Introduction

I will present the protocols and methods used to implement network communications in the Data Acquisition System of the KLOE experiment.

The following items will be covered:

- experiment requirements;
- DAQ architecture;
- DAQ process model;
- messaging system and command flow;
- data transmission;
- some performance measurements;
- conclusions.

Experiment Requirements

The KLOE experiment is planned to run at the beginning of 1998 at the Φ-factory DAΦNE in Frascati.

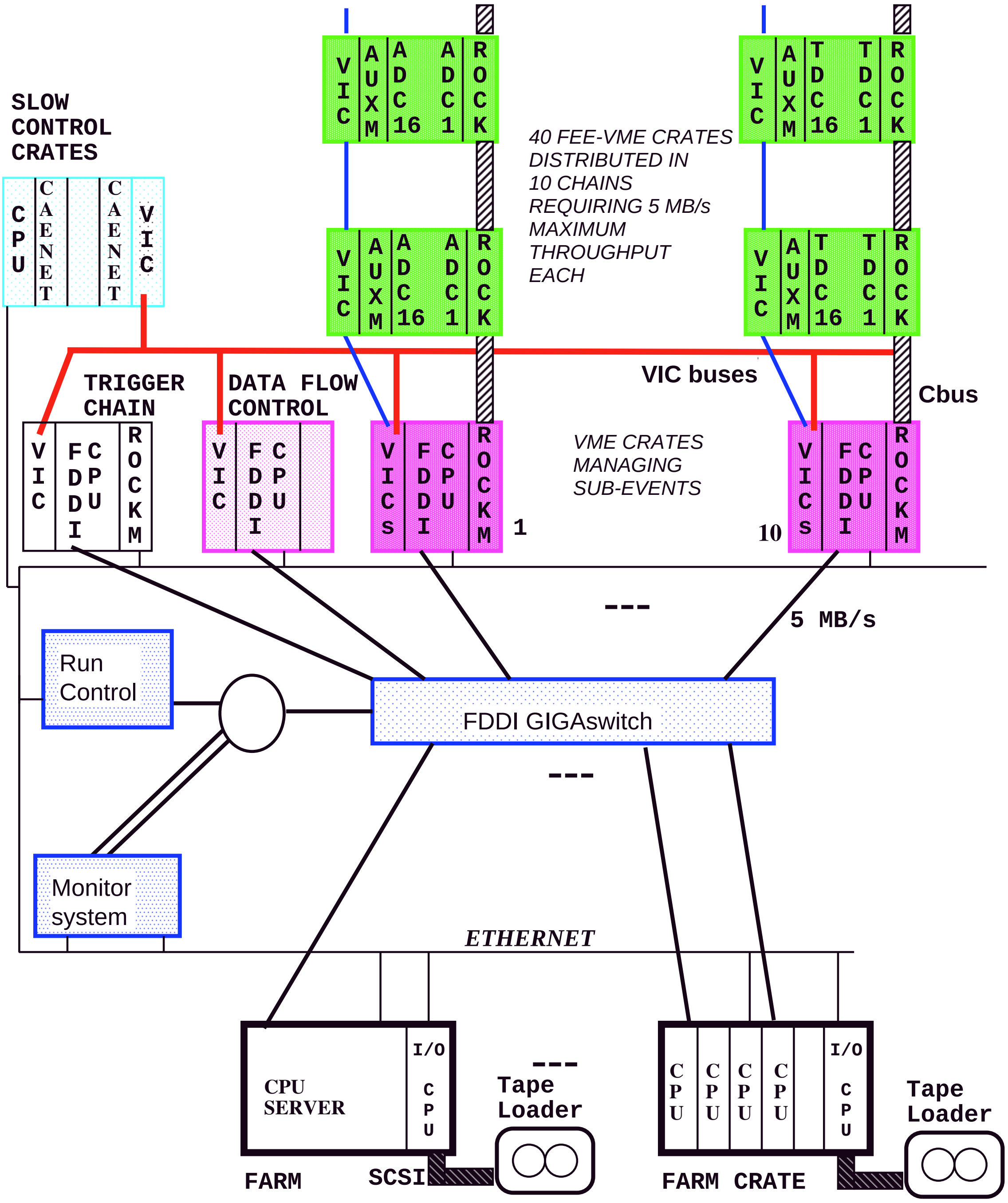
KLOE will perform CP-violation studies at sensitivity of $\mathcal{O}(10^{-4})$

At DAΦNE full luminosity ($10^{33} cm^{-1} sec^{-1}$), the data rate from the detector is estimated at $10^4 events/s$.

The estimated average event size is 5 kbytes $\Rightarrow$

Total throughput $\simeq$ 50MBytes/s.

KLOE DAQ Architecture



Communication protocols

KLOE Network Management, Data Flow Control

link: FDDI

protocol: SNMP

Command distribution, DAQ Monitoring

link: Ethernet & FDDI

protocol: SNMP

Event Data Transmission

link: FDDI

protocol : TCP/IP

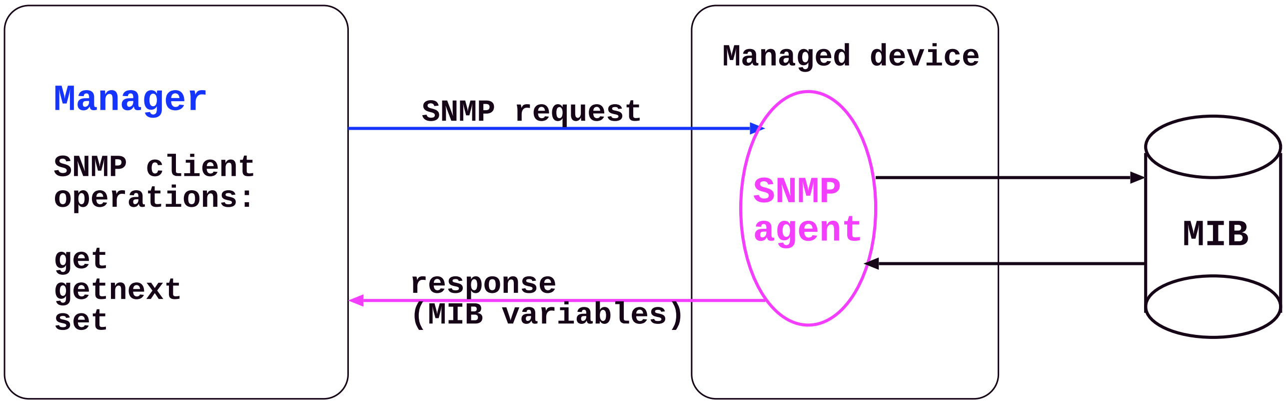
FDDI dedicated links through the Gigaswitch.

Ethernet local network interconnected through bridges and isolated by the rest of the laboratory network.

SNMP protocol

Simple Network Management Protocol

- Standard for administration of non uniform interconnected networks.
- An agent (server) on each managed network device communicates with a manager (client) .
- Information about standard network objects stored in a virtual database, the Manager Information Base .
- The MIB (tree structure) can be extended to hold private information.
- The agent sw is public domain (e.g. by CMU) and can be modified to implement special features.



Process Template

All KLOE Data Acquisition Processes conform to a pre-defined skeleton (UNIDAQ like), that implements inter-process communication.

Initialization:

Msg Queue creation;

Shared Memory subscription;

Space allocation for variables (user and system vars);

User Init .

Main Loop:

Process Event ;

Process Command (user and system commands);

Idle time .

Interrupt handling:

Extract commands from MsgQ;

Message System

Use standard UNIX facilities (message queues, shared memories and signals) for message passing inside the local node.

A private specialized SNMP Agent, the Command Server , runs on each node and takes care of communication between the local processes and the rest of the network.

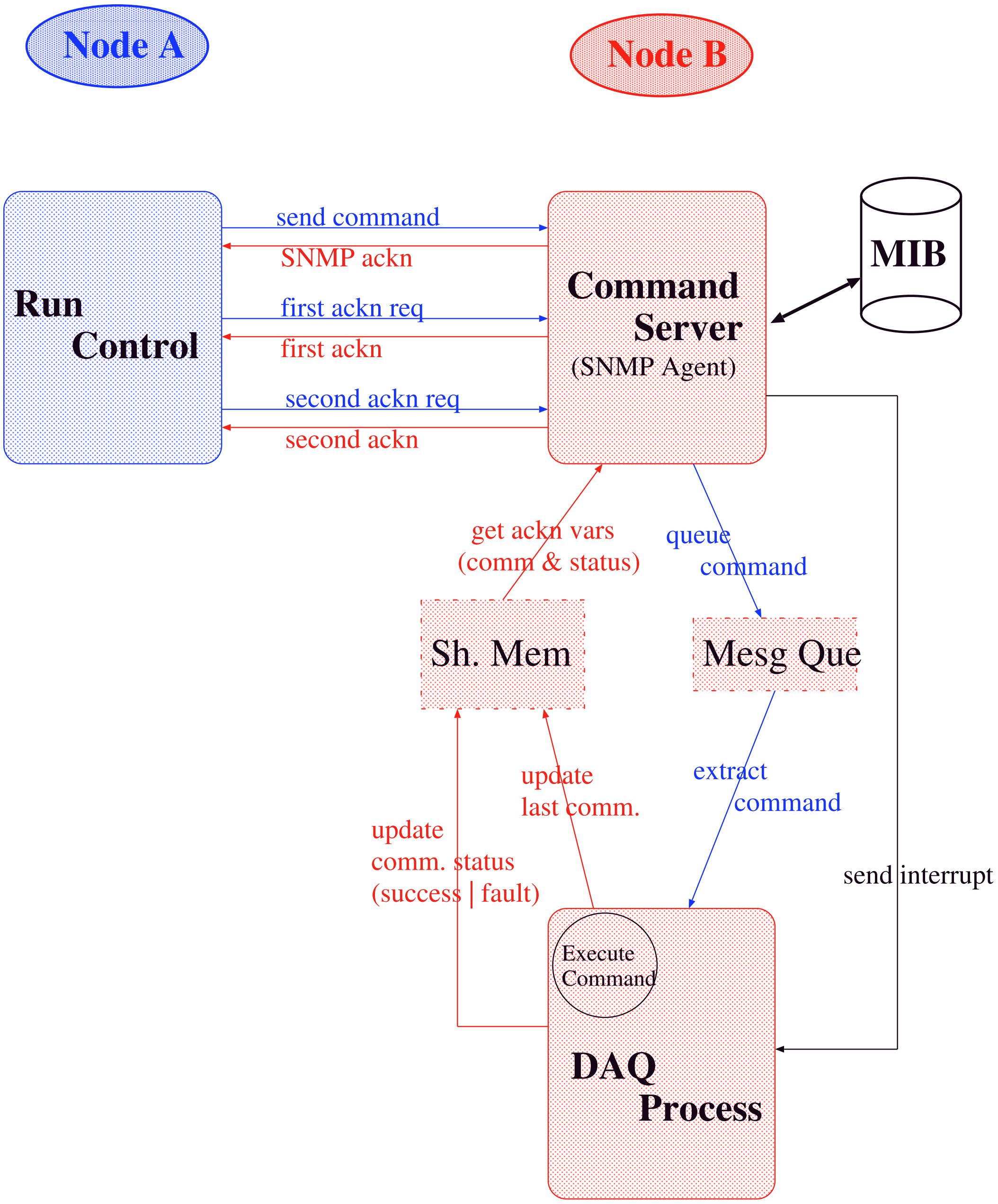
A private extension to the standard MIB allows to manage the KLOE DAQ processes as remote devices and to make their variables visible to SNMP clients.

Double level of acknowledgement for remote command execution:

1) the command has been received;

2) the command has been executed, with success or failure.

Message System



Performance measurements (on HP9000/735):
send a remote command and get complete ackn: < 30 ms;
send a command on local node: < 1.2 ms.

Command Distribution

The defined command syntax allows:

- to multicast messages to a list of recipients,
- to force command execution by the processes in the list to be sequential or parallel,
- to broadcast a command to all DAQ processes running on a node.

Sequential execution :
the command is sent to the next process in the list only if the previous one has acknowledged successful completion.
Parallel execution :
the command is sent to all processes in the list and a global acknowledgement is expected.

Data Flow Control

The Data Flow Control is in charge of the assignment sub-event $\leftrightarrow$ farm CPU, optimizing the traffic in the GIGA switch and the load of farm processors.

The DFC builds a network map and creates an assignment table, distributed to the VME-CPU through the VIC shared memory.

SNMP traps from the farm processors signal the DFC that a new table has to be created (e.g. farm input queue status changed).

Data Transmission

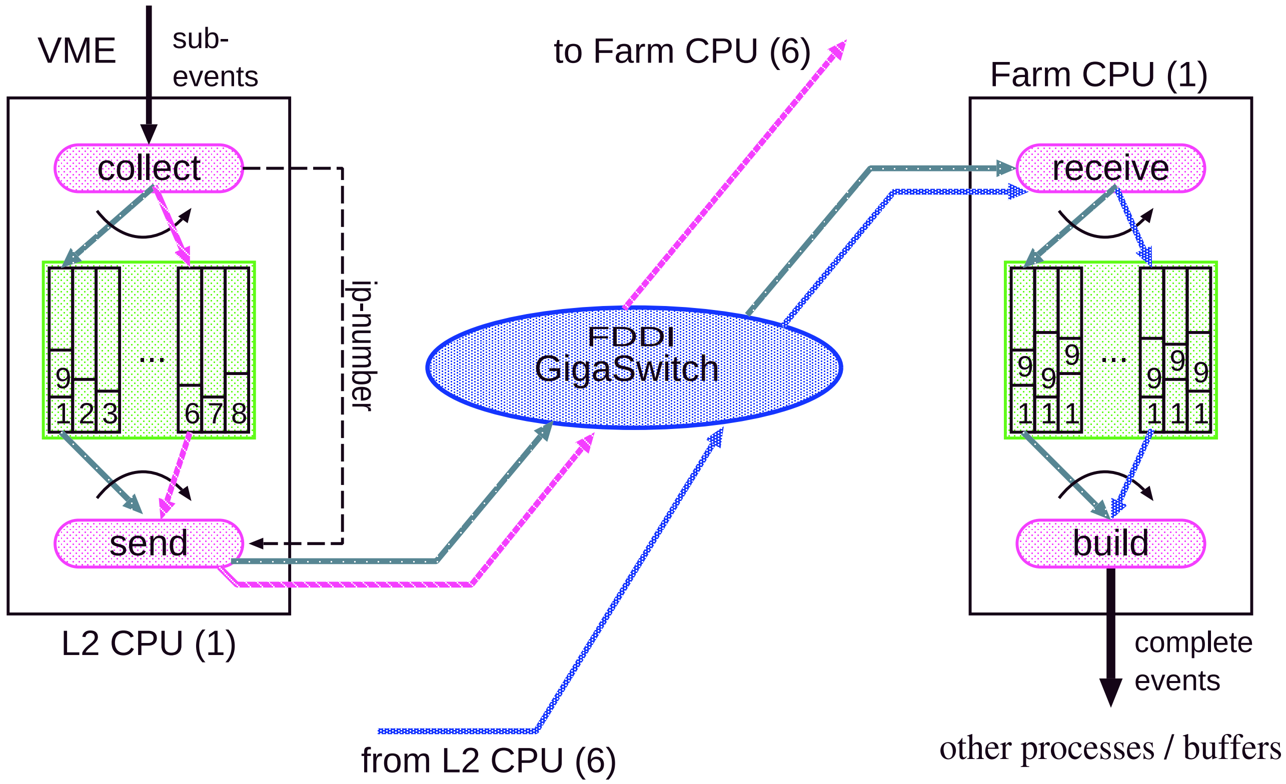
Four processes, besides DFC, are directly involved in the transmission of Event Data:

collector & sender at VME-CPU level,
receiver & event builder at farm level.

A Multiple Circular Buffer acts as a parallel pipe between two processes, reducing data copies when possible.

- Collector: reads packets from the ROCKM FIFO; builds a string of fixed number of sub-events in a specific stream of the Circ Buff, according to DFC destination assignment.
- Sender opens TCP/IP connections, extracts packets from Circ Buff and sends to destinations.
- Receiver accepts TCP/IP connections, gets sub-event packets and fills a stream in the Circ Buff for each connection.
- Event builder extracts sub-events (now belonging to the same trigger nr.) from the Circ Buff and builds complete events.

Event Data Flow



Performance tests

- Optimize TCP parameters for event building.
- Test configuration with N senders and N receivers.
- Test the FDDI GIGAswitch under load.
- Test parallel event transfer and event building using TCP/IP.

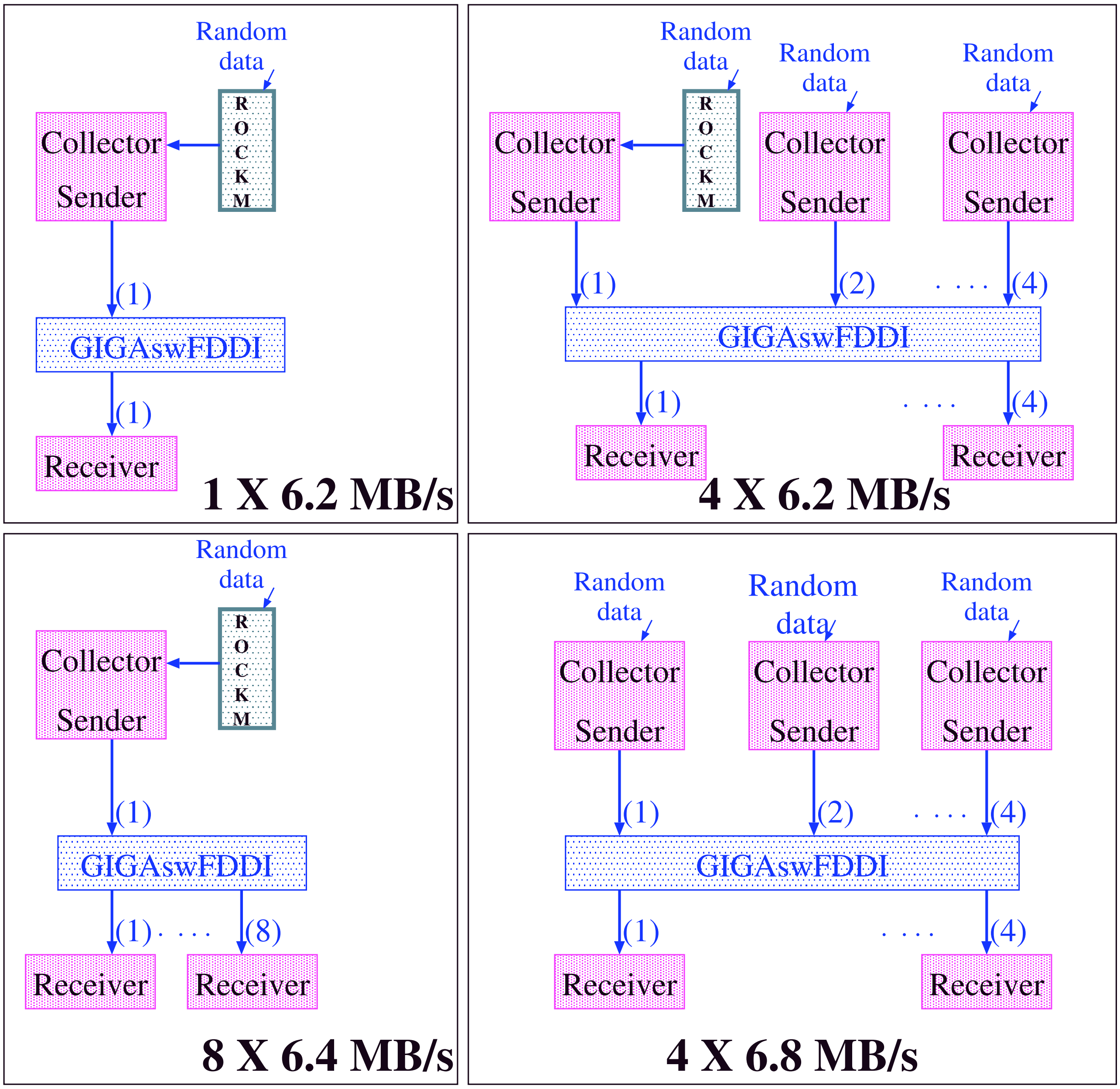
Tests with 10 CPUs (HP/HP-UX, Alpha/OSF1, PowerPC/LynxOS) connected to the GIGAswitch.

Different TCP/IP implementations use different algorithms to avoid data losses (e.g. slow down when insufficient window size is foreseen); the use of similar protocol implementation at both sides should improve throughput.

One CPU available at test time with collector reading sub-events from the ROCKM FIFO.
Collectors reading random data used for $N \times N$ tests.

Sub-event size = 0.5 KB, 100 sub-events per stream.

Test Results



TCP send buffer size = 8 KB gives best results,
16 KB is sufficently good.

The measured throughput is compatible with our requirements!

Conclusions

SNMP can usefully be used as remote communication protocol, not only for network management;

however, the modularity of our software easily allows to try different implementations for message system protocol.

The event data transmission with TCP/IP over FDDI and GIGAswitch works fine.

The software for event transmission based on Multiple Circ Buff has proved to be very efficient and fast.