Application layer
2 basic models

- client-server model
- producer-consumer
- publisher-subscriber
- Services and VMD
- Coffee machine

No more change

Coin

Select coffee

Coffee

Select sugar
• VMD and server
Confirmed Service

- MMS Provider
  - Service-Rq
  - Service-Cnf

- Underlying profile

- MMS Provider
  - Service-Ind
  - Service-Rsp

- Server
Example

Client  MMS  Server

Start Paint1
● Unconfirmed service

Client

Service-Ind

MMS Provider

Underlying profile

MMS Provider

Server

Service-Rq

C/S 5
C/S 6

- Unconfirmed service

Client

Server

Paint tank empty

IN2P3  octobre 2002  Jean-Pierre Thomesse
C/S 7

- Performs really the required services
- Manages the resources, the access
- Programmed by the device vendor
- A single executive function for a VMD
“Usual” computing objects

- Variables and Types
- Semaphores
- Events
- Programs and Processes
- Domains
- Journals
- Operator station
- VMD itself
- Associations
MMS = abstracts objects

Each object is the instance of a class

Each object:
  - Characteristics.
  - Set of operations.

All are implemented in the VMD

Contribute to model the VMD
Producer - Consumer
MPS- production

- Producer

Local Write

Speed

IN2P3  octobre 2002  Jean-Pierre Thomesse
MPS-consumption

- Consumer

Local Read

Speed
Read and write

- Services primitives

Production

Write-Rq(Ident, value)
Write-Cnf (Ident, result)

Distribution

Consumption

Read-Rq(Ident)
Read-Cnf(Ident, value)
**Services primitives**

**Production**
- Write-Rq(Ident, value)
- Write-Cnf (Ident, result)

**Distribution**

**Consumption**
- Read-Rq(Ident)
- Read-Cnf(Ident, value)

Ethernet or variant
Read and write

- Services primitives

Production

Write-Rq(Ident, value)
Write-Cnf (Ident, result)

Distribution

token

Consumption

Read-Rq(Ident)
Read-Cnf(Ident, value)
Read and write

- Services primitives
  Production
    - Write-Rq(ident, value)
    - Write-Cnf(ident, result)
    - Sent-Ind(ident)
  Distribution
    - WorldFIP
      - PDU, ident
    - PDU, value
  Consumption
    - Read-Rq(ident)
    - Read-Cnf(ident, value)
    - Sent-Ind(ident)
Read and write

- Producer
  - Local Write
  - Speed
  - Application layer
  - Data link
  - Physical layers

- Consumer
  - Local Read
  - Application processes
  - Speed "copy"
  - Application layer
  - Data link
  - Physical layers
Periodic and Aperiodic Traffics

Scheduling by table
Example

\[\begin{array}{cccccc}
D & A & B & C & A & D \\
C & A & B & C & A & D \\
B & C & B & C & B & C \\
\end{array}\]

Max Load

Time

IN2P3

octobre 2002

Jean-Pierre Thomesse
MAC Basic principle

- A bus arbitrator
- For a given variable, stations are
  - Producer
  - Consumer
  - Other
- Centralized access
- Source addressing by the identifier of a variable
Exchanges - 1

- First stage

Bus arbitrator

Producer

Consumer

Consumer

Phy. L.

1234

1234

1234

1234

IN2P3  octobre 2002  Jean-Pierre Thomesse
Exchanges - 2

- Second stage

Bus arbitrator
Producer
Consumer
Consumer

Phy. L.
Exchanges - 3

- Third stage

Bus arbitrator

Producer

Consumer

Consumer

Value of "1234"

IN2P3 octobre 2002 Jean-Pierre Thomesse
Fourth stage
● Exchange in four stages
● Two types of frame
  – 1st to transmit the identifier of the variable
  – ID-DAT (acronym for Identifier of Data)
  – 2nd to transmit the value of the variable
  – RP-DAT (acronym for Response to Identifier of Data)
● ID-DAT always sent by the bus arbitrator
● RP-DAT always sent by a producer
- Periodic load evaluation
- Microcycle
  - variable transfer time
  - number of variables
- Macrocycle
  - organized according to a static scheduling
- Free time slots
  - depend on the load and on the scheduling
Example

Max Load

<table>
<thead>
<tr>
<th>Time</th>
<th>A</th>
<th>A</th>
<th>A</th>
<th>A</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>C</td>
<td>B</td>
<td>A</td>
<td>C</td>
<td>D</td>
</tr>
<tr>
<td>D</td>
<td>C</td>
<td>B</td>
<td>A</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>A</td>
<td>A</td>
<td>B</td>
<td>A</td>
<td>A</td>
<td>A</td>
</tr>
</tbody>
</table>

IN2P3  octobre 2002  Jean-Pierre Thomesse
- Bus arbitrator table
• Periodic traffic
  – Piloted by the BA
  – Only a producer may send a frame
  – Respect of the periods
  – Wait for the next microcycle if free slots

• Aperiodic traffic
  – Take place in free slots
  – How to initiate and manage this aperiodic traffic?
- **Aperiodic exchange of identified variables**
  - Request by an application layer to obtain or to send new values of variables
  - This application layer must be associated to a producer station of at least one variable

- **Aperiodic exchange of messages**
  - Request by an application layer to send a message
  - This application layer must be associated to a producer station of at least one variable
**Bus arbitrator table**

<table>
<thead>
<tr>
<th>IDc-Rq</th>
<th>D</th>
<th>C</th>
<th>B</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
<td>A</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>B</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>B</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>D</td>
<td>C</td>
<td>B</td>
<td>A</td>
</tr>
</tbody>
</table>
Bus arbitrator table
Aperiodic traffic synthesis

- Aperiodic exchanges
- First stage:
  - request for an aperiodic exchange (by a producer)
- Second stage:
  - sending right to express the need (by the BA)
- Third stage:
  - aperiodic exchange itself initiated as periodic traffic by the BA
**Bus arbitrator table**

<table>
<thead>
<tr>
<th>IDc-Msg</th>
<th>D</th>
<th>C</th>
<th>B</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
timeliness

- Buffers can uncouple
  - data production,
  - data access,
  - data transfer,
  - data consumption

- Data age may be unknown

- Timeliness features for data transfer between buffers
timeliness

production transmission reception consumption

production transmission consumption

IN2P3 octobre 2002 Jean-Pierre Thomesse
timeliness

- **Sender timeliness**
  - Residence
  - Update
  - Synchronous
  - Transparent

- **Receiver timeliness**
  - Residence
  - Update
  - Synchronous
  - Transparent
Assessment based upon the time that a data unit has been resident in a buffer.

Write-date                Residence Time                Read-date

End of time window
Assessment based upon the time interval between a synchronising event and the moment the buffer is written.

- Update-Time
- Synchro-event
- Writing-date
- End of time window
synchronous

- Assessment based upon the time intervals and timing relationships between
  - a synchronising event
  - the moment when the buffer is written
  - the moment the buffer is read
Conclusion

- needs for
  - client-server mechanisms
  - producer-consumer mechanisms
  - producer(s)-consumer(s) mechanisms

- timeliness
  - depends essentially on MAC
  - attributes / qualifiers according to the models