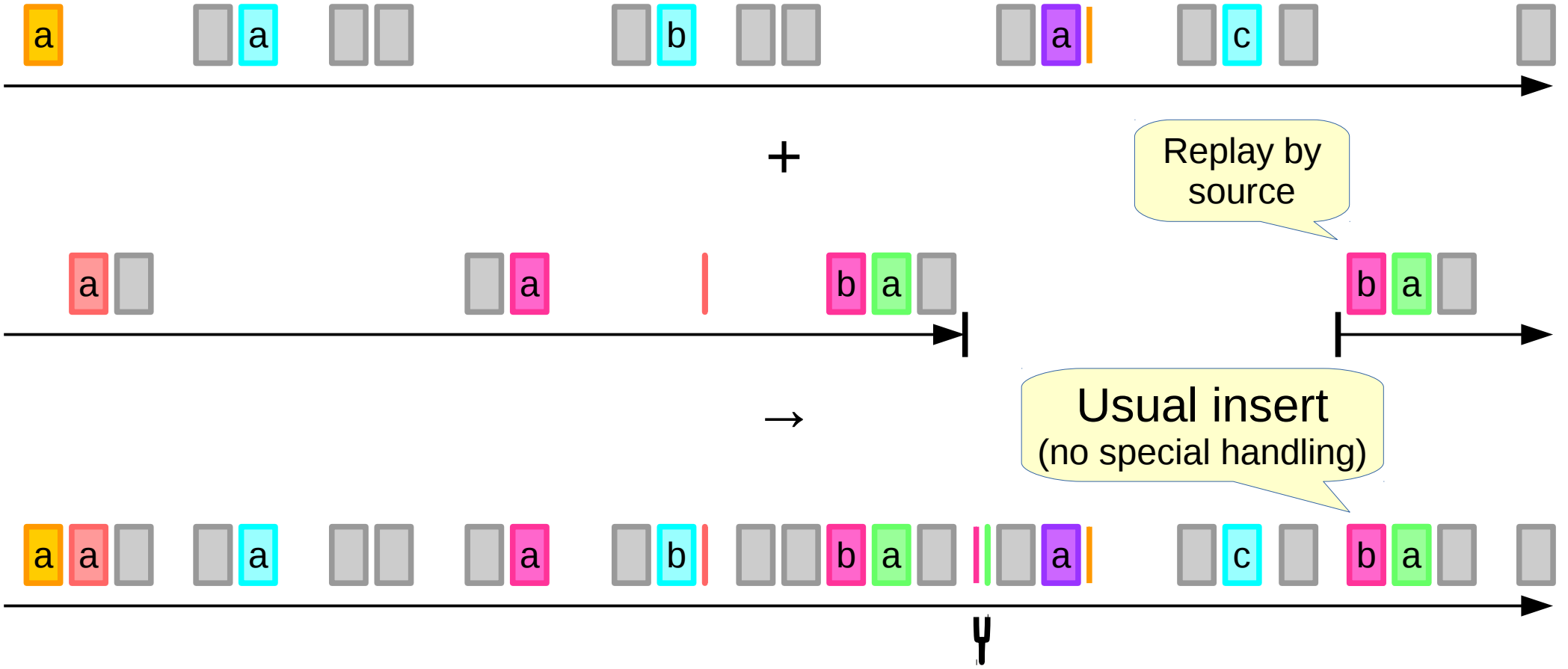
Output stages keep track (network)



Merging / time sorting

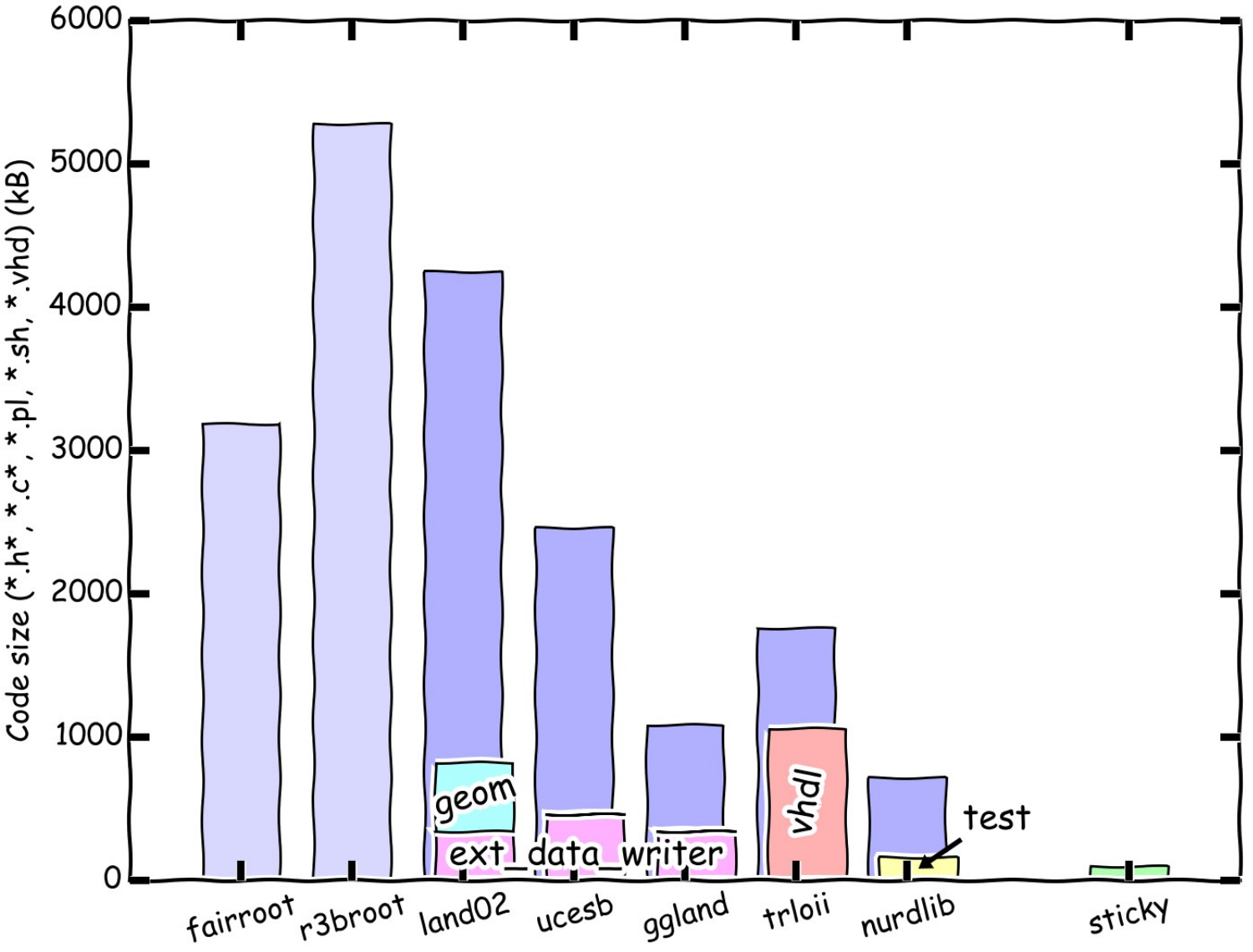


Merging – loss of source

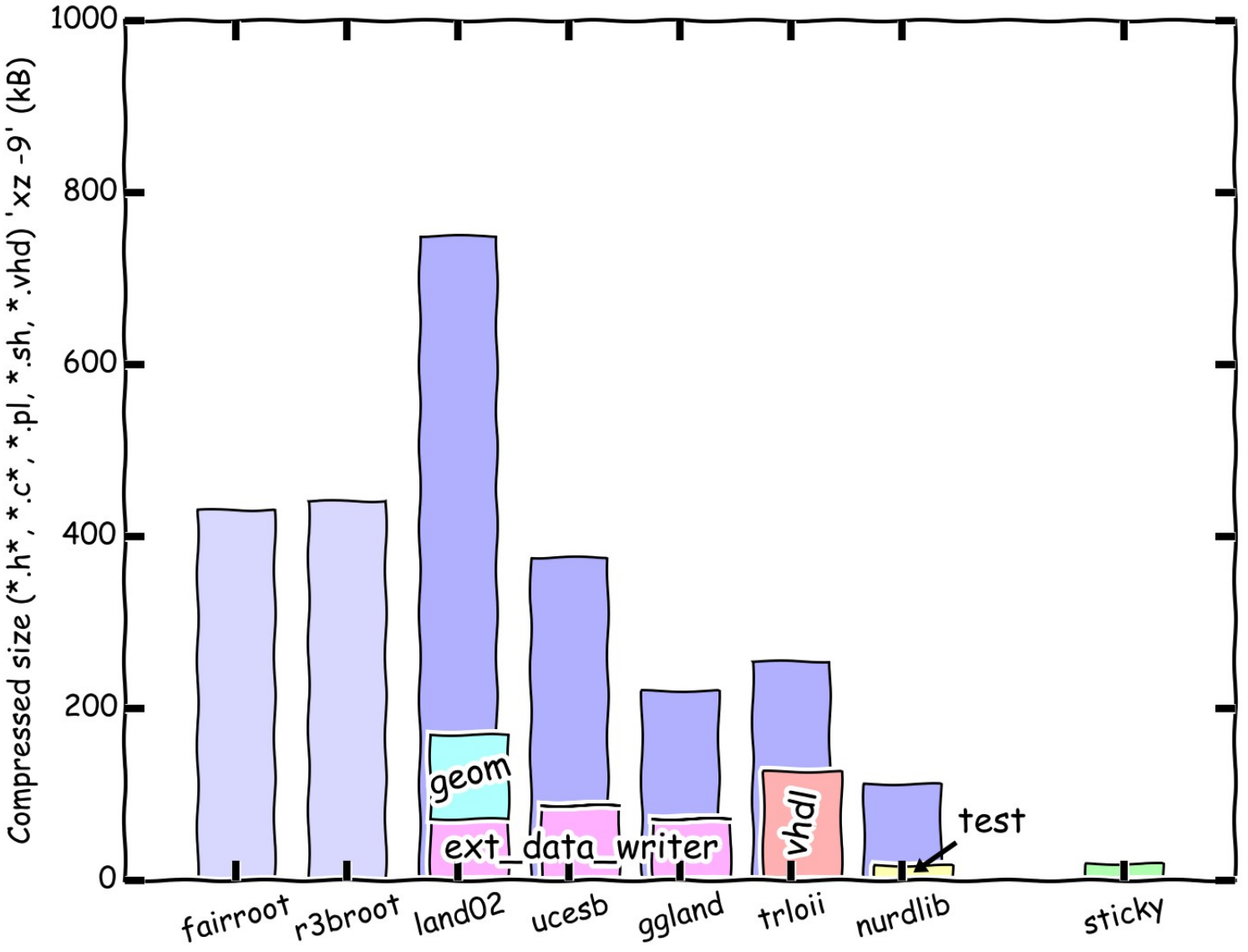


Revoke active sticky subevents that lost their source

Complicated?



Complicated? (comparing compressed)

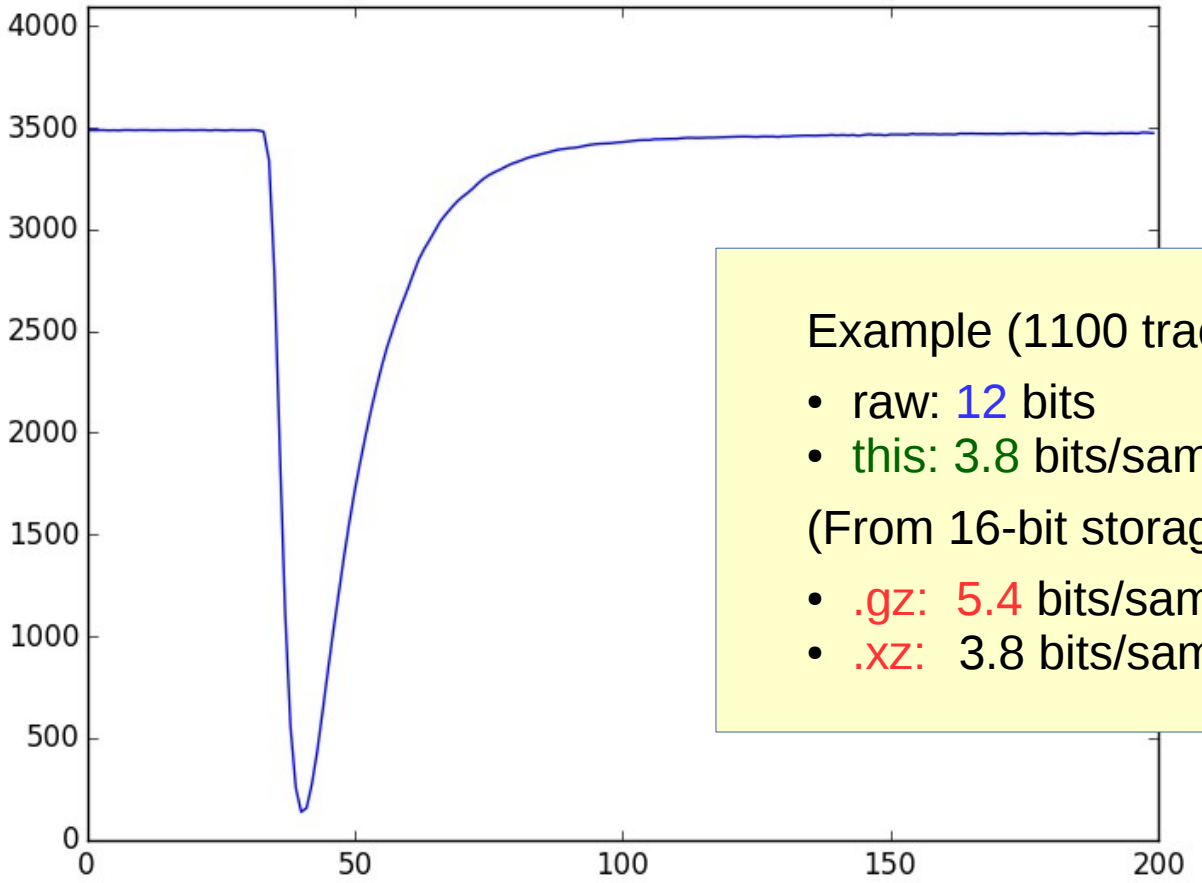


with Giovanni Bruni

Trace Compression

Work-in-progress!!!

- FPGA-friendly
- Reduce:
  - bandwidth
  - storage



Example (1100 traces):

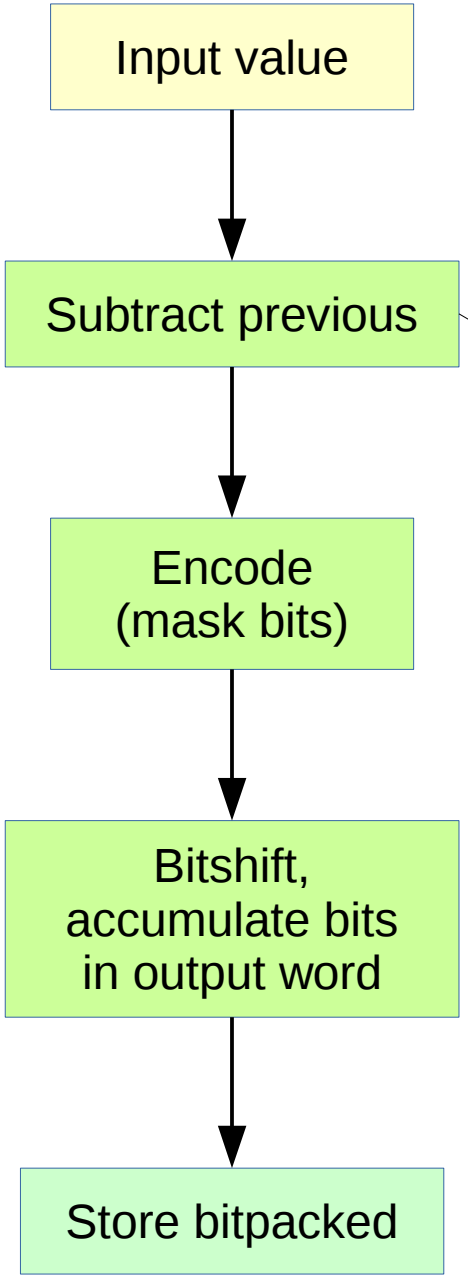
- raw: 12 bits
- this: 3.8 bits/sample

(From 16-bit storage:)

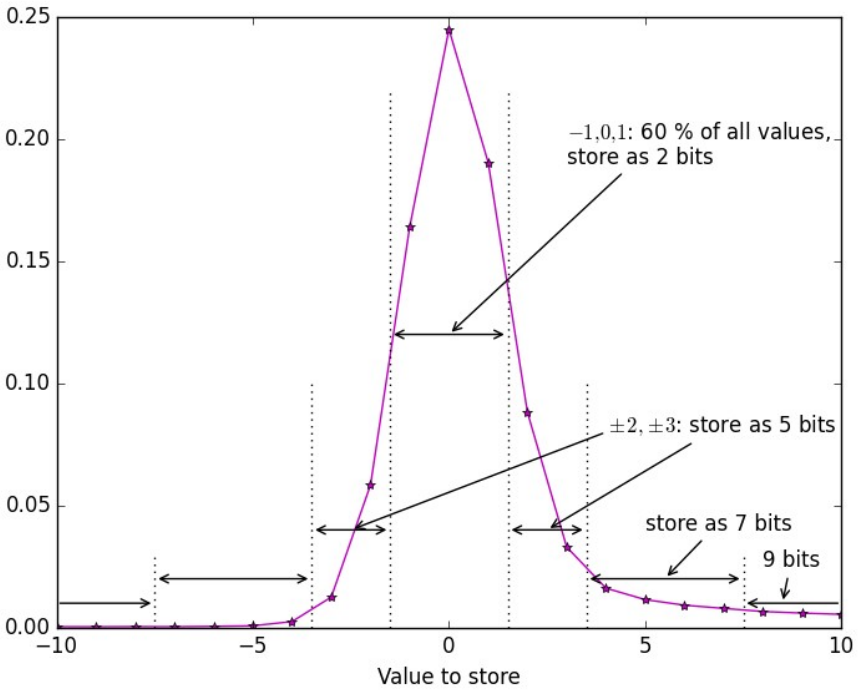
- .gz: 5.4 bits/sample
- .xz: 3.8 bits/sample

Strategy

For each sample:



Centre values to store around 0



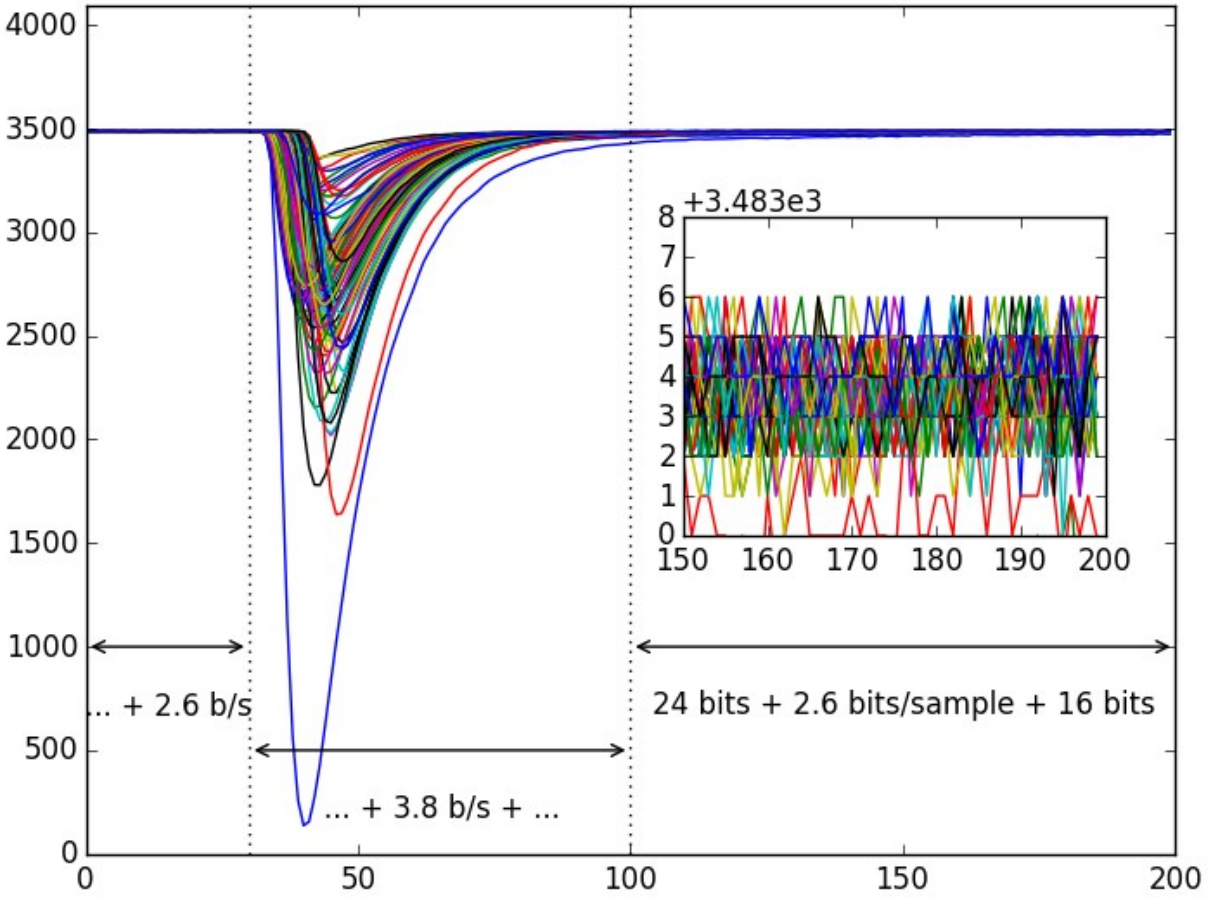
Length proportional to # significant bits
(~ 2x, due to length-encoding prefix)
(inspired by Huffman code)

FPGA code

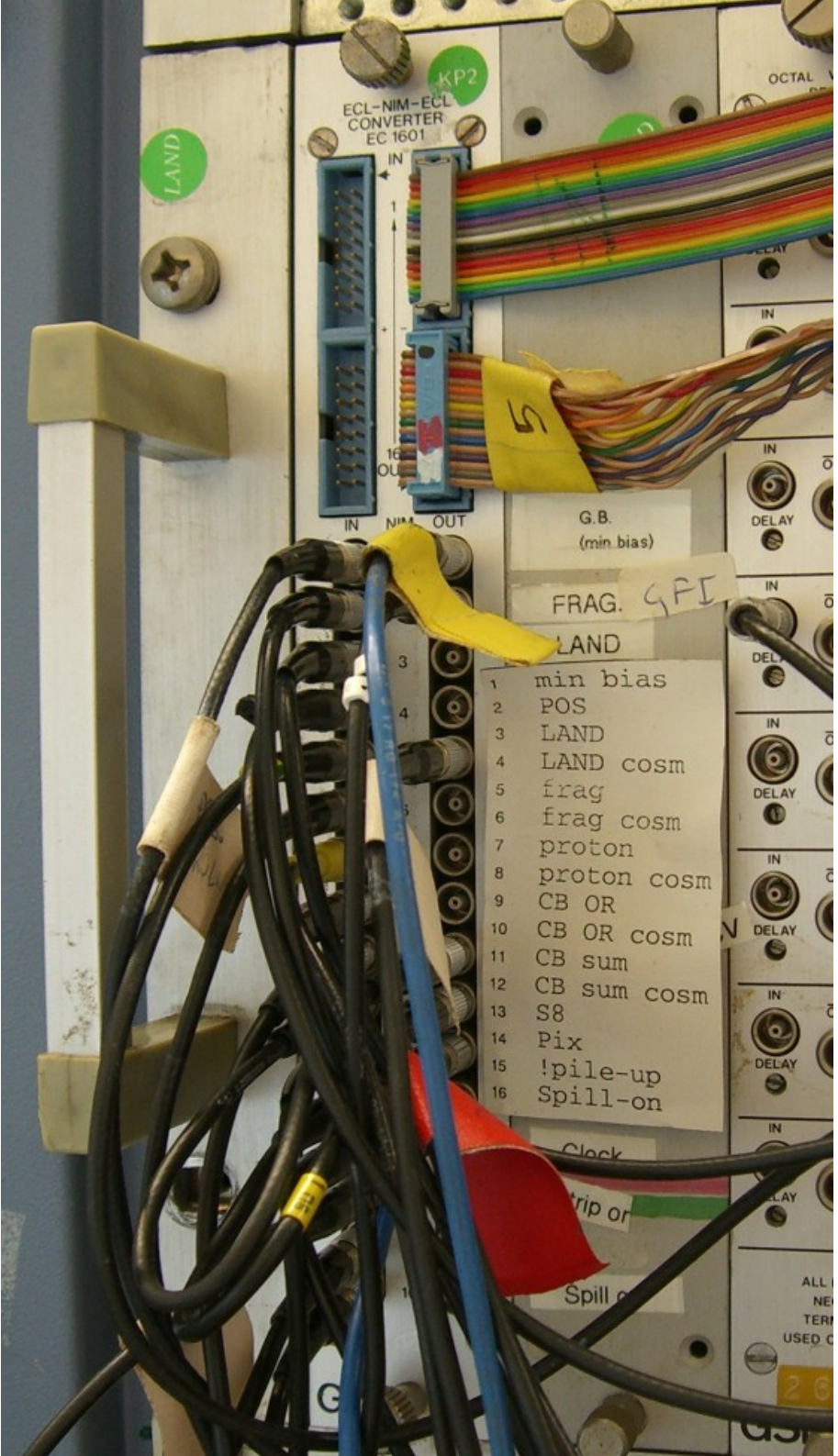
```
entity logcode_compress is
  generic(bits    : integer);
  port (clk       : in  std_logic;

        reset     : in  std_logic;

        input     : in  std_logic_vector(bits-1 downto 0);
        in_word   : in  std_logic;
        get_last  : in  std_logic;

        output    : out std_logic_vector(31 downto 0);
        out_word  : out std_logic
        );
end logcode_compress;
```

- VHDL
- 1 sample/clock cycle (@100 MHz easily)
- Tested in testbench, data compared to C implementation.
- 550 LUTs on virtex4 (compiled, not tested)
  - Dominated by large barrel shifter.
  - (Could be smaller (~1/3), but with ~10-30 cycles/sample.)

Some results



- Drawback: current version requires noise-level adaption
- Working on new encoding scheme:
  without drawback - slightly less efficient, even simpler (de)coding



Fine!

Thank you!

Lots of FUN !

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

Live by the compiler timing messages!