Data
Acquisition
Backbone

Jörn Adamczewski-Musch, Hans G.Essel, Nikolaus Kurz, Sergey Linev
GSI, Experiment Electronics: Data Processing group

DAQ overview

Acquisition backbone

DABC

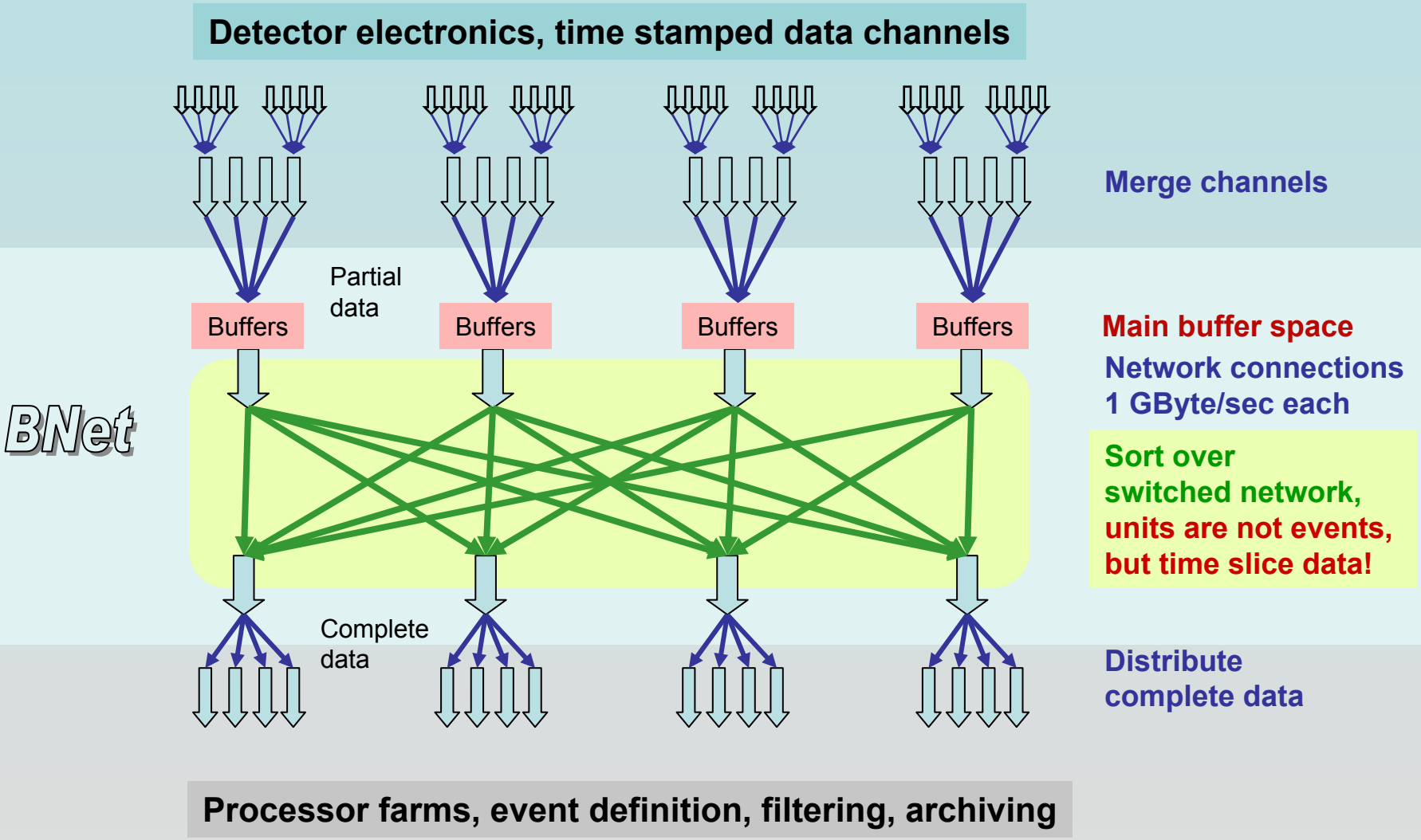
Performance

Use cases

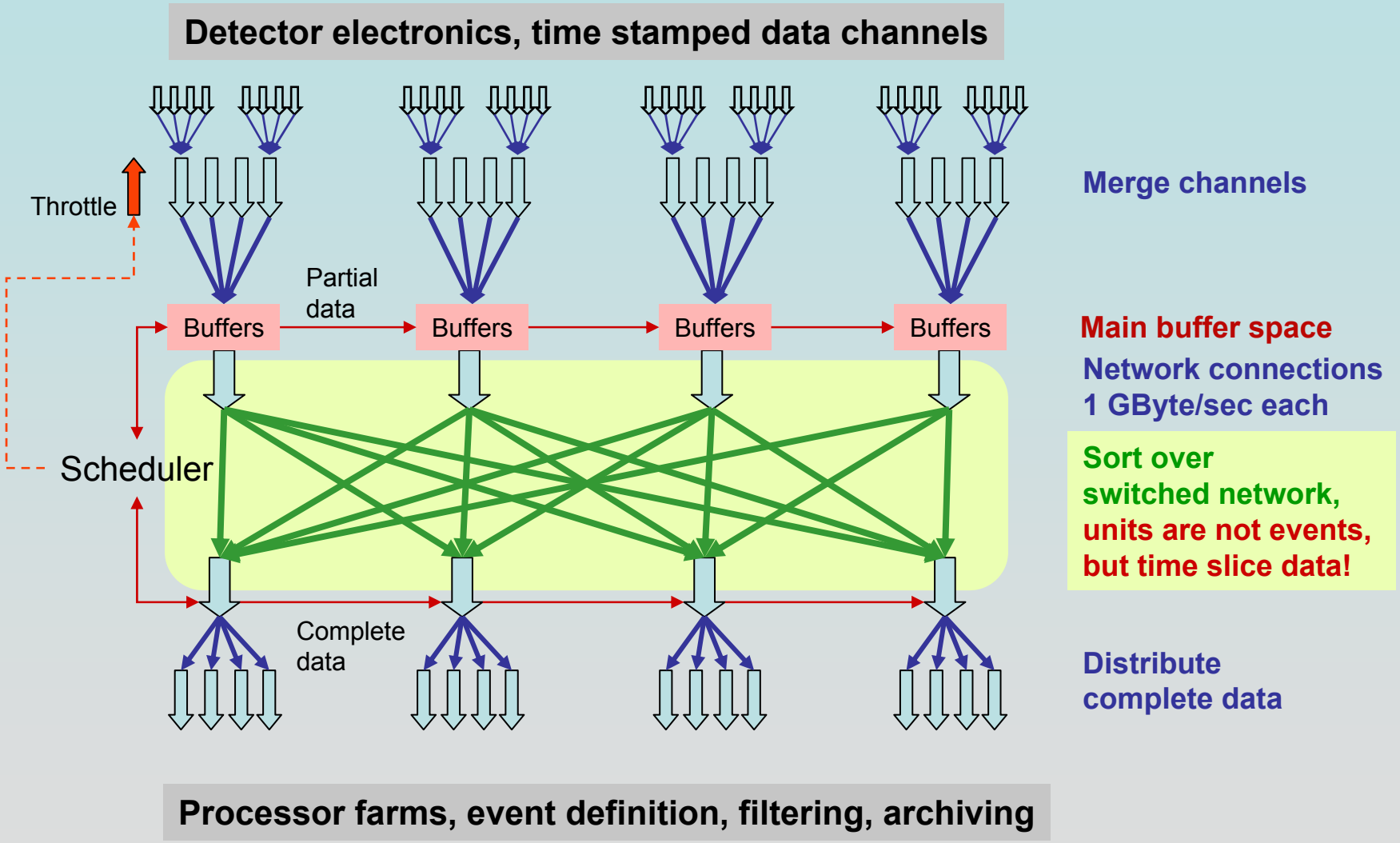
For discussion

Work supported by EU RP6 project JRA1 FutureDAQ RII3-CT-2004-506078

Data flow principle

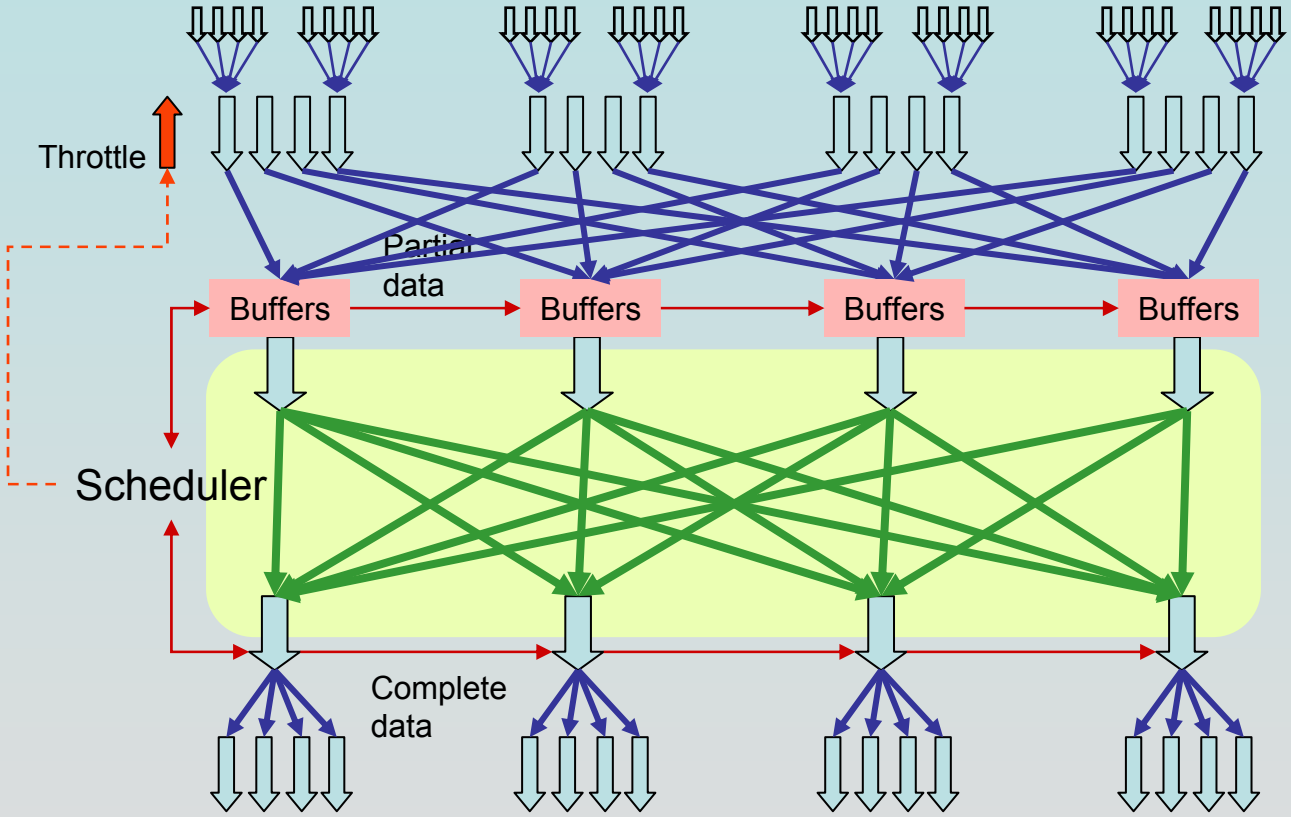


Data flow principle



Data flow principle

Detector electronics, time stamped data channels



Merge channels

optimize for partial readout

Main buffer space

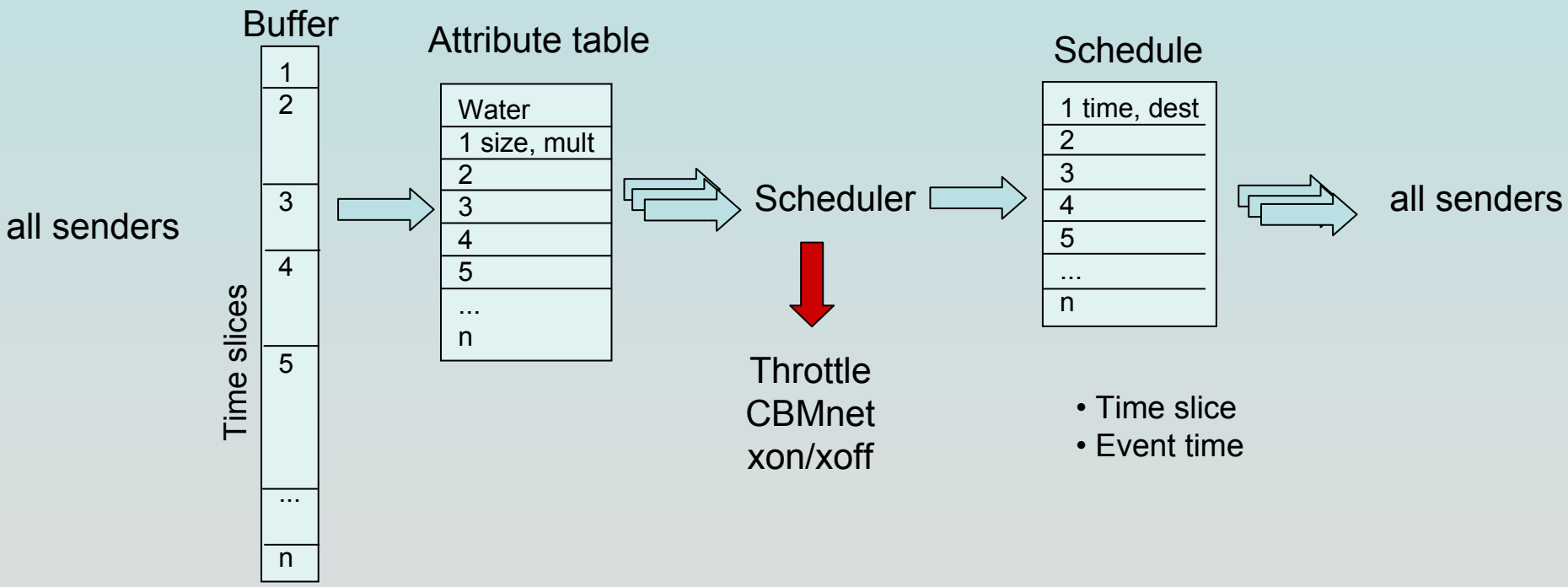
Network connections 1 GByte/sec each

Sort over switched network, units are not events, but time slice data!

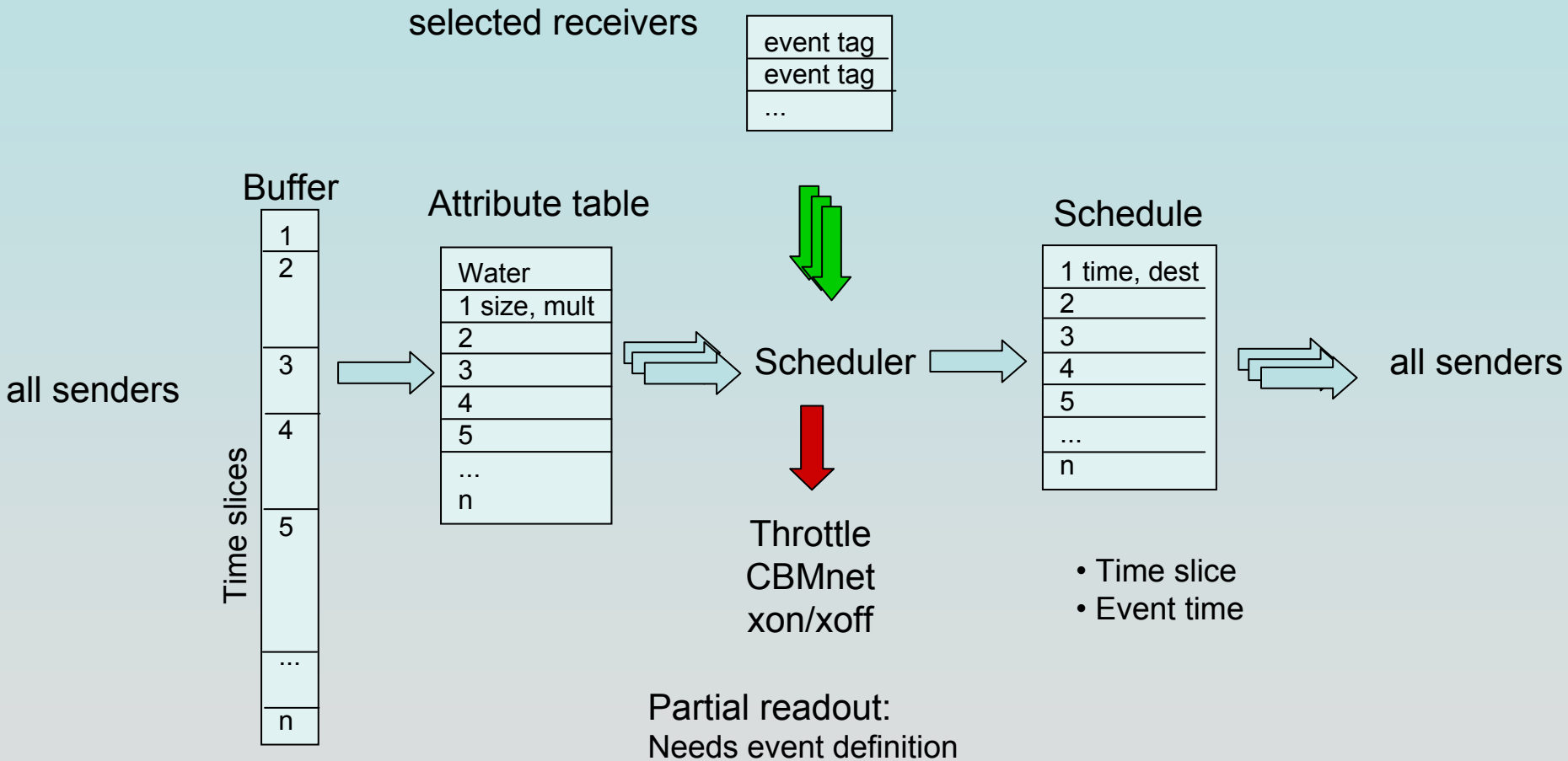
Distribute complete data

Processor farms, event definition, filtering, archiving

Scheduling


Buffer
Time slices
1
2
3
4
5
...
n
all senders
Attribute table
Water
1 size, mult
2
3
4
5
...
n
Scheduler
Throttle
CBMnet
xon/xoff
Schedule
1 time, dest
2
3
4
5
...
n
• Time slice
• Event time
all senders

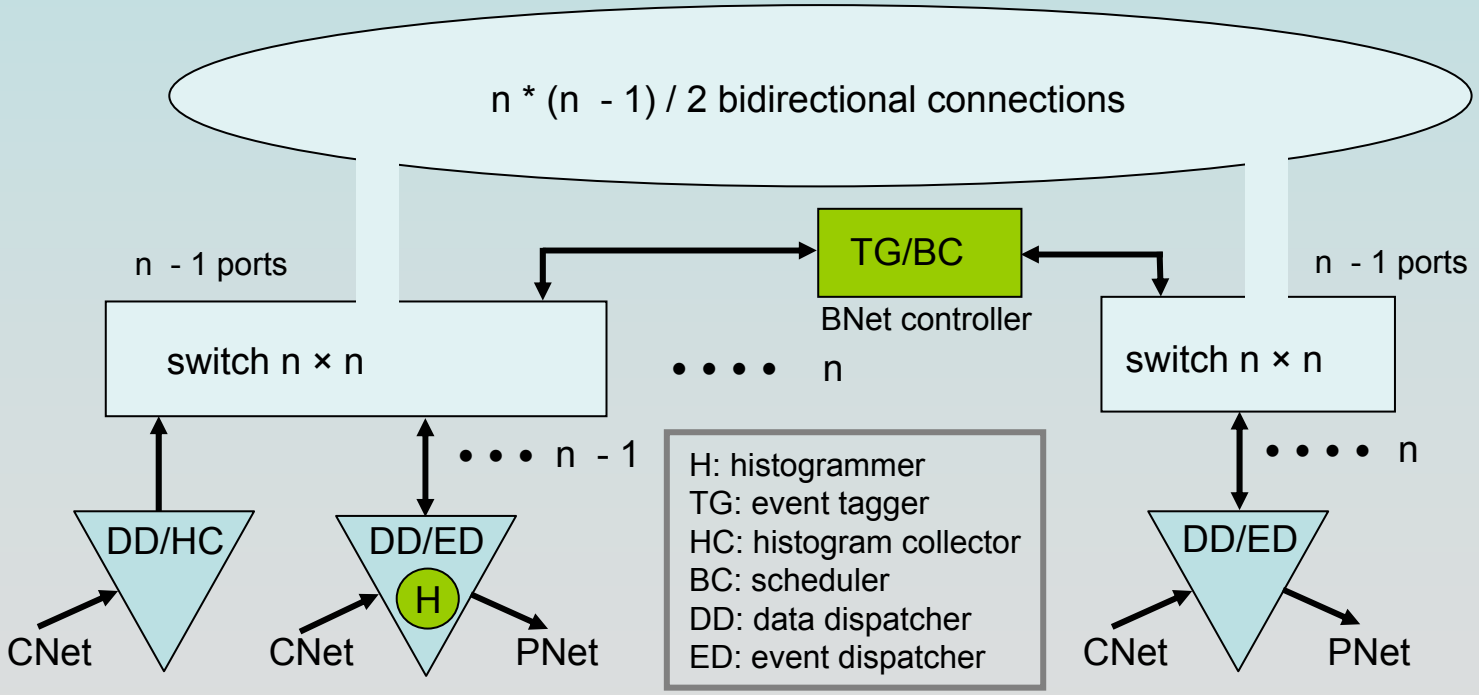
Scheduling



Partial readout:
Needs event definition
Feed back selected event times to scheduler
More granular schedule based on Δtime instead of slices

Bi-directional links (all senders are receivers)

144 double-nodes need **12 12x12** switches with **66** bi-directional connections
Data transfer in time slices (determined by epochs or event streams)
Bandwidth reserved for histogramming and scheduling (traffic shaping)
1 GByte/s point to point



Real Time 2005

What's about

What is a Data Acquisition Backbone Core ?

What is a Data Acquisition Backbone Core ?

A **Data Acquisition** is:

- Digitizing **signals** from detectors
- Identify data by triggers or time stamps
- Combine data of events or time intervals
- Filtering, **archiving**, controls, monitoring

What is a Data Acquisition Backbone Core ?

A **Data Acquisition** is:
- Digitizing signals from detectors
- Identify data by triggers or time stamps
- Combine data of events or time intervals
- Filtering, archiving, controls, monitoring

A **Data Acquisition Backbone** is:
- Combine data of events or time intervals over fast networks
- Connect multiple data sources and multiple data processors
- Flow control

What is a Data Acquisition Backbone Core ?

A **Data Acquisition** is:
- Digitizing signals from detectors
- Identify data by triggers or time stamps
- Combine data of events or time intervals
- Filtering, archiving, controls, monitoring

A **Data Acquisition Backbone** is:
- Combine data of events or time intervals over fast networks
- Connect multiple data sources and multiple data processors
- Flow control

A **Data Acquisition Backbone Core** is:
- A C++ written software running on Linux
- Data sources supported by plug-ins
- Data processors supported by plug-ins
- Scheduling supported by plug-ins
- Fast network event building (InfiniBand, GE, other)
- Filtering, archiving, monitoring by plug-ins
- Controls

Software packages developed:

1. Simulation with SystemC (flow control, scheduling)
   - Meta data on data network

2. Real dataflow core (round robin, with/without sychronization)
   - Linux, InfiniBand, GB Ethernet
   - Simulates data sources

3. **Data Acquisition Backbone Core DABC (includes dataflow core)**
   - **Controls, Configuration, Monitoring, GUI ...**
   - **Real data sources**
   - **General purpose DAQ framework**

Recall motivation for DABC

2004 → EU RP6 project JRA1 FutureDAQ*
2004 → CBM FutureDAQ for FAIR

Intermediate demonstrator

1996 → MBS future
50 installations at GSI,
50 external
http://daq.gsi.de

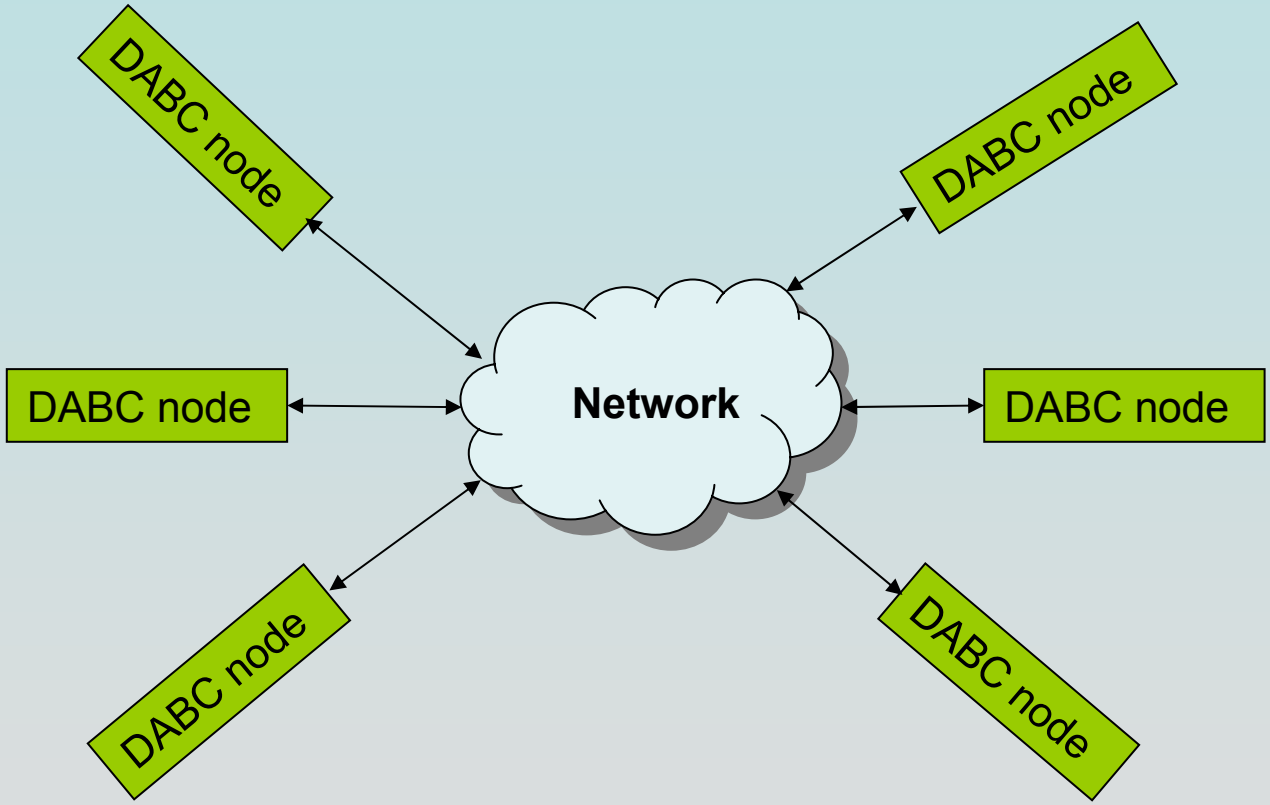
Requirements
- build events over fast networks
- handle triggered or triggerless front-ends
- process time stamped data streams
- provide data flow control (to front-ends)
- connect (nearly) any front-ends
- provide interfaces to plug-in application codes
- connect MBS readout or collector nodes
- be controllable by several controls frameworks

Use cases
- Detector tests
- FE equipment tests
- Data transport
- Time distribution
- Switched event building
- Software evaluation
- MBS event builder
- General purpose DAQ

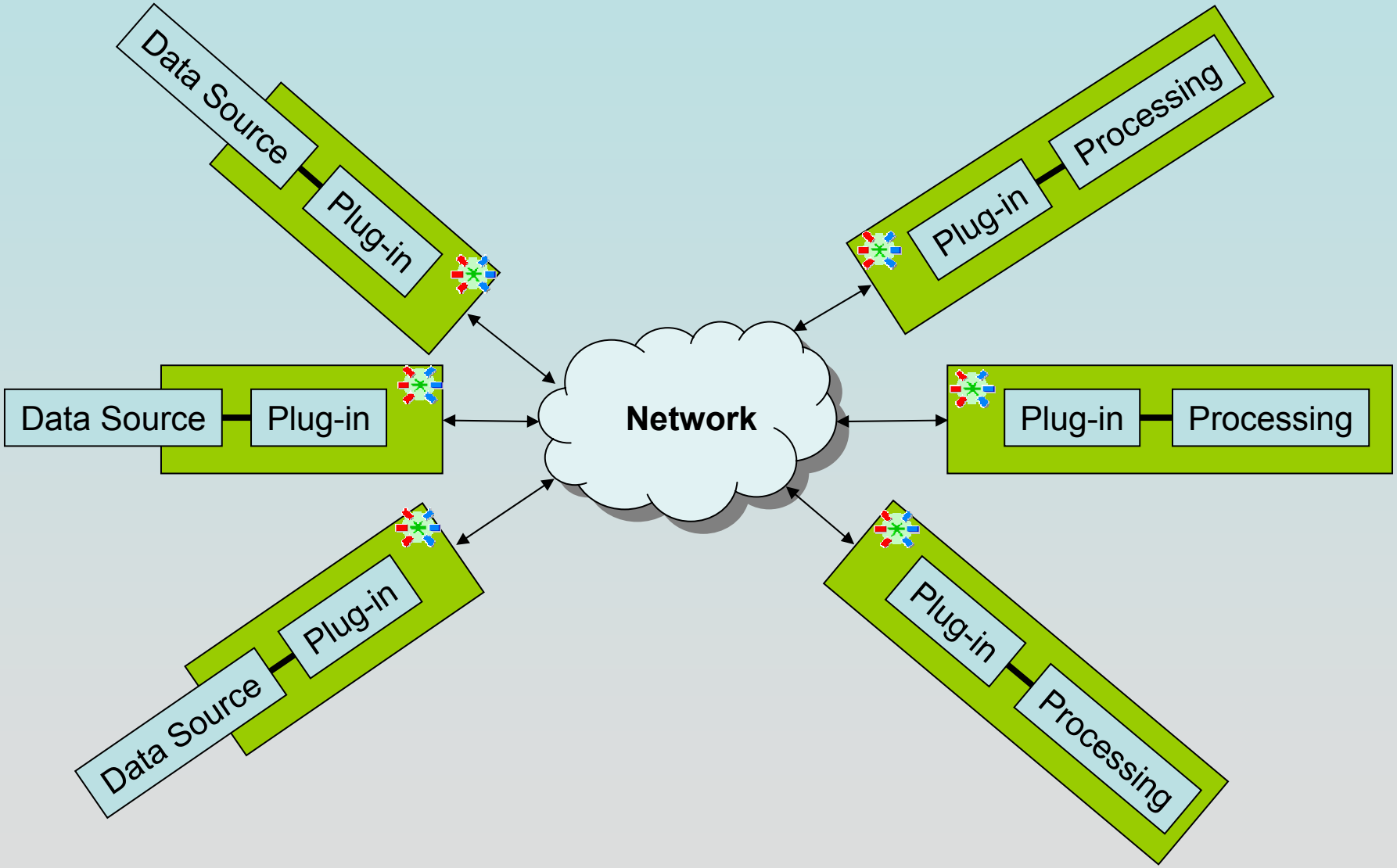
Data
Acquisition
Backbone
Core

* RII3-CT-2004-506078

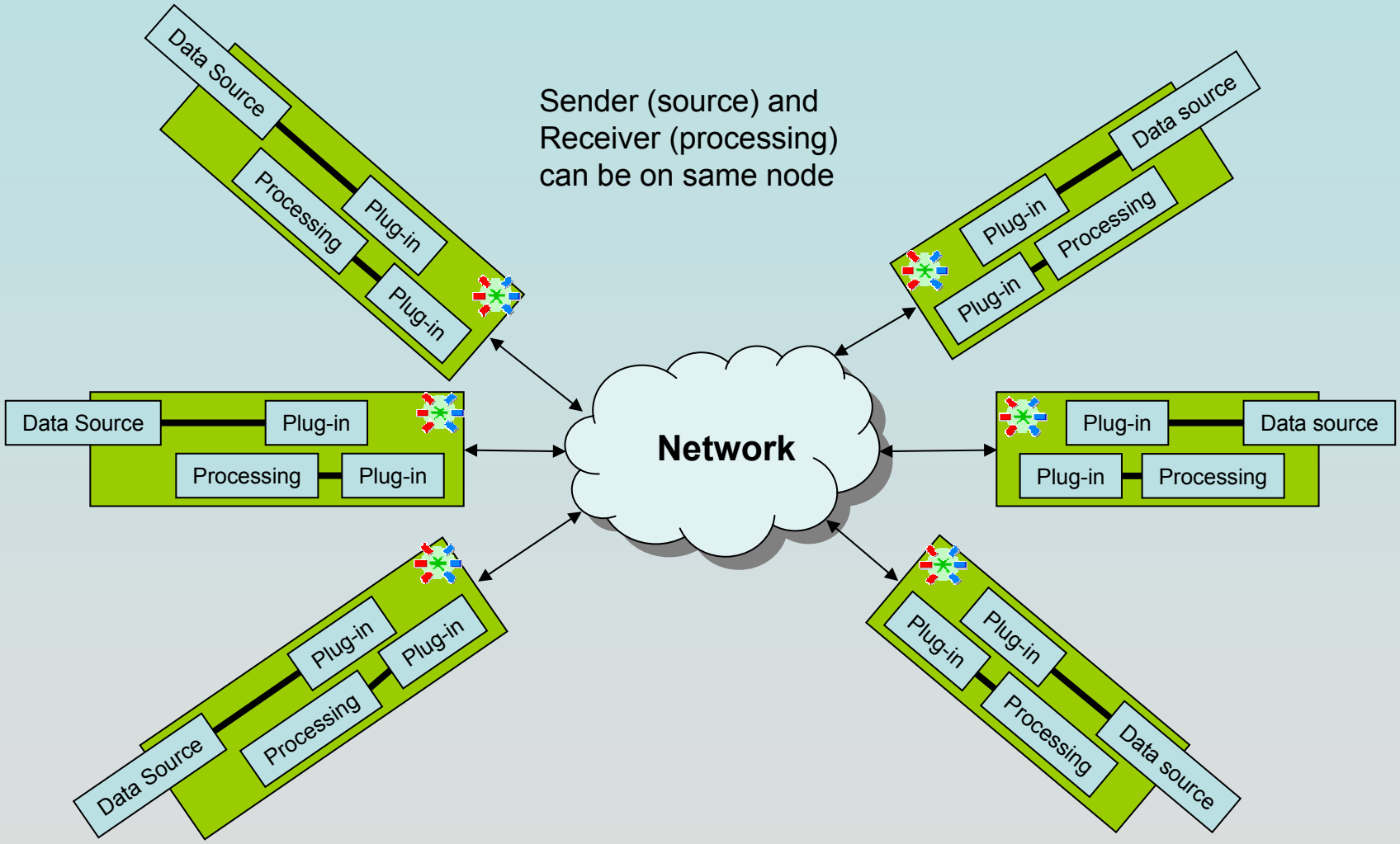
Data Acquisition Backbone Core



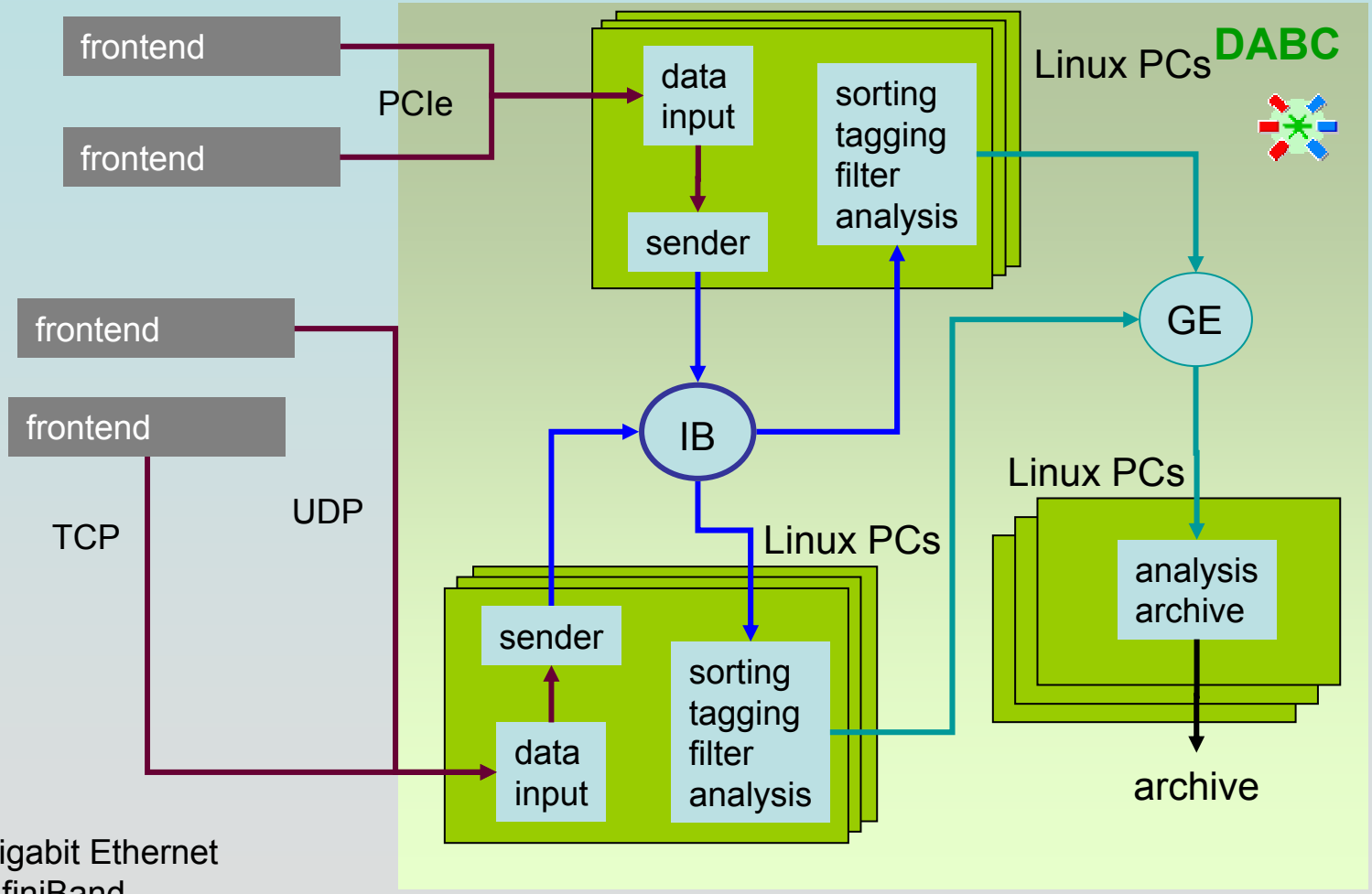
Data Acquisition Backbone Core



Data Acquisition Backbone Core



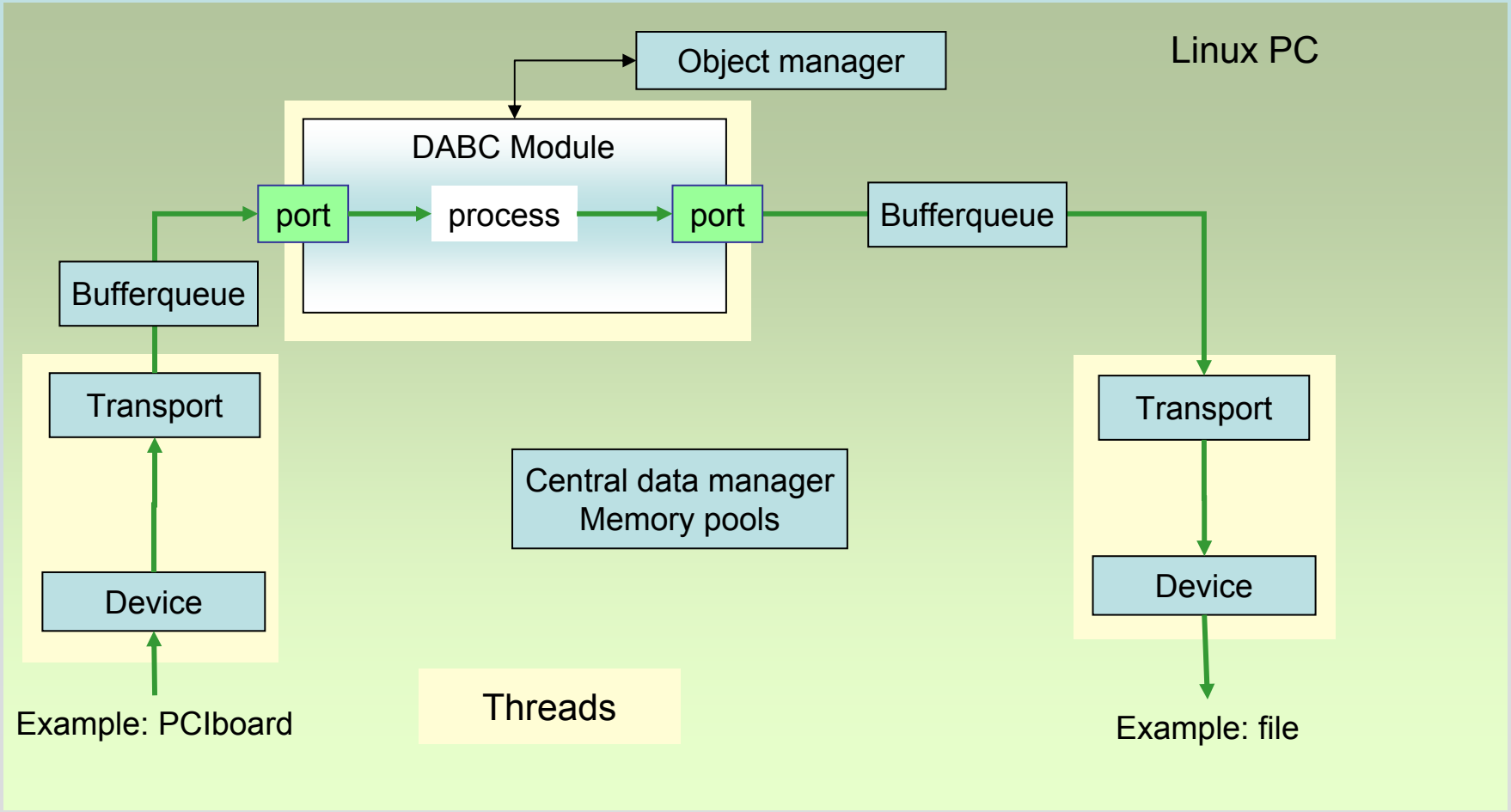
DABC design: global overview



GE: Gigabit Ethernet
IB: InfiniBand

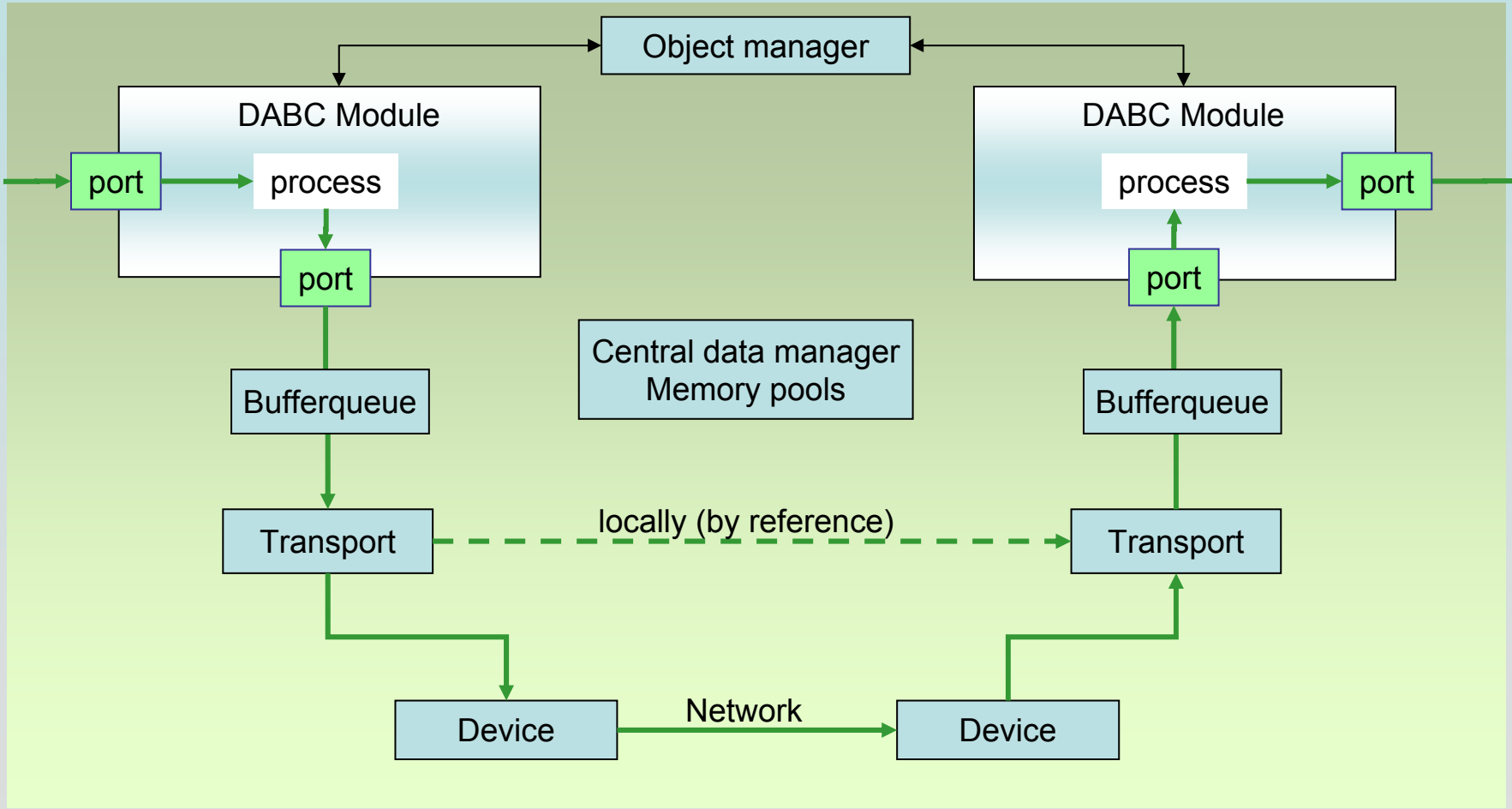
DABC design: Software objects of data flow engine

A *module* processes data of one or several data streams.
Data streams propagate through ***ports***, which are connected by ***transports*** and ***devices***

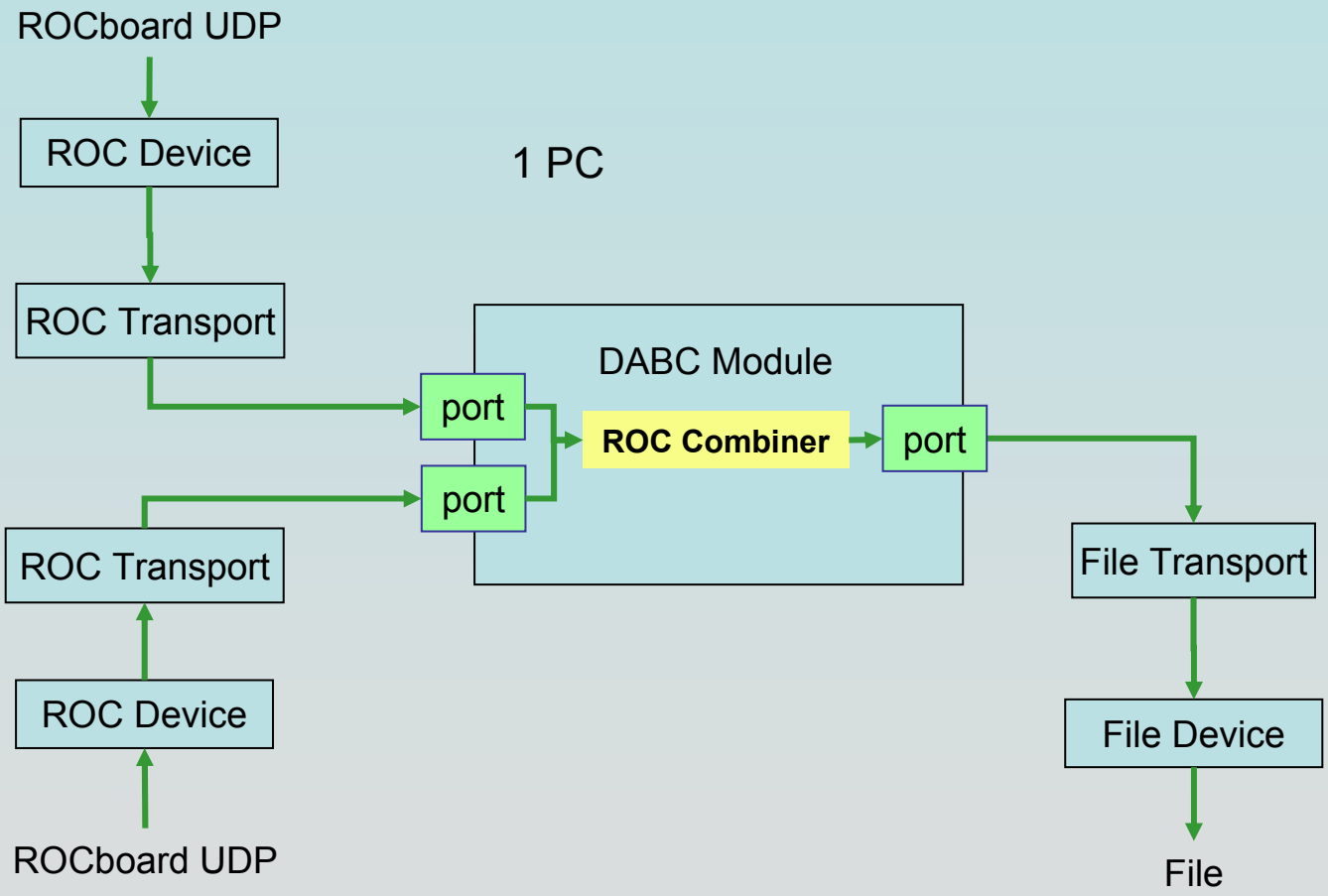


DABC design: Software objects of data flow engine

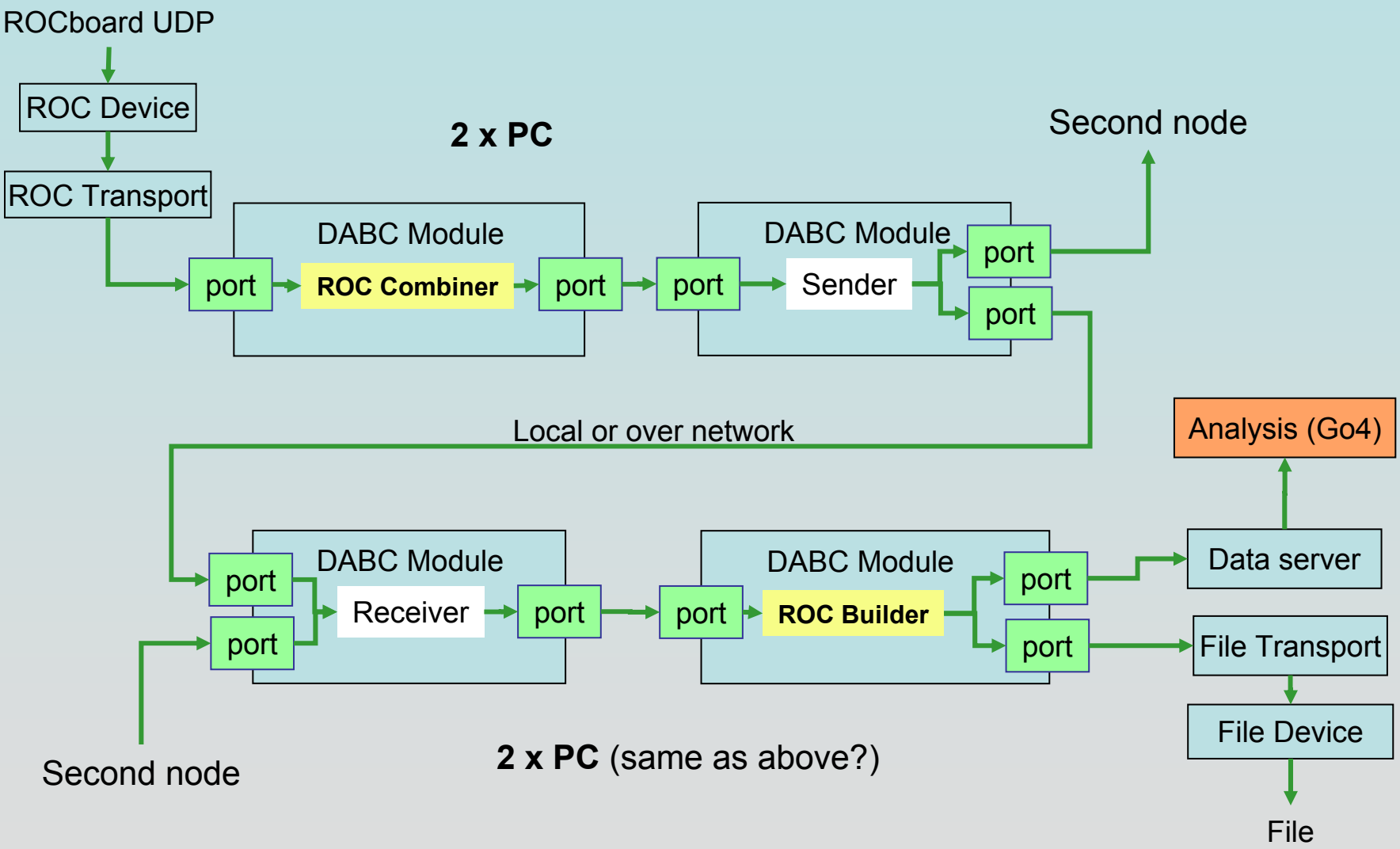
A *module* processes data of one or several data streams.
Data streams propagate through ***ports***, which are connected by ***transports*** and ***devices***



DABC example: Local event building for ROC

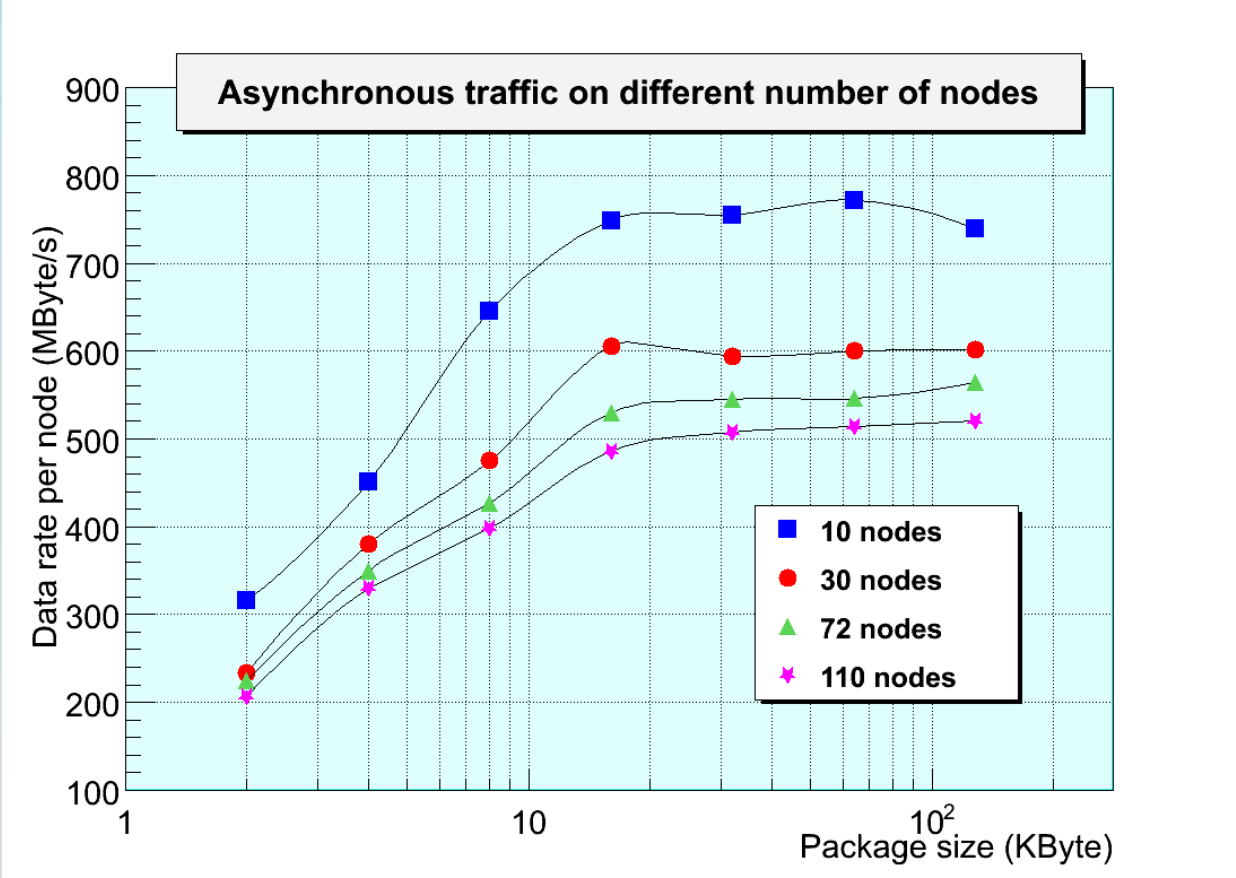


DABC example: Network event building for ROC



Scaling of performance (Mainz)

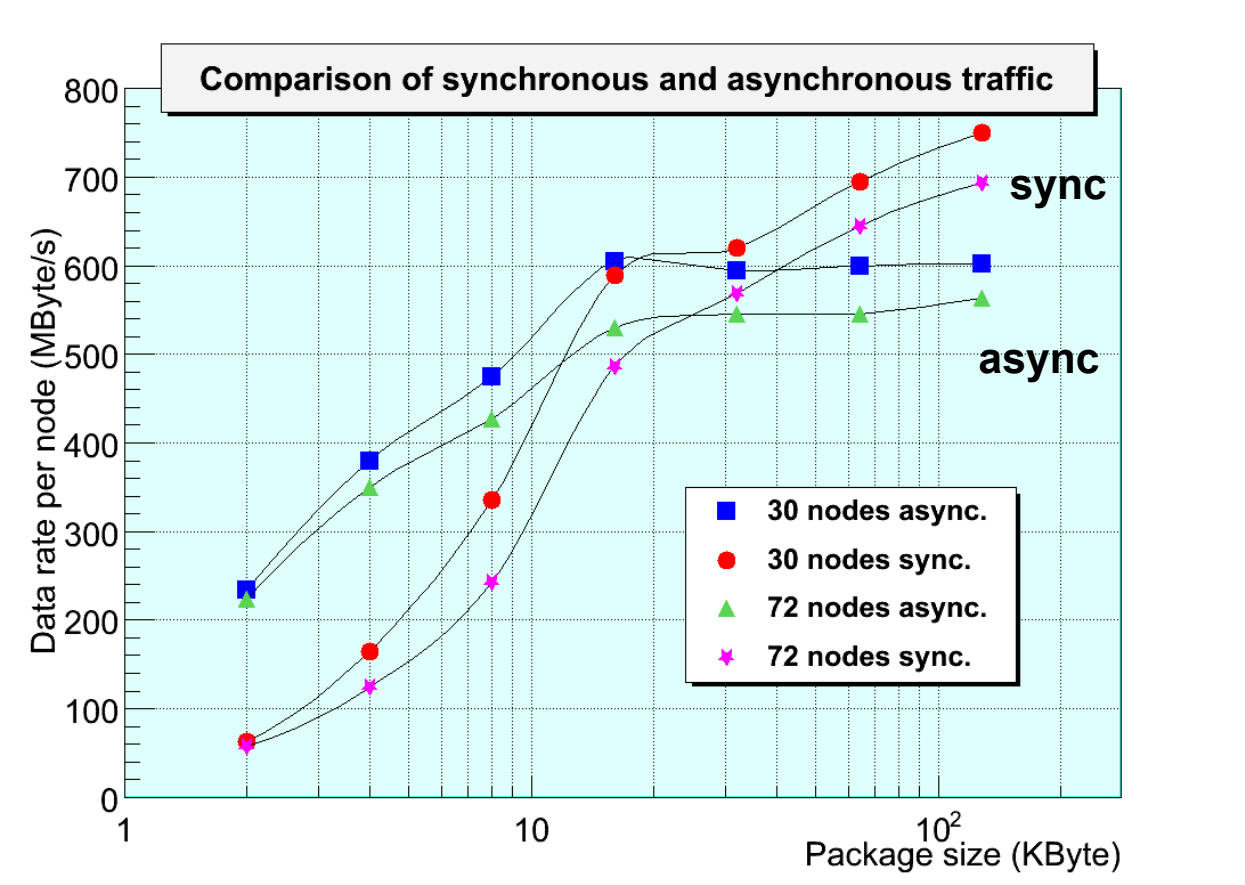
Infiniband performance



We thank Klaus Merle and Markus Tacke at the
Zentrum für Datenverarbeitung der Johannes Gutenberg Universität, Mainz,
for providing resources and support for the large-scale measurements.

Scaling of performance (Mainz)

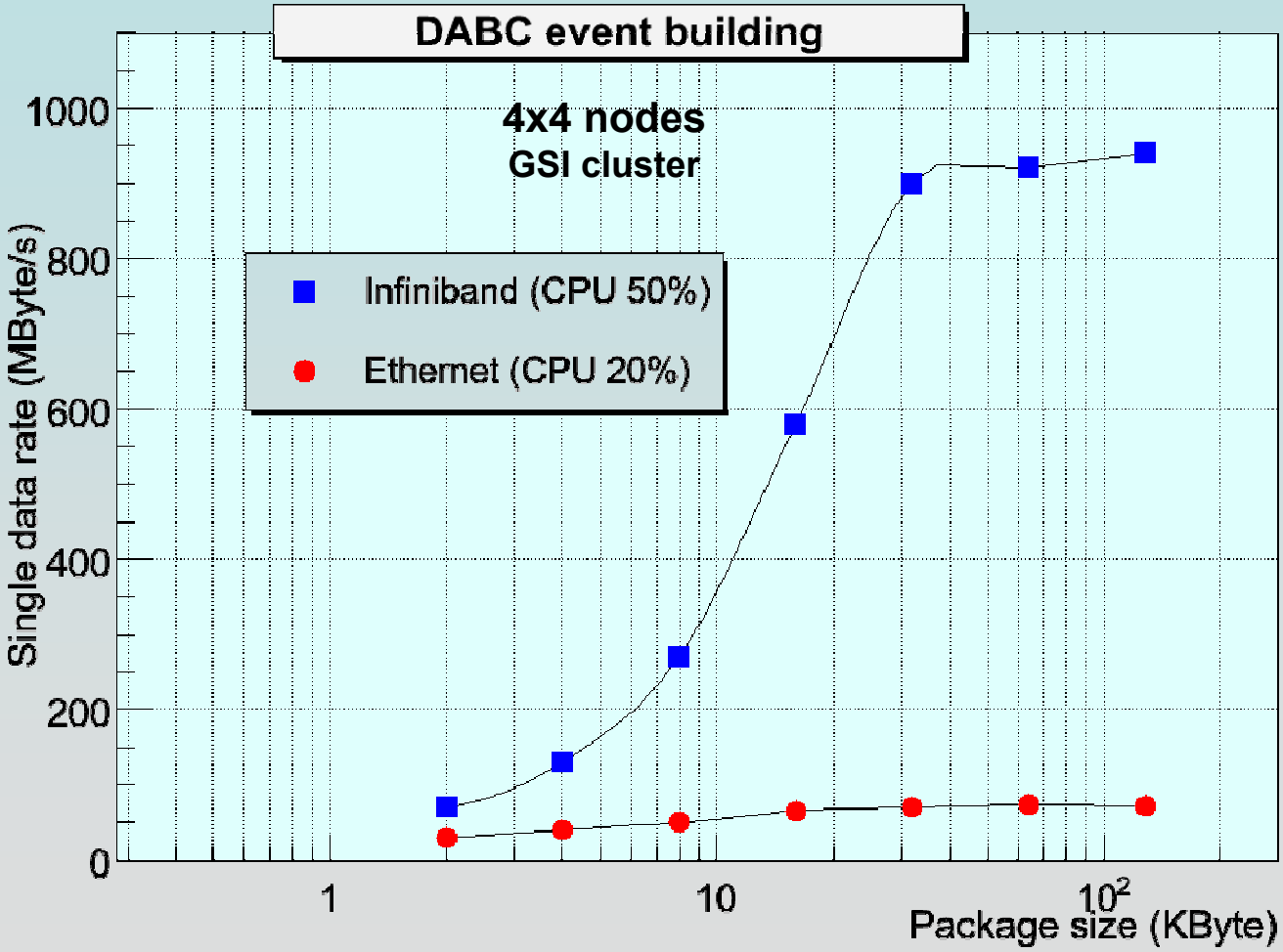
Infiniband performance



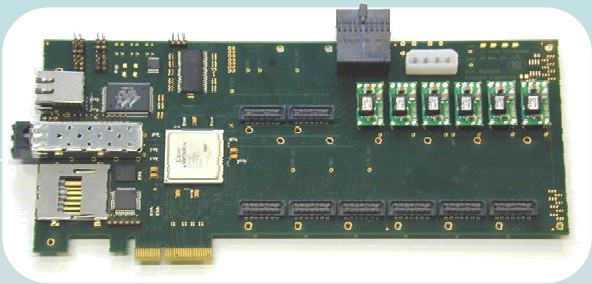
We thank Klaus Merle and Markus Tacke at the
Zentrum für Datenverarbeitung der Johannes Gutenberg Universität, Mainz,
for providing resources and support for the large-scale measurements.

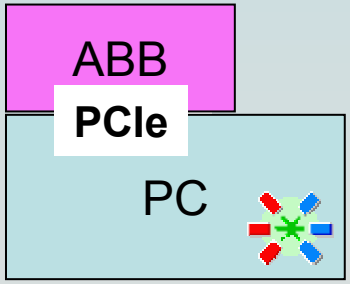
Dataflow engine and event building rate

Infiniband performance



Use case example: Frontend components test





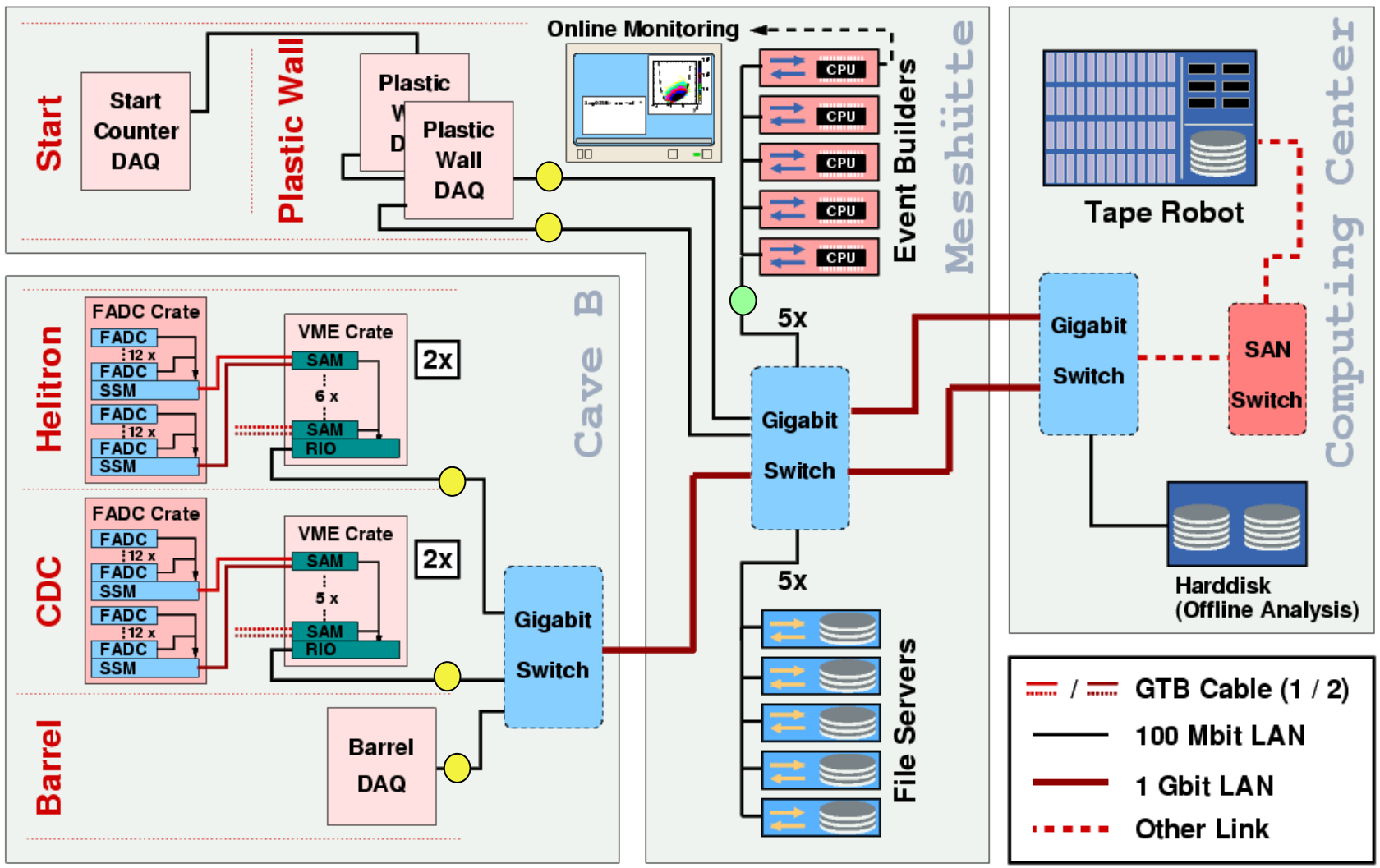
***Active Buffer* board *:**
PCI express card

The goal:

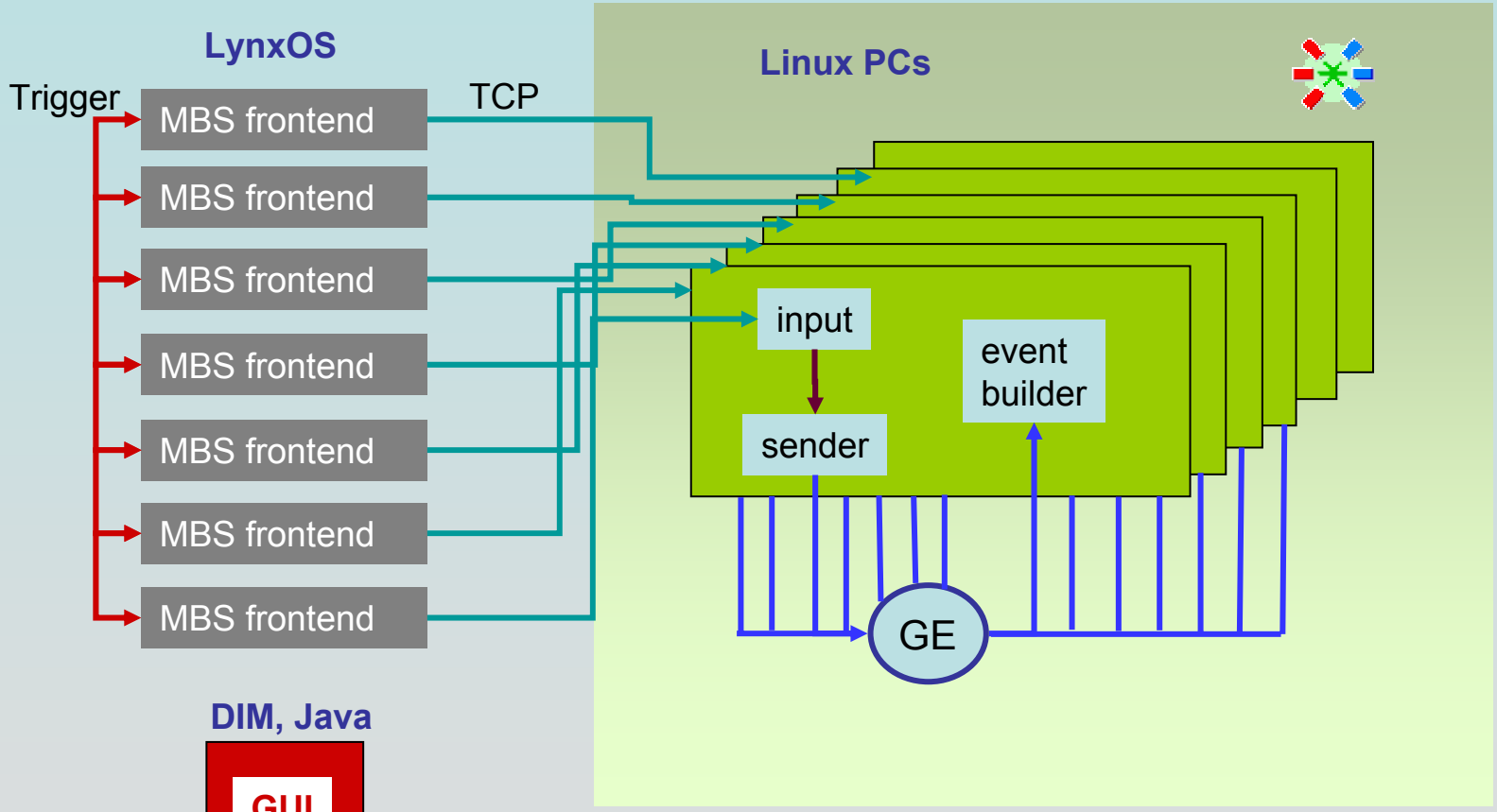
- Driver tests
- Data flow tests
- 400-600 MB/s

* A.Kugel, G.Marcus, W.Gao, Mannheim University Informatics V

Use case example: Event builder for MBS (FOPI)

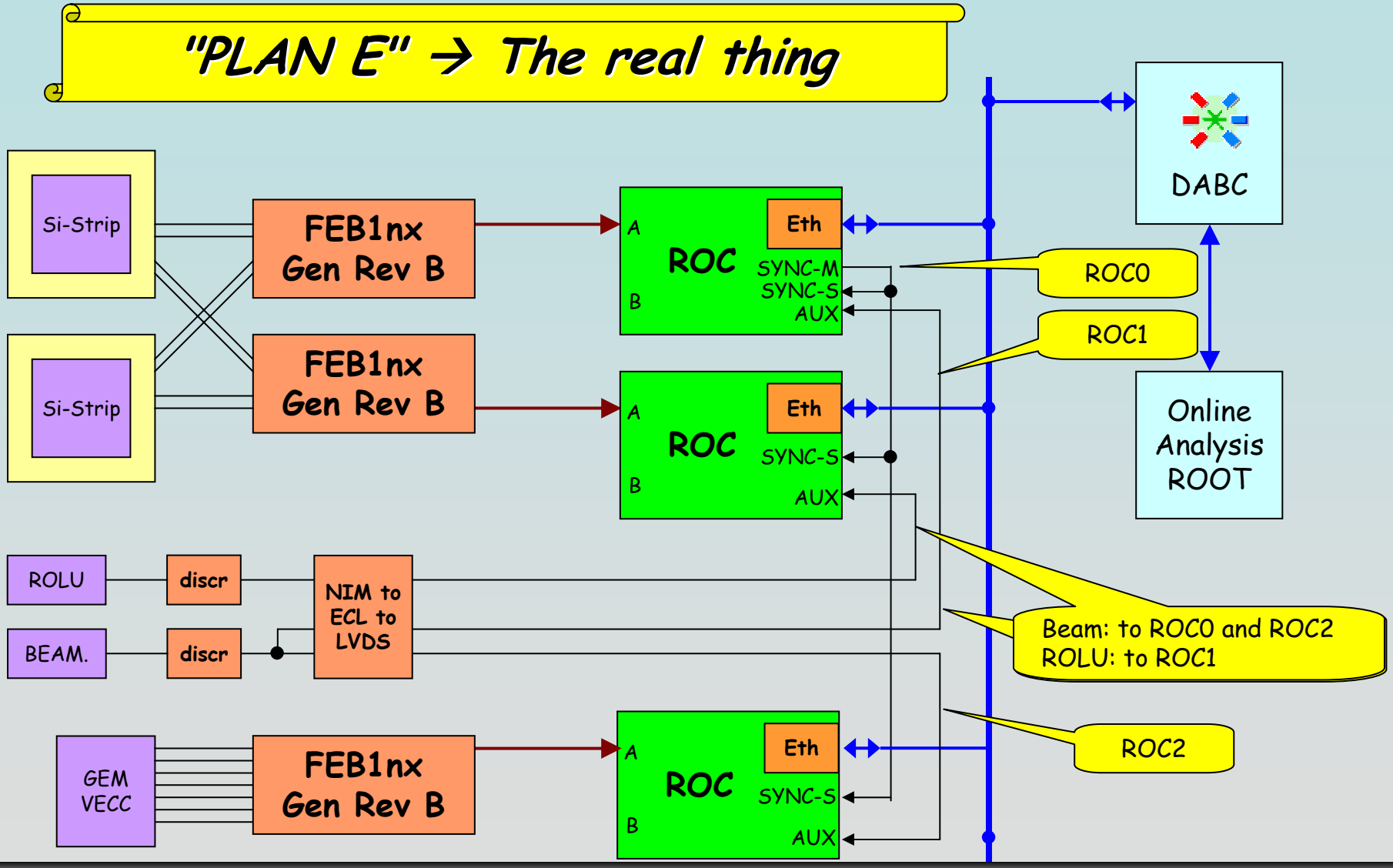


Use case example: Event builder for MBS (FOPI)



GE: Gigabit Ethernet

Config: September Test Beam @ GSI



DABC Controls

MbsLauncher

OK: MBS tasks ready

Name server:	lxg0523.gsi.de
User name:	goofy
Password [RET]:	
Master node:	x86g-4
Servers:	3
System path:	/daq/usr/goofy/mbswork/v51
User path:	v50/x86/newmbs
Script:	script/remote_exe.sc
Command:	m_rising v50/x86/newmbs . x86g-4 x86-7
Launch file:	MbsLaunch.xml

Commands

- MbsStream_serv
- MbsTo
- MbsTransport
  - CloseFile
  - ConnectRfio
  - DisableTcp

Command ConnectRfio, scope MBS

node (C , REQ) :

-diskserver

-archiveserver

Infos

```
X86-7: MsgLog
X86-7: Rate
X86-7: Rising
X86-7: Transport
X86-7: Util
X86-7: xxx0074.lmd size: 100 MB, filled: 97 MB  97% closed
X86-7: Events:  731659936, MBytes: 1463319, E/s:  1709, MB/s:  3.50
```

319, E/s: 1710, MB/s: 3.50

MbsMonitor

- X86-7:Acquisition Running
- X86-7:BuildingMode Immediate
- X86-7:EventBuilding Working
- X86-7:FileOpen W:File closed
- X86G-4:Acquisition Running
- X86G-4:BuildingMode Immediate
- X86G-4:EventBuilding Working

X86-7/MSG		
DataRateKb[KB/s]	3500,0	16000
EventRate[Ev/s]	1709,0	10000
FileFilled[%]	97,0	100
StreamRateKb[KB/s]	0,0	16000
StreamsFull[n]	0,0	1
X86G-4/MSG		
DataRateKb[KB/s]	3503,0	16000
EventRate[Ev/s]	1710,0	10000
StreamRateKb[KB/s]	0,0	16000
StreamsFull[n]	0,0	20

```
X86-7:Transport
X86-7:xxx0074.lmd size: 100 MB, filled: 97 MB  97% closed
X86-7:Events:  731659936, MBytes: 1463319, E/s:  1709, MB/s:  3.50
X86G-4:Transport
X86G-4:No file
X86G-4:Events:  731659744, MBytes: 1463319, E/s:  1710, MB/s:  3.5
Current node: X86G-4
X86-7,X86G-4
```

RateMeters

Layout

X86-7:MSG DataRateKb 3500,0 (0 – 16000)	X86-7:MSG EventRate 1709,0 (0 – 10000)	X86-7:MSG FileFilled 97,0 (0 – 100)
X86G-4:MSG DataRateKb (0:16000, Time)	X86G-4:MSG DataTrendKb 3503,0 (0 – 1935025)	X86G-4:MSG EventRate 1710,0 (0 – 10000)
X86G-4:MSG StreamRateKb 0,0 (0 – 16000)	X86G-4:MSG StreamsFull 0,0 (0 – 10)	

Parameters

ID	Node	Application	Parameter	Type	Current	Set value	Show
00128	X86G-4	MSG	Rising	info	Rising	-	✓
00129	X86G-4	MSG	Run	char		-	
00130	X86G-4	MSG	StreamKeep	int	0	-	
00131	X86G-4	MSG	StreamMbytes	int	0	-	
00132	X86G-4	MSG	StreamRateKb	rate	0.0	-	✓
00133	X86G-4	MSG	StreamScale	int	0	-	
00134	X86G-4	MSG	StreamSync	int	0	-	
00135	X86G-4	MSG	StreamTrendKb	rate	0.0	-	
00136	X86G-4	MSG	StreamsFull	rate	0.0	-	✓

DIM servers: DIS_DNS X86G-4:MSG X86-7:MSG X86G-4:PRM

Go4 v3.4-0 @lxg0517

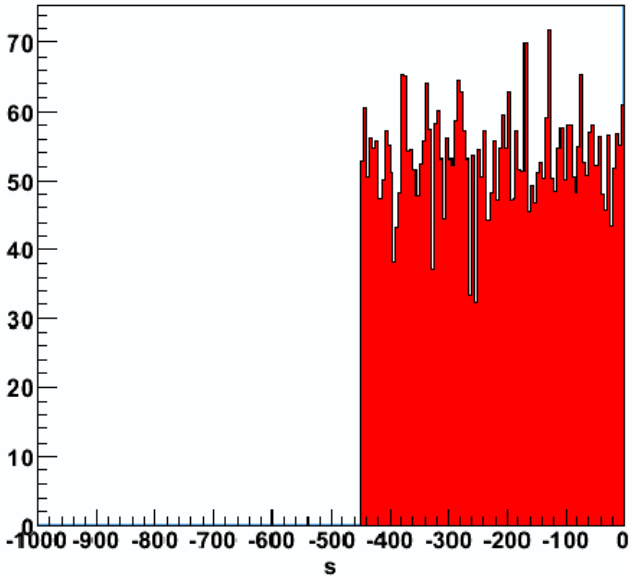
File Tools Analysis Settings Windows Help

2 s | All items

Panel1 : [DataRate.Sender-TrendingAverage]

File Edit Select Options | Apply to all | AutoScale

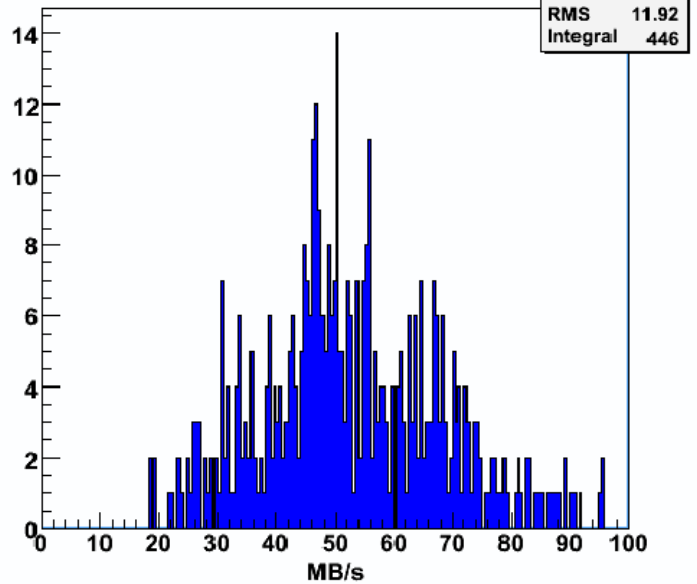
MB/s 16:07:37



Panel2: [DataRate.Receiver-StatsFast]

File Edit Select Options | Apply to all | AutoScale

counts 16:07:39



Name

- Workspace
  - histo1
  - DABC
    - lxi011.gsi.de:1970-Worker:42
      - DataRate.Receiver-Trending.
      - DataRate.Receiver-Trending.
    - lxi009.gsi.de:1970-Worker:42
      - DataRate.Sender-TrendingFa
      - DataRate.Receiver-Trending.
      - DataRate.Receiver-StatsFast
      - DataRate.Sender-TrendingA..
      - DataRate.Receiver-Trending.
      - DataRate.Receiver-StatsAver
    - lxi010.gsi.de:1970-Worker:42
      - DataRate.Receiver-Trending.
      - DataRate.Receiver-Trending.

DIM dns server

lxg0517.gsi.de

Mi Okt 22 16:07:39 2008

☑ more...

Inspector

Rate

☑ histogramming on

200 bins

☑ each sample

☑ average 5 s

DABC Nodes

Context	State	Ratemeters	Service list
lxg0517.gsi.de:1969	Running	☐ Show	☐ Dump
lxi009.gsi.de:1970	Running	☑ Show	☑ Dump
lxi010.gsi.de:1970	Running	☑ Show	☐ Dump
lxi011.gsi.de:1970	Running	☑ Show	☐ Dump
lxi012.gsi.de:1970	Running	☑ Show	☐ Dump

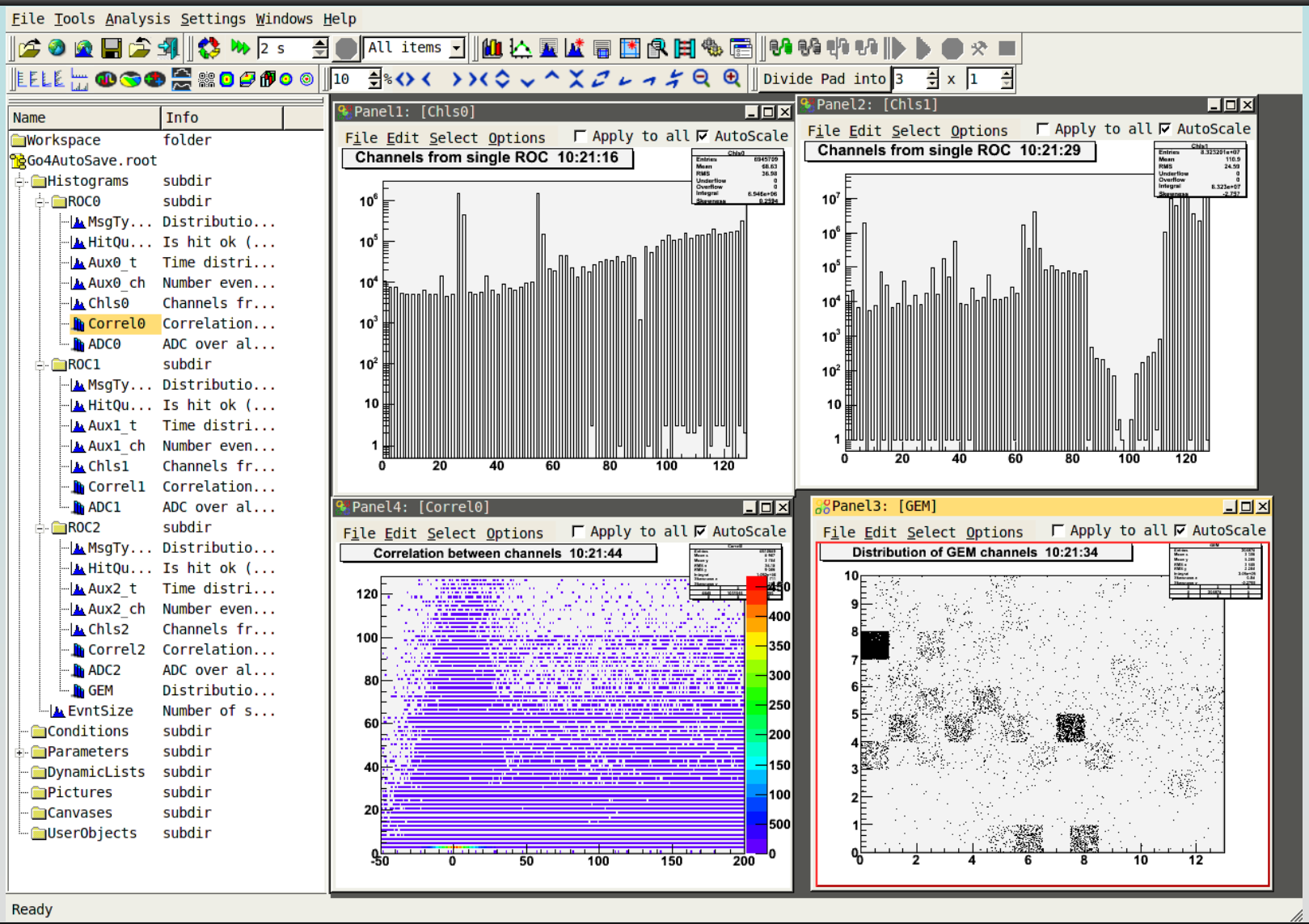
Print DIM services of the nodes checked as 'Dump' to terminal

Ratemeter View

Name	Value	Trending	Statisti
lxi009.gsi.de:1970/Worker:42/DataRate.	63.9993 MB/s	☑ histogram	☑ histo
lxi009.gsi.de:1970/Worker:42/DataRate.	63.3675 MB/s	☑ histogram	☐ histo
lxi010.gsi.de:1970/Worker:42/DataRate.	49.6184 MB/s	☑ histogram	☐ histo
lxi010.gsi.de:1970/Worker:42/DataRate.	49.8574 MB/s	☐ histogram	☐ histo
lxi011.gsi.de:1970/Worker:42/DataRate.	50.1071 MB/s	☑ histogram	☐ histo
lxi011.gsi.de:1970/Worker:42/DataRate.	50.0979 MB/s	☐ histogram	☐ histo
lxi012.gsi.de:1970/Worker:42/DataRate.	64.0035 MB/s	☐ histogram	☐ histo
lxi012.gsi.de:1970/Worker:42/DataRate.	63.9918 MB/s	☐ histogram	☐ histo

Ready

Go4 analysis screen shot N-XYTER test beam



J.Adamczewski-Musch, H.G.Essel, N.Kurz, S.Linev

DABC state of project

Achieved

- Infrastructure
- Data flow engine
- Generic Java GUI
- PCIe support
- ROC event building
- MBS event building
- MBS DIM control
- Plug-in Interface implementation
- **Version 1.0 installed at GSI**

Todo

- GUI (continuously)
- Data formats
- Combiner/Time sort
- Event building
- Scheduler
- **Final packaging**
- **Documentation**
- **Public distribution (Jan 2009)**

Resources and time

- People of data processing group
  - H.G.Essel
  - J.Adamczewski
  - N.Kurz
  - S.Linev
- People of controls group
  - maybe one FTE
- People from CBM
  - hopefully

- **CBM required in 2008 a data taking system**
  - **Test beam September: DABC and Go4 were ready and run**
  - Start with small system, grow on demand
  - Preliminary controls

- NUSTAR
  - Combination of MBS with other front-end equipment
- MBS for FOPI: a first test bed

Discussion

- DABC as designed and implemented
  - scales to O(10) nodes with 2x700 MByte/s per node.
  - can be used as general purpose DAQ framework
  - can combine triggered (MBS) and triggerless (non-MBS) front-ends
  - can be used for all detector and front-end tests
- For larger systems we need
  - more powerful controls integrating all DAQ subsystems
  - more powerful configuration system
  - investigations in network topologies for performance
  - failure recovery and fault tolerance
  - DABC kernel still could be used
- We need more detailed requirements for
  - bandwidth resources
  - computing resources
  - filters based on partial data

Discussion

- Keep on track of industrial main stream
  - Maintenance
  - Hard to keep advantage of propriety solutions over decades
  - Multiple CPUs per data connection
- Avoid hardware dependencies
  - Hardware replacements necessary over time
- Design system to grow on demand
  - Full system may be installed in phases.
- Design subsystems replacable
  - Due to progress in technologies and growing requirements it might be necessary to replace subsystems.

Discussion

FE systems

To do:
Merging
Time sorting
Formatting

Attributes

B-Network

To do:
Sorting
Schedule

Attributes

Event definition

P-Network

To do:
Processing
On-line analysis codes
Filter
Distributing

Event definition

Matrix elements = 1000 ± 400

Buffers ---------->

Events ---------->

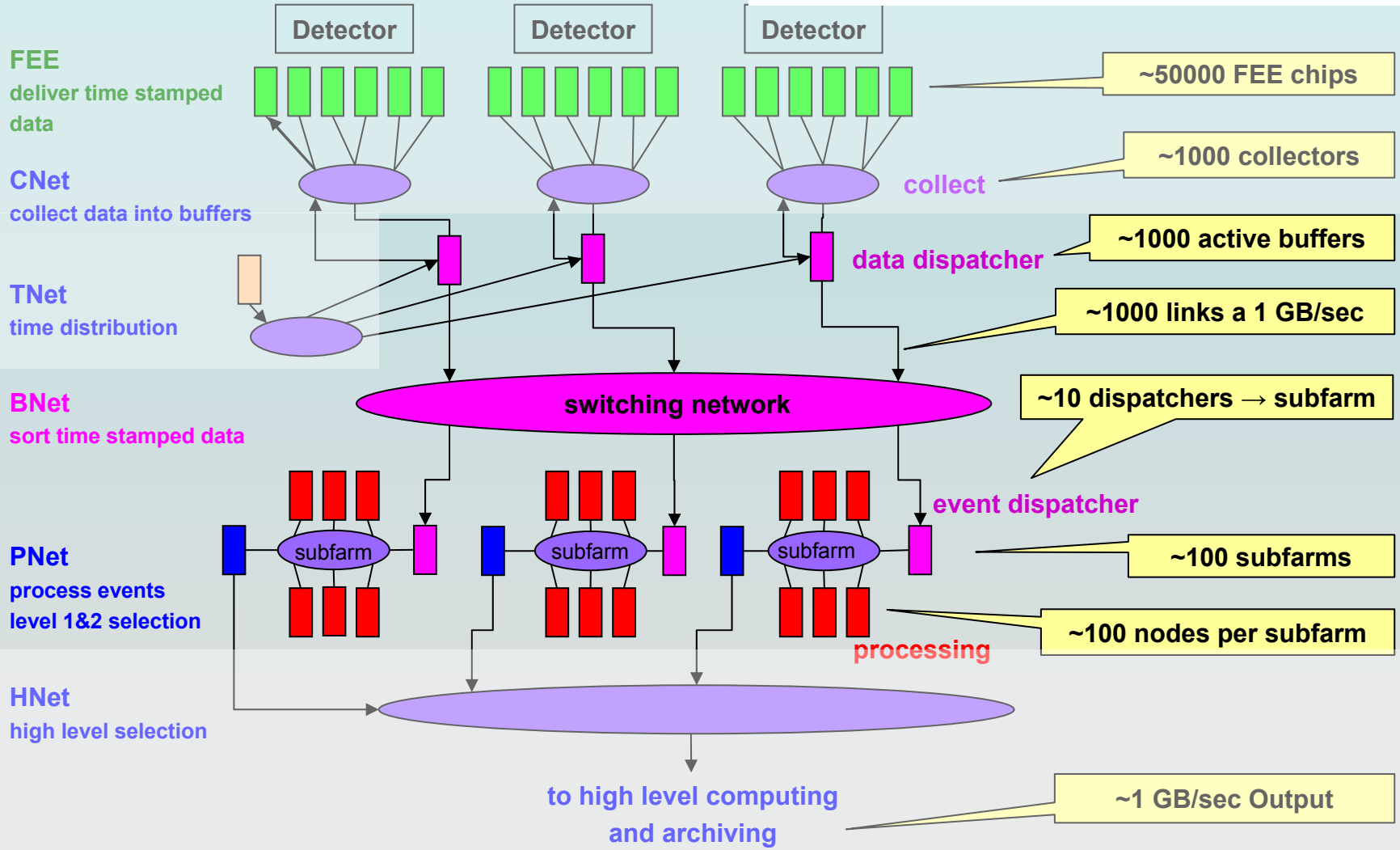
1085	1096	956	987	1302	806	903	1421	941	1153	1070	934	997	1088	772	1194	918	880	668	1054
1334	924	1093	1157	844	1079	790	1285	938	855	792	1023	1220	987	1017	1016	1220	1353	1194	925
986	988	1024	814	1223	1006	947	1263	1124	888	824	1091	697	811	1350	1206	928	1002	1221	871
1068	1020	1338	934	765	958	900	883	793	905	372	886	1043	1024	897	1095	617	1125	918	1359
724	600	1268	757	1003	855	1267	931	845	1024	709	1089	883	1185	639	1059	854	961	1006	884
1036	1174	1067	1031	1300	1154	1071	1064	1082	989	1395	755	905	832	913	1167	1125	1254	955	638
861	770	948	556	698	1054	1022	652	875	839	1038	1013	1266	748	975	914	645	1076	1175	946
1068	854	886	1112	1261	985	988	1004	1243	1304	931	1046	995	1424	632	1247	802	1097	603	919
653	1271	1155	953	1078	825	1017	947	500	1145	901	836	1181	633	642	1233	829	771	1055	887
1005	1326	664	1084	984	1225	791	945	420	1139	758	672	791	1192	1115	1011	1289	1220	1007	762
778	969	881	1023	1016	809	705	801	1105	1753	972	689	890	1395	877	1237	793	531	886	708
806	815	1145	1195	1246	804	751	895	1060	889	662	1032	878	1280	928	1031	832	1065	1141	986
868	1017	941	985	1109	978	1204	1011	1045	648	1109	1051	907	867	639	805	1284	918	1226	744
1069	1297	1117	1019	1207	1309	1206	1155	801	1046	936	810	893	884	714	1139	657	805	1099	1180
1122	1277	783	874	831	1094	875	996	874	1047	1109	931	1203	604	1121	939	1140	1048	908	965
777	922	1073	1120	986	1232	938	1008	860	948	687	779	1036	912	793	1058	983	582	541	764
670	1310	986	1017	983	1150	949	1087	794	860	1303	1184	1216	1321	1036	996	1386	933	1294	1533
1096	819	1178	719	1144	971	1176	818	771	1119	844	1314	1184	1021	870	1020	952	921	1417	818
856	964	1320	790	1030	1435	1290	909	1176	1297	565	1189	1145	549	860	1049	1068	976	931	793
994	1121	1129	1026	859	1004	1192	1105	930	1440	805	1121	1398	902	1107	775	1092	629	975	805

Barrel slots = 20

CBM data acquisition sketch

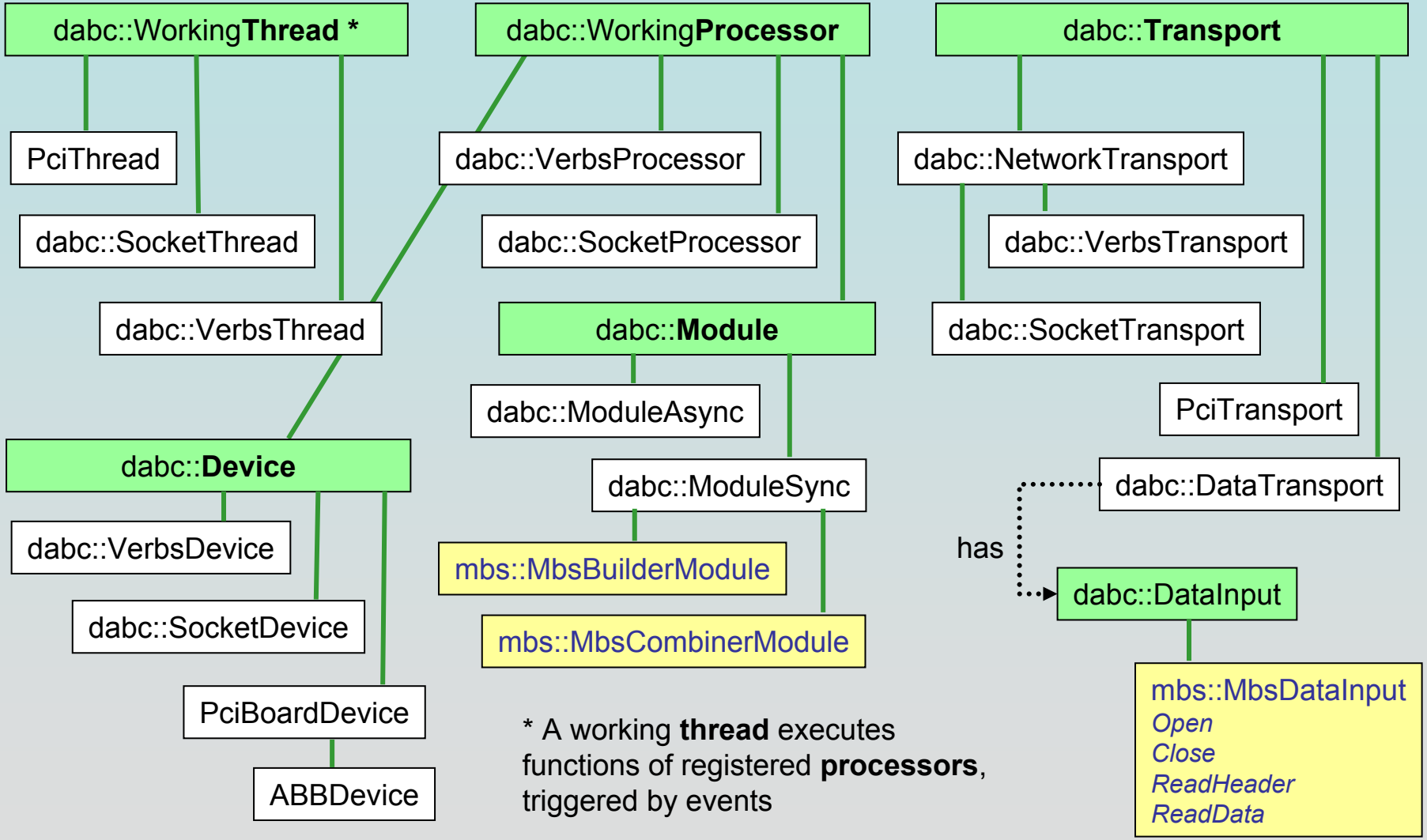
W.F.J.Müller, 2004

Triggerless, time stamped data flow concept

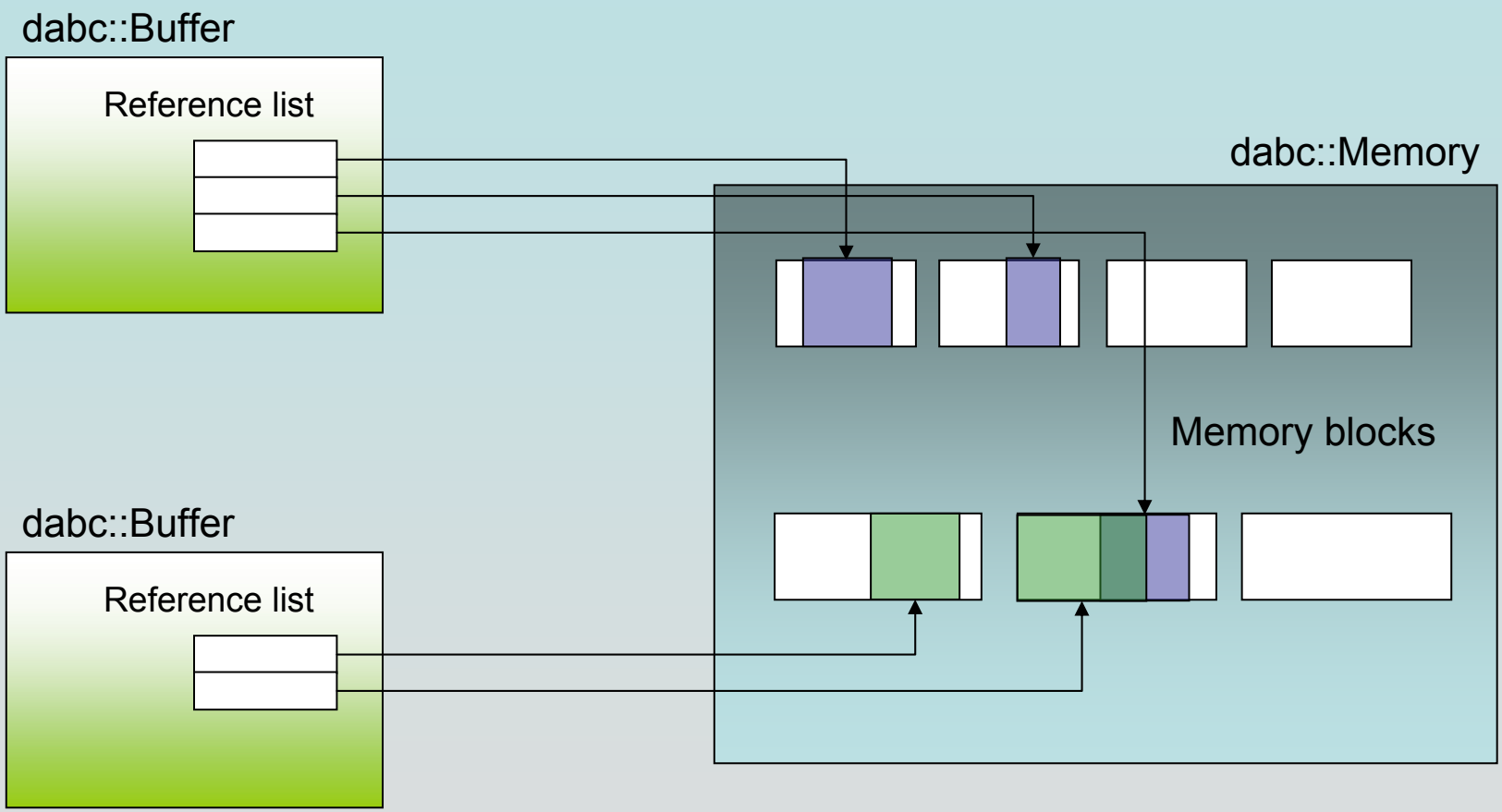


DABC compoments design

MBS example plug-ins



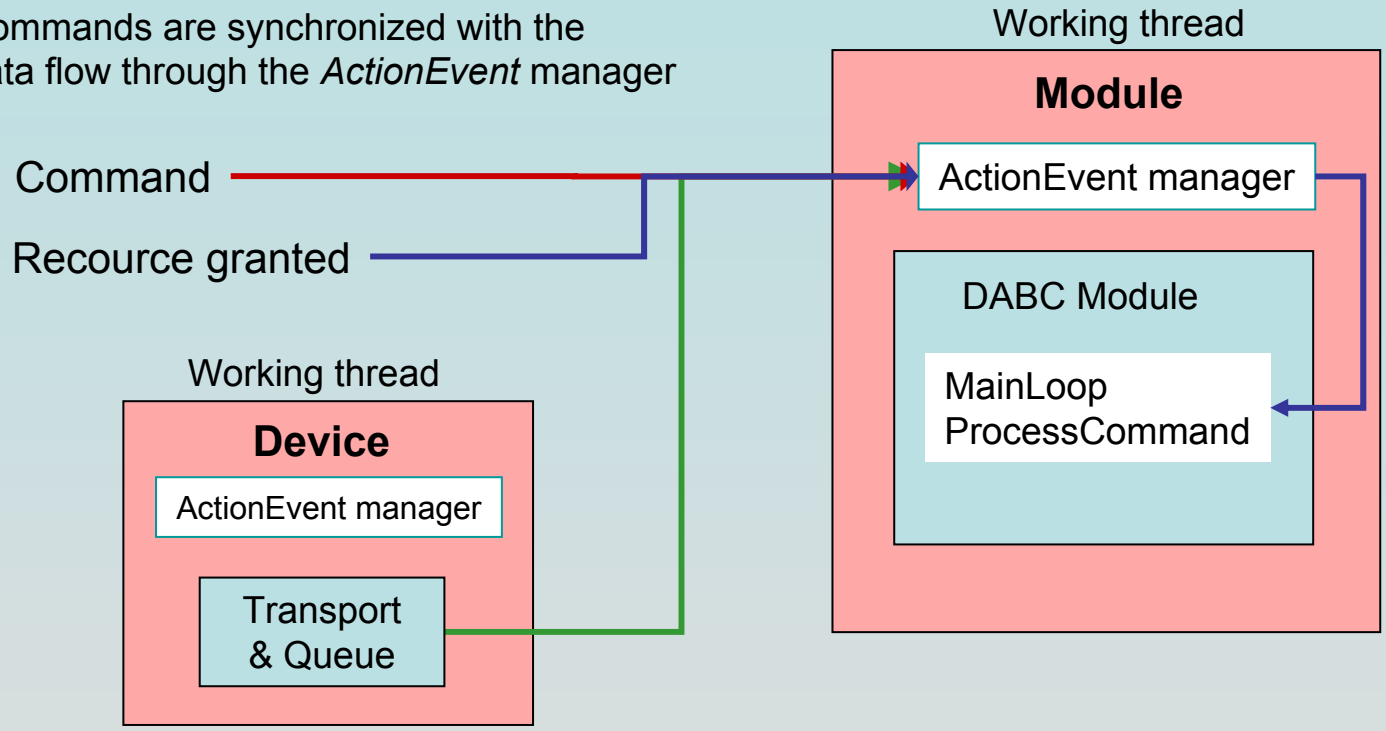
DABC smart buffers



Multiple references of resources (blocks)
Copy buffer without copy of data
PCI/DMA → buffer → IB/DMA

DABC design: Threads and synchronization

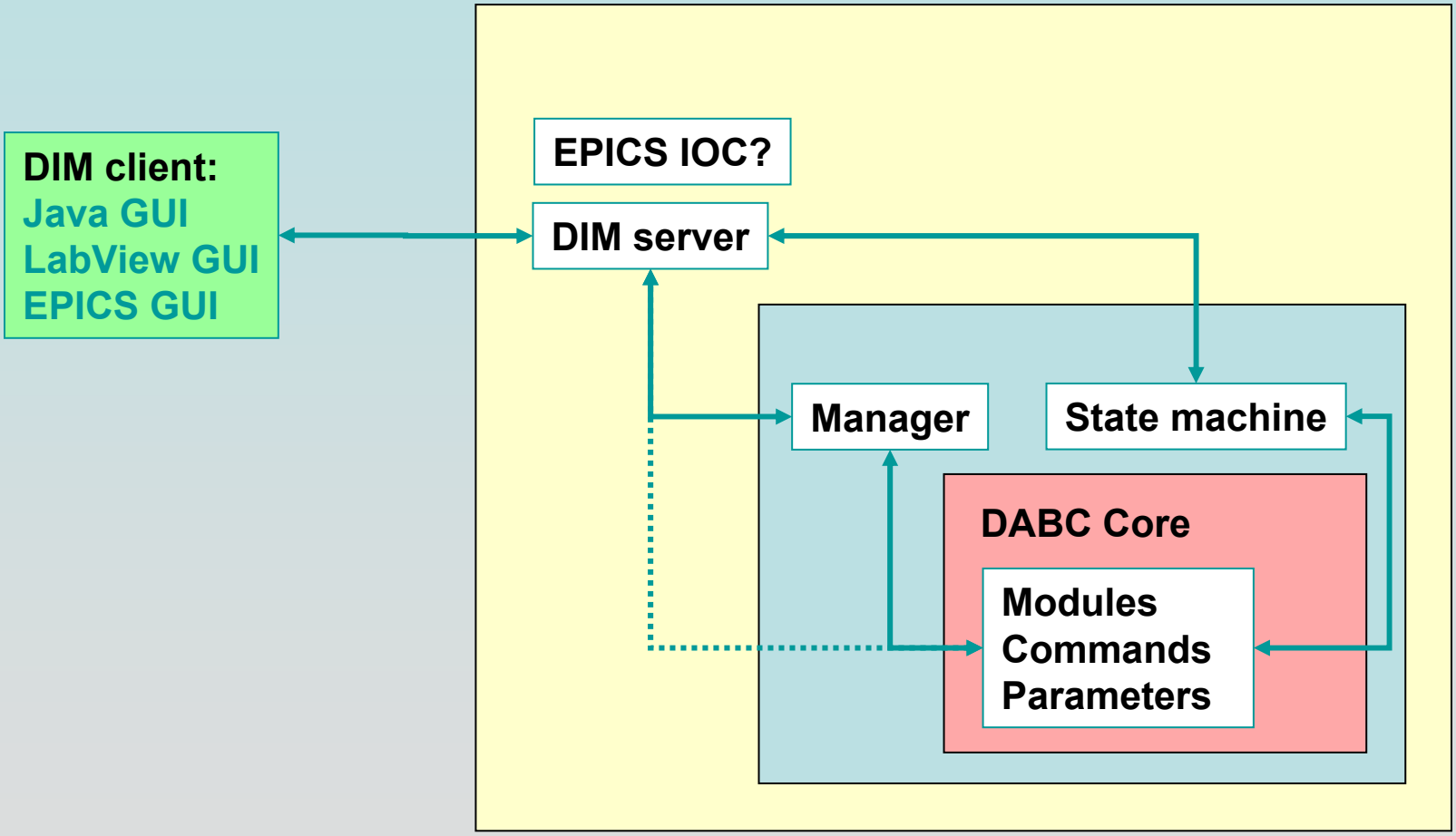
Commands are synchronized with the data flow through the *ActionEvent* manager



A device, i.e. socket device controls one port (transport).
Once a queue buffer is filled, the transport signals the *ActionEvent* manager, which in turn calls the *MainLoop* function of the associated module, **or**
If *MainLoop* was waiting for resource or buffer, it continues.

DABC design: Controls & monitoring communication

DABC core decoupled from XDAQ environment!



DABC Setup

- **Configuration via XML files**
- ***ApplicationPlugins* (entry point to application libraries)**
  - Call application factories
- **Application factory classes**
  - CreateDevice
  - CreateModule
  - Device->CreateTransport (Module->GetPort)
  - ConnectPorts (Module1->GetPort, Module2->GetPort)
  - CreateMemoryPool
- **State commands**
  - Startup
  - Initialize
  - Run / Stop
  - Hold / Resume
  - Shutdown

DABC: Commands and parameters

- **Commands**
  Objects with command description (XML) and *ProcessCommand* function.
  Name string: **/ server / node / application / type.thread.name**
  - server: DIM namespace
  - node: name:ID (port)
  - application: namespace::name:ID
  - type: DEV, MOD, POOL, PLUG
  - thread: name of module or device or...
  - name: command (description by related parameter record)
- **Parameters**
  Same name structure as above
- **Parameter records**
  Recognized by GUI, graphical presentation
  - Status
  - Rate
  - Histogram
  - Command description
  - and more