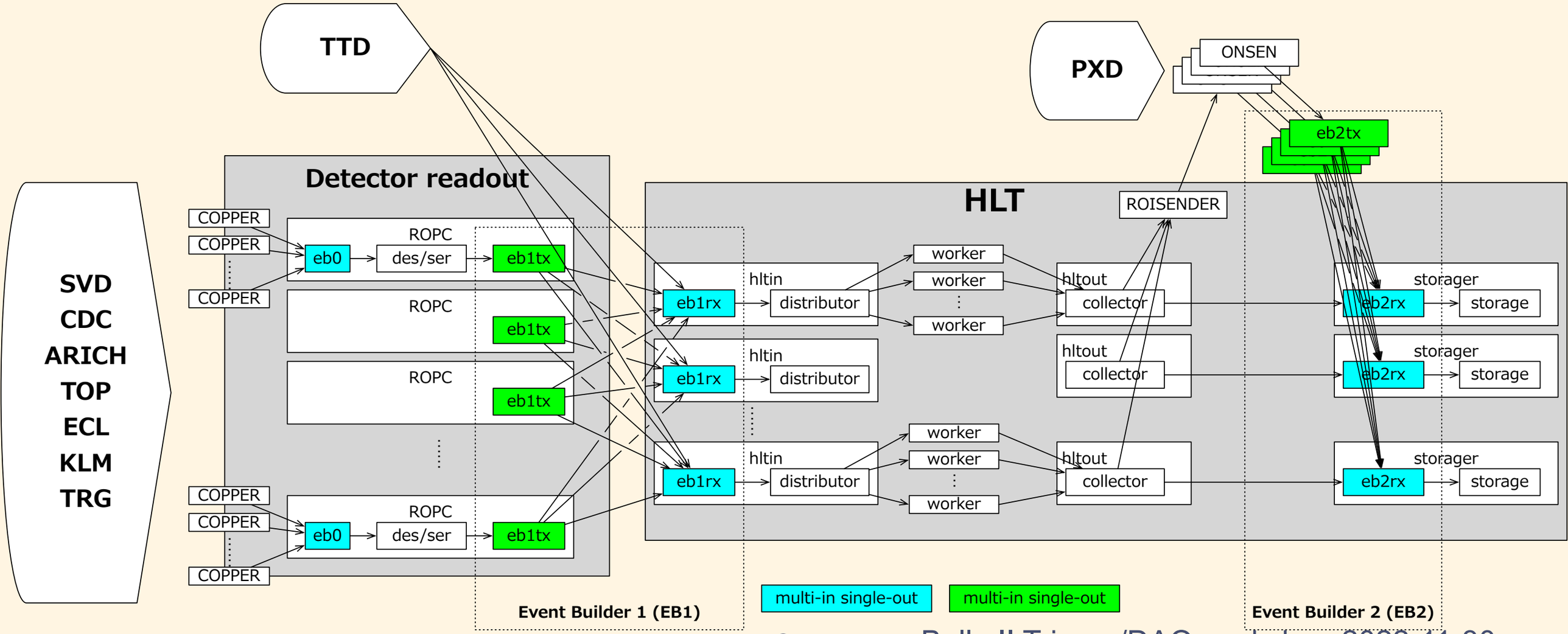
event builder

yamagata

Event builder

- consists of widely spread software agent and network switches



Components

- **Multi-in single-out**
  - eb0, eb1rx, eb2rx
  - upstream depends on which detectors, coppers are included.
  - receive data from up, concatenate them, then send to down
  - check mismatch by event number in header
- **Single-in multi-out**
  - eb1tx, eb2tx
  - downstream depends on which HLTs are included
  - receive data from up, choose destination by event number and send the destination

Moving to PCIe40

- eb0 is not needed any more.
- eb1tx must be rewritten to read 5 TCPs cyclically.
- rewritten again for throughput faster than 10Gbps
  - removing thread interlock by inproc ZMQ
- rewritten again for new PCIe40 reader by Dmytro-san, which sends data via ZMQ

Partial SALS

- To reduce the DAQ dead time caused by SALS
- keep components READY except ABORTed component
- event builder components dies when the peer process dies.
- The process connected to event builder also die.
- single des_ser death → single eb1tx death → all eb1rx death → all HLT death & all eb1tx death → all des_ser death

Requirement on event builder

- Even when any peer process is aborted, other sockets must be kept connected.
- When the aborted process becomes ready, data flow should be resumed without any action.
- call it as "connection keeping" in this slide

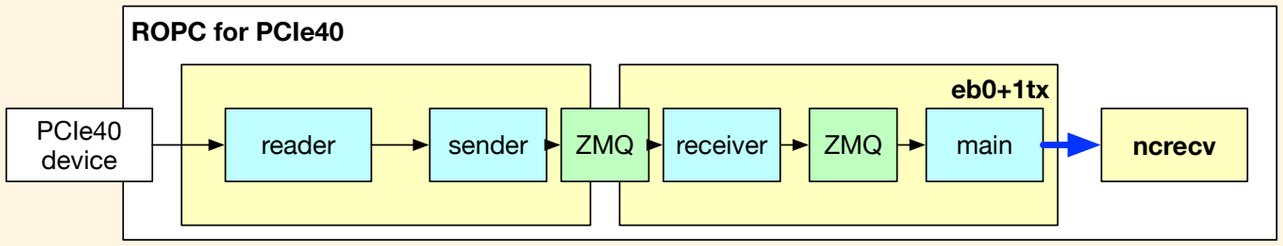
event builder after LS1

- connection keeping to upstream
  - eb1tx for new PCIe40 reader is OK (ZMQ)
  - eb1rx, eb2tx, eb2rx will be (ordinary TCP)
- limited connection keeping to downstream
  - only eb1rx - HLT (ordinary TCP)
  - handling eb1tx downstream is difficult

ZMQ vs ordinary TCP

- **ZMQ is performant and auto re-connectable**
- unable to manage individual socket accepted by a single port
  - need to re-design eb1tx-eb1rx connection to replace ordinary TCP by ZMQ
  - perhaps eb1tx should have dedicated port per HLT
  - unable to remove one of them even if it is connected to remote zombie process
- unable to find the reason of EOF, connection failure

With new PCIe40 reader



- ~5.4kHz, 1750MB/s with CRC calculation

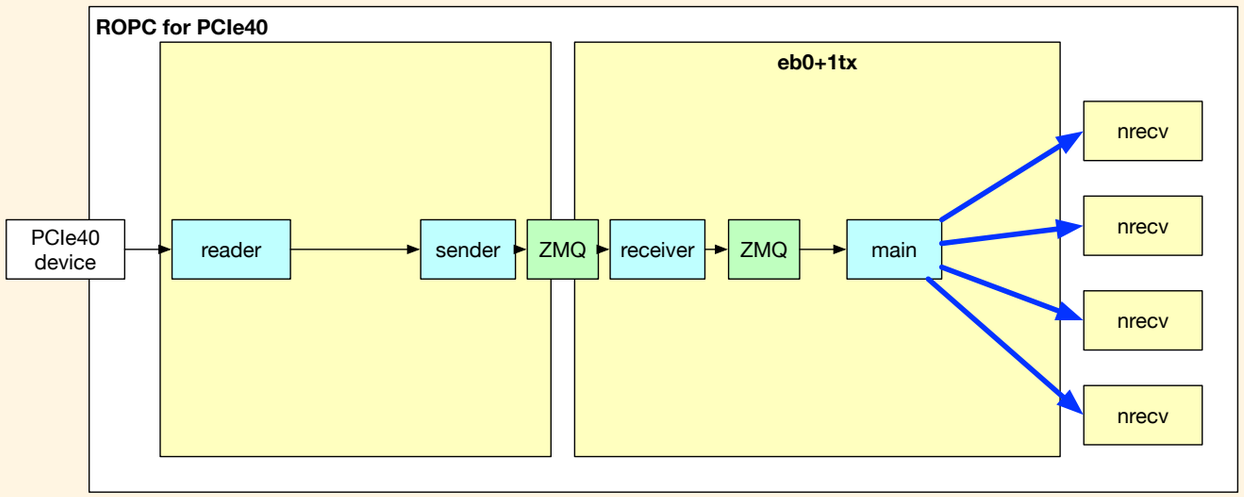
```
[yamagata@ttd7 ~]$ trigft -23 pulse 11500
trigft version 2015073000
pulse trigger rate 11522.145 Hz
exp 0 run 183 sub 0 started
```

```
700000 5417.240172 Hz 1741.837736 MB/s
800000 5416.988097 Hz 1741.756685 MB/s
900000 5451.859967 Hz 1752.969246 MB/s
1000000 5434.029523 Hz 1747.236117 MB/s
2000000 5455.286421 Hz 1754.070975 MB/s
3000000 5437.318497 Hz 1748.293640 MB/s
4000000 5435.606428 Hz 1747.743148 MB/s
5000000 5465.569747 Hz 1757.377434 MB/s
6000000 5441.214361 Hz 1749.546301 MB/s
7000000 5460.727027 Hz 1755.820325 MB/s
8000000 5463.915434 Hz 1756.845513 MB/s
9000000 5520.673334 Hz 1775.095221 MB/s
```

```
top - 13:32:06 up 19 days, 8 min, 16 users,  load average: 16.97, 18.29, 17.36
Tasks: 525 total,   2 running, 519 sleeping,   4 stopped,   0 zombie
%Cpu(s): 44.6 us, 16.5 sy,  0.0 ni, 37.4 id,  0.0 wa,  0.7 hi,  0.7 si,  0.0 st
MiB Mem :  31843.7 total,  19786.3 free,   4805.4 used,   7252.0 buff/cache
MiB Swap:  15257.0 total,  14308.9 free,    948.1 used.  26555.0 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
1413551 yamagata  20   0 5012408   2.2g   6596 S 956.6   7.0 421:14.78 sweb_receiver
1413494 yamagata  20   0 1653148 690616   6256 S 175.8   2.1  76:00.39 eb0+1tx_for_pci
1413500 yamagata  20   0   62944  44040   2672 R  82.1   0.1  35:23.23 ncrecv
1422558 yamagata  20   0   65956   5576   4344 R   0.3   0.0   0:00.11 top
 876566 yamagata  20   0   89672   9784   8180 S   0.0   0.0   0:00.19 systemd
```

Assigning 2CTX and 4 ncrecv




```
100000 1808.102142 Hz 581.369930 MB/s (ncrecv1)
100000 1795.866705 Hz 577.435797 MB/s (ncrecv2)
100000 1793.922442 Hz 576.810646 MB/s (ncrecv3)
100000 1797.482985 Hz 577.955489 MB/s (ncrecv4)
```

- Totally 7.193kHz, 2311MB/s

```
top - 16:02:51 up 19 days,  2:39, 16 users,  load average: 12.48, 8.88, 10.18
Tasks: 527 total,   3 running, 520 sleeping,   4 stopped,   0 zombie
%Cpu(s): 61.5 us, 19.3 sy,  2.1 ni, 15.5 id,  0.0 wa,  0.4 hi,  1.1 si,  0.0 st
MiB Mem :  31843.7 total,  20671.3 free,   3868.7 used,   7303.8 buff/cache
MiB Swap:  15257.0 total,  14311.5 free,    945.5 used.  27491.6 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
1452933 yamagata  20   0 4684728   1.8g   6436 S  1288   5.8   5:56.04 sweb_receiver
1452874 yamagata  20   0 1255596 118992   6256 R 249.7   0.4   1:04.89 eb0+1tx_for_pci
1452900 yamagata  25   5   62944  14368   2644 S  32.5   0.0   0:07.64 ncrecv
1452891 yamagata  25   5   62944  14632   2644 S  30.8   0.0   0:07.79 ncrecv
1452896 yamagata  25   5   62944  18316   2772 R  30.8   0.1   0:07.77 ncrecv
1452904 yamagata  25   5   62944  19400   2592 S  28.8   0.1   0:07.46 ncrecv
1448785 yamagata  20   0   65956   5512   4284 R   1.0   0.0   0:04.92 top
1385665 yamagata  20   0  163788   5568   4252 S   0.3   0.0   0:00.38 sshd
```

2 ZMQ Context is enough, 4 is too much for 20Gbps

- It is consistent that 10Gbps data flow fully exhaust single core already we know
  - already SVD is reaching 10Gbps per ROPC
- We will have 25Gbps ether on ROPC, 4 is too much for our usage.
- Important: Multiple HLT units must be used to 3Gbps as HLT design is 10 units for 3GB/s.

ZMQ is certainly performant, but EB will use ordinary TCP also

- Until the redesign of connection between eb1tx-eb1rx
- To enable partial replacement of EB components
- To identify the reason of a connection break

Strategy of connection keeping

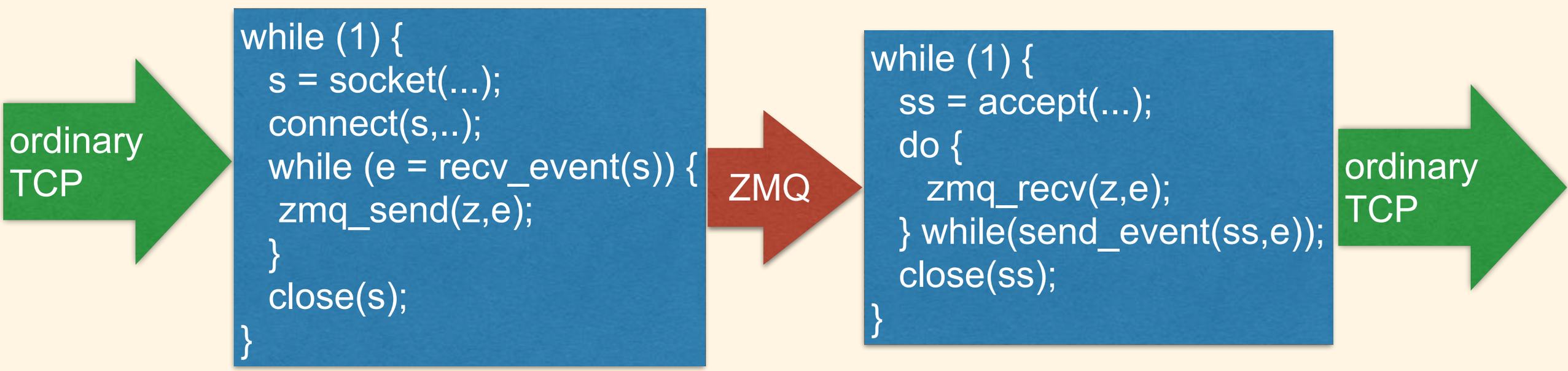
- avoid exit on any error on socket
- When an upstream has been unexpectedly closed
  - stop dataflow until full-reconnection
  - discard partial event
  - reconnect
- When a downstream has been unexpectedly closed
  - stop dataflow until full-reconnection
  - discard partial event

Connection keeping by ZMQ

- avoid exit on any error on socket
- When an upstream has been unexpectedly closed
  - stop dataflow until full-reconnection
  - discard partial event (ZMQ framing internally does)
  - reconnect
- When a downstream has been unexpectedly closed
  - stop dataflow until full-reconnection
  - discard partial event (ZMQ framing internally does)

Tentative agent to keep connection

- only applicable to single TCP



handle active connection for upstream

handle passive stream for downstream

May drop event on a reconnection

For partial SALS

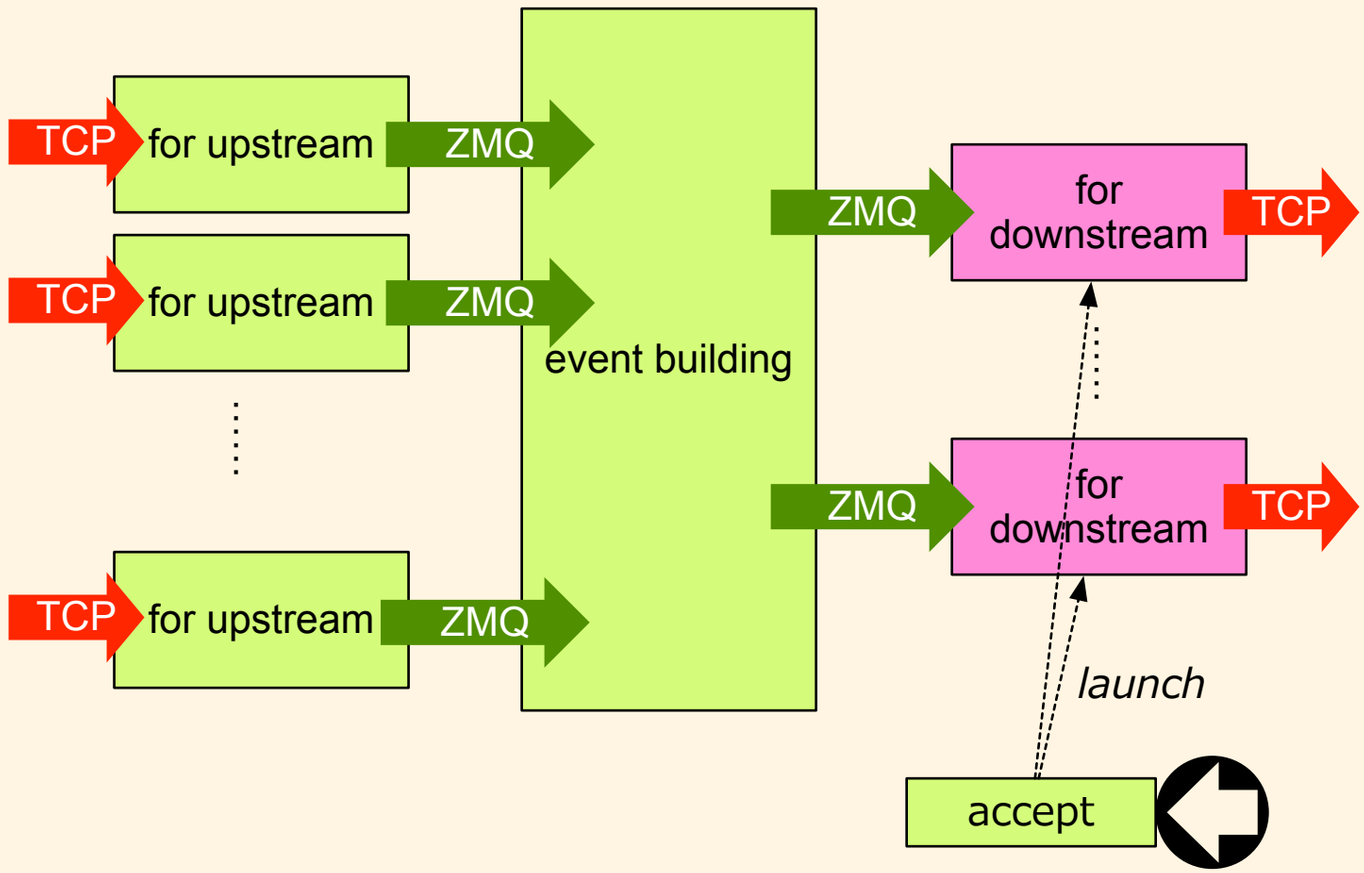
- Which subsystems are targeted ?
  - detector / HLT / TTD ?
  - The targeted NSM node shouldn't contain eb1txd/eb1rxd because it will be killed on ABORT ... some new subsystem is needed.
    - eb1rsvd1, eb1rsvd2, ... on all ROPCe40 ROPC and eb1hlt1, ... on all HLT units .. are acceptable?
    - EB1 can't be organized in single NSM node because they are not connected to single network.
- Currently partial-detector-SALS is important to reduce deadtime (?)
- Does Include/exclude operation requires global SALS ? I hope so

When sub-run change happens?

- SEU?
  - Is trigger stopped and reseted to zero?
- intentional partial reset of PCIe40 reader
- Intentional partial reset of any HLT unit
- inclusion/exclusion change requires SALS I think. Is it OK?
- EB uses the event number assigned by COPPER/PCIe40 to choose the destination HLT unit.
  - It must be consistent on all PCIe40 even after the partial restart.
  - Is it reset to zero on partial restart?

thread model

- event building
- thread for upstream
- thread for downstream
- thread to accept



ZMQ socket exists until ABORT

TCP socket exists on remote-ABORT

threads exists until ABORT

threads exits on remote-ABORT

event building

- makes ZMQ(PUSH/PULL) to upstream and downstream
- launches acceptor threads
- waits for all downstream becomes ready (for compatibility of current behavior)
- launches upstream threads
- doesn't care partial SALS

Thread for upstream

```
while (1) {
    if (receive_event() != fail) {
        sends_event_to_ZMQ();
    } else {
        close_socket();
        connect_upstream(assigned_remote_host);
    }
} /* forever until abort */

/* self reconnect for partial SALS */
```

Thread for downstream

```
while (1) {
    zmq_recv(&built_event);
    if (send_event(built_event) == fail))
        pthread_exit();
}
/* exit on partial SALS */
```

Thread to accept

```
sl = socket; listen(sl); bind(sl);
for (int i=0; i<Ndown; i++) rem_sock[i] = accept(sl);
sort_rem_sock_by_IP_address();
for (int i=0; i<Ndown; i++) rem_addr[i] = getpeername(rem_sock[i]);
for (int i=0; i<Ndown; i++) launch_downstream_thread(i);
/* now ready to start event building */

while (1) { /* catch reconnection after partial SALS of downstream */
        sa = accept(s); ra = getpeername(sa);
        i = find_index_by_remote_addr(ra);
        close(rem_sock[i]); rem_sock[i] = sa;
        launch_downstream_thread(i);
} /* until ABORT */
```

relatively complex

event builder after LS2

- use ZMQ eb1tx/eb1rx connection
  - About eb1rx - HLT connection, not sure
- Should be individual NSM node
  - runcontrold for each detector (e.g. rcdc1) will not manage eb1txd
  - runcontrold for HLT will not manage eb1rxd
- TODO
  - NSM interface

Summary

- To avoid cascaded death, ZMQ will be internally used in event builder.
- But not directly applicable to current EB scheme
- Need to re-design for LS2
  - each connection must have individual port number to distinguish connection.