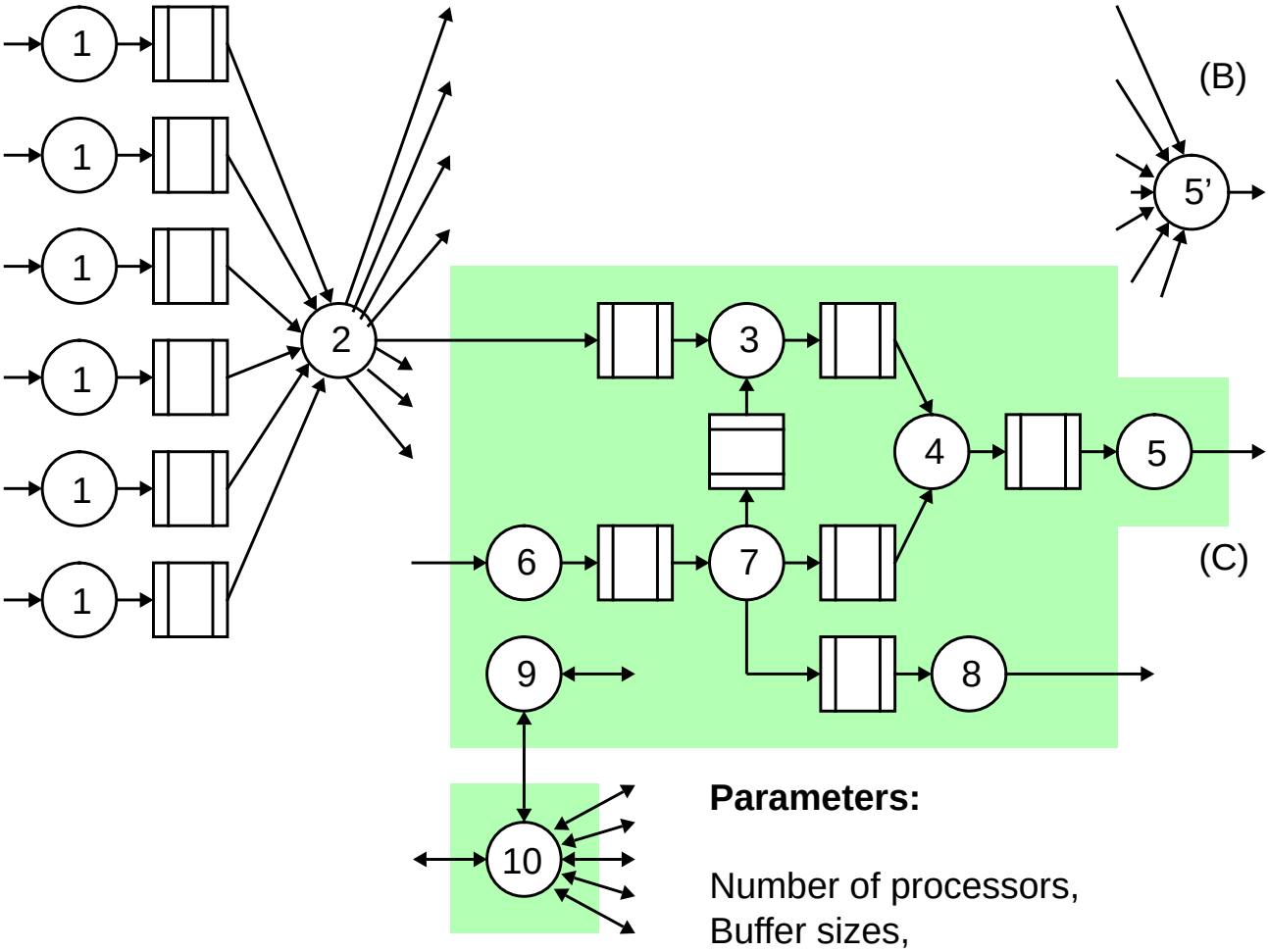
Model and parameters



Parameters:

Number of processors,
Buffer sizes,
Processing times (from demo's)

Processes: **Time:**

1. Receive ROI fragment
2. Merge ROI fragments into complete ROI message
3. Get Proc_ID, allocate processor and build RIOR (SF)
4. Send ROIR(SF) or T2DR to output interface
5. Arbitrate and broadcast RIOR or T2DR (B)

5'. Send ROIRSF to selected processor or broadcast T2DR (C)

6. Receive LVL2 decision (GOUTR)
7. Process LVL decision, return Proc_ID, build T2DR
8. Communicate with LVL3
9. Monitor performance and errors
10. Communicate with DCS

Data flow control

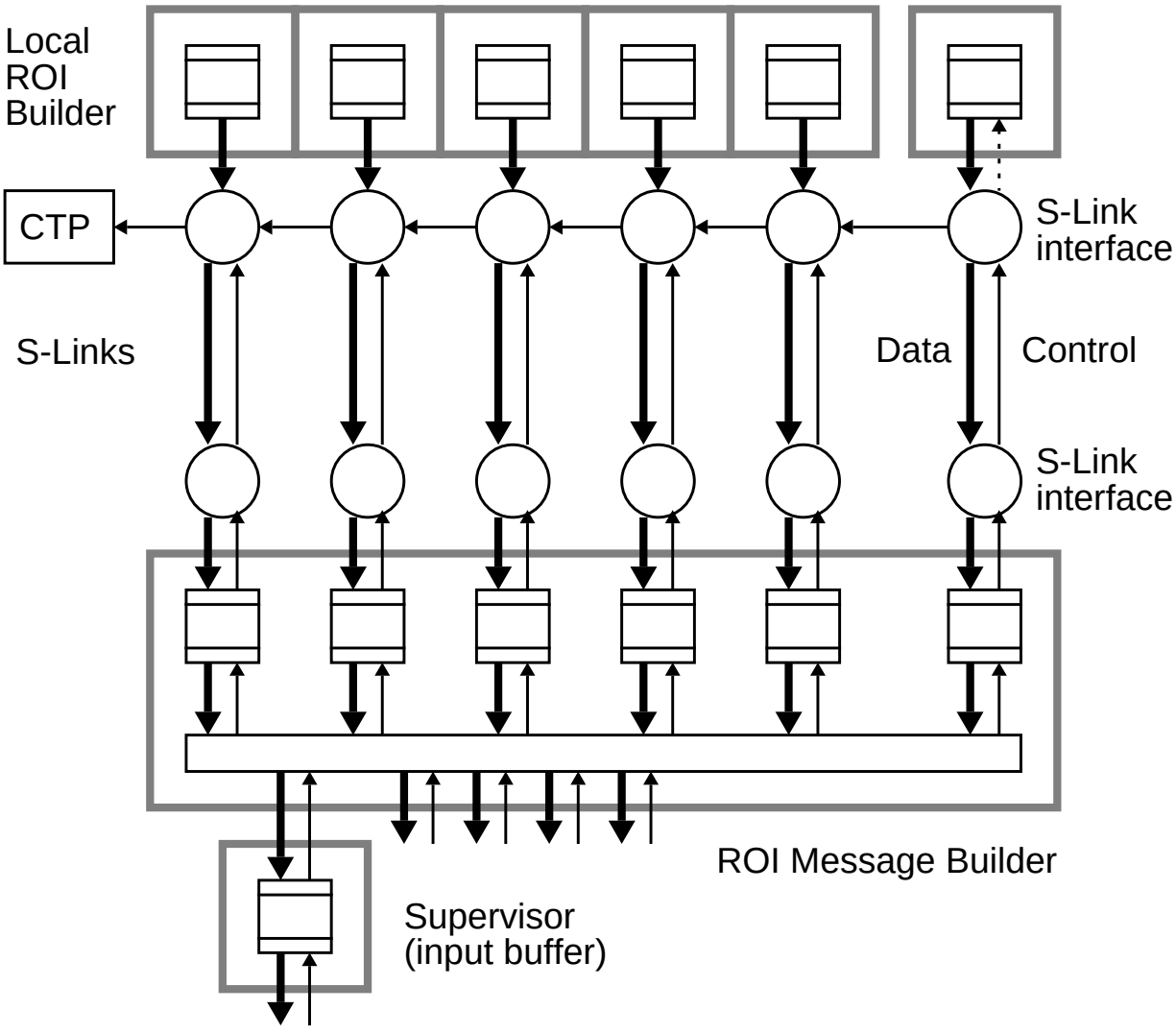
Disabling of LVL1_Accept generation is foreseen in CTP of LVL1

It is more naturally receive this throttle signal from single LVL2 supervisor then from multiple ROD's

Single Steward is an obvious place

As soon as supervisor become distributed (multiple supervisors) "throttle" generation is not a Steward task any more

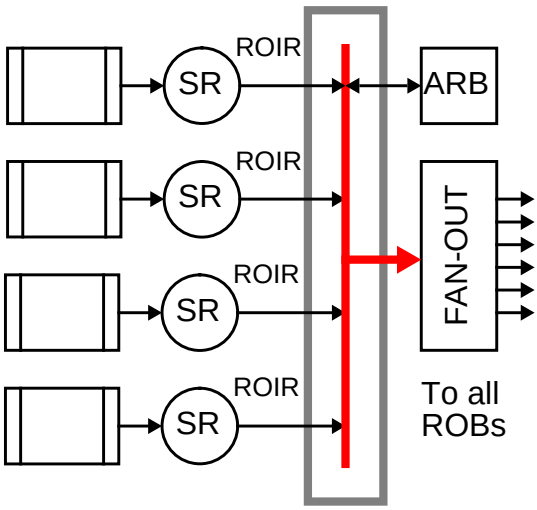
ROI Message Builder may be a natural place to generate Throttle for the LVL1 using data flow control mechanism in S-Link



Broadcast to ROB's

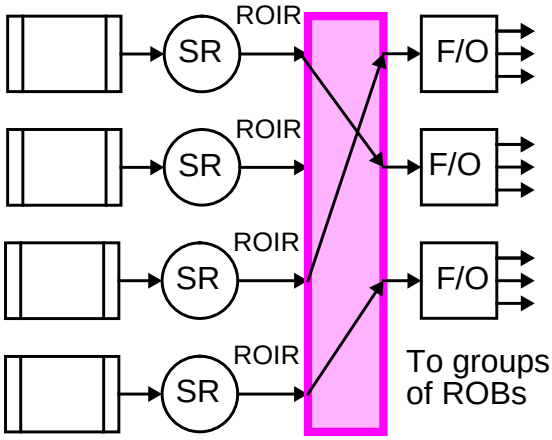
ROIR ("Local Global"/"Push"/"Parallel") 24 Bytes * 100 kHz=2.4 MB/s

Broadcast (single output with fan-out)



BUS/MUX

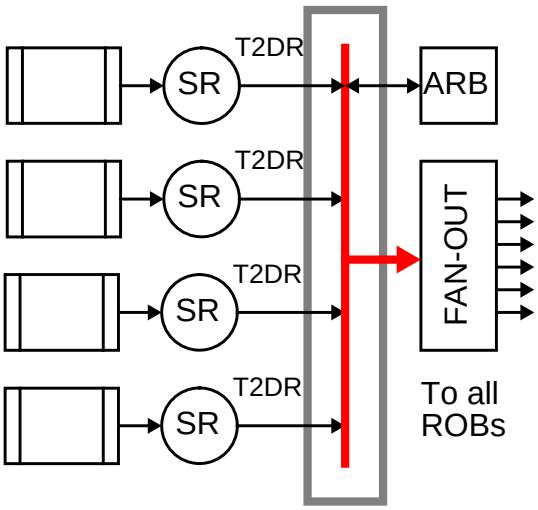
Sent to selected ROBs (switch)



SWITCH

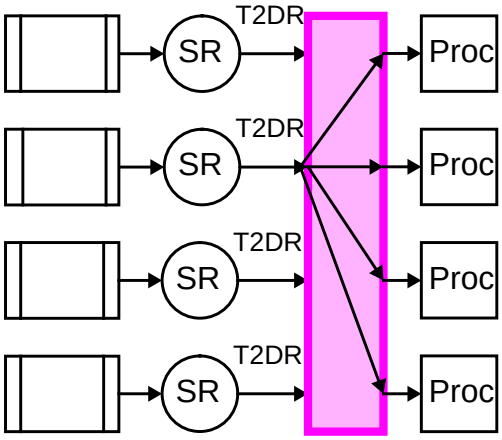
T2DR Broadcast 408 Bytes * 1 kHz = ~0.4 MB/s

"Local Global"/"Push"/"Parallel"



BUS/MUX

"Single Farm"/"Pull"/"Sequential"



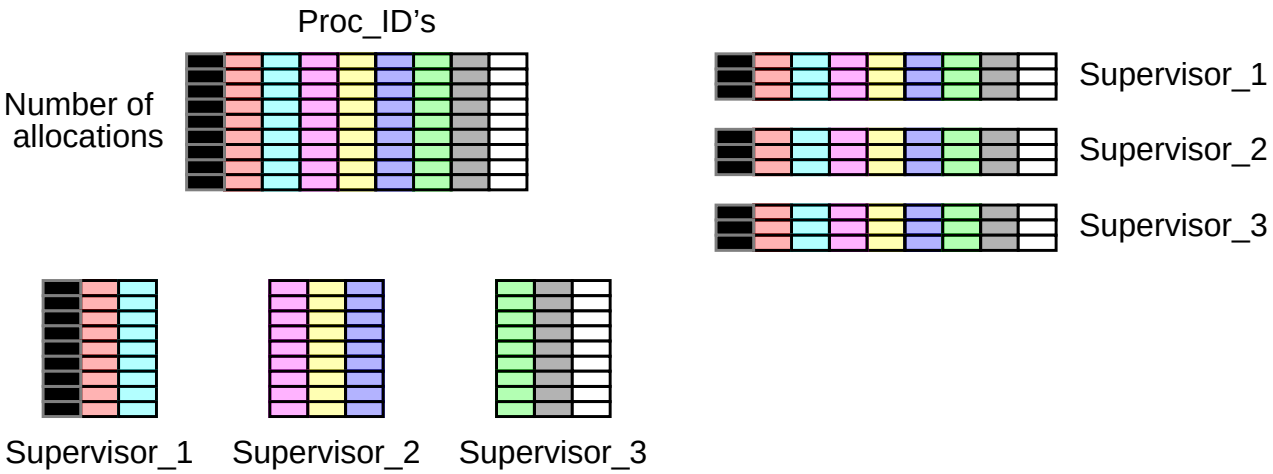
ATM SWITCH

Distributed resource management (2)

Demonstrators - single Steward, all PRoc_ID's in one place

Distributed supervisors - distributed Proc_ID's management (efficiency)

Proc_ID distribution between multiple supervisors:

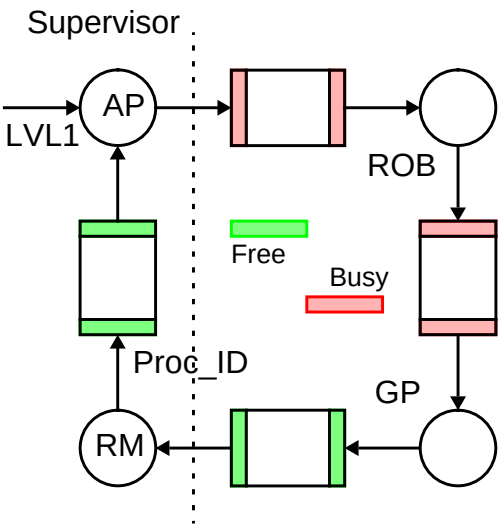


Supervisors may allocate only a group of processors (Proc_ID's)
Number of allocations = Alloc_N

Supervisors may allocate any processor but
Number of allocations = Alloc_N / Number_of_supervisors

Distributed resource management (1)

Processor allocation schemes (J. Vermeulen):



Single allocation:

Free List - return upon response, most ineffective

Multiple allocation without feedback:

Round Robin, Random - next event -> next PRoc_ID

Static list, potential overflow in buffers

Expected Free - return after “assigned” time

Potential overflow in buffers (virtual feedback)

Multiple allocation with feedback:

Updated Round Robin - list update on response

Update algorithm? processor waiting time

Self allocation by processor:

Self Allocation - event request with Proc_ID

Effectiveness - responsibility of processors...

Multiple allocation with feedback (Updated Round Robin) **seems most effective**

Number of allocations (Alloc_N) allowed
- size of the input buffer in processor

Allocation - need feedback from processor and some algorithms

Initially all processors are available for the allocation in the free list

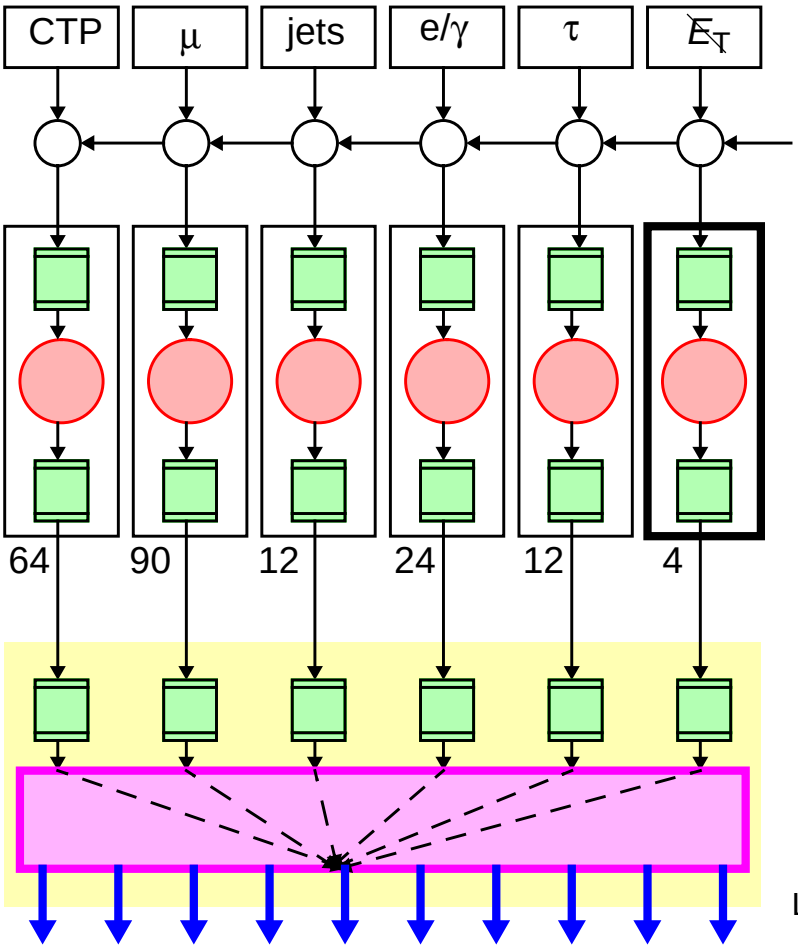
Keep in the free processor list Proc_ID + Alloc_N

After each allocation -> Alloc_N = Alloc_N - 1

Upon response from processor -> Alloc_N = Alloc_N + 1

Proc_ID position in the list is not static
- function of response time?
- how difficult to handle

Input from LVL1 - data distribution



LVL1 trigger

L1_A 100 kHz

Raw LVL1 ROI data

"Local" subtrigger **ROI Builders**

Data compression and formatting

Bytes/L1_A

6 S-Links

ROI Message Builder (Bus/Switch)

LVL1

~200 bytes/event

~2 MB/s per processor

Inputs to Supervisor ROI processors (assume 10)

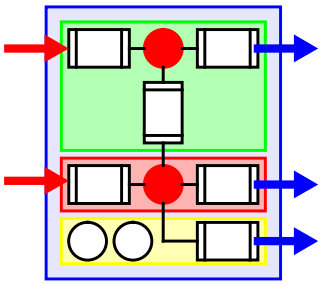
- on every L1_Accept raw LVL1 ROI data are transferred to "local" sub-trigger **ROI Builders** (CTP, muon, ...)

- raw ROI data are compressed according to the algorithm, e.g. - muon - classification of RoIs on a Pt basis - select (max) 16 ROIs with highest Pt thresholds

- compressed ROI data are sent over 6 S-Links to **ROI Message Builder** (Bus/Switch - Custom/ATM/SCI/FC/Raceway)

- **ROI Message Builder** merges ROI fragments from "local" ROI Builders (fragments order?) into complete ROI message and pass it to one of supervisors

Process to processor mapping / distributed processing

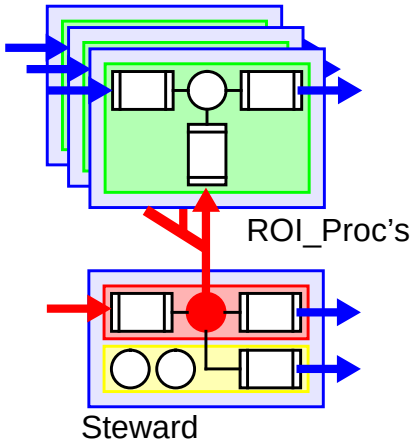


All processes running on one general purpose programmable processor (board) with shared memory communication

I/O bottlenecks, heavy load on processor

Several dedicated processors (Transputer/DSP/FPGA) on the board running in parallel with message exchange via links:

I/O bottlenecks, less load per processor

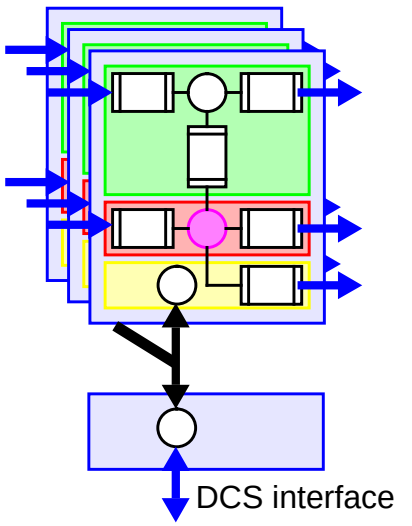


Conceptual design: partially distributed I/O (RIO_Proc), resources handled in one place (Steward)

Heavy load on Steward processor
I/O bottlenecks - VME, input from Globals

Possible (not the best) solution - multiple Stewards - distributed input from Globals, still problems:

Steward - ROI_Proc communication (VME)
Distributed resource management
(Proc_ID's <=> multiple Stewards)



Another possible (may be better) solution - several processors (supervisors) working in parallel on different events, no inter-processor (inter-board) communication, role of Steward - interface to DCS (configuration, monitoring, parameters loading, etc.) and may be interface to LVL3

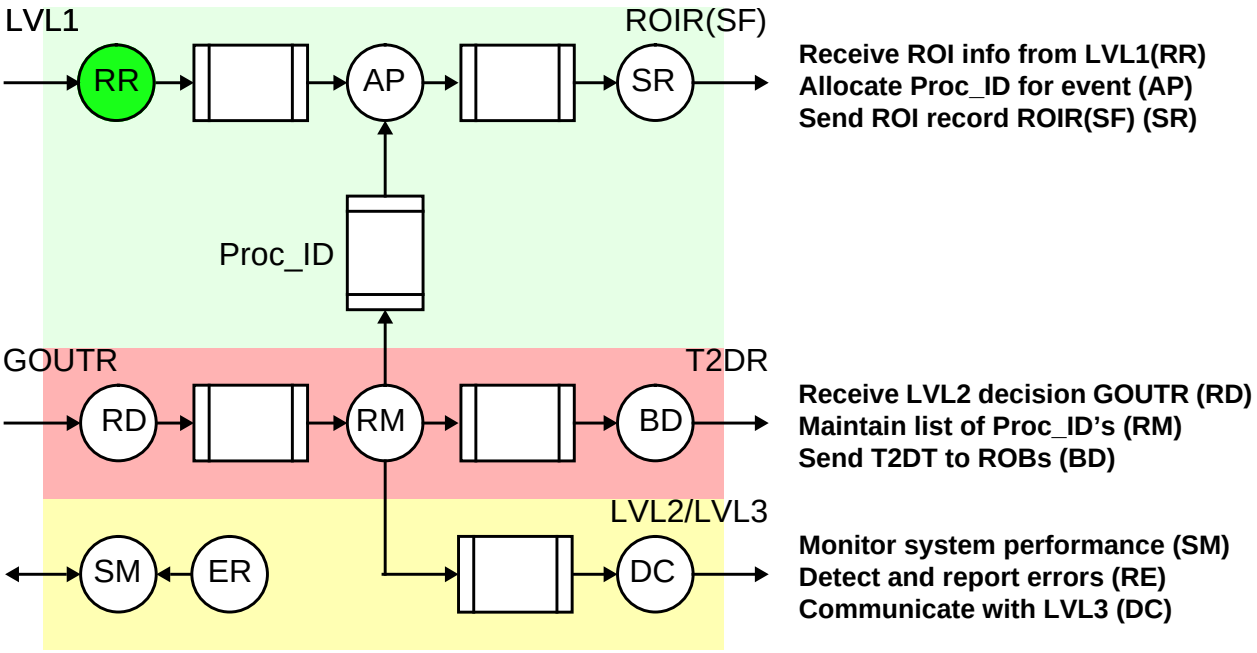
Distributed resource management
(Proc_ID's <=> multiple supervisors)

LVL1 - Supervisors
Supervisors - ROB's
Distributed resource management

Processes in Supervisor

Correspond to the Supervisor tasks

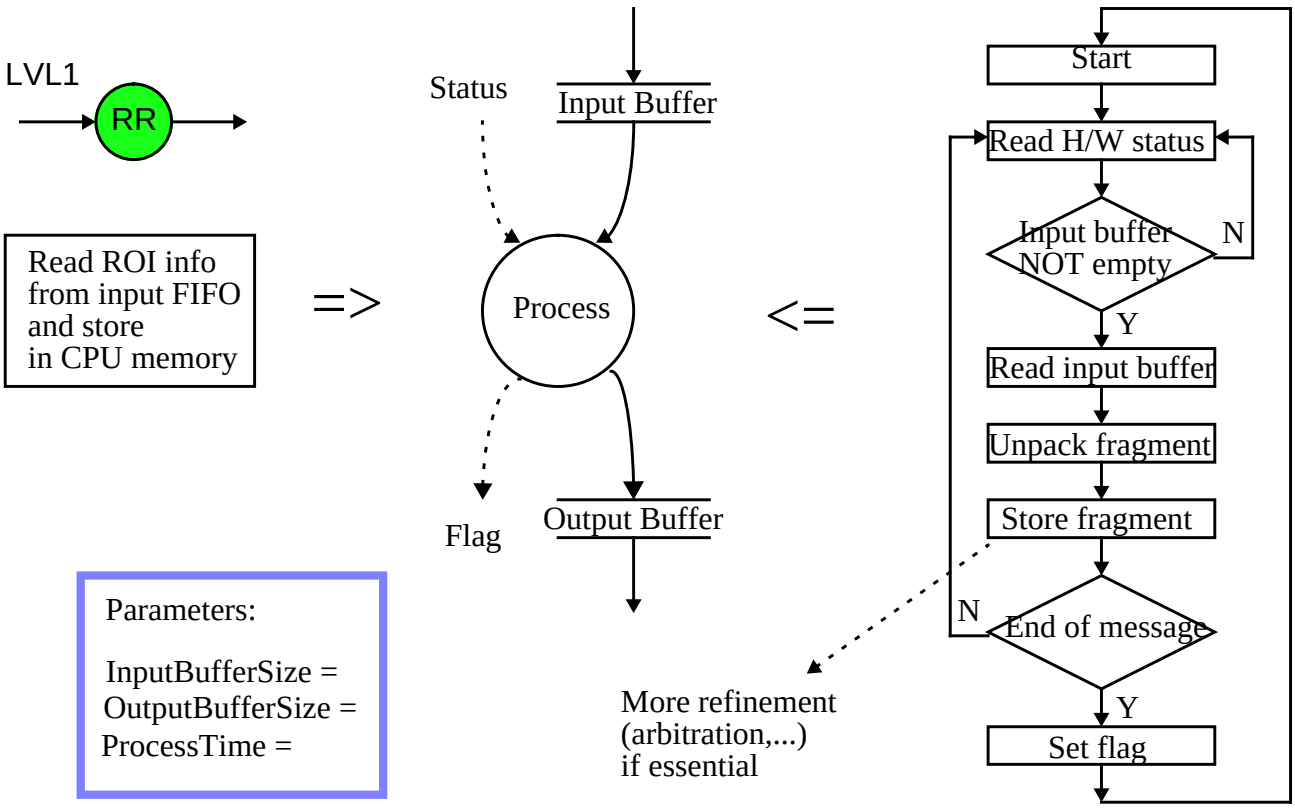
Event independent



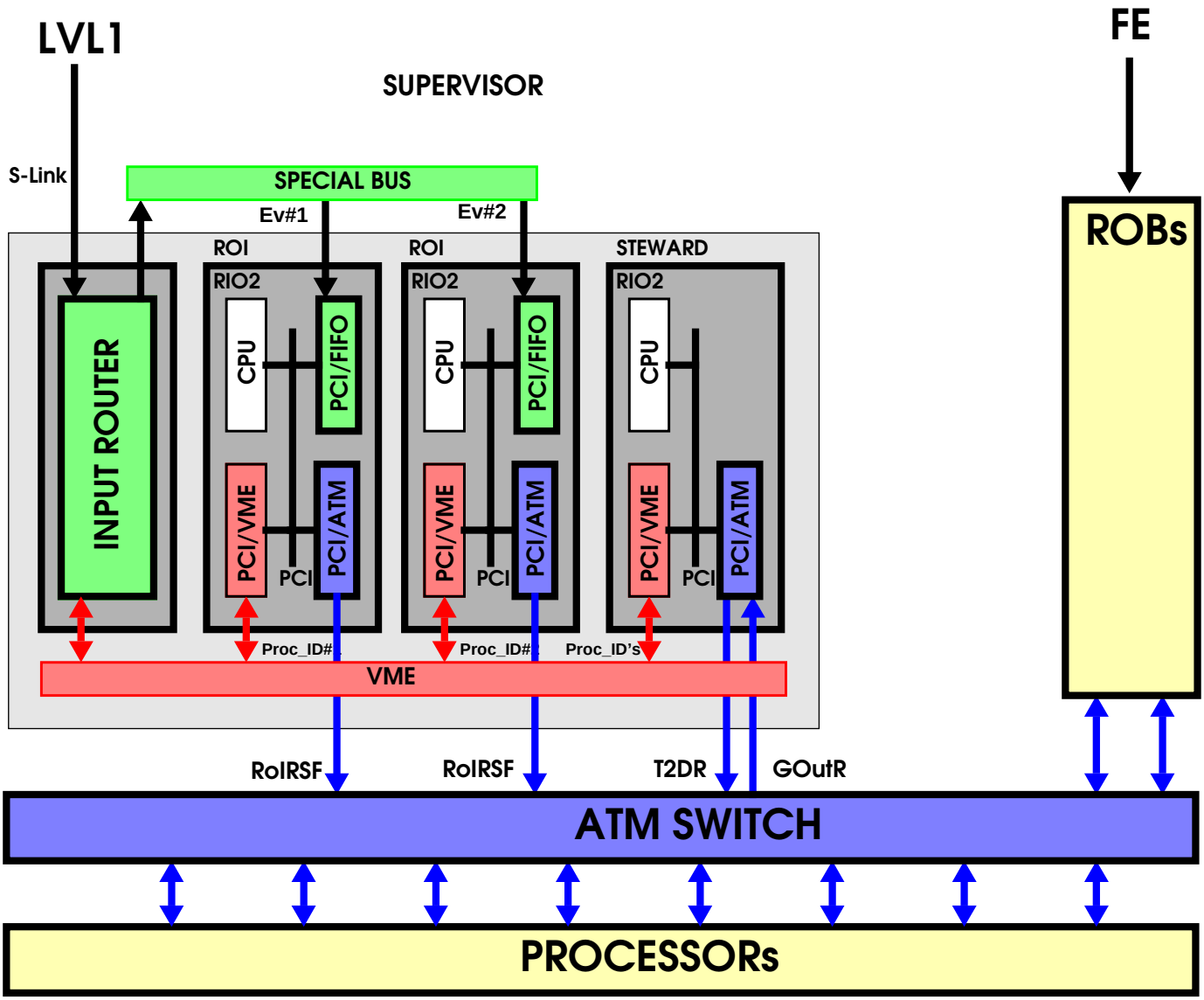
Receive ROI info from LVL1(RR)
Allocate Proc_ID for event (AP)
Send ROI record ROIR(SF) (SR)

Receive LVL2 decision GOUTR (RD)
Maintain list of Proc_ID's (RM)
Send T2DT to ROBs (BD)

Monitor system performance (SM)
Detect and report errors (RE)
Communicate with LVL3 (DC)



Supervisor for DemoC



Custom H/W:

Input Router (S-Link / Special Bus)

Special Bus

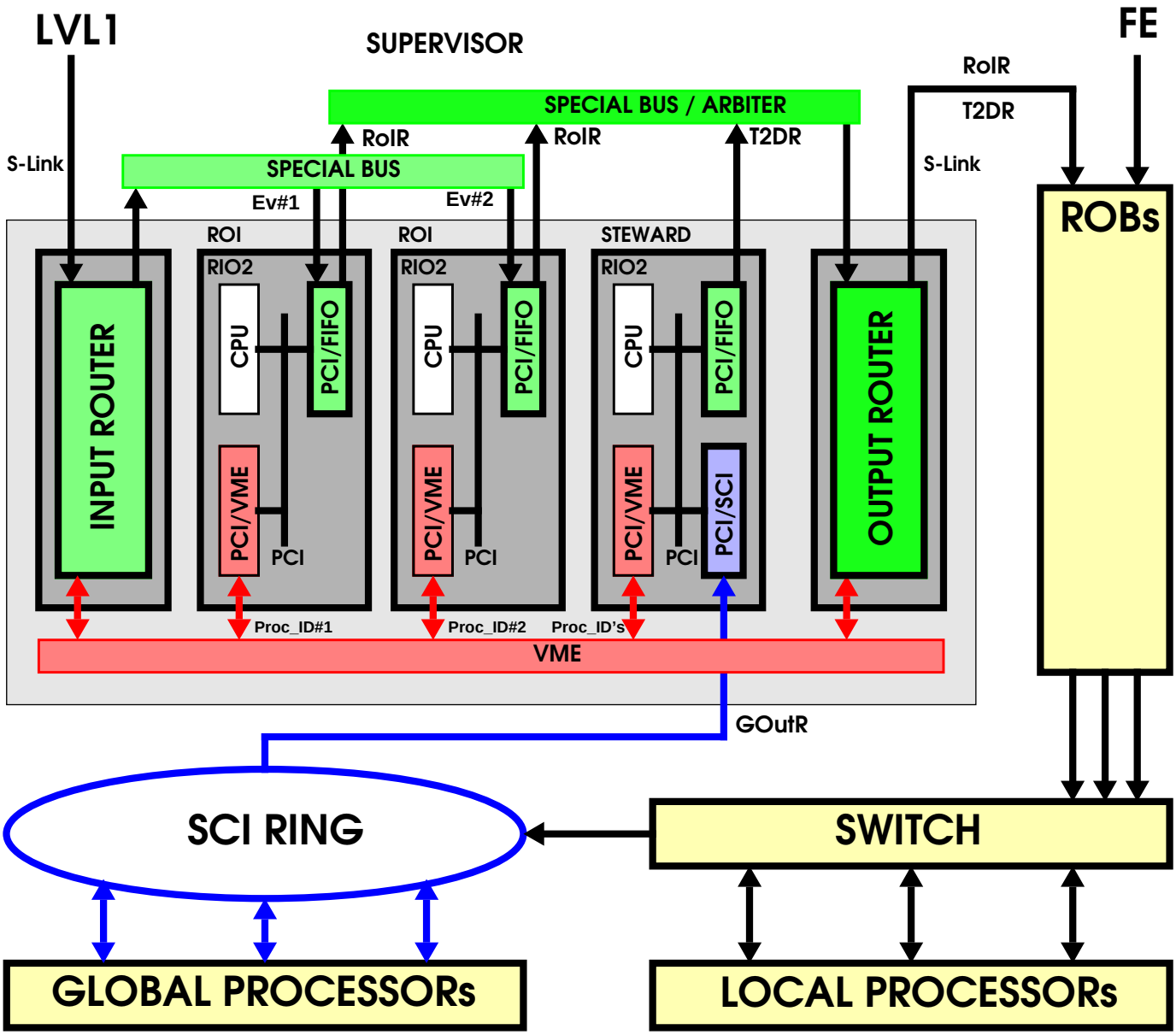
Data I/O PMC (Special Bus / PCI)

Commercial H/W:

CES RIO2 (ROI, STEWARD)

CES ATM PMC

Supervisor for DemoB



Custom H/W:

Input Router (S-Link / Special Bus)

Special Bus, Arbiter

Data I/O PMC (Special Bus / PCI)

Output Router (Special Bus / # S-Link)

Commercial H/W:

CES RIO2 (ROI, STEWARD)

Dolphin PCI/SCI PMC (PC) card

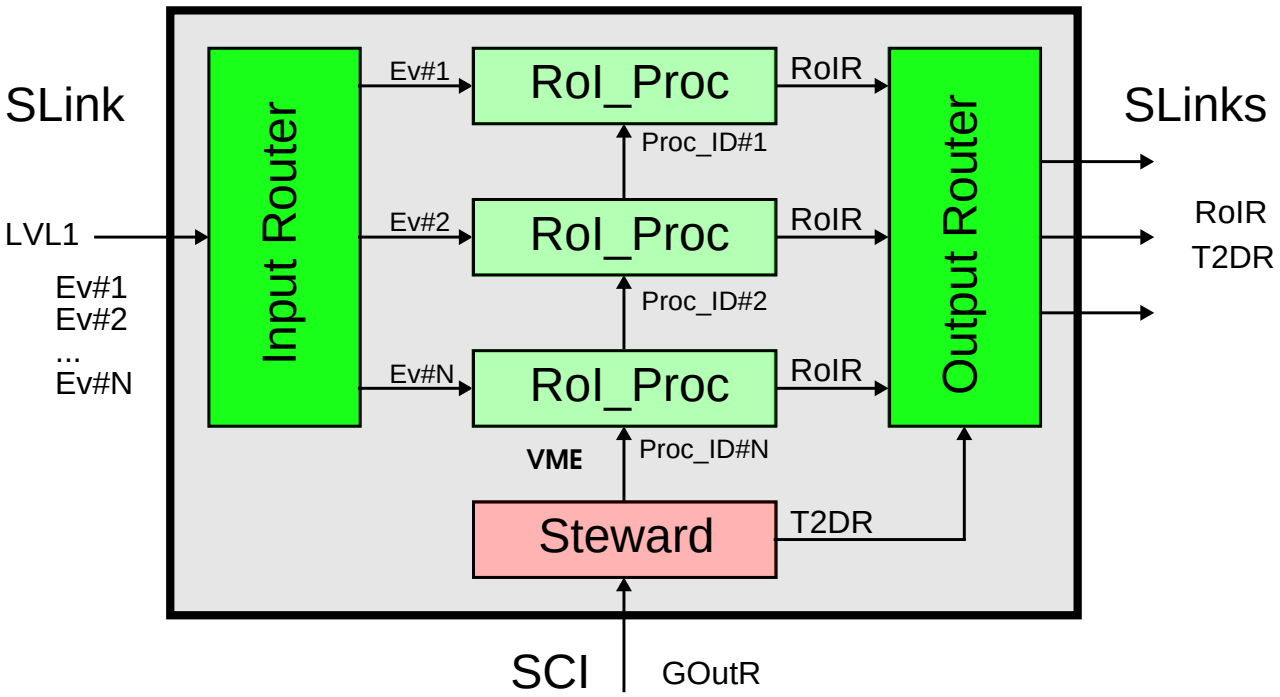
CERN PCI/S-Link PMC

CERN PCI/DS-Link PMC

Level-2 Supervisor: Conceptual design

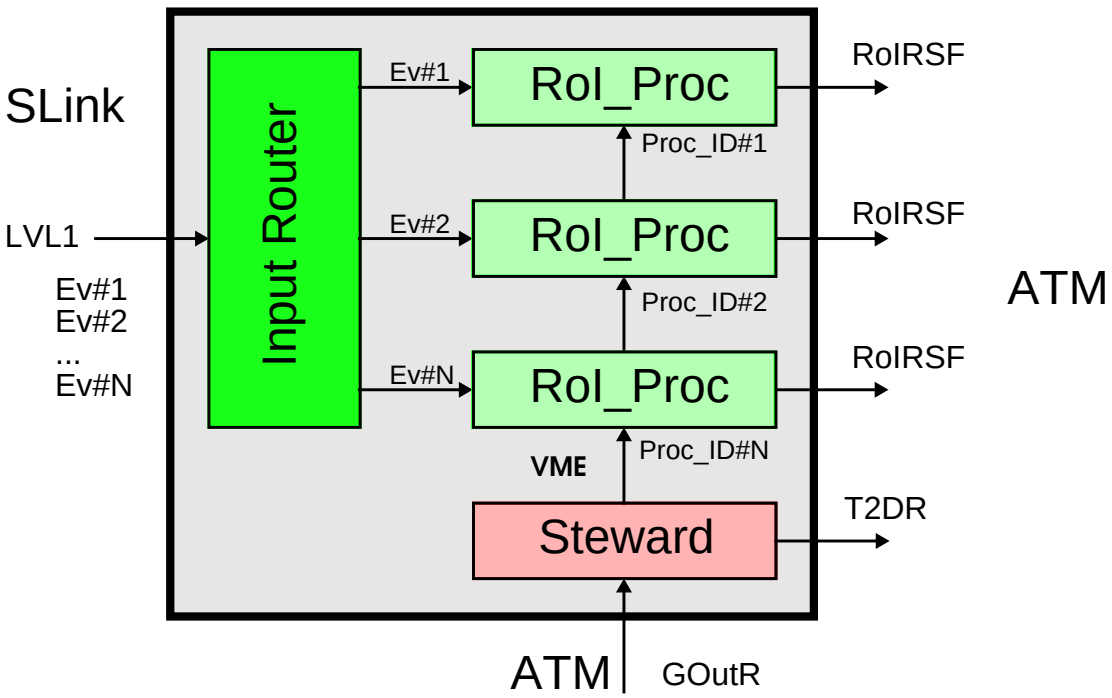
(http://sgi3.hep.anl.gov:8001/l2concep.html)

DemoB



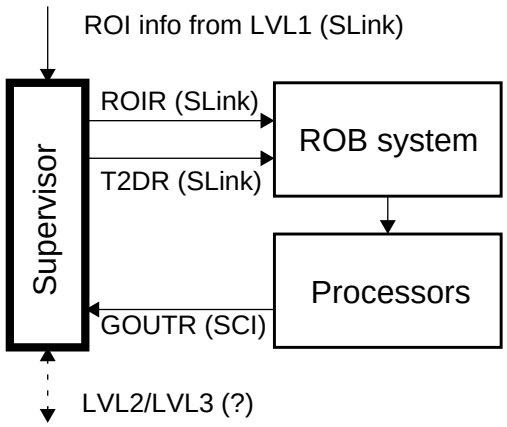
GUIDELINES:
- Commercial Hardware (RIO2, PMC)
- Event Parallel Operations - RoI_Proc

DemoC

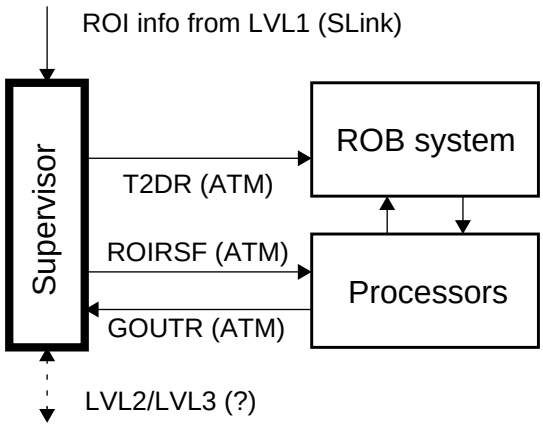


LVL2 trigger system architecture under consideration

“Local Global”/”Push”/“Parallel” (DemoB)



“Single Farm”/”Pull”/“Sequential” (DemoC)



ROB system - ~100 crates housing ROBs. There is about 100 processor clusters.

Supervisor tasks:

- Receive ROI information from the level-1, assign free level-2 processor(s) for this event and send ROI record to the ROB’s (ROIR) or to the selected processor (ROIRSF).
- Receive the level-2 decision (GOUTR) and communicate the results to the ROB’s (T2DR) to either drop the event or keep it for further processing, send a level-2 decision to the level-3 trigger system.
- Maintain lists of level-2 resources (free and allocated).
- Monitor system performance and report error conditions.
- Maintain event processing status in the "Local Global"/"Push”/ “sequential" case - the most complicated for the Supervisor - ???

Cannot be executed on one processor - need tasks distribution

Supervisor modelling

“... The supervisor is a complex system in its own right, so it will be the subject of detailed modelling which goes beyond the scope of LVL2 paper models...”

S.George (RHBNC)
J.R Hubbard (Saclay)
J.C.Vermeulen (NIKHEF)

- Several simple models of the Supervisor exist, they need to be refined (J.Vermeulen - SIMDAQ-C++, N.Madsen - MACRAME, etc...)
  - No complete model
  - Times allocated to each algorithm are guestimates
- Input from demonstrator program - timing, algorithms, bottlenecks
  - Single LVL1 input - potentially overloaded
  - Single Steward - potentially overloaded
  - Global Processor to Steward input - contention (many to one)
  - Steward to ROI_Proc path - VMEbus is slow
- Different ideas - need confirmation
  - Input from LVL1 - data compression
  - Distributed input from LVL1
  - Distributed resource management
  - Data flow control
- “Paper model” of the Supervisor
  - Processes and their interaction
  - Process to processor mapping
  - Model parameters
  - Average processing times, data transfer times etc.

Supervisor Paper Modelling