PDP 4PS : Periodic Delayed Protocol for Partitioned Systems

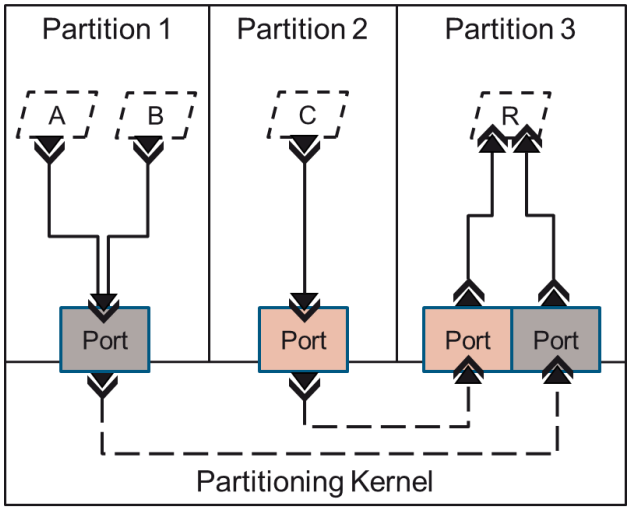
Authors: Antoine Jaouën
Etienne Borde
Laurent Pautet
Thomas Robert

firstname.lastname@telecom-paristech.fr

Introduction & Issues
Context

Safety Critical systems

- Under standard specification (ARINC, CENELEC-EN 50128)
- Partitioned systems (space & time segregation of applications)
- Certification requirements



N→1 Inter-Partition Communication mechanism

- Usage : Data Fusion, Triple Modular Redundancy
- Issues : **Natively not supported by ARINC**

  Non-Propagation Non disturbance of non-faulty partitions by faulty partitions (i.e. port overflow, message loss)

  Message Identification Ability to identify the origin of each message

Introduction & Issues
The Periodic Delayed Protocol (PDP)

Objectives : Deterministic N→1 mechanism

- Message consumption order known
- Execution time & memory overheads minimized (*e.g.* No locks)
- Tight estimation of sufficient memory space

Communication Model

- Periodic task set
- One message produced per job
- Message available at the sender's job deadline
- PDP available next receiver's job
- Messages ordered with jobs deadlines

Communication Mechanism

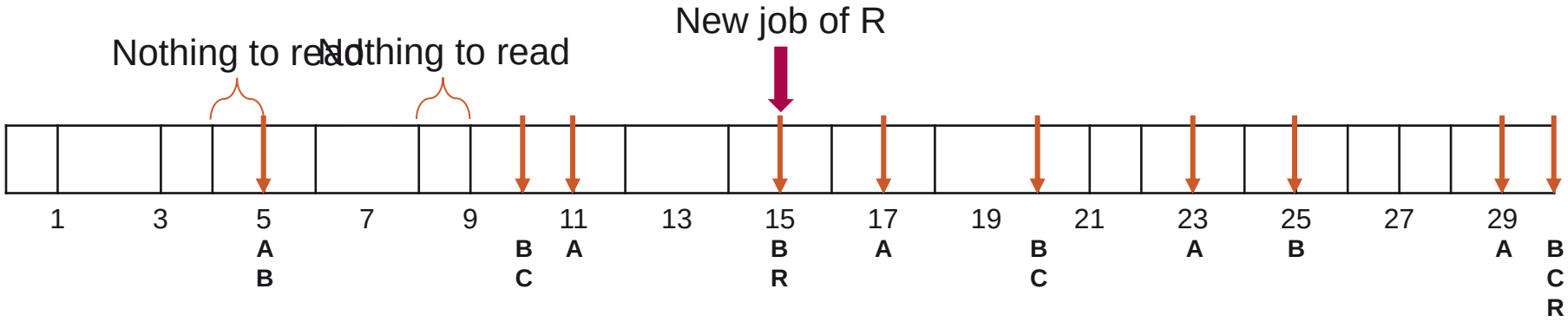
- Wait-free shared circular queue (aka. PDP buffer)
- One pre-defined slot per message
  - ⇒ **Message Identification**
- One pre-defined message per slot
  - ⇒ **Non-Propagation**

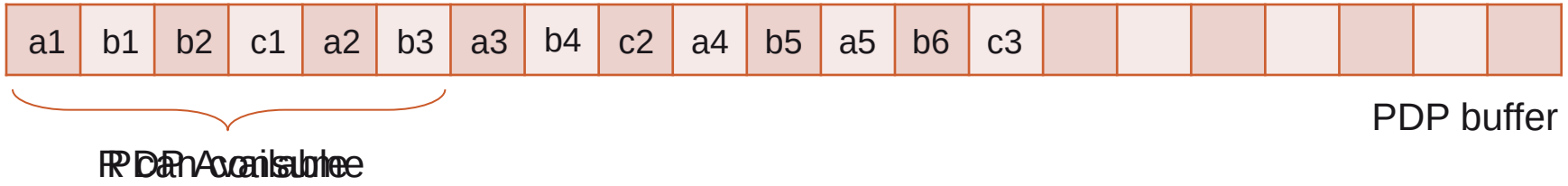
Introduction & Issues
PDP - Example

Task	A	B	C	R
P	6	5	10	15
D	5	5	10	15
C	2	1	1	2

Key:

A — Deadline of A





Introduction & Issues
PDP - Example

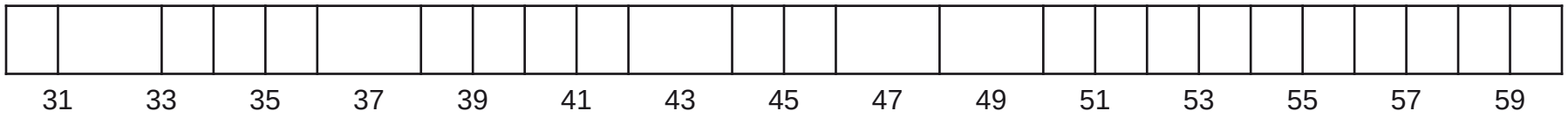
Task	A	B	C	R
P	6	5	10	15
D	5	5	10	15
C	2	1	1	2

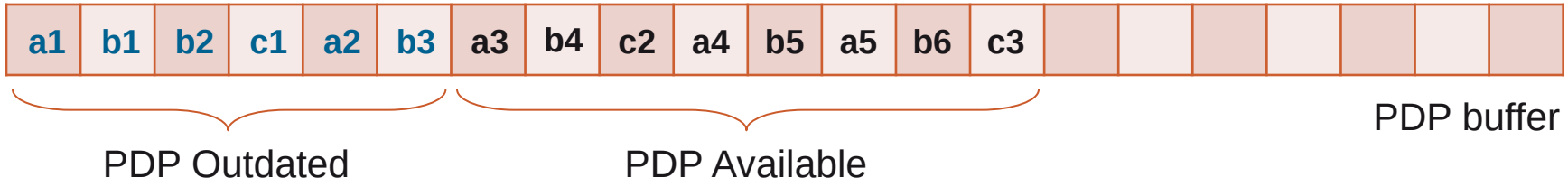
Key:

Deadline of A

A

New job of R





Introduction & Issues
PDP adaptation to partitioned systems

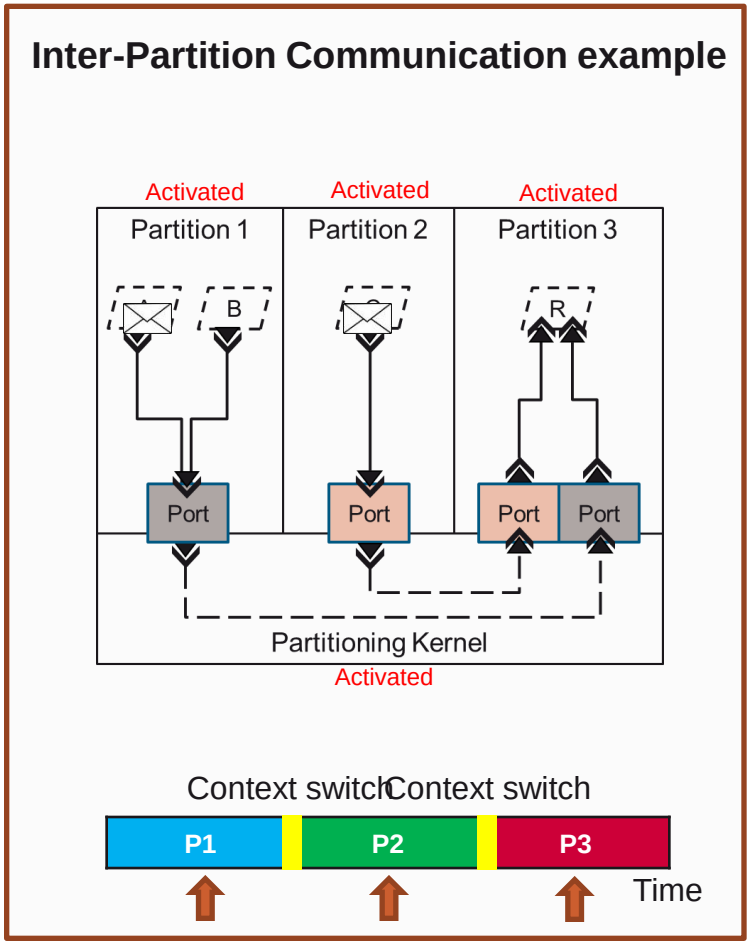
I) Implementation issue

Space & time segregation

- PDP : for shared memory systems
- PDP : direct transfer to the receiver queue

II) Design issue

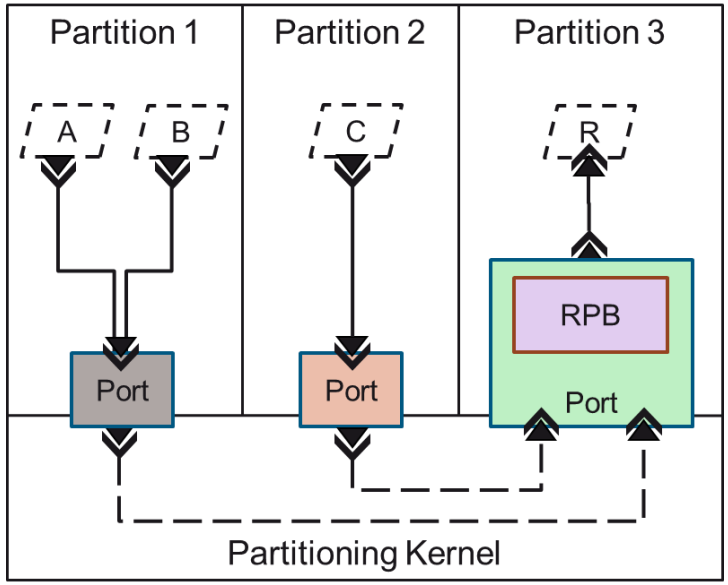
- How do we deal with disjoint memory ?
- How do we allow the PDP semantic
- How do we maintain the **Non-Propagation** & **Message Identification** properties



PDP 4PS : For Partitioned Systems
General Methodology

- Message Conditioning Before Transfer
  - Store msg in sender memory space
  - Prevent from msg overproduction
    - **Non-Propagation**
  - Provide sender IDs to msg
    - **Message Identification**
- Message Actual Transfer
  - Done before next receiver activation
- Message Conditioning After Transfer
  - Store msg in receiver memory space
  - Insert msg in the RPB
    - **Non-Propagation**
    - **Message Identification**

General architecture



PDP 4PS

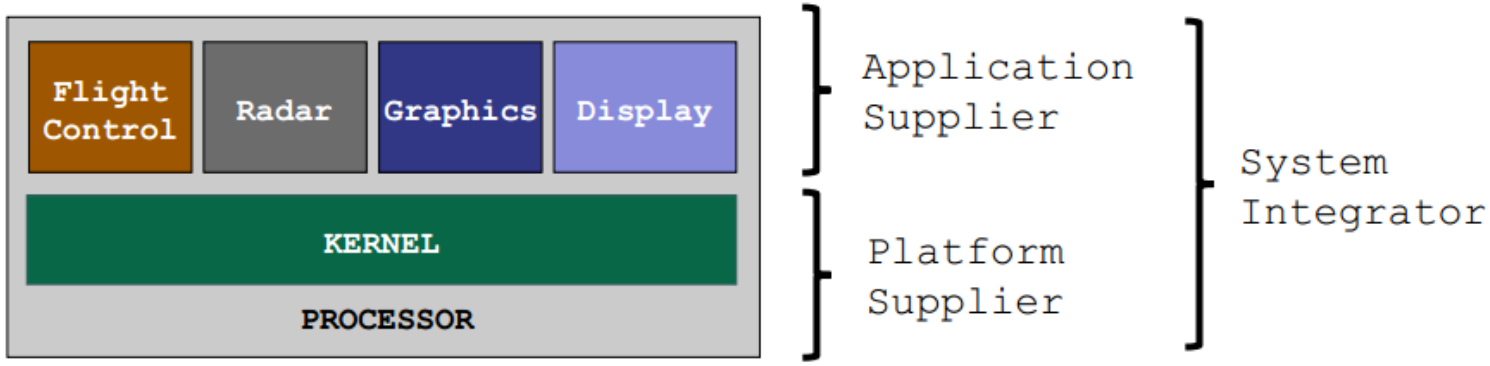
4PS – For Partitioned Systems

Features:

- **Role separation (DO-297)**
- **XML configuration**

Two architectures:

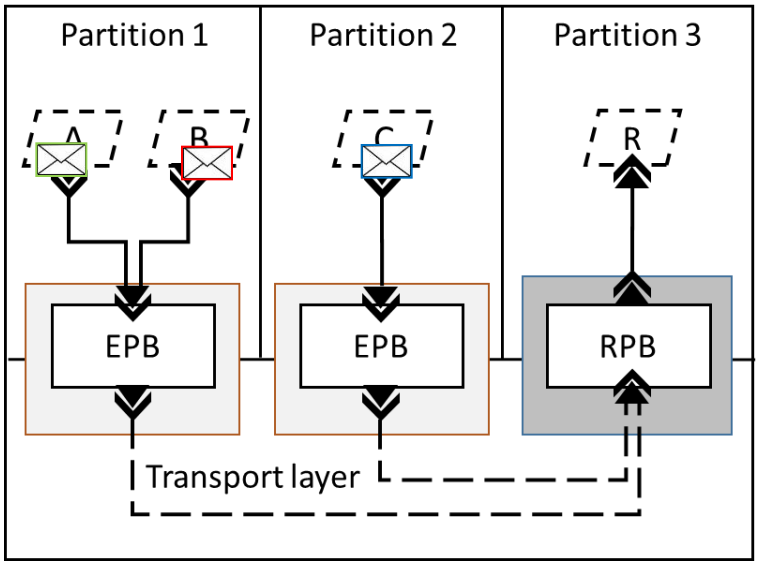
1. For platform suppliers
   - **Native implementation**
   - **Wait-free access protocol**
2. For application suppliers
   - **Reduce maintenance effort**



PDP 4PS : For Platform Supplier
Methodology

- Message Conditioning Before Transfer
  - Store msg in **EPB** under PDP policy
  - Check slot state before insertion
    - **No overproduction**
      - **Non-Propagation**
- Message Actual Transfer
  - By the **Partitioning Kernel**
  - Execute at receiver partition activation
  - Transfer only PDP available msg
    - **Statically determined**
      - **Message Identification**
- Message Conditioning After Transfer
  - Insert the msg in the RPB by the **Partitioning Kernel**

Platform supplier architecture

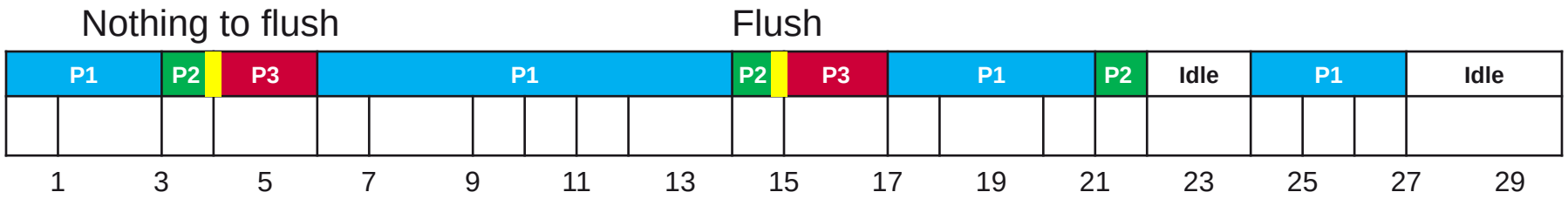


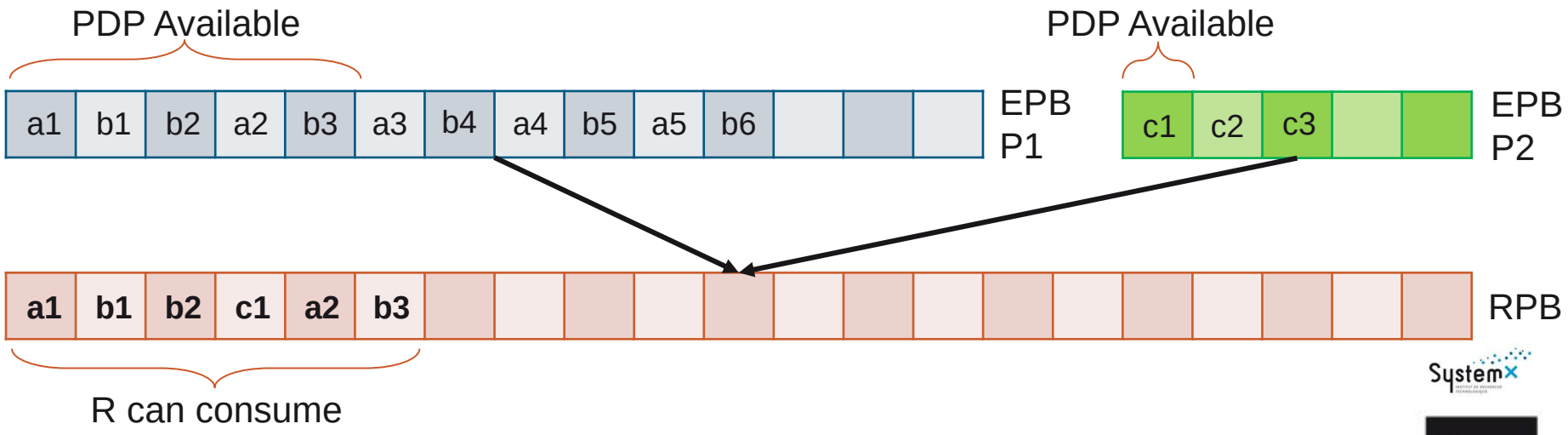
Transfer and insertion by Partitioning Kernel

PDP 4PS : For Platform Supplier
An example

Task	A	B	C	R
P	6	5	10	15
D	5	5	10	15
C	2	1	1	2

Partition	Task
P1 (RMS)	A, B
P2	C
P3	R

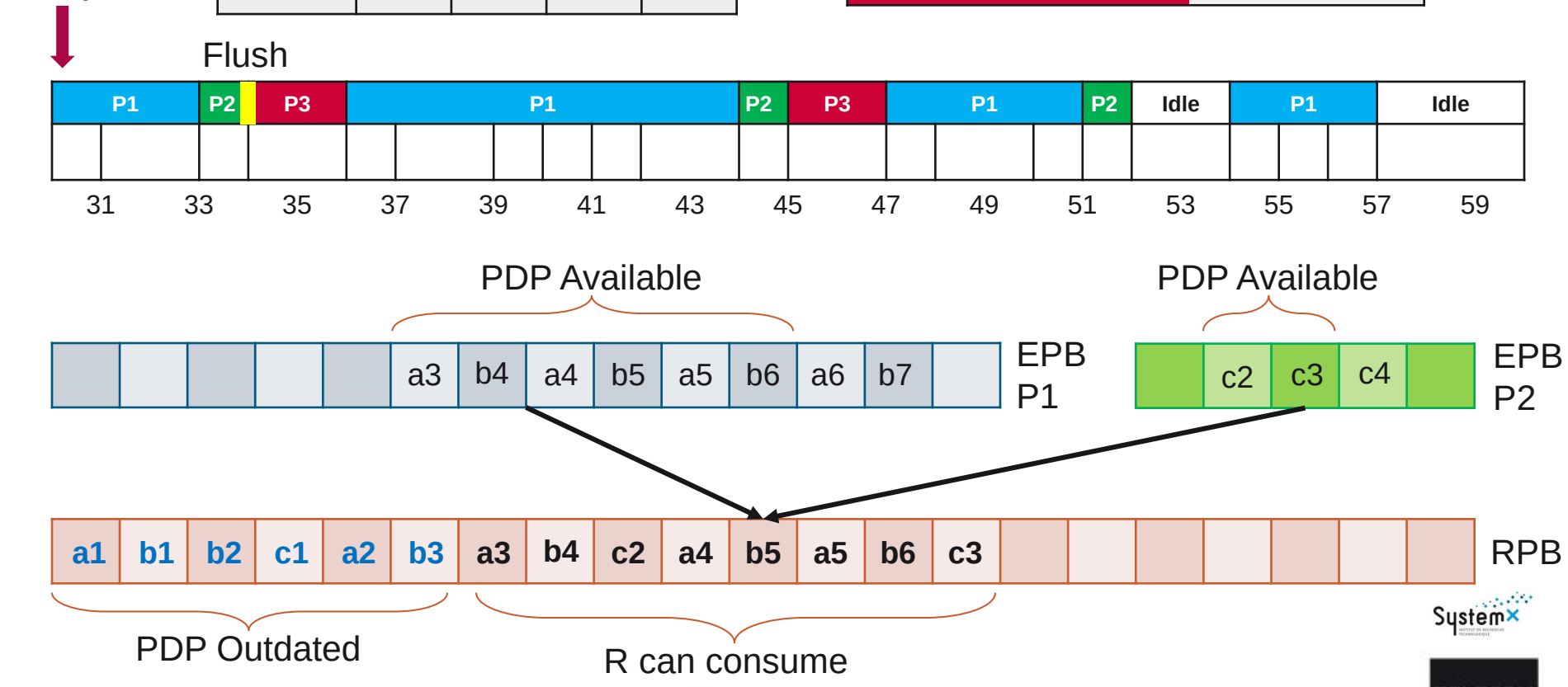




PDP 4PS : For Platform Supplier
An example

Task	A	B	C	R
P	6	5	10	15
D	5	5	10	15
C	2	1	1	2

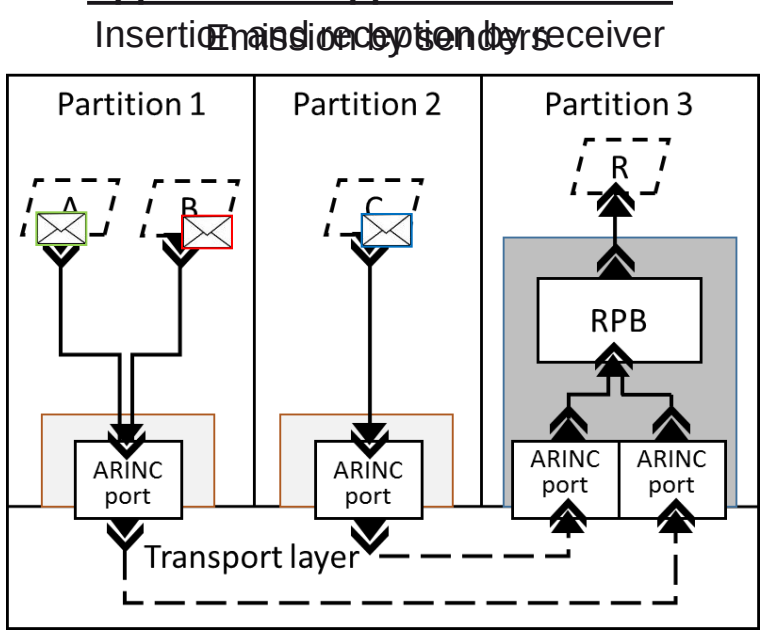
Partition	Task
P1 (RMS)	A, B
P2	C
P3	R



PDP 4PS : For Application Supplier
Methodology

- Message Conditioning Before Transfer
  - Store msg in **Partitioning Kernel sending port**
  - Maintain the number of sent msg by job
    - **No overproduction**
      - **Non-Propagation**
- Message Actual Transfer
  - **Partitioning Kernel original** transfer policy (e.g. FIFO)
  - Done before next receiver activation
- Message Conditioning After Transfer
  - Store msg in **Partitioning Kernel receiving ports**
    - **Message Identification**
  - Insert the msg in the RPB by the **receiver**

Application Supplier architecture



Transfer by Partitioning Kernel

Conclusion

- **N→1 Inter-Partition Communication mechanism**
  - Main properties
    - Periodic task set
    - Specific communication model
    - **Deterministic** message delivery order
    - **Non-Propagation** and **Message Identification** properties
  - Two architectures
    - For Platform supplier : wait-free
    - For Application supplier
  - Experimented on POK (Partitioned Open Kernel)

- **Further works**
  - Model transformation to produce kernel or application configuration
  - Adaptation to direct memory accesses

Thank you