Load balancer

y.ma@riken.jp

20170721

Outline

- Introduction
- Description of the problem
- Discrete event simulation, Queuing theory…
- Summary & todo

Introduction

- Load balancing for electricity…
- Load balancing for web server:
  - Caching, content delivery network…
- Load balancing for trigger-less DAQ:
  - Monitor status of EPN
  - Optimize the usage/coupling between FLP&EPN
  - Exception handling: broken node, data quality…

Definition of two scenarios

Number of FLP = M; Number of EPN = N;

- Scenario I:

If EPN processing time is fast enough,
then M>N and network may become the bottle neck.
For instance, 50 FLP needs at least 10 EPN in order to
have total throughput of 10GB/s.
Total network throughput = 10×10gbps ≈ 10GB/s

- Scenario II: (used in the following discussion)

If EPN processing time is NOT fast enough,
then M<N and network throughput is not a problem anymore.
For instance, 50 FLP conjugates with 300 EPN.
Total network throughput = 50×10gbps ≈ 50GB/s

General property of the problem

Number of FLP = M; Number of EPN = N;
Data notation i-j:
i = ith Frame ID, from 1 to # of events
j = jth Frame Segmentation
(FLP Node ID = Frame Segmentation)

Data Matrix

$$\begin{bmatrix} x_{11} & x_{12} & x_{13} & \cdots & x_{1m} \\ x_{21} & x_{22} & x_{23} & \cdots & x_{2m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & x_{n3} & \cdots & x_{nm} \end{bmatrix}$$

buffered in 2nd EPN; complete data frame, ready for processing

buffered in 3rd FLP
keep at least N fragments
in order to fill N EPNs
Assume FLP has big enough RAM...

Definition of *data matrix*

Data Matrix

$$\begin{bmatrix} x_{11} & x_{12} & x_{13} & \dots & x_{1m} \\ x_{21} & x_{22} & x_{23} & \dots & x_{2m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & x_{n3} & \dots & x_{nm} \end{bmatrix}$$

Properties of ***data matrix***:

- Bandwidth ≤ M×10gbps;
  - if all M FLPs are sending data to EPNs non-blocking, the bandwidth = M×10gbps;
  - if one or more FLPs are blocked because of congestion on the same EPN, bandwidth < M×10gbps;
- To achieve high data throughput, the number of EPN receiving data (denoted by Q) must have Q ≥ M;
- FLP buffer depth ≥ Q

Demonstration I: FLP M=3, EPN N=6

Step 1

$$\begin{bmatrix} x_{11} \end{bmatrix}$$

Step 2

$$\begin{bmatrix} x_{11} & x_{12} \end{bmatrix}$$

Step 3

$$\begin{bmatrix} x_{11} & x_{12} & x_{13} \end{bmatrix}$$

...

Step 18

$$\begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \\ x_{41} & x_{42} & x_{43} \\ x_{51} & x_{52} & x_{53} \\ x_{61} & x_{62} & x_{63} \end{bmatrix}$$

The slowest approach (bandwidth = 10gbps):

- All 3 FLPs are queued for the same EPN;
- The queue iterates over 6 EPNs one by one;
- It takes 18 steps is complete 6 EPNs but 3 steps for one EPN

If the amount of data is very small but processing time is extremely long, one can adapt this approach.

Demonstration II: FLP M=3, EPN N=6

Step 1

$$\begin{bmatrix} x_{11} & & \\ & x_{22} & \\ & & x_{33} \\ & & \\ & & \\ & & \\ & & \end{bmatrix}$$

Step 2

$$\begin{bmatrix} x_{11} & & x_{13} \\ x_{21} & x_{22} & \\ & x_{32} & x_{33} \\ & & \\ & & \\ & & \\ & & \end{bmatrix}$$

Step 3

$$\begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \\ & & \\ & & \\ & & \\ & & \end{bmatrix}$$

Step 6

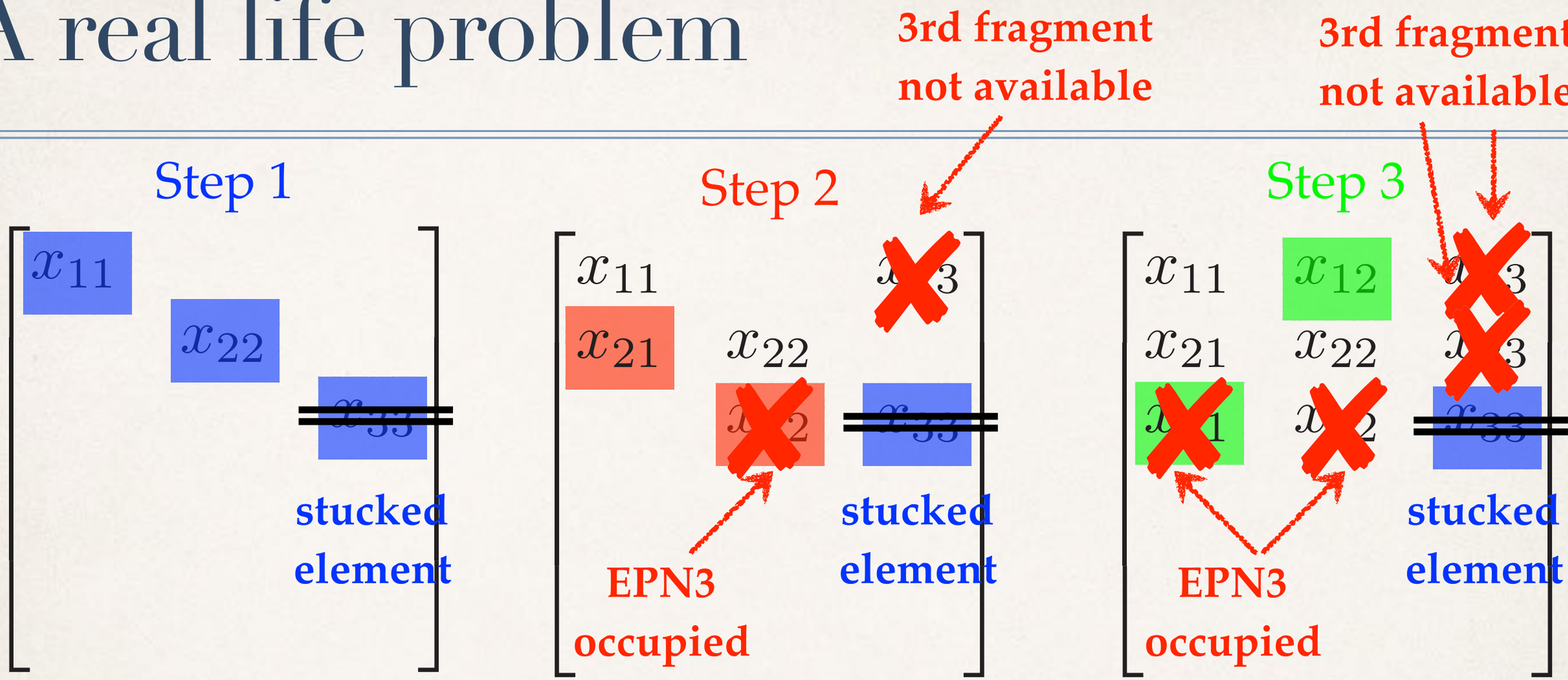
$$\begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \\ x_{41} & x_{42} & x_{43} \\ x_{51} & x_{52} & x_{53} \\ x_{61} & x_{62} & x_{63} \end{bmatrix}$$

The high bandwidth approach :(bandwidth = 3×10gbps)

- Each FLP is sending data to a different EPN;
- FLP iterates over the same subgroup of M=3 EPNs;
- It takes 3 steps to complete 3 EPNs; 6 steps for 6 EPNs

If everything works like a precise mechanical clock,
our job is done and we can go home now.

A real life problem

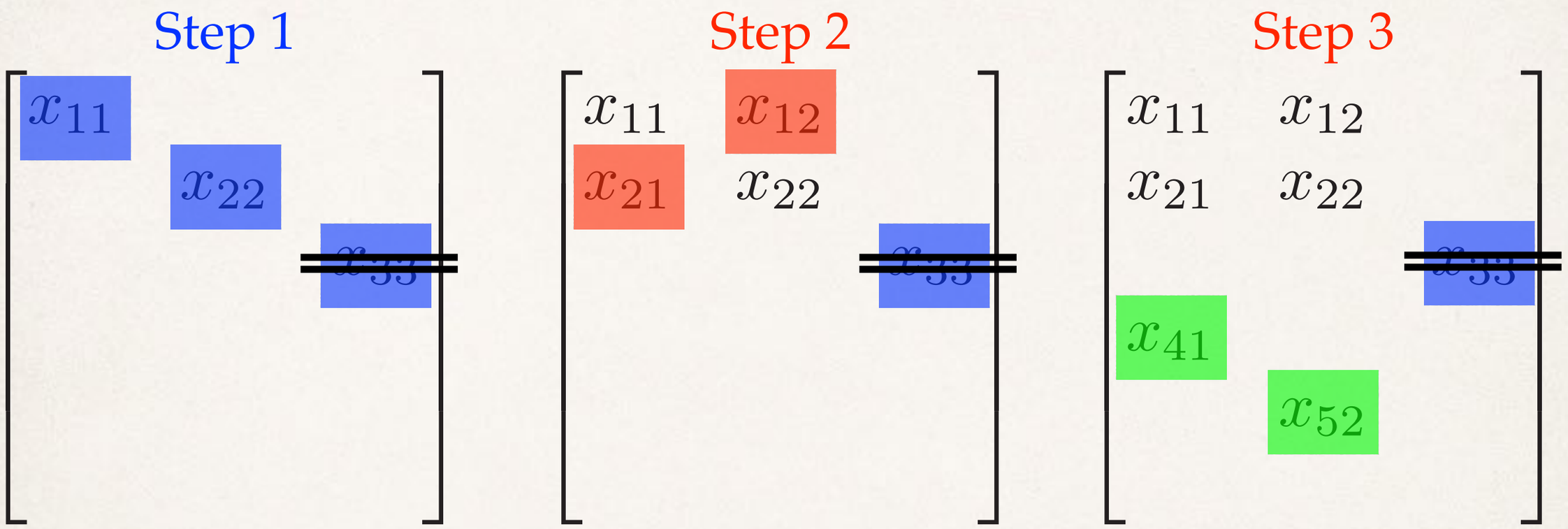


Some facts:

- The size of each frame segment is different i.e. the size of x_{ij} follows a gaussian distribution(?)
- Each FLP-EPN links has different speed, which is gaussian from our previous TCP/IP throughput study

The completion time of each step = the slowest one of x_{ij}

Optimization: load balancing



Motivation:

- Each FLP visits one available EPN without waiting for the completion of the sub group i.e. break the set defined in “step”;
- The EPN has less missing frame fragments has higher priority. An early EPN completion can help for data processing.

Weighted sorting problem: M lists of N×M elements;
weight = 1, missing fragment; 2, EPN RAM capacity
Guarantee for the fast total EPN build rate.

One more problem: centralized vs distributed

- How to monitor/collect FLP EPN status to a "control center" and dispatch to the FLPs in time?

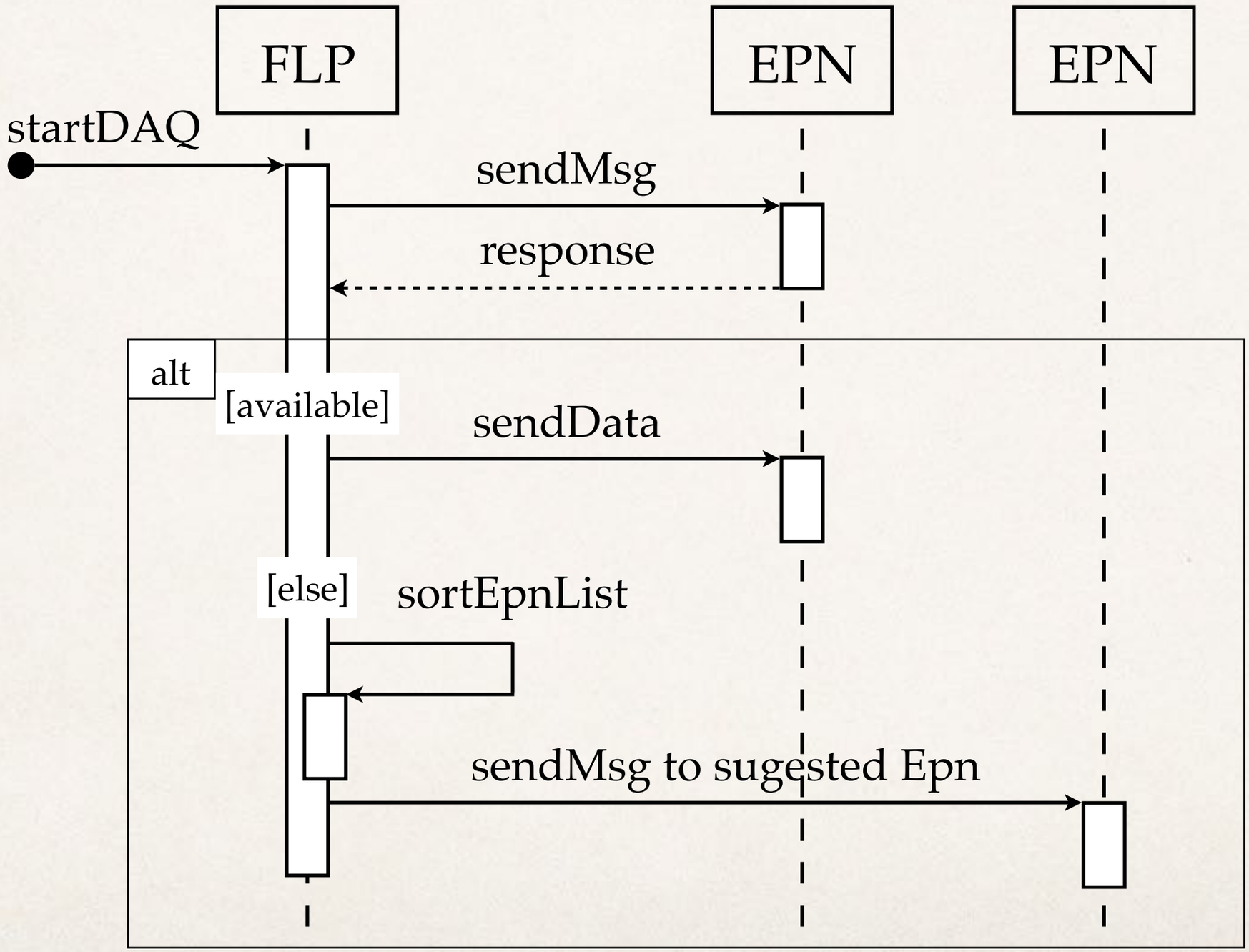
- We need *a centralized system* where FLP and EPN status monitoring is *instantaneous and reliable*, for instance, a PCI based unit.

- A fully programmable switch to implement the weighted sorting algorithm? Is such thing available?

What can we do with distributed system?

- Each FLP talks to one EPN to ask for its status;
  - If this EPN is waiting for data, it tells FLP to start data transition;
  - If this EPN is currently receiving data from another FLP, it tells FLP that it has been occupied; FLP sort ENP visit order based on EPN feed back information

What can we do with distributed system?



What can we do with distributed system?

- Each FLP performs sorting algorithm based on partially available EPN information

FLP 1 ⟶

$$\begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \\ x_{41} & x_{42} & x_{43} \\ x_{51} & x_{52} & x_{53} \\ x_{61} & x_{62} & x_{63} \end{bmatrix}$$ EPN 1 ↓

When FLP_i visits EPN_j,
its message arrival time, data size and so on
will be recorded into x_ij.
When another FLP_k visits EPN_j,
EPN_j will feed back with ALL stored information
including FLP_k itself.
For FLP_k, the jth row of the information matrix
is updated to help FLP_k makes decision.

Data structure held by each ELP node:

- it is a snapshot for each EPN status taken upon each visit;
- the snap shot can be outdated because of other FLP's action

Simulation setup: data structure

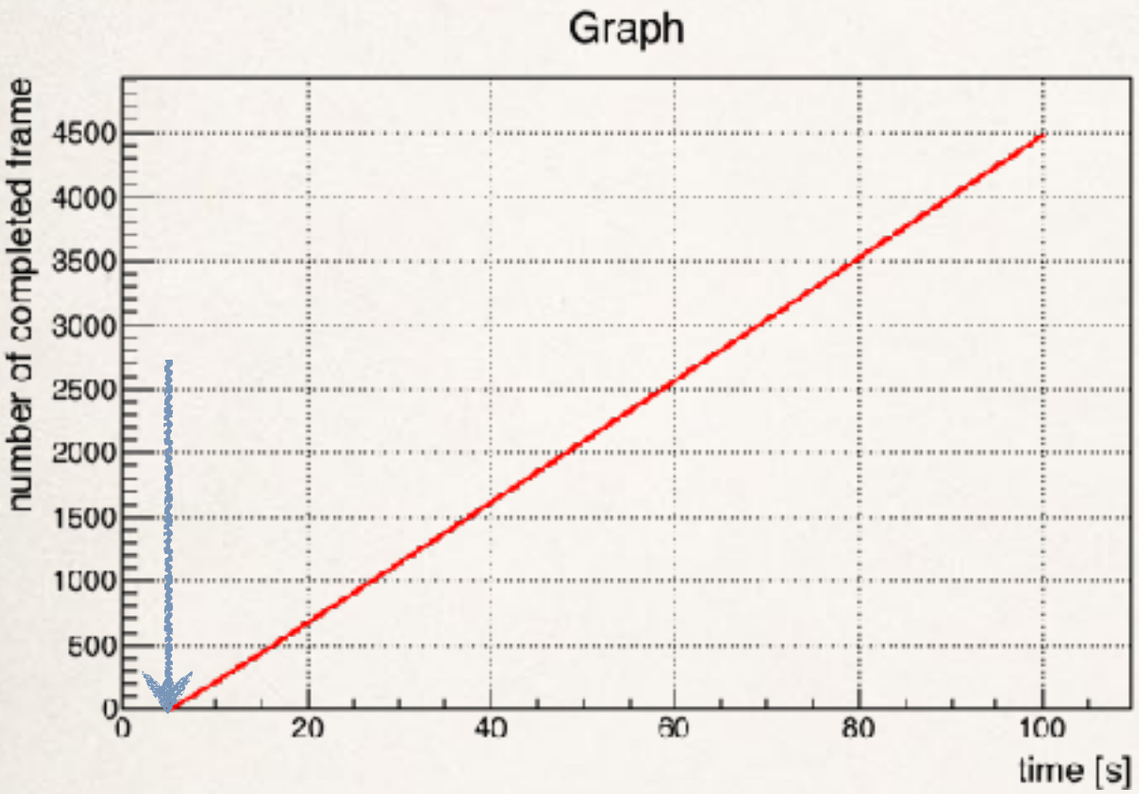
```
// data structure for flp info stored inside epn
typedef struct
{
    Int_t   frame_id;
    std::vector< Int_t > binding_flag;
    std::vector< Double_t > binding_time;
    std::vector< Double_t > msg_arrival_time;
    std::vector< Double_t > data_arrival_time;
    std::vector< Int_t > data_amount;
} flp_info_t;

// data structure for epn info stored inside epn
typedef struct
{
    Int_t   frame_id;                // frame id
    Int_t   status;                  // true: pr
    Int_t   processing_start_time;   // message
    Int_t   processing_stop_time;    // message
} epn_info_t;

// data structure for epn machine
typedef struct
{
    flp_info_t              flp_info;
    std::vector< flp_info_t > flp_info_record;
    std::vector< epn_info_t > epn_info_record;
} epn_t;

// data structure for flp machine
typedef struct
{
    Int_t       status;                  //
    Double_t    due_time;                //
    std::vector< epn_t > epn;            //
    std::vector< Int_t > sorted_epn_list;  //
} flp_t;
```

```
FLP [ 0 ]
PRINTING FLP ..............................
flp.status =                    SENDING_MSG
sorted epn list =               3 0 1 2
EPN[ 0 ]:
PRINTING EPN ..............................
current flp info :
frame id =                      UNSET
binding_flag =                  TRUE FALSE
binding_time =                  UNSET UNSET
msg_arrival_time =              20309.9 124.531
data_arrival_time =             UNSET 20249.9
data_amount =                   UNSET UNSET
EPN[ 1 ]:
PRINTING EPN ..............................
current flp info :
frame id =                      UNSET
binding_flag =                  TRUE FALSE
binding_time =                  UNSET UNSET
msg_arrival_time =              40902.4 20377.9
data_arrival_time =             UNSET 40891.8
data_amount =                   UNSET UNSET
EPN[ 2 ]:
PRINTING EPN ..............................
current flp info :
frame id =                      UNSET
binding_flag =                  TRUE FALSE
binding_time =                  UNSET UNSET
msg_arrival_time =              62792 41034.2
data_arrival_time =             UNSET 61667.2
data_amount =                   UNSET UNSET
EPN[ 3 ]:
PRINTING EPN ..............................
current flp info :
frame id =                      UNSET
binding_flag =                  UNSET TRUE
binding_time =                  UNSET UNSET
msg_arrival_time =              82589.3 61837.4
data_arrival_time =             UNSET UNSET
data_amount =                   UNSET UNSET
```

Simulation setup: assumptions

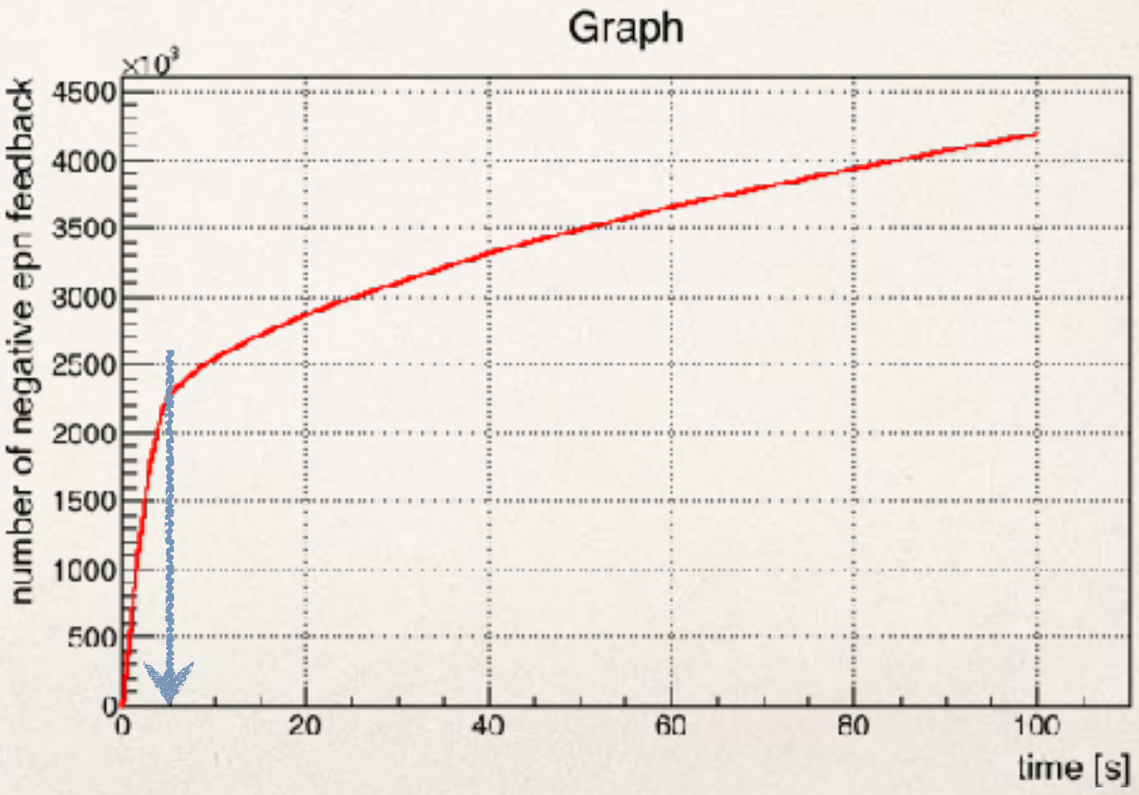
- Use measured buffer speed: speed = 20±2 ms
- Use Message queue speed: response time = 300±100 μs
- FLP: M = 250; EPN: N = 1500
- FLPs have large enough buffer; EPN processing not considered here
- simulation step = 10 μs time slice; when two events fall into the same time slice, count from FLP[0]…FLP[249]

Simulation setup: results



First complete data frame
at ~5 second = 250 seg/50;
Too many traffic for neg_msg,
to be improved by algorithm
(load balancing)

Very simple algorithm:
polling and rotation
~45 data frame/s;
slightly lower than 50 Hz



Summary & Todo

- Defined the load balancer problem as maximizing the *data matrix* completion rate
- Illustrated some typical scenario
- For a centralized system, i.e. with instant response time, a weighted sorting algorithm will be enough
- For a distributed system like O2, a self-maintained job list is more plausible
- Exception control by including more realistic factors: FLP RAM, EPN processing speed…
- seed rotation, sub divided detector groups, delayed flp msg, deep learning, predict other flp's actions and its effect for epn status, then make decision, Bayesian?

- Backup slides

- If almost ready, wait. Otherwise, find itself something to do
- Fluctuations in the neg msg
- Flp arrival time and its data amount: start with typical tcp speed, update with measured ones along time; kinds of feed back and learning? Sort the epn list with neural network?
- Keep 10k data frame history on epn for neural network?
- Break M flp and N epn into sub groups? Helpful for reduce congestion and flp ram! But each sub group can handle part of data processing! Merge these results for later stage global tracking; many things grow nonlinearly as dimension such as sorting and probability of congestion
- Each epn concentrates on one event id till it's completed
- Exceptional control

Queuing theory and DAQ

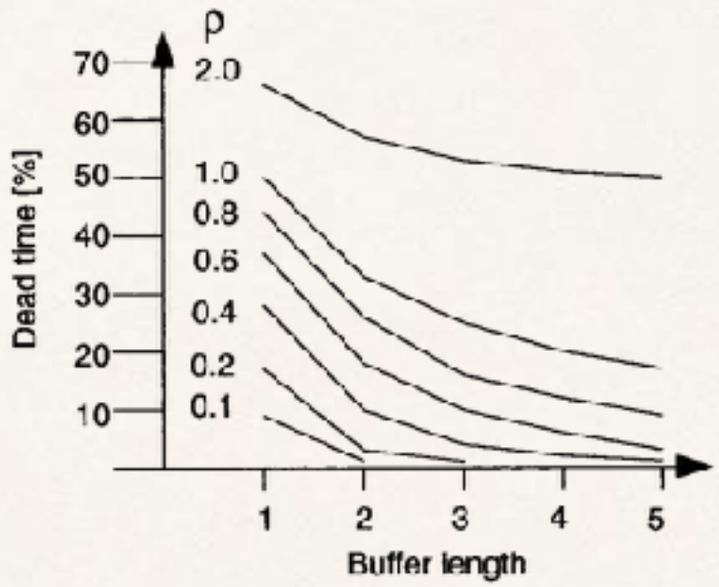


Fig. 1.11. Dead time as a function of buffer length and processor speed. ρ is the ratio of service rate to arrival rate. $\rho = 1$ means average arrival time equals average service time. In this case the dead time drops from 50% to 16.6% if five buffers are used.

R. Fruehwirth ***et al.***,
Data Analysis Techniques for High-Energy Physics 2nd ed.

Queuing theory and DAQ

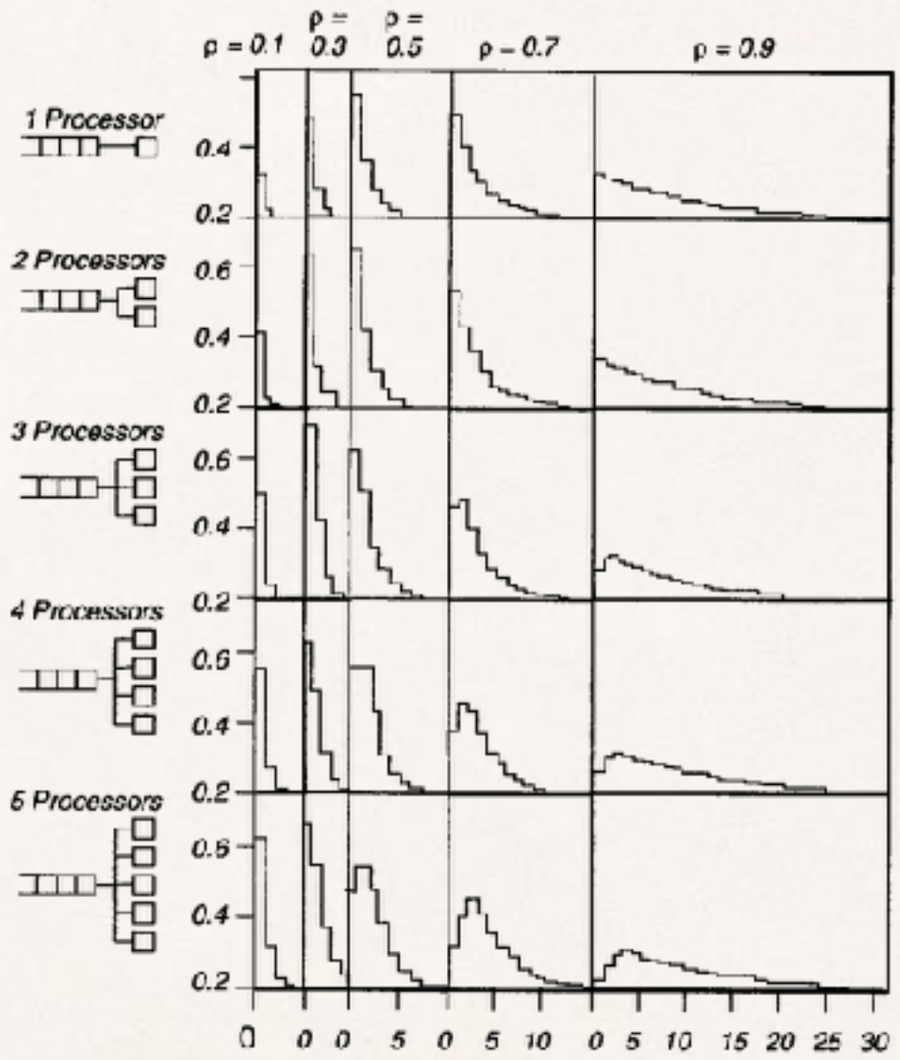


Fig. 1.14. Queue lengths for different systems. One to five processors are used to process data. From left to right the processors are slower or the input rate gets higher. One can imagine that the queue length goes to infinity if the service rate is equal to the arrival rate.

R. Fruehwirth *et al.*,
Data Analysis Techniques for High-Energy Physics 2nd ed.

Queuing theory and DAQ

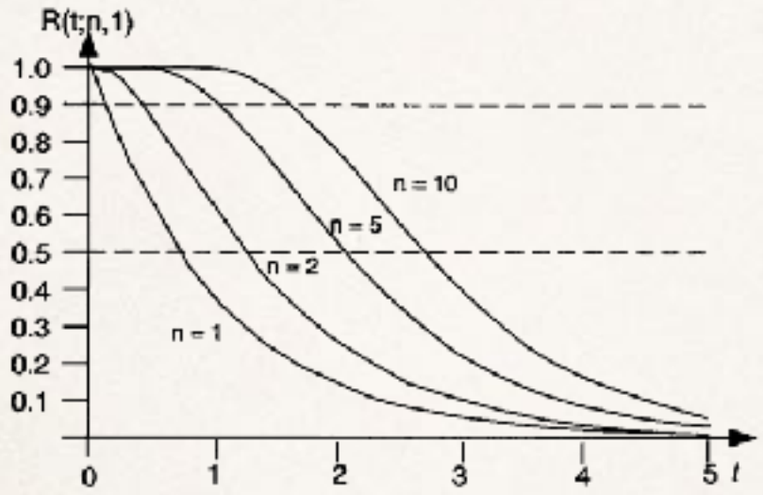


Fig. 1.23. Reliability of coupled systems with hot connection. The components have equal failure rates. On average one does not gain very much for long periods of time, but the time with a reliability at 90% increases by a large factor.

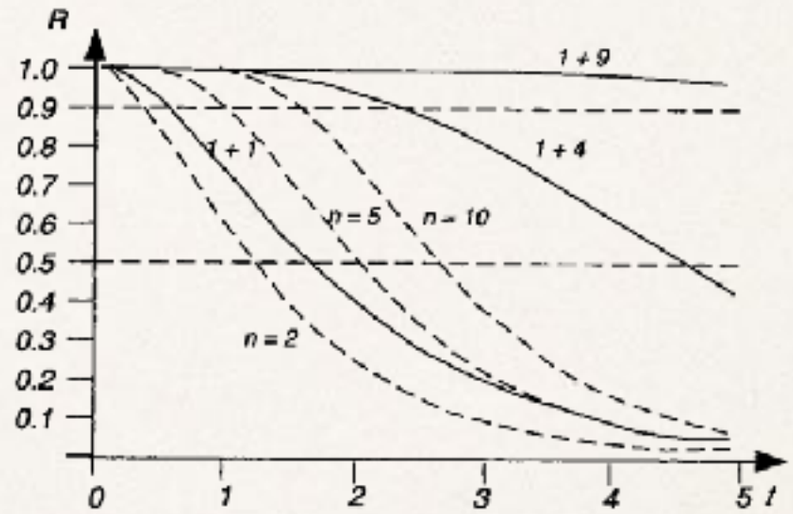
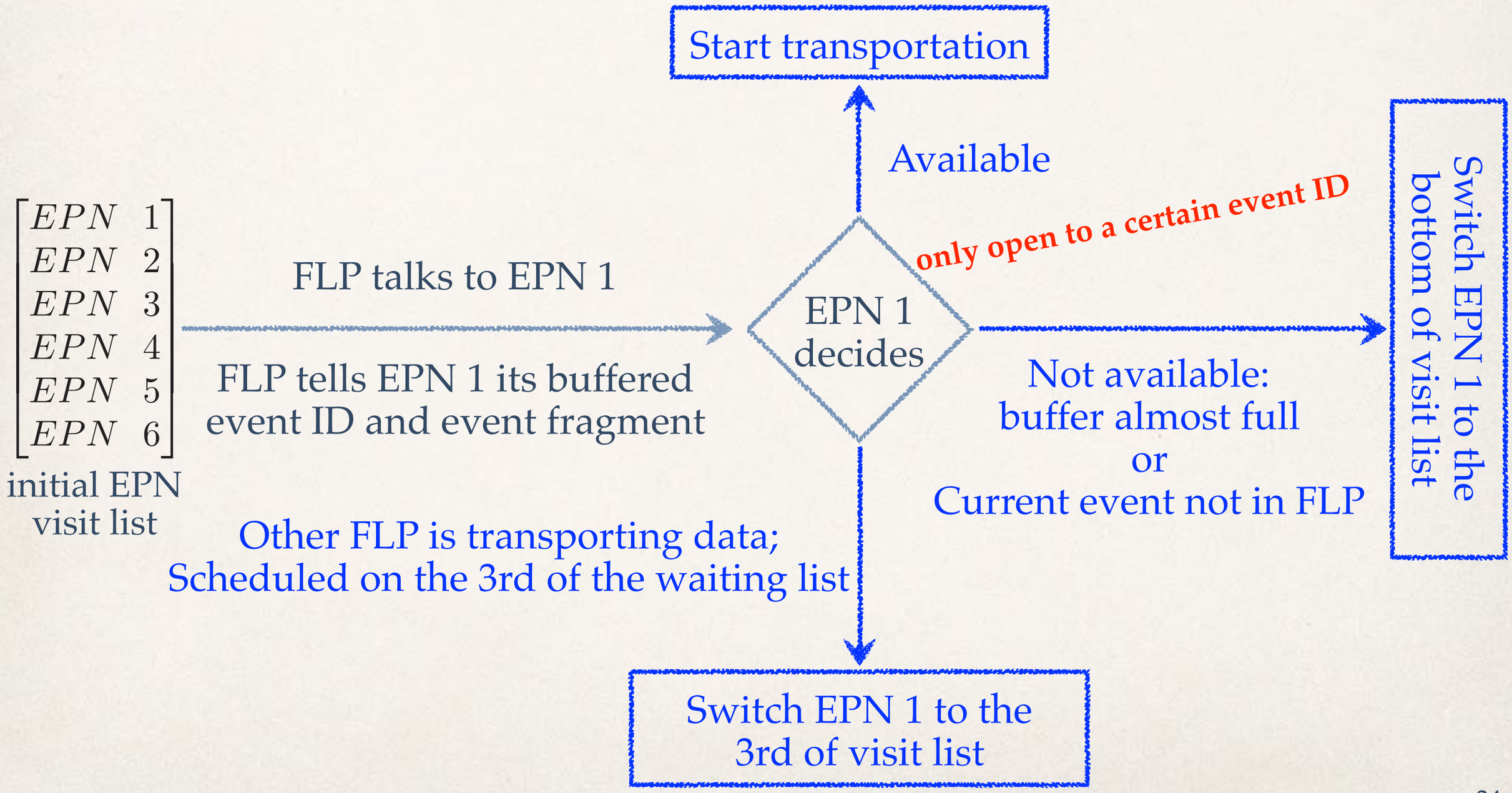


Fig. 1.24. Reliability of systems with cold connections. The reliability of systems with cold connections or stand-by components is higher than for systems with hot connections, assuming that stand-by components do not age. The dashed line shows the corresponding systems with hot connections.

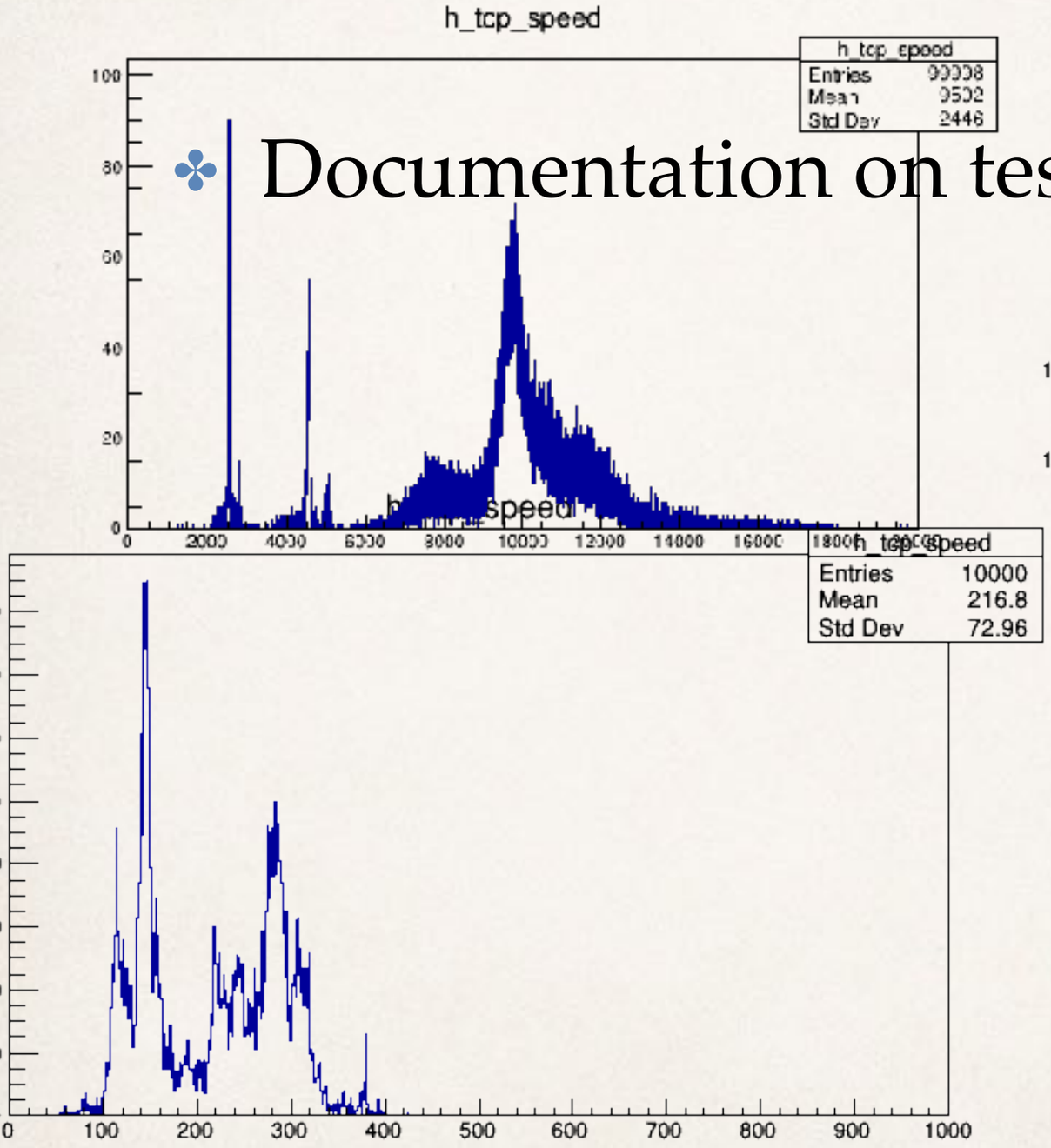
R. Fruehwirth *et al.*,
Data Analysis Techniques for High-Energy Physics 2nd ed.

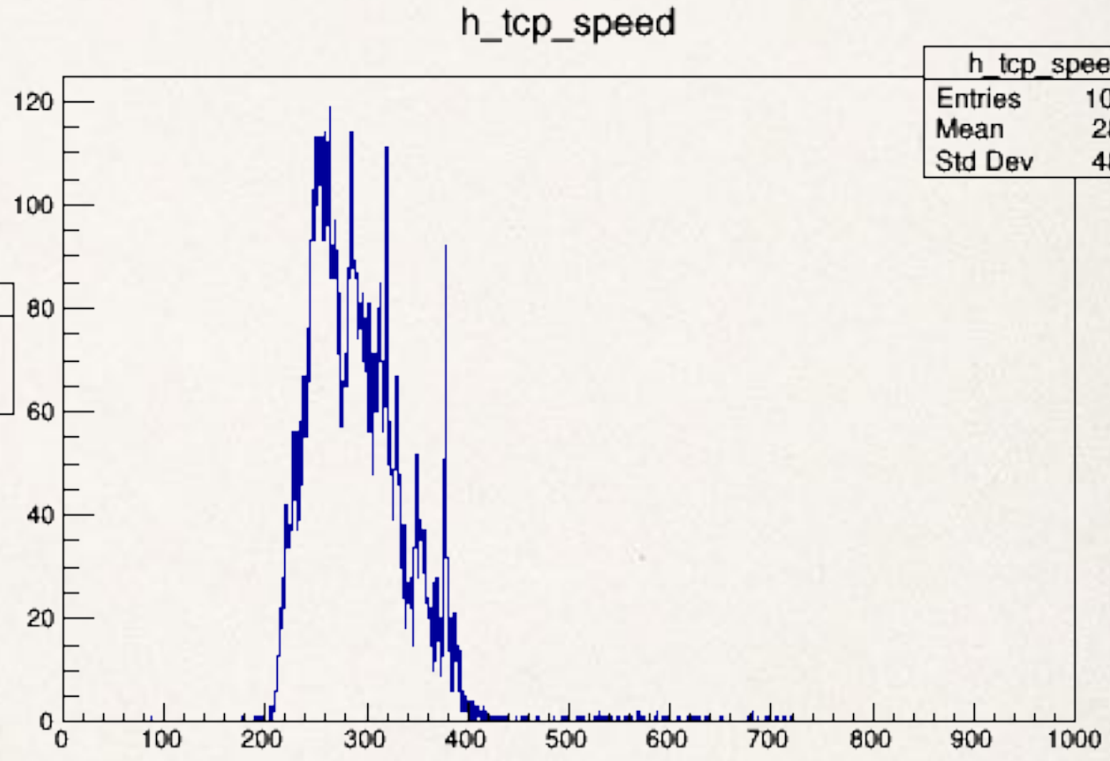
What can we do with distributed system?

Illustration for the work flow of the *ith FLP*

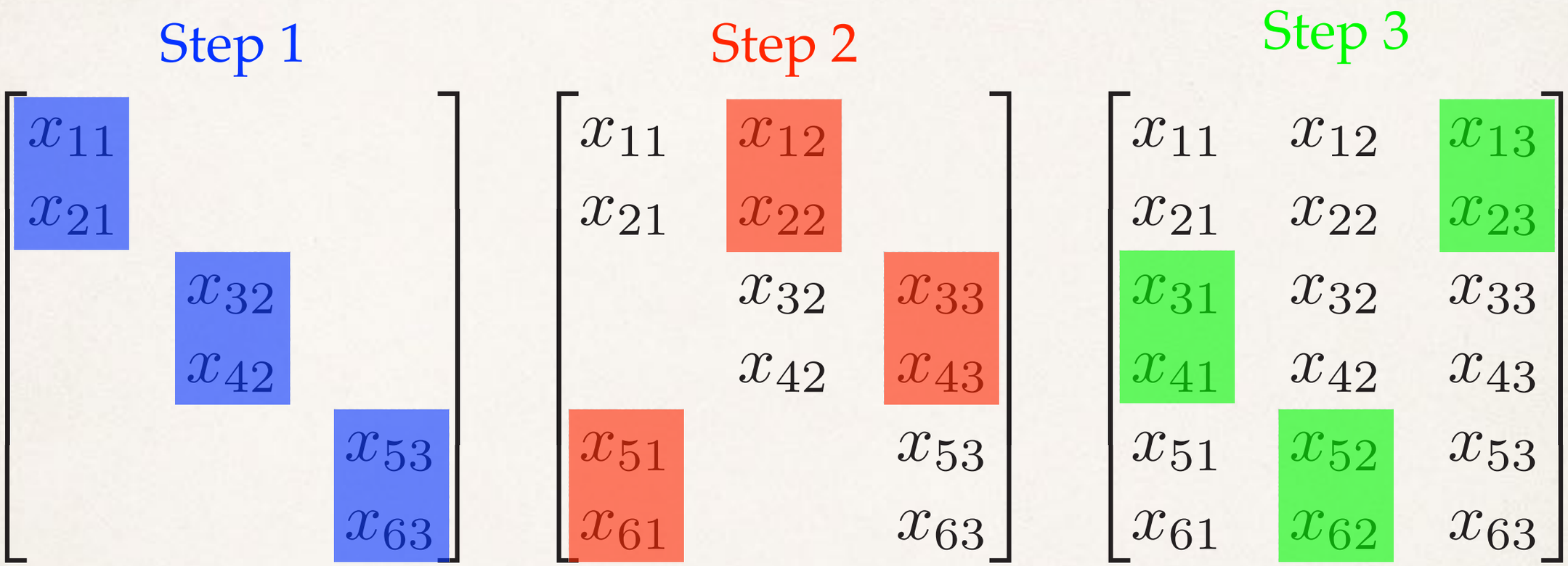


- Documentation on test method: 100 to 300 mus





Demonstration II: FLP M=3, EPN N=6



If everything works like a precise mechanical clock,
a simple iteration over M loops will get the job done!

In the above example, # of FLP = 3, # of EPN = 6.

Three iterations on FLP loops complete the task.

6 full data frame constructed for EPN to process.

General property of the problem

Situation

$$\begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \\ x_{41} & x_{42} & x_{43} \\ x_{51} & x_{52} & x_{53} \\ x_{61} & x_{62} & x_{63} \end{bmatrix}$$

Load balancer

Find a way to the goal

Goal

$$\begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \\ x_{41} & x_{42} & x_{43} \\ x_{51} & x_{52} & x_{53} \\ x_{61} & x_{62} & x_{63} \end{bmatrix}$$

Real life problem:
what if certain ENP nodes are down/too slow?
EPN visit table to optimize the data table
kinds of conditional sorting problem;
what's the limit on response time?

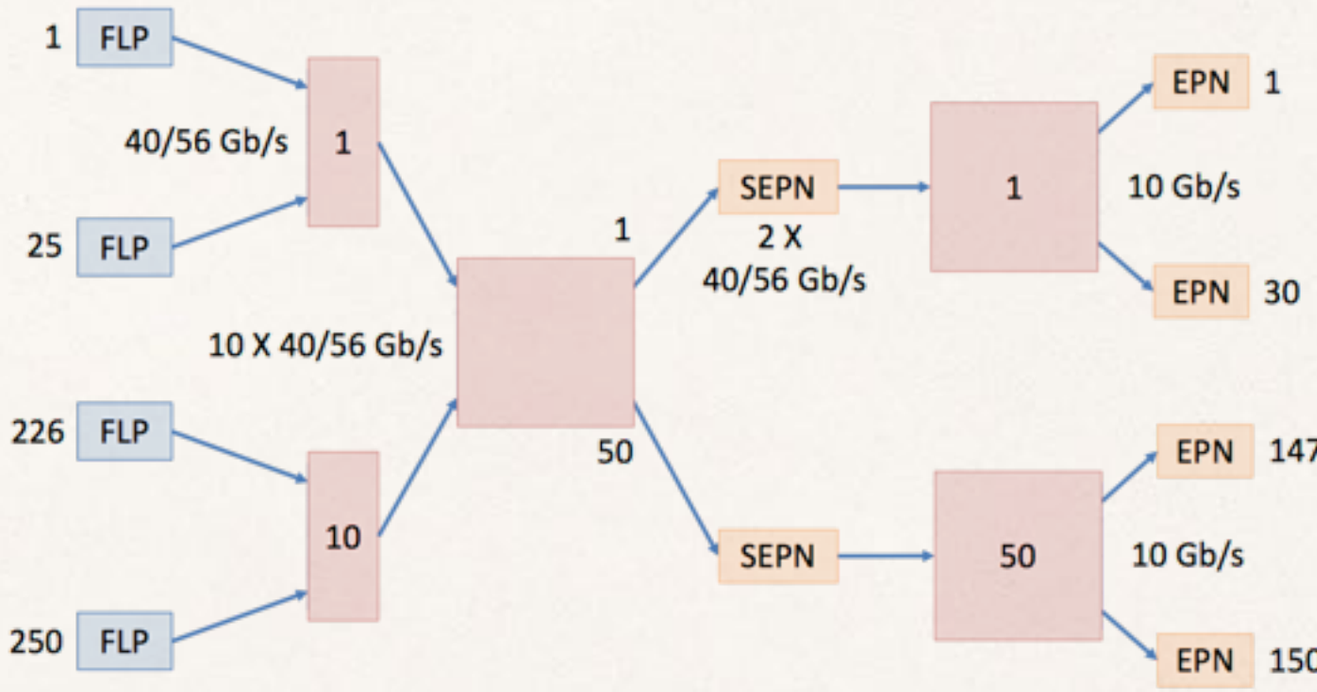


Figure 5.5: FLP-EPN network layout with SEPNs in charge of assembling and distributing the TFs to the EPNs.

Table 5.1: Hardware characteristics for FLP and EPN nodes.

Type	Number of Nodes	Input bandwidth Peak (Gb/s)	Input bandwidth Average (Gb/s)	Output bandwidth Peak (Gb/s)	Output bandwidth Average (Gb/s)
FLP	250	100	50	40	20
EPN	1500	10	2.7	0.48	0.33

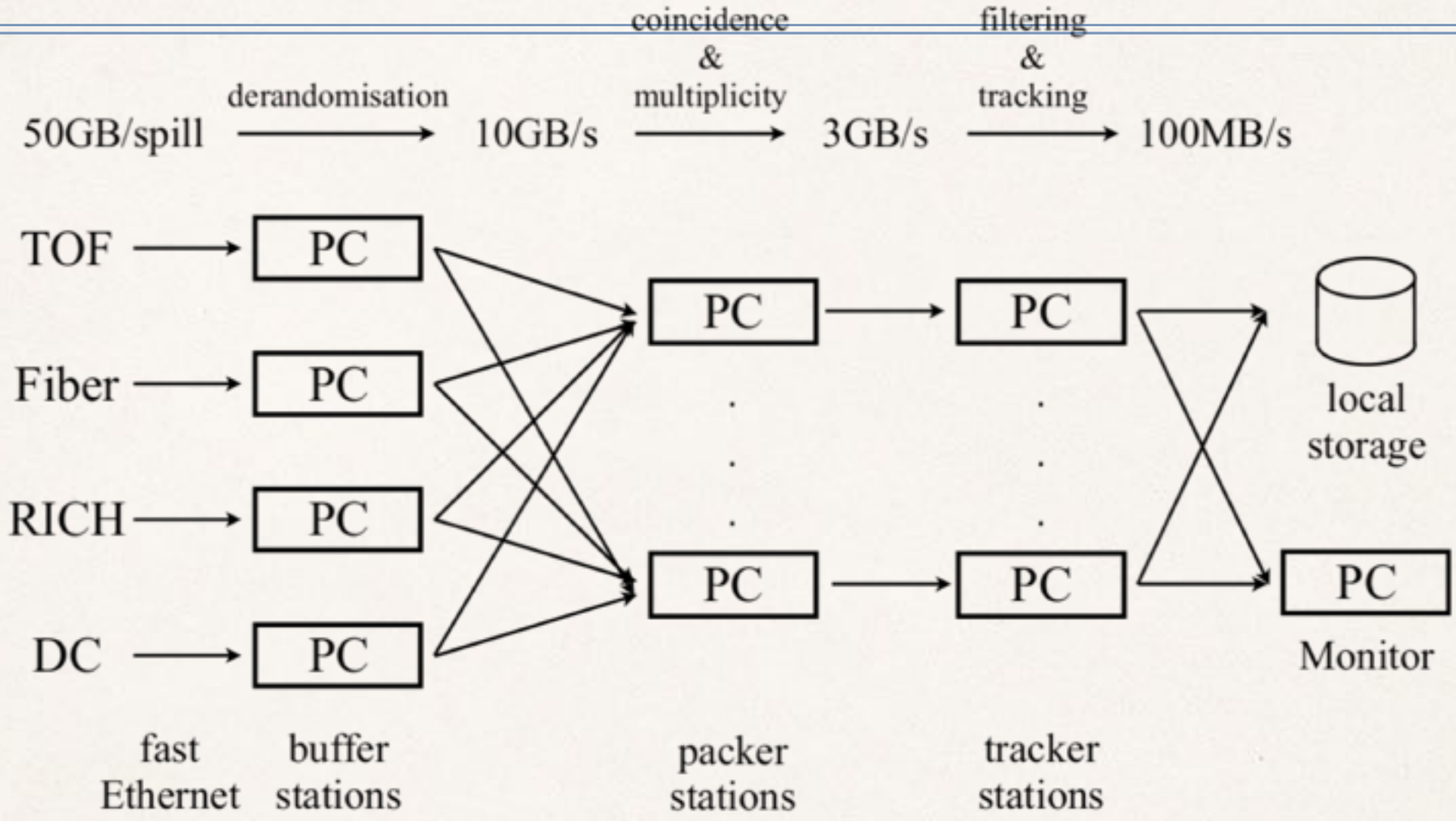


Fig.3, Schematic diagram of the proposed trigger-less DAQ. PC stands for the computing unit consisting multiple CPU and GPU cores.

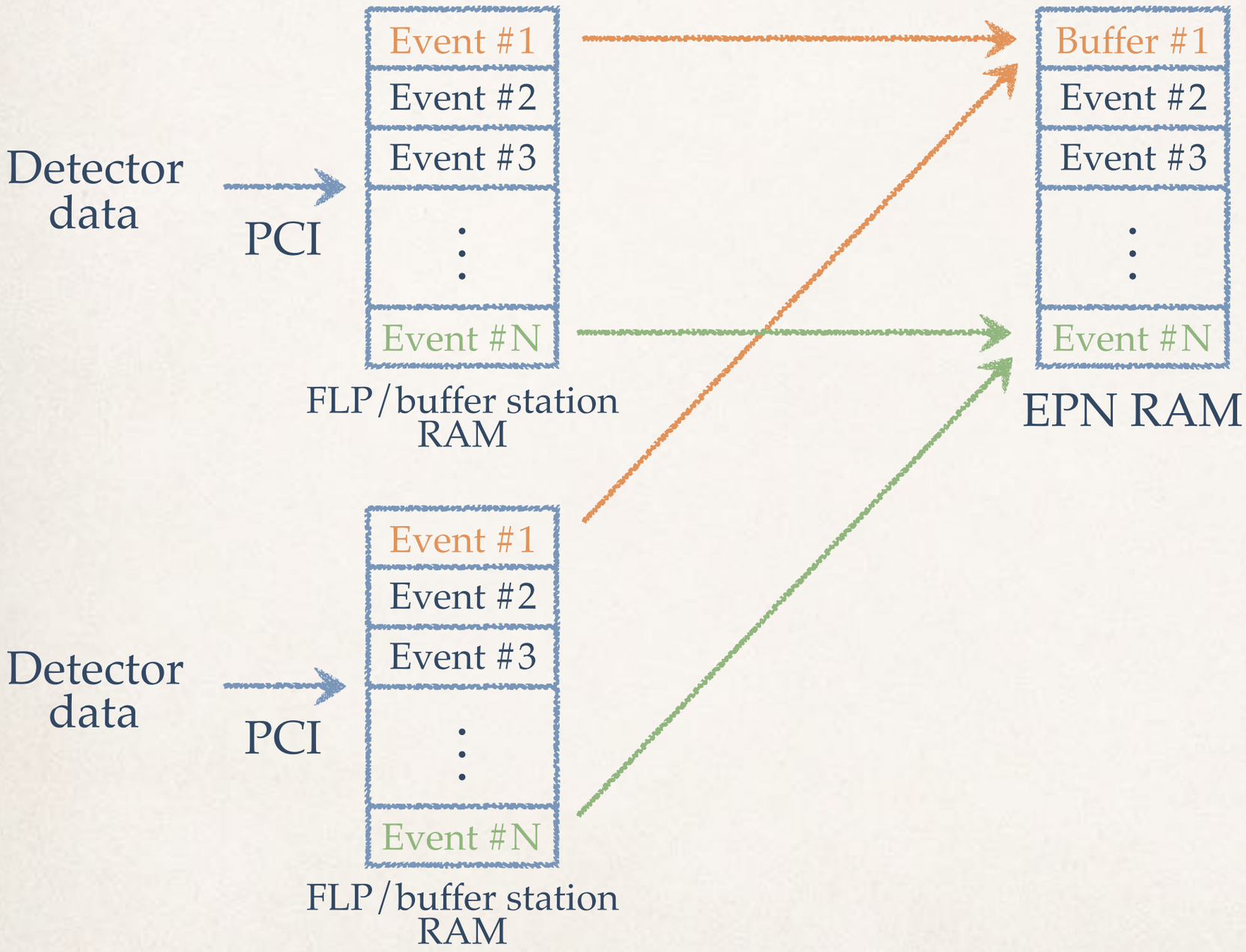

Detector data
PCI
Event #1
Event #2
Event #3
Event #N
FLP/buffer station RAM
Buffer #1
Event #2
Event #3
Event #N
EPN RAM
Detector data
PCI
Event #1
Event #2
Event #3
Event #N
FLP/buffer station RAM

Illustration of Scenario I

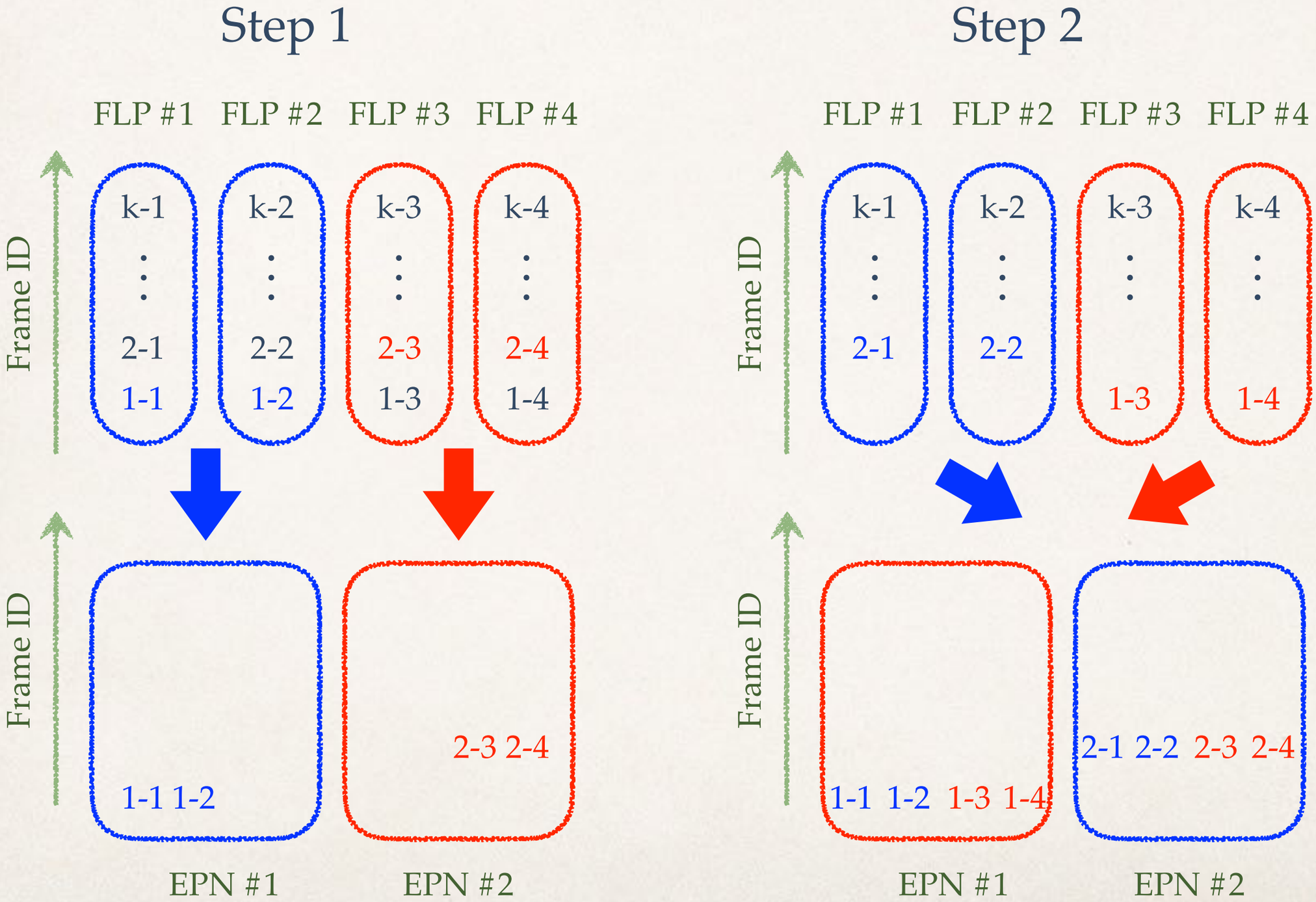


Illustration of Scenario II