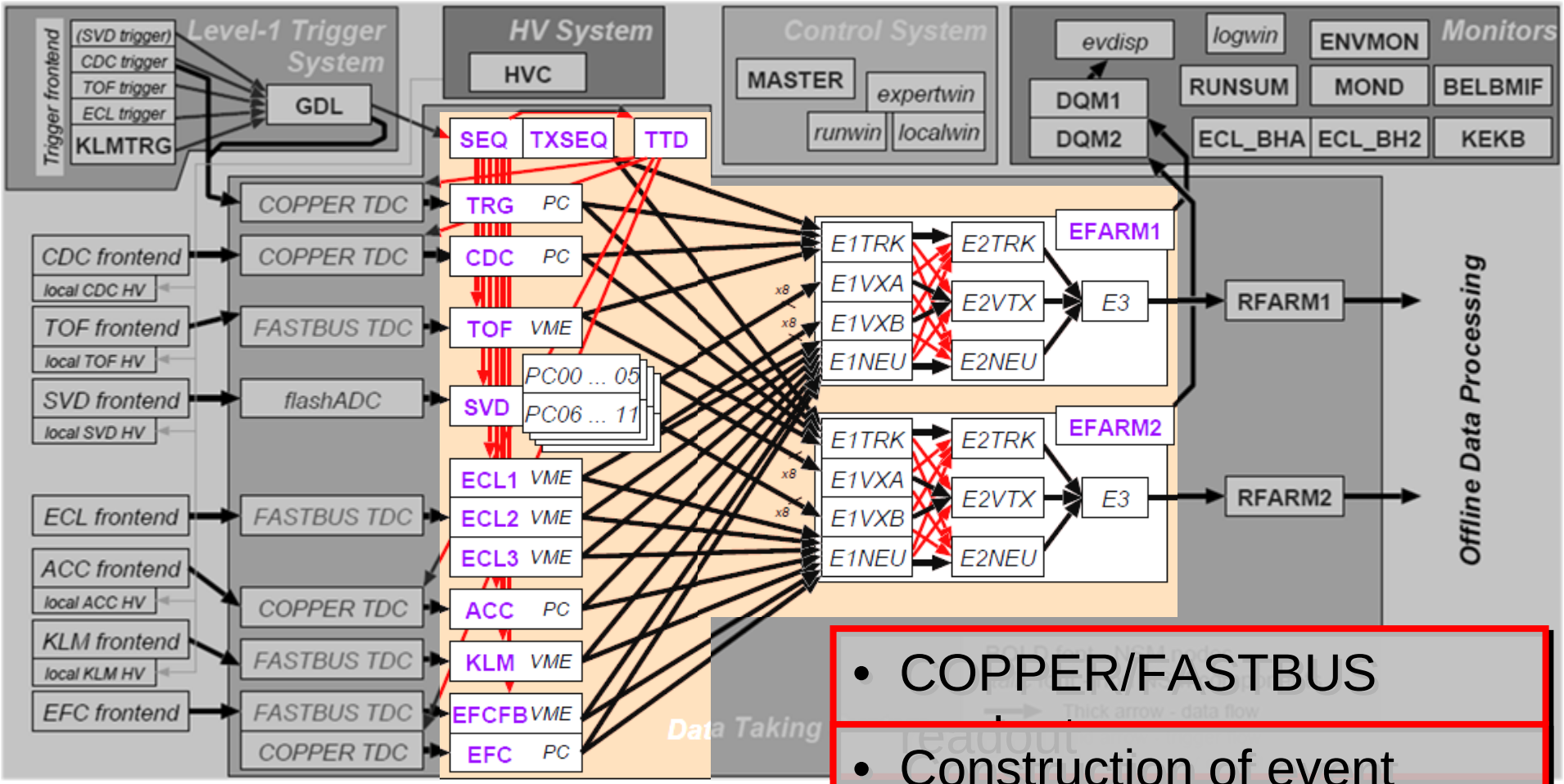
SuperBelle Event Building System

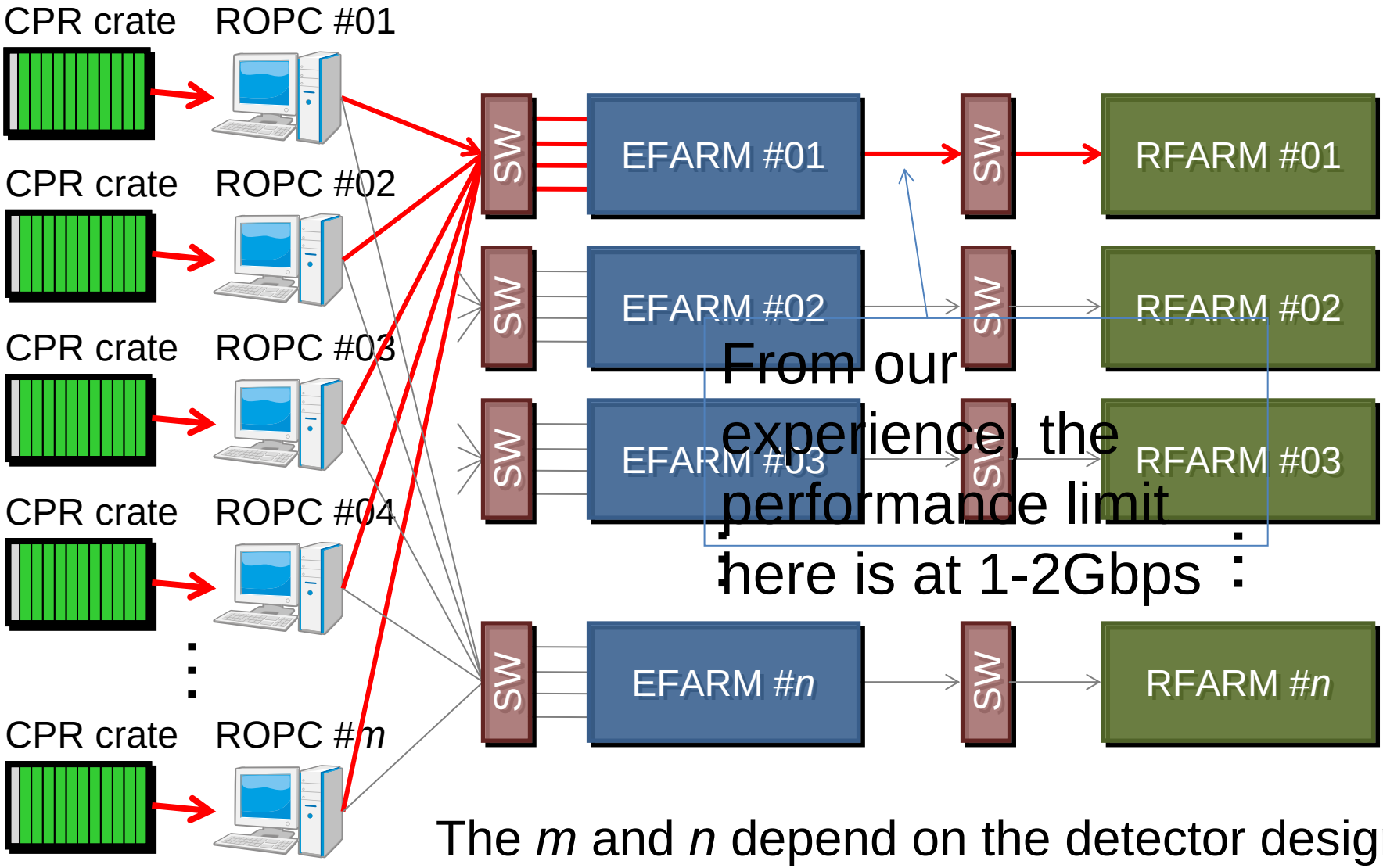
Takeo Higuchi (IPNS, KEK)
S.Y.Suzuki (CC, KEK)

Belle Event Builder



- COPPER/FASTBUS
- Construction of event record

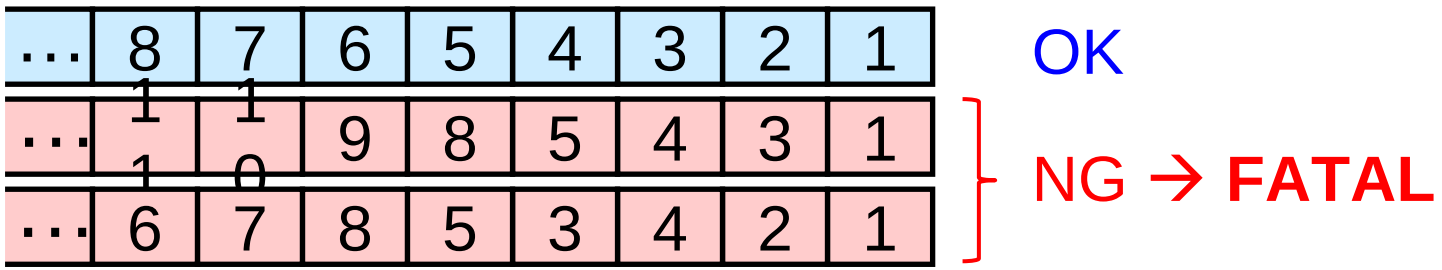
Configuration of SB Event Builder



The *m* and *n* depend on the detector design.

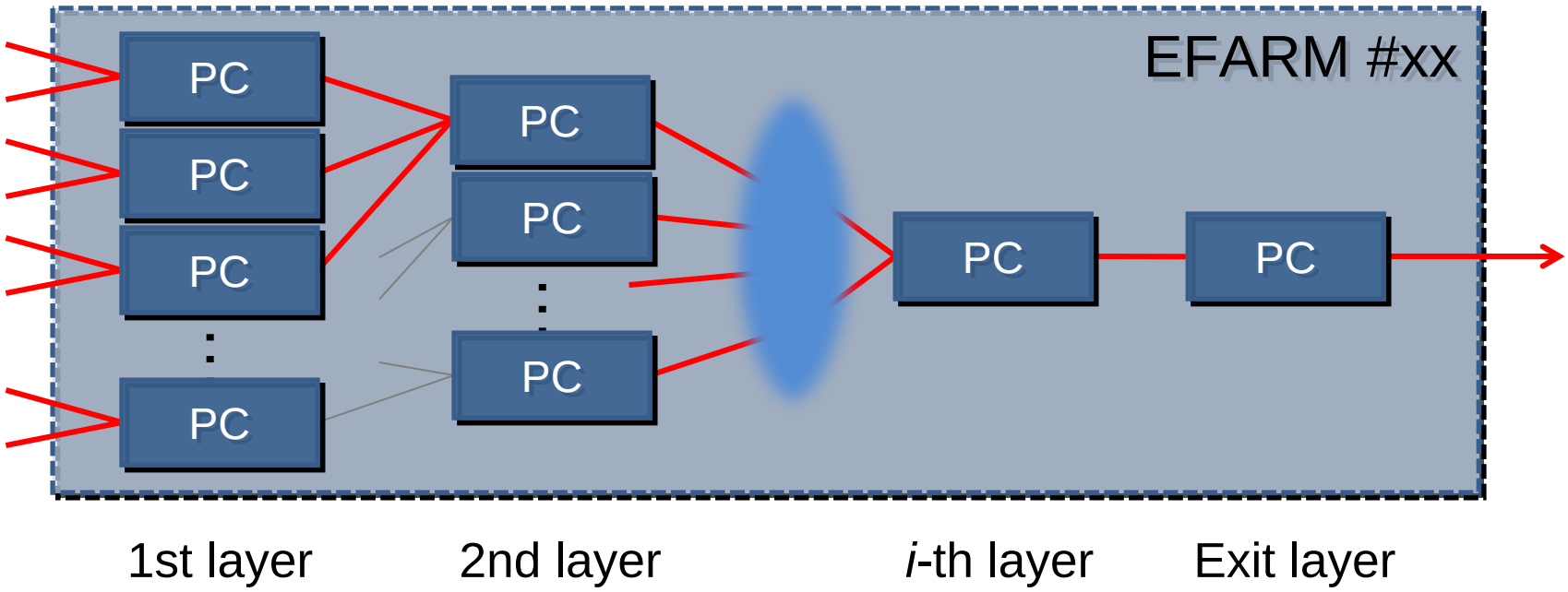
Regulation on FINESSE by EB

- No event# skip / No event# shuffle
  - Any kind of data record must be sent out from the FINESSE upon the L1 trigger.
    - At least a capsule with event# tag (and empty body) must be generated even for a totally zero-suppressed event or an L2-aborted event.
  - Event # by the FINESSE must be increased monotonically by 1.



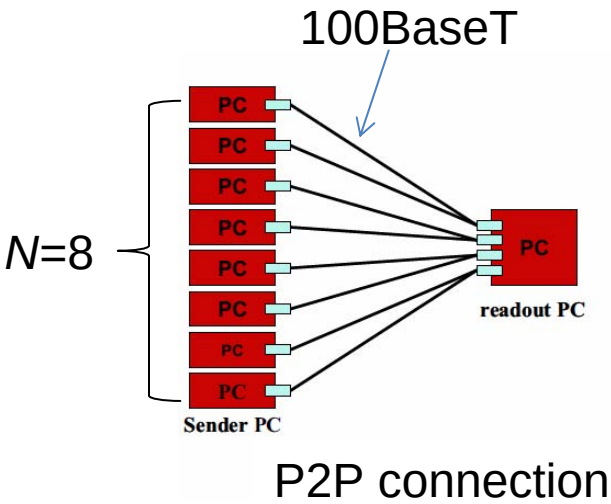
Inside EFARM #xx

- Multi-layer event building
  - Similar to the present EFARM configuration.

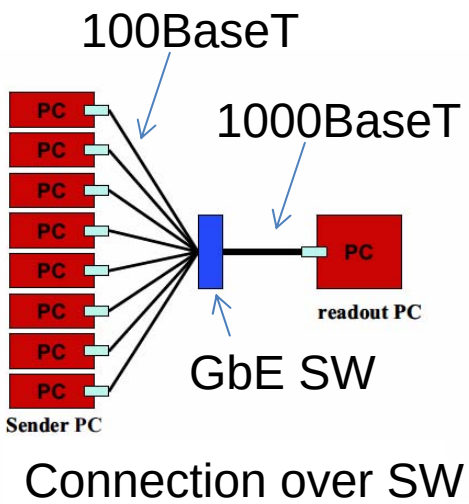


Performance Study Inside EB (1)

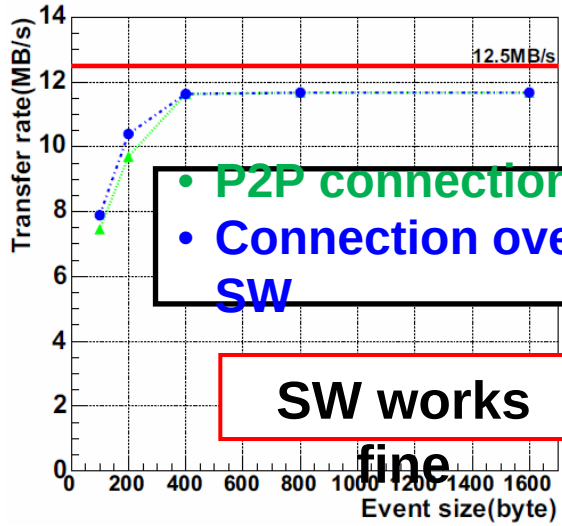
Master thesis of K.Ito (2005)



P2P connection

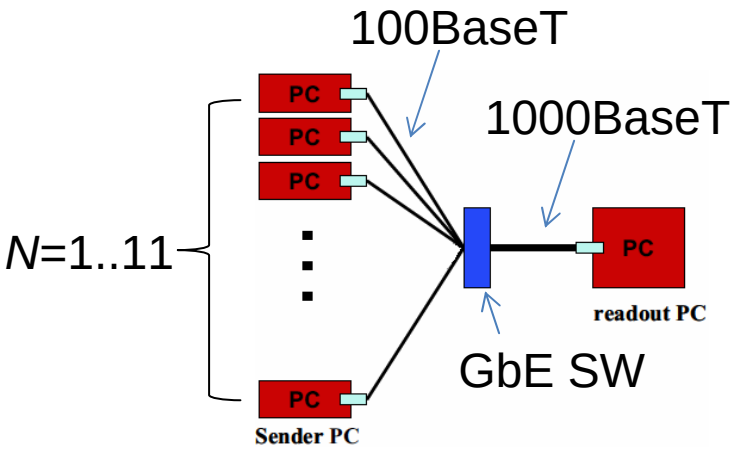


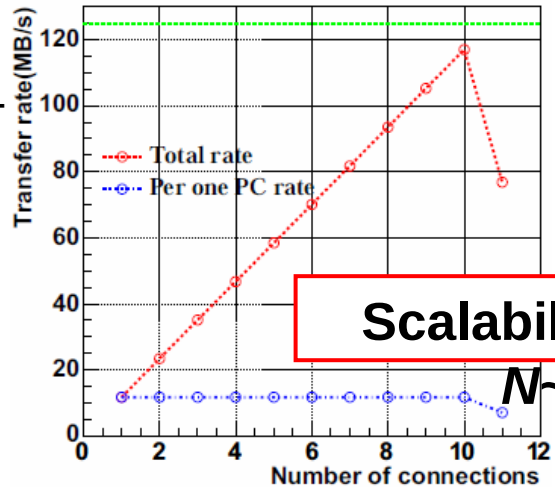
Connection over SW



- P2P connection
- Connection over SW

SW works fine



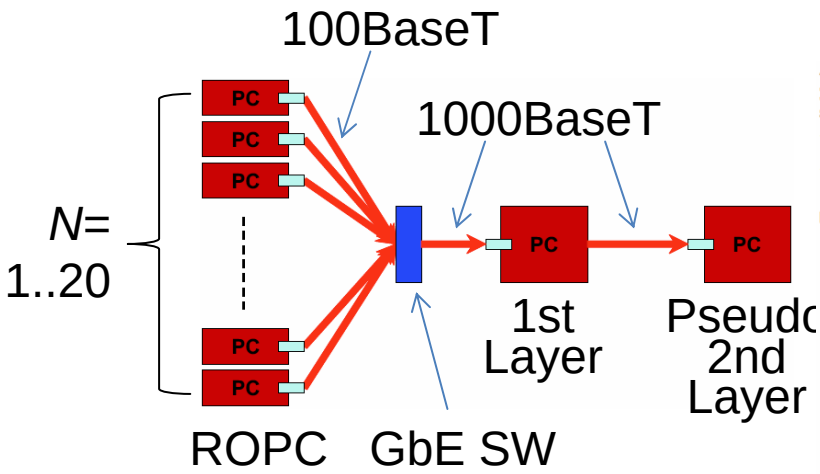


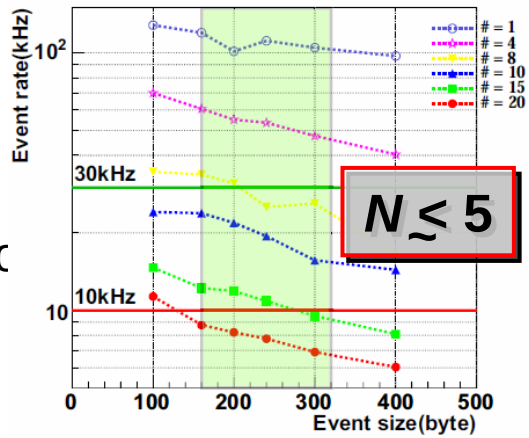
@800Bytes/ev

Scalability up to *N*~10

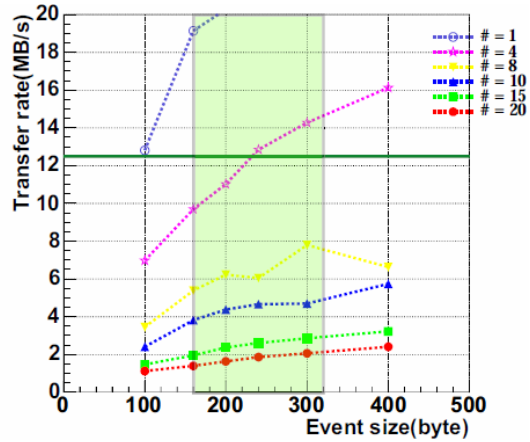
Performance Study Inside EB (2)

Master thesis of K.Ito (2005)





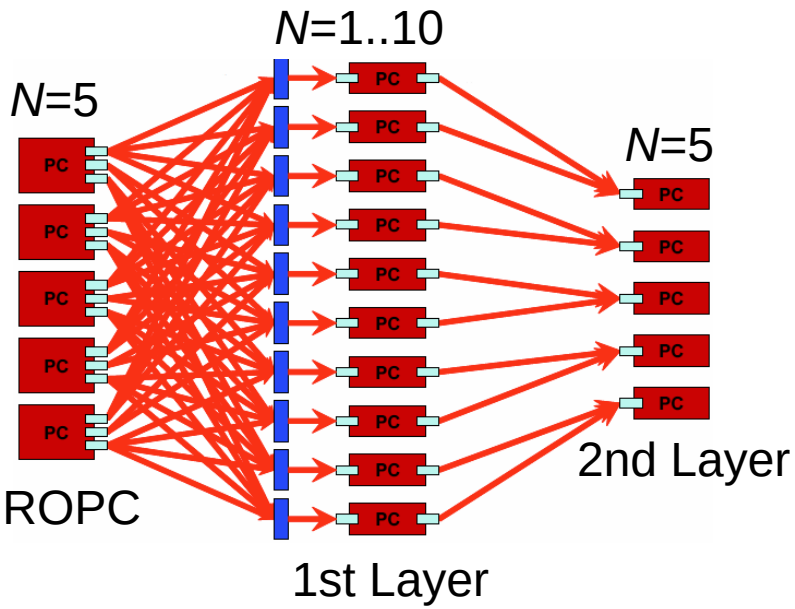
Event rate

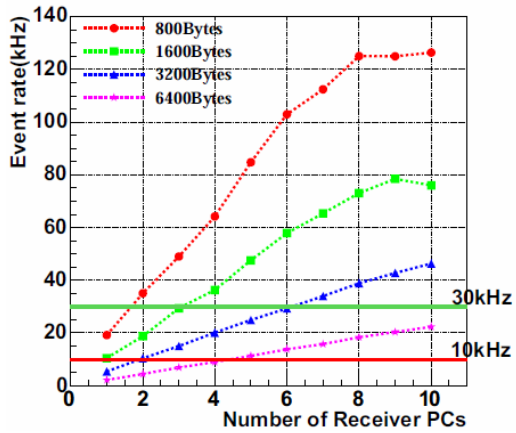


Total data transfer rate

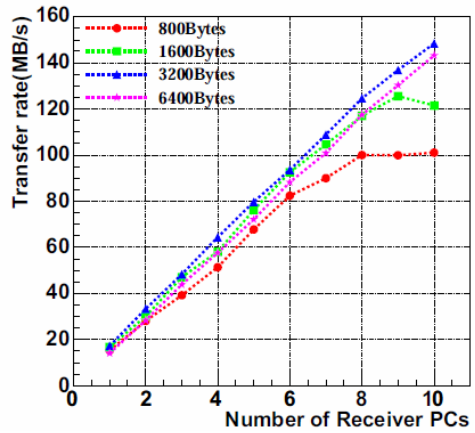
Performance Study Inside EB (3)

Master thesis of K.Ito (2005)

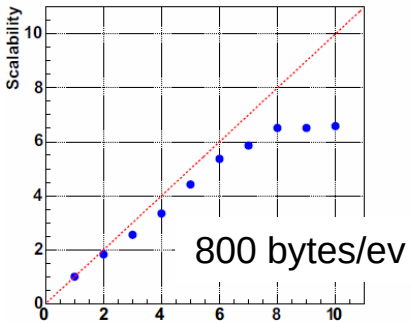


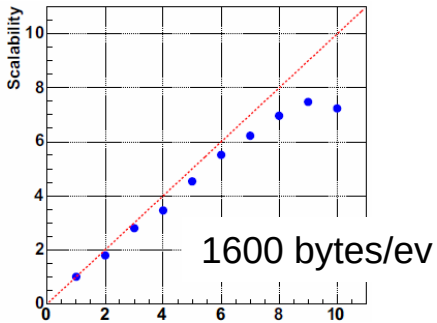


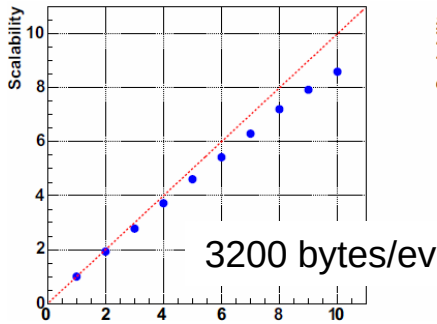
Event rate

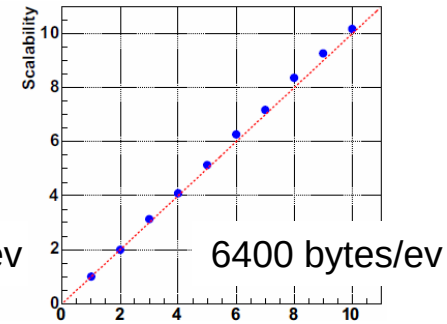


Total data transfer rate









$$Scalability = \frac{EventRate(\#of\,ReceiverPC = 1-10)}{EventRate(\#of\,ReceiverPC = 1)}$$

vs. # of 1st later PCs

Concepts of Implementation

- Switch from house-made technologies to generally existing technologies

Less effort for documentation

  - E.g. (1): event buffering
    - Now: House made (shared memory + semaphore).
    - SB: Socket buffer within Linux kernel.
  - E.g.(2): state machine
    - Now: All EB nodes are managed by NSM.
    - SB: **`rwho`** or **`finger`**+**`.plan`**
      Only one NSM node to respond to master.
      This collect all EB nodes' status via those commands.

Concepts of Implementation

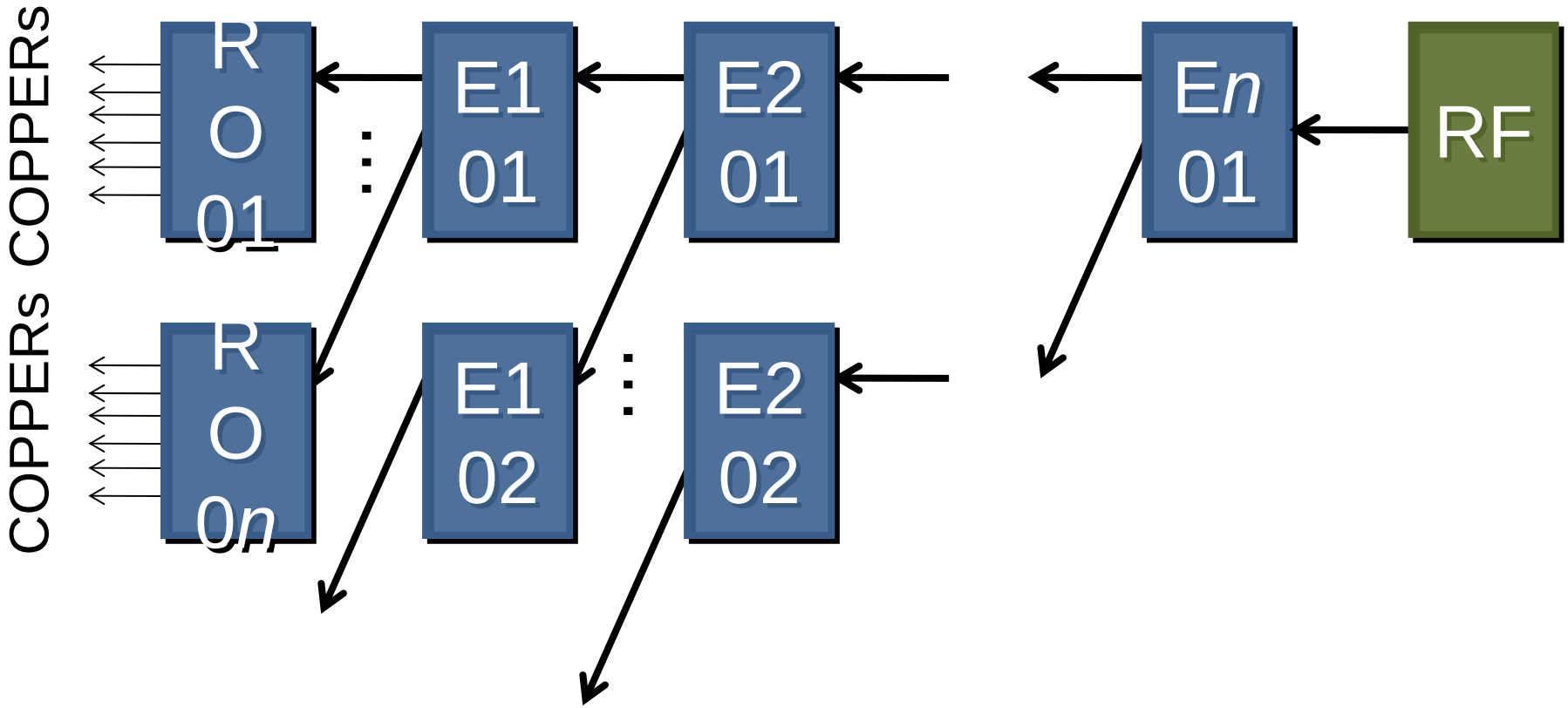
- Quit from complexity

Less effort for trouble shooting

  - NSM: Only one NSM node to respond to master.
  - D2: Own error reporting / logging → **`syslog`**.
  - No (or minimum) configuration file:
    the present EB requires large effort to add new ROPCs
    for its multiple and complicated configuration files...
  - Common software for all EB PCs in each layer.

We have just started the conceptual design.
Following slides includes many unimplemented ideas.

Network Link Propagation



Network connections propagate at **RUN START**
from DOWNstream to UPstream.

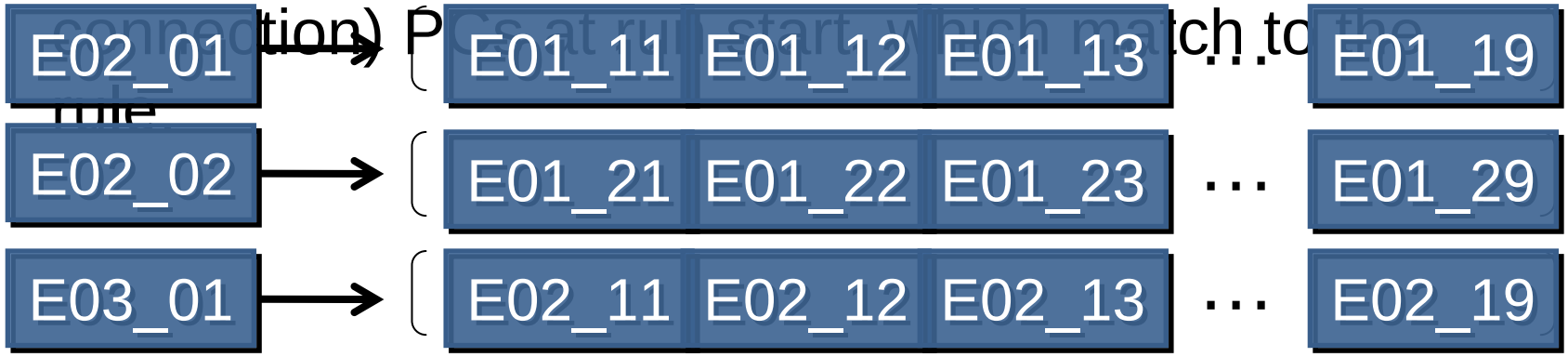
→ # or ports awaiting for connection can be unity.

Determination of PCs to Connect

Recall "no (or minimum) configuration file"

- Idea #1: Determine EB PCs to connect from its own hostname + rule.
  - EB PC connects to all available (= accepting connection) PCs at run start which match to the rule.

E.g.:

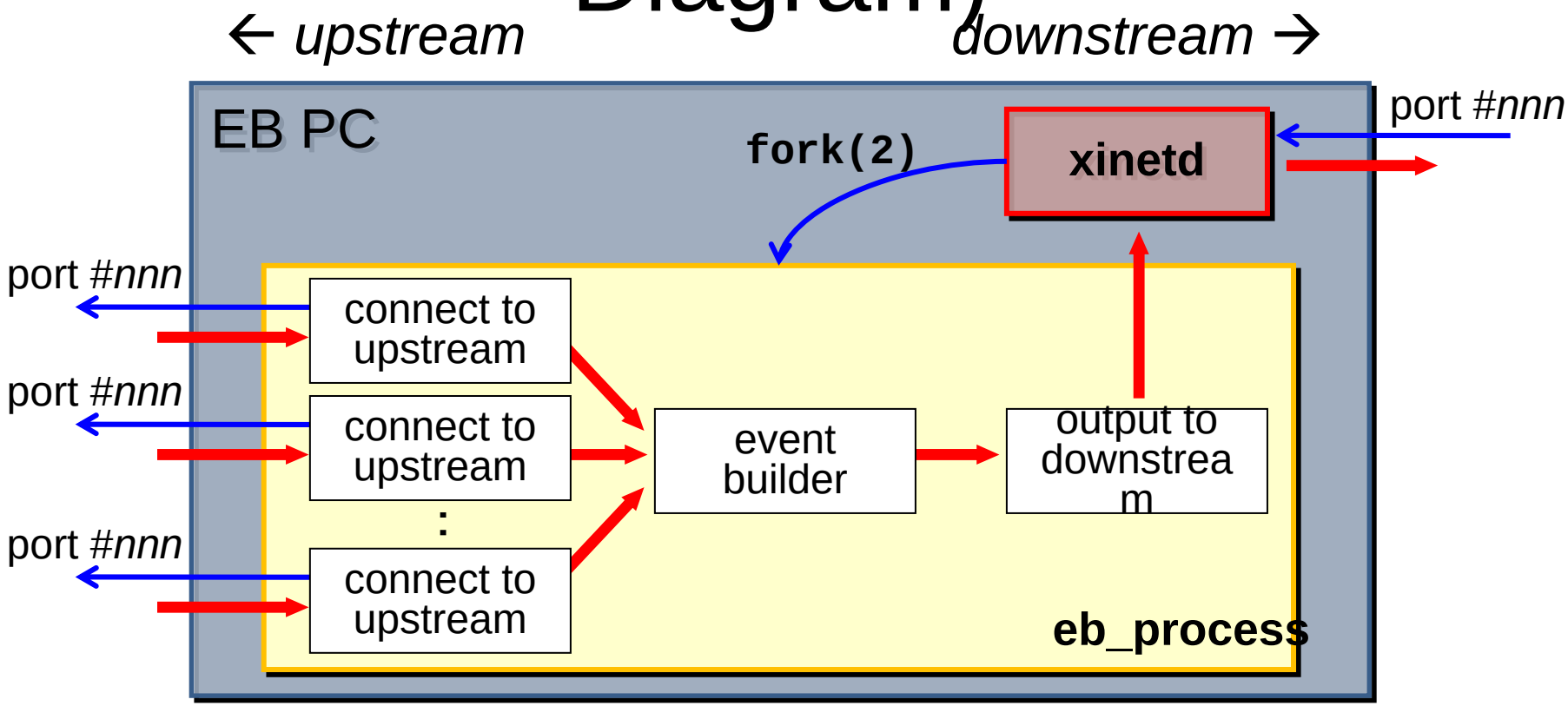


  - Two new schemes must be created:

Determination of PCs to Connect

- Idea #2: “EB_CONFIG server”.
  - EB_CONFIG server knows which EB PCs are booted up and which are not.
  - EB_CONFIG server knows which subsystems are included and which are excluded (told from MASTER).
  - EB_CONFIG server knows which EB PCs belong to which subsystem (from a configuration file).
    - But this is what we like to avoid.

Inside EB PC (Process Diagram)

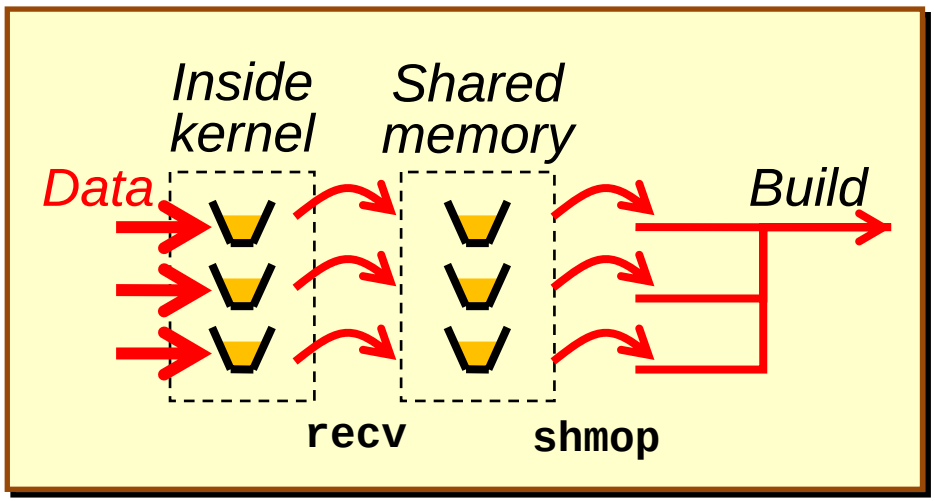


¶ An EB PC in the exit layer may have to output the built-up record to one of multiple destinations (RFARM PCs), which consequently means the EB PC accepts multiple connections from each destination PC. We have found a way to pass socket descriptor to another process. This enables us to perform event building, destination selection, and data transmission in a single process.

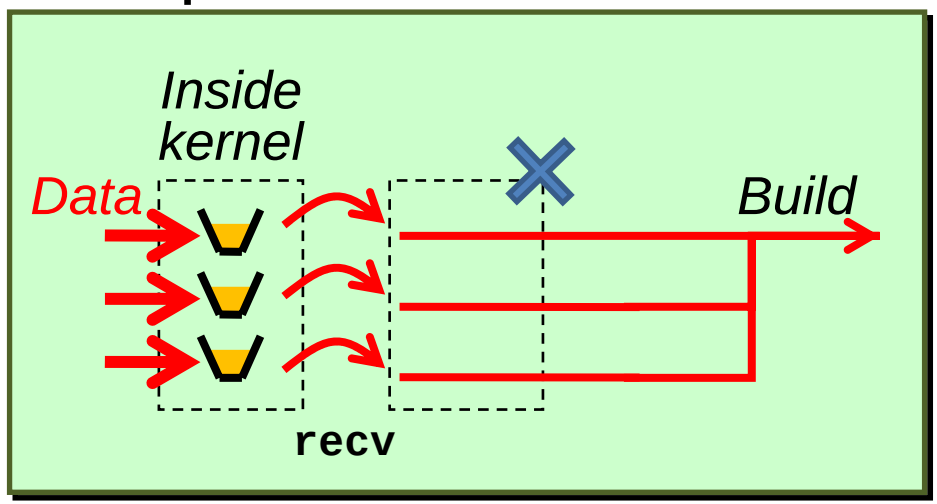
Input Buffer

- Use Linux kernel buffer
  instead of house-made buffering scheme
  - Much easier implementation/maintenance.
  - Unlike when the present EB developed, very large buffer (~80MB) can be allocated in the kernel now.

Now



SuperBelle

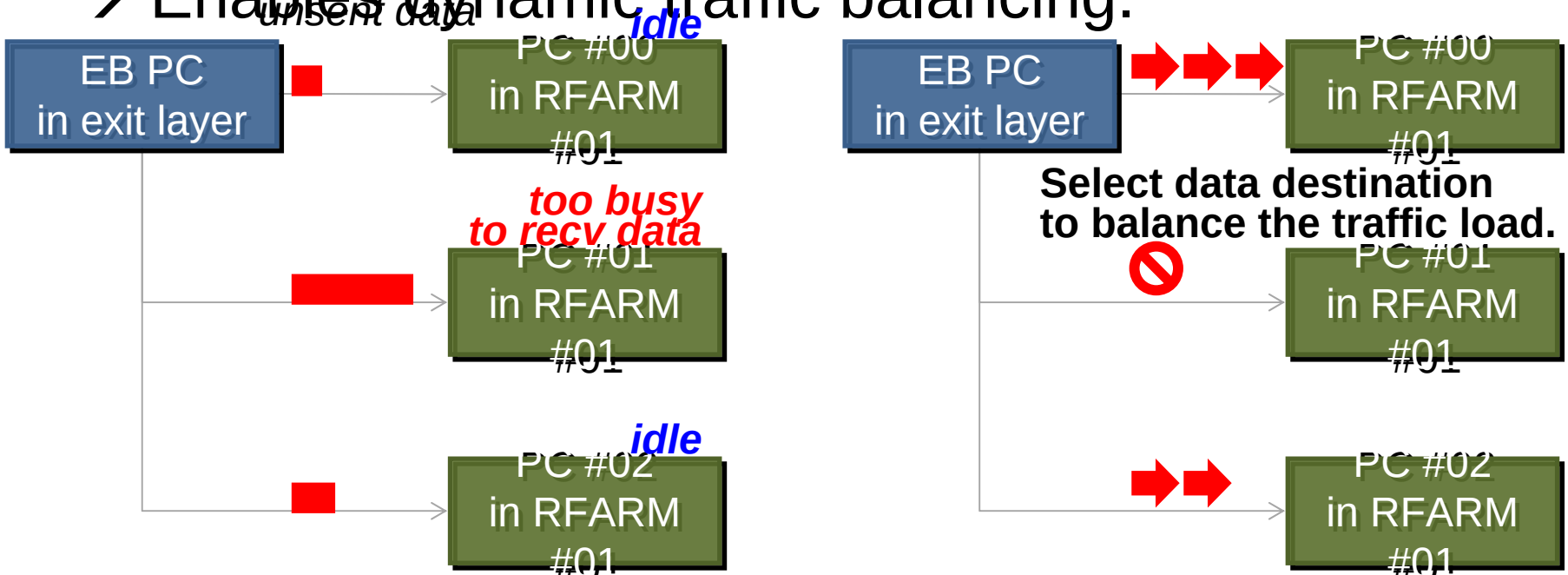


Output Buffer

- Recent Linux provides information about how much unsent data remain in the kernel output buffer.
  - Established by SK DAQ team (Hayato Yamada).

```
ioctl(
   sd, SIOCOUTQ, &val);
```

→ Enables dynamic traffic balancing.



State Machine / Status Collection

- Possible statuses of EB PC

From downstream to the EB PC

Accepting side
Not ready
Ready
Partially
Fully connected

← **Ready to run** →

From the EB P to upstream

Connecting side
Not ready
Ready
Partially
Fully connected

- Collection of EB PC statuses and EB Configuration

EF ARM side **rwho is used.**

CDAQ side

Collection of EB PC statuses

Query: which EB PC is ready? to connect? included?

PC: EBNSM

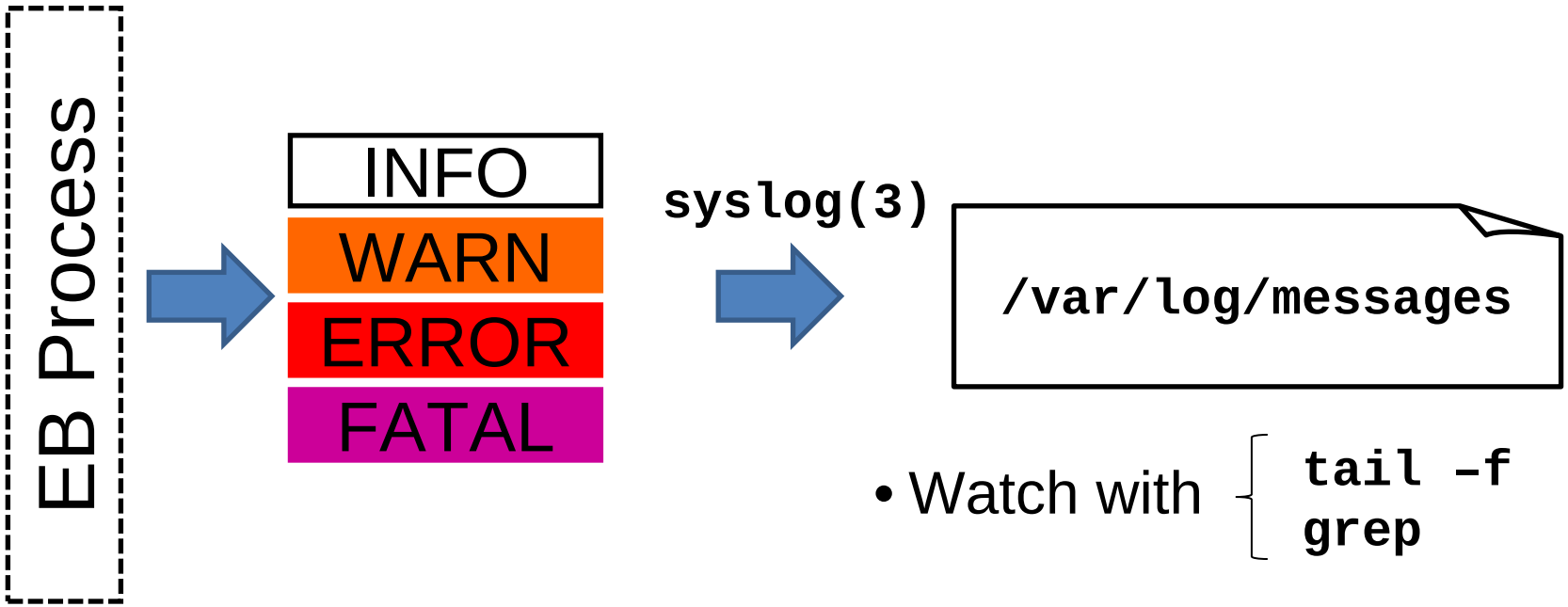
EB_CONF server

nsmd

NSM command: EB_CONFIG …

NSM reply: READY, NOTREADY

Error Reporting and Logging



- Watch with { `tail -f` / `grep` }
- Throw error messages to the NSM master.

BASF Substitution

- Separation of BASF from EFARM
  - Present EFARM strongly depends on BASF; BASF modules build up event records.
    → May cause BELLE_LEVEL problem.
  - SuperBelle EFARM is not to use BASF to build up event.
- Substitution of BASF
  - BASF-like substitution for online data quality check and/or L3 trigger use will be provided.

```
const int
 (*userfunc[])(unsigned int *data, size_t size);
```

Summary

- We have started the design of event building farm for SuperBelle.
- Regulation on the FINESSE is given from the EFARM.
- Several performance studies about network data link are already made to input to the new EFARM design.
- The new EFARM fully utilizes pre-implemented Linux functions instead our own invention.
- The new EFARM tries to be self-configured to be free from configuration-file nightmare.