Towards an HLA Run-time Infrastructure with Hard Real-time Capabilities (10E-SIW-011)

Pierre Siron
DMIA Department,
Université de Toulouse, ISAE

Jean-Baptiste Chaudron
ONERA/DTIM/SER
ONERA, Centre de Toulouse

Martin Adelantado
ONERA/DTIM/SER
ONERA, Centre de Toulouse

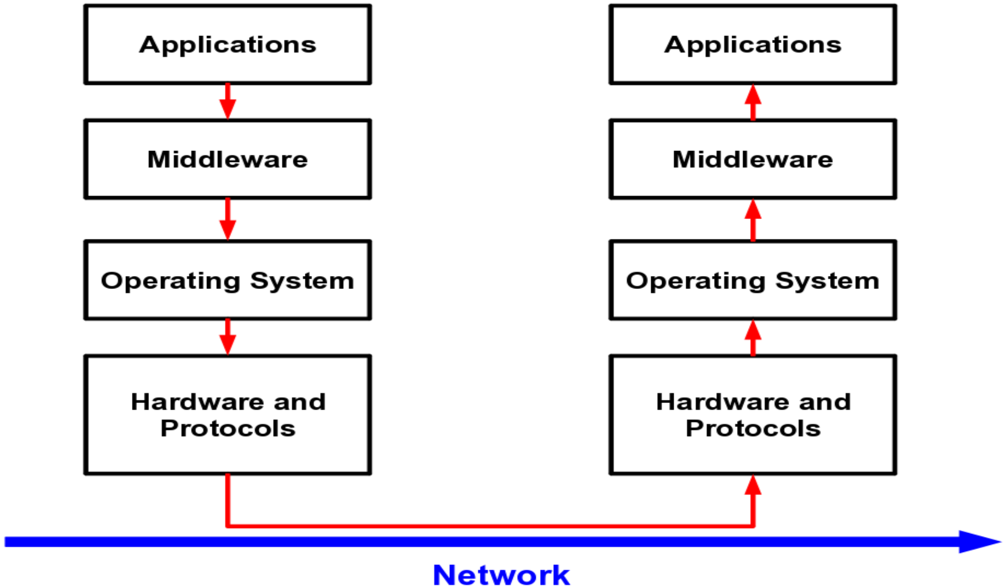
retour sur innovation

INTRODUCTION (1/4)

- Real Time Systems
  - Real time systems are defined like systems in which the correctness of the system ***not only depends on the logical results of computation***, but ***also*** on the time at which these results are produced:
    - *Hard Real Time:* a missed deadline is catastrophic (Command and Control Systems,...)
    - *Soft Real Time:* could accept an error rate for deadlines (Multimedia System, ...)
  - ***Always return right results in right times (deadlines predefined)***.

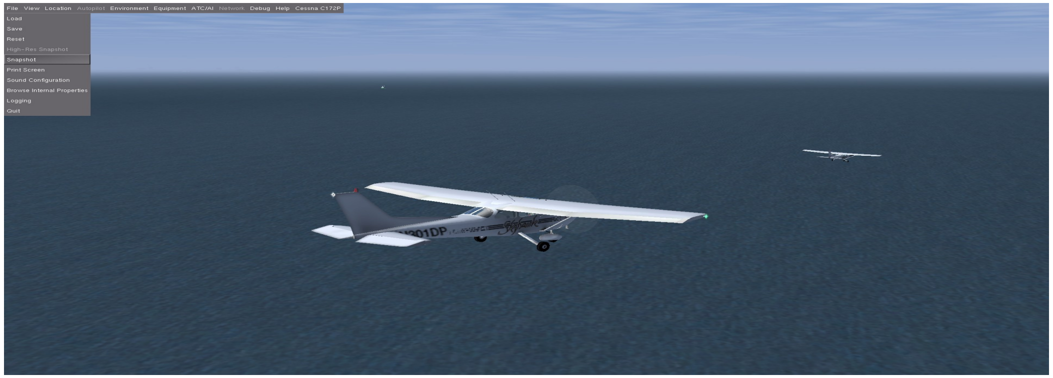
- Distributed Systems
  - Emergence of computer networks technologies;
  - A distributed system consists of ***different autonomous computers*** that communicate through ***a global (or local) network***;
  - The computers interact with each other in order to achieve a *global common goal*.

- Middleware Level
  - Development of standards (CORBA, RPC,...) to face consistently to problems involved by distribution (heterogeneous computers, network protocols):
    - *HLA standard* for distributed simulations (1.3 / IEEE 1516 / Evolved).
  - ***Middleware*** in computing terms is used to describe a software agent acting as an ***intermediary*** between different distributed processes:
    - *Run Time Infrastructure (RTI)* is the HLA compliant middleware.

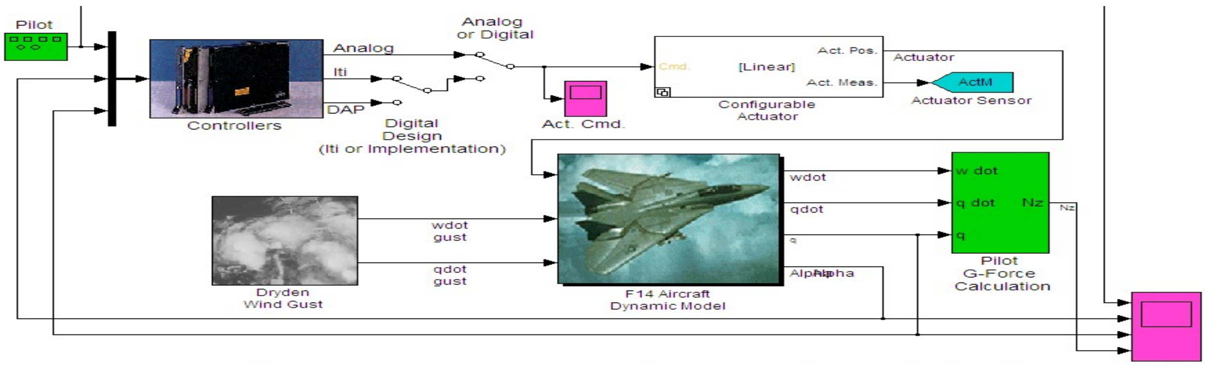


INTRODUCTION (3/4)

- <u>Targeted Applications</u>
  - Formation flying simulation (Xplane, Flight Gear, MS Flight Simulator,...)
    - ➔ Communication *between each simulator* with HLA



  - Hardware-in-the-loop and embedded systems simulations
    - ➔ Connecting *sensors* and *actuators* with HLA



P.Siron, M.Adelantado,
JB.Chaudron (July 12, 2010)

INTRODUCTION (4/4)

- Our goals
  - To use HLA standard to allow communication between several distributed process with timing constraints (real time tasks);
  - To understand *weaknesses* and *strengths* of this technology for real time using domain bibliography, different experiments and test cases;
  - To suggest a *formal model* in order to validate every Real time simulation compliant with HLA standard:
    - With a given RTI and an underlying operating system and hardware.

- Plan

Problem Description
- *HLA for Real Time ?*
- Action levels
- CERTI

(1)

Related Work
- Periodic Federates
- Time Management use
- Real Time RTI vision

(2)

Formal Model
- Basic Assumptions
- Precedence constraints
- WCET Evaluation

(3)

Illustration
- Original Test case
- Applying technique
- Results

(4)

- <u>To adapt current middleware standards for real time</u>
  - Traditional standards and middlewares architectures for distributed computing are not very suitable for supporting hard real time constraints;
  - Research community ***tries to adapt current middleware standards*** to include real times properties:
    - *RT CORBA Specifications* (*ref:* ***ORBOS/99-02-12***).
  - Works for hard real time HLA are ***less advanced*** than CORBA ones:
    - Different kind of works (R.Fujimoto&T.McLean, A.Boukerche, H.Zao, ...);
    - *No specifications for real time* in HLA standard documents.

- <u>HLA does not currently address hard real-time simulation</u>
  - HLA does not provide interfaces to specify ***end to end prediction requirement*** for federates;
  - HLA does not allow ***the management of underlying Operating(s) System(s)*** and ***Networks Protocols*** in term of priority or ressource;
  - HLA only supports ***two transportation types***:
    - the *reliable* one (usually encoded with TCP/IP network protocol);
    - the *best-effort* one (usually encoded with the UDP/IP network protocol).

- **Temporal properties for a real-time simulation are obtained from a complex combination of:**
  - the *application structure* **(4)**;
  - the *HLA middleware* used (implementation in the chosen language) **(3)**;
  - the *infrastructure software implementation* (operating system and communication protocols) **(2)**;
  - the *physical infrastructure* of execution (type of computers, network type and distribution topology) **(1)**;
- **Also:**
  - The *formal model* needed to validate the simulation **(5)**.

- **First Choices:**
  - We consider usual single monoprocessor system **(1)** ;
  - This system is running under *Linux Red Hawk* operating system (Posix Real time compliant) **(2)**;
  - We use CERTI HLA compliant RTI **(3)**.

Formal Level (5)
Formal method
to validate the system

Application Level (4)
Simulation model,
Task type,...

Middleware Level (3)
Standard, implementation

Software Level (2)
Operating System,
Programming language,...

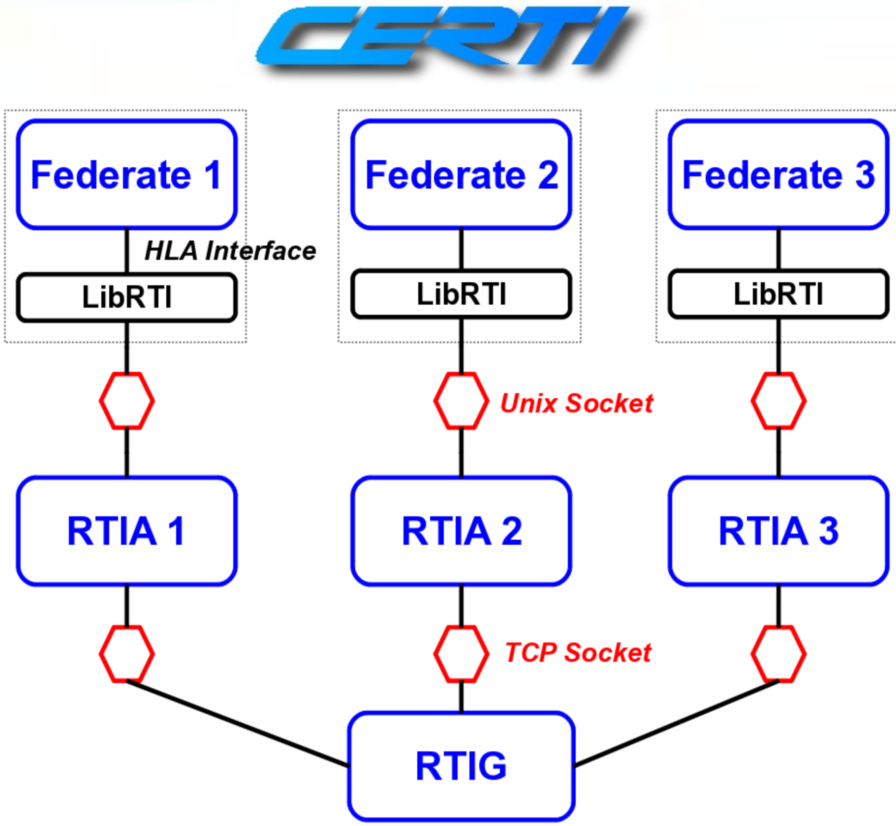
Hardware Level (1)
Processor, memory,
network,...

- *Open Source RTI* managed and maintained by Onera team (GPL):
  - *ref:* ***09S-SIW-015***.

- Developed in ***C++***;
- Architecture of communicating processes with ***TCP*** and ***UDP*** protocols;
- Available under ***Linux***, ***Unix*** and ***Windows*** operating systems.

- *Fully compliant* with 1.3 standard;
- *Not fully compliant* with IEEE 1516:
  - Work in progress.

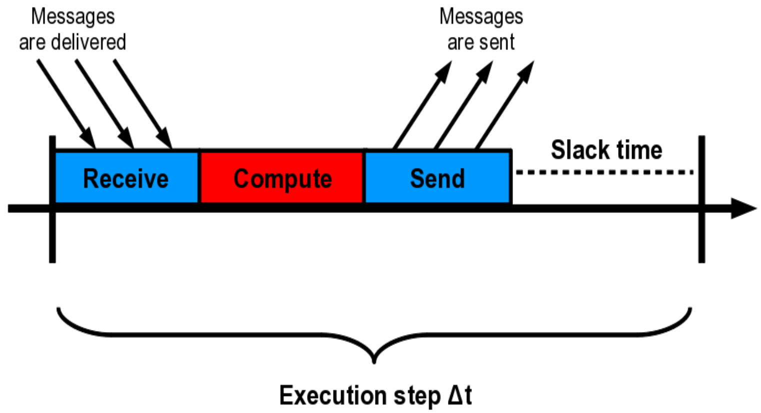
- Available at address:
  - *http://www.cert.fr/CERTI/*



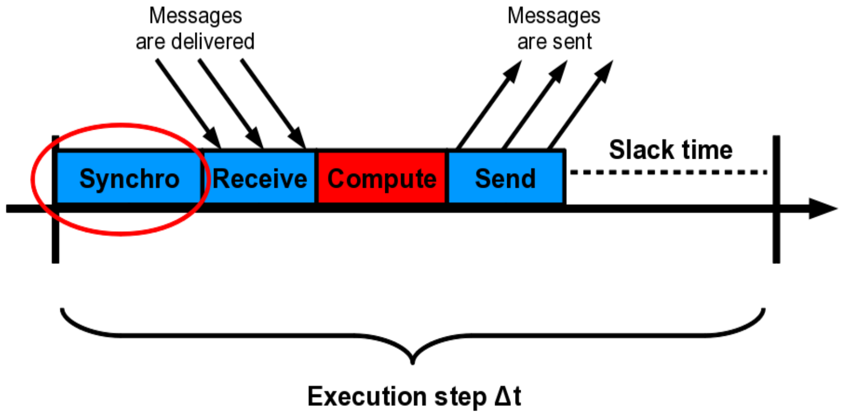
Middleware Level (3)
Standard, implementation

Repeatability within the simulations

- Concept introduced by Fujimoto and McLean;
- Federates repeat the ***same pattern*** of execution periodically (time step noted *Δt*).



- Each step is the execution of 4 phases:
  - (1) a *reception* phase;
  - (2) a *computation* phase;
  - (3) a *transmission* phase;
  - (4) a *slack time* phase.



- Onera's studies show the necessity of adding a *synchronization* phase that could be done by 3 techniques:
  - (1) Consulting an hardware clock;
  - (2) Sending an interaction which rhythms the simulation;
  - (3) Using time management algorithms.

Application Level (4)
Simulation model,
Task type,...

- Time management mechanisms
  - One of the main benefits of this simulation standard HLA;
  - To allow a ***consistent global time*** throughout the simulation and ***prevents causal anomalies*** effects;
  - Different kinds of algorithms:
    - First generation: *Null Message Algorithm* (K.M.Chandy&J.Misra);
    - Second Generation: *Global Virtual Time Algorithm* (*F.*Mattern).
- Usefulness of Time Management for real time simulation ?
  - To ensure respect of deadlines;
  - To keep ***consistency*** between the different federates cycles during their execution.
- Limitations for real time
  - ***First Generation:*** Latency due to *null message exchange* between federates (depend on *lookahead* parameter);
  - ***Second Generation:*** LBTS computation cannot generally be guaranteed to complete *within a bounded time* (Transient messages cause an LBTS computation to be aborted and retried).

- Different techniques allow a ***better use of system resources*** and also a ***higher reactivity*** for RTI services (HLA Compliant Middleware):
  - Real time Optimized RTI
    (1) *Multi-threaded* synchronous process for RTI (A.Boukerche, H.Zao,...);
    (2) Global scheduling services in RTI;
    (3) Use a real-time operating system to allow *preemptive priority scheduling* and *deterministic system call:*
      → For our test we use *Linux Red Hawk* Operating System.
  - Use of specific HLA service (given by the RTI)
    (1) Time management explained before (R.Fujimoto&T.McLean,...);
    (2) Data Distribution Management (A.Boukerche,...).
  - Distributed Case:Use of specific QoS communication protocols
    (1) RSVP protocol (H.Zao,...);
    (2) VRTP protocol (D.Bruzman&M.Zyda,...).

- Scheduling theory
  - A periodic task is defined as a quadruplet *< ri , Ci , Di , Pi >*:
    - (1) ***ri*** is the time of initial activation of the task;
    - (2) ***Ci*** is the worst case execution time;
    - (3) ***Di*** is the deadline;
    - (4) ***Pi*** is the period.
  - The principle is to verify that every set of real time tasks will ***respect its timing constraint*** on a given software/hardware.

Formal Level (5)
Formal method to validate the system

- No related work has linked *Scheduling theory* and *Distributed simulations*
  - Real-time simulations are usually validated by experiments rather than by formal model and schedulability analysis;
  - Here we describe a ***preliminary work*** focusing on achieving hard real-time properties for HLA federations running on a **single computer**;
  - Our ultimate objective is to achieve real-time capabilities for ***distributed HLA real time federations*** executions.

(1)Federate-RTIA pair = ***one real time task***;

(2) RTIG process is the highest priority task on the processor, it ***only runs when it's needed*** (for federates communication);

(3) The tasks therefore share the same reference time (the ***CPU clock)***:

- *Synchronization* by consulting this clock.

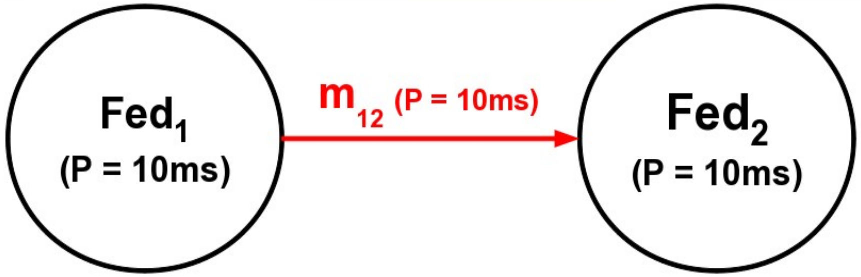
(4) Tasks communicate via service call **updateAttributesValues()**, we assume that the receiver federate is waiting for callback **reflectAttributesValues()** in the reception phase;

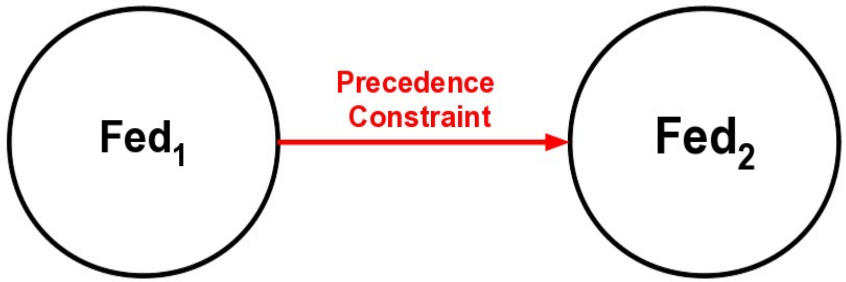
(5) We focus on ***Static scheduling algorithms***:

- Priorities for each task is calculated *before the simulation*.

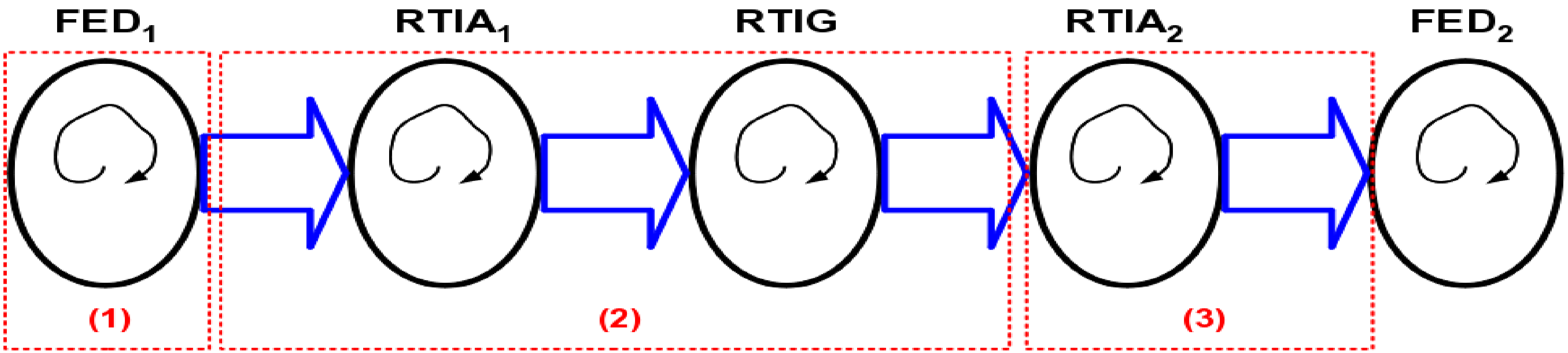
1. Creation
⇒ Federation is created by one of the federates ⇒ Others federates join federation
2. Initialization
⇒ Each federate states his intentions of publication and subscription ⇒ Establishment of time management policy (if used) ⇒ Synchronization phase between federates (if used) ⇒ Registration of simulated objects
3. Simulation Loop
⇒ Time Advance (for each federate if used) ⇒ Receipt of updates to subscribed data ⇒ Local computations (for each federate) ⇒ Sent updates to data published
4. Termination
⇒ Remove registered objects ⇒ Disable time management policy (if used)
5. Suppression
⇒ All Federate leave the federation ⇒ Destruction of federation by creator federate

- One Federate (and its RTIA) is a ***real time periodic task***;
- Federates communicate by using HLA principles:
  - **updateAttributeValues();**
  - **sendInteractions().**
- These communications could be represented by ***periodic messages***;
- These periodic messages are taken into account by ***a simple precedence constraint*** between tasks:
  - Sender and Receiver must run at the *same period*;
  - The task who produces the message must have *higher priority* than the receiver.





Formal Level (5)
Formal method
to validate the system



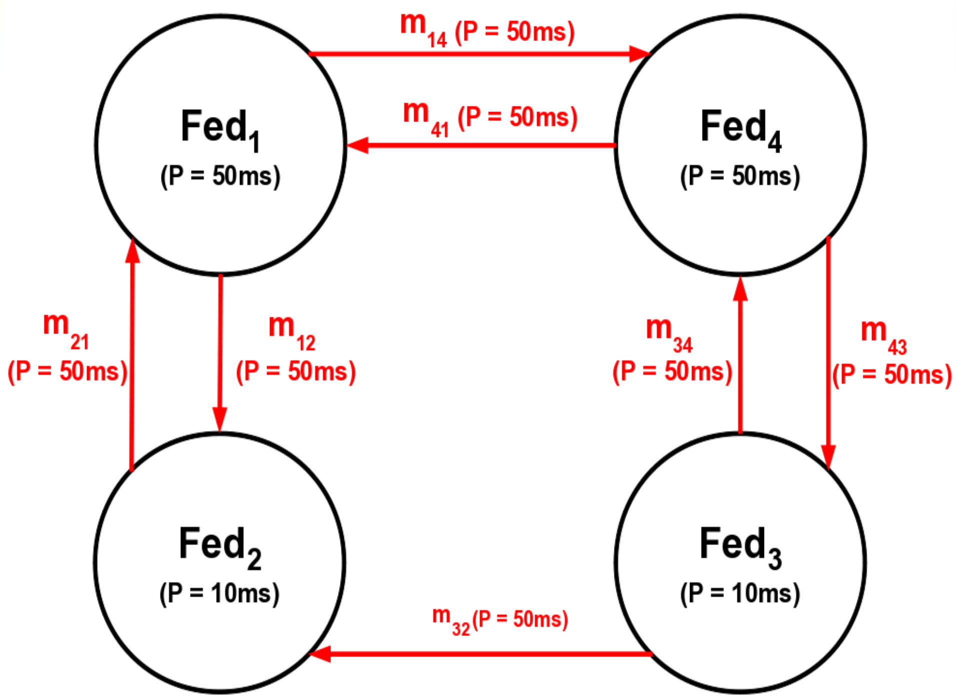
- ***WCET*** is a ***key parameter*** for scheduling analysis:
  - Determine the value of C_i parameter for a real time task.

- A task (Federate-RTIA) consists of three phases (No synchronisation phase):
  - a phase of receiving the data (RTIA compute and socket use) **(3)**;
  - a computation phase of the new data (only federate proper computation phase) **(1)**;
  - a phase of transmission of this new data (RTIA compute, socket use and RTIG also) **(2)**.

- WCET will be equal to ***the sum of the WCET for each phase***:
  - C_i = ***WCET(Receive) + WCET(Computation) + WCET(Send)***

• Test case from the collaborative work between ONERA and CNES:

- *ref:* ***08E-SIW-061***;
- Test case extracted from a *satellite formation flying* experiment;
- Simulation representing a set of *embedded systems*;
- Set of federates running periodically and exchanging messages periodically.



- System composed of 4 real time tasks:
  - Simple Precedence rules could not be apply;
  - Deadline = Period;
  - Execution time Ci = *10% x Period;*
  - We assume this execution time takes into account *every WCET phase* explained before.

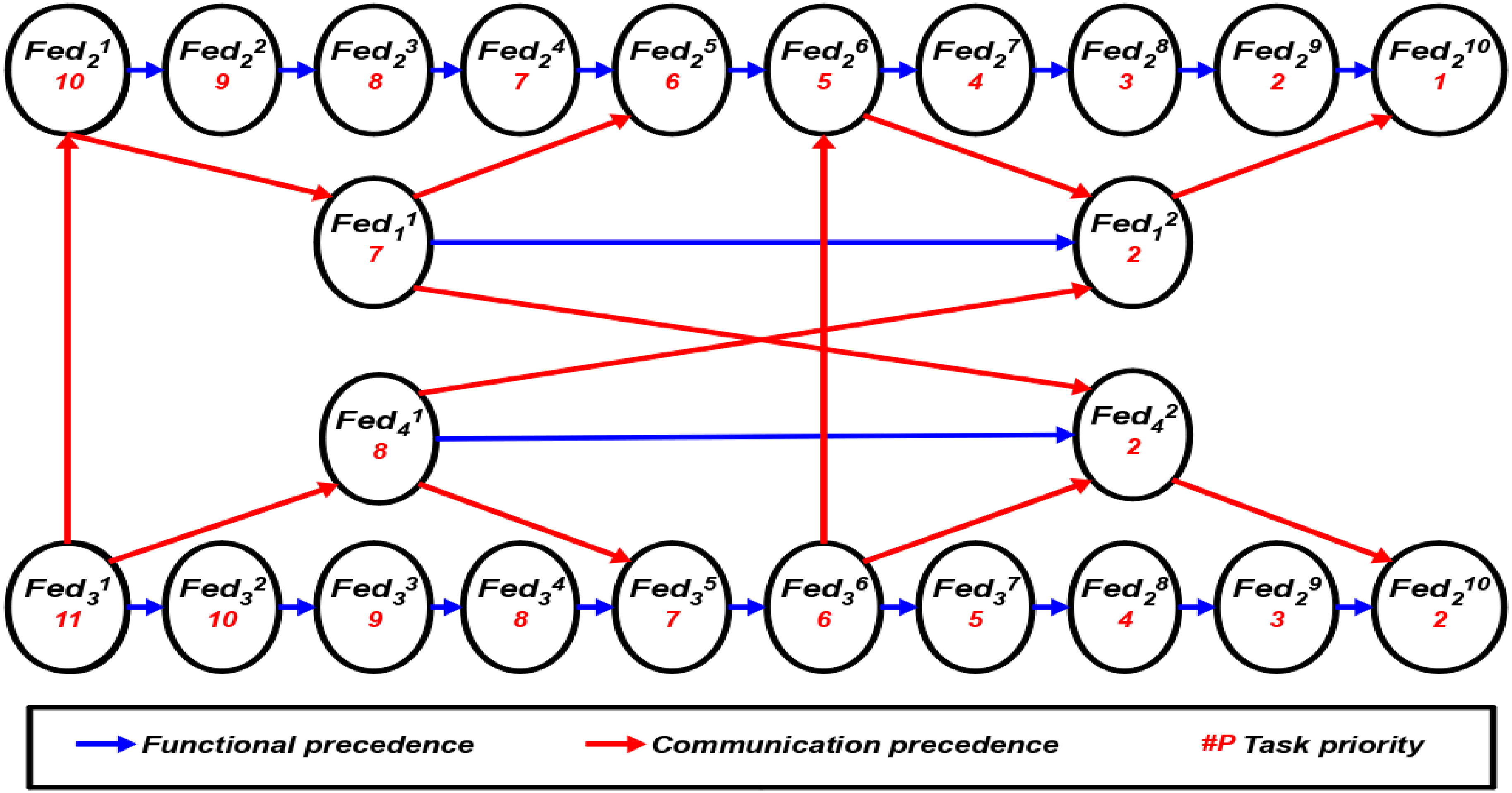
1. Fed_1 : <0, 5, 50 , 50> ;
2. Fed_2 : <0, 1 , 10 , 10> ;
3. Fed_3 : <0, 1 , 10 , 10> ;
4. Fed_4 : <0, 5 , 50 , 50> .

- Divide each previous task into a ***set of subtasks***;
- The period of each subtask is equal to the hyper-period for the set of basic tasks:
  - hyper-period is the *lcm* (least common multiple) of all tasks periods;
  - Test case: *lcm = 50ms* (12 subtasks).

- Cyclicity problem between ***Fed1*** and ***Fed4***:
  - They *run at the same period* and they *exchange messages at the same period too* (could take into account any precedence between them);
  - Solution: We extend the subdivision by taking 2 times hyper-periods *2xlcm = 100ms*.

- We obtain a set of ***24 subtasks*** each with the same period and their proper deadlines:
  - We can apply *Deadline Monotonic* techniques to schedule this set of tasks;
  - We can apply *Simple Precedence constraint* techniques.

1. $Fed_1^1 : <0, 5, 50, 100>$;
2. $Fed_1^2 : <0, 5, 100, 100>$;
3. $Fed_2^1 : <0, 1, 10, 100>$;
4. $Fed_2^2 : <0, 1, 20, 100>$;
5. $Fed_2^3 : <0, 1, 30, 100>$;
6. $Fed_2^4 : <0, 1, 40, 100>$;
7. $Fed_2^5 : <0, 1, 50, 100>$;
8. $Fed_2^6 : <0, 1, 60, 100>$;
9. $Fed_2^7 : <0, 1, 70, 100>$;
10. $Fed_2^8 : <0, 1, 80, 100>$;
11. $Fed_2^9 : <0, 1, 90, 100>$;
12. $Fed_2^{10} : <0, 1, 100, 100>$;
13. $Fed_3^1 : <0, 1, 10, 100>$;
14. $Fed_3^2 : <0, 1, 20, 100>$;
15. $Fed_3^3 : <0, 1, 30, 100>$;
16. $Fed_3^4 : <0, 1, 40, 100>$;
17. $Fed_3^5 : <0, 1, 50, 100>$;
18. $Fed_3^6 : <0, 1, 60, 100>$;
19. $Fed_3^7 : <0, 1, 70, 100>$;
20. $Fed_3^8 : <0, 1, 80, 100>$;
21. $Fed_3^9 : <0, 1, 90, 100>$;
22. $Fed_3^{10} : <0, 1, 100, 100>$;
23. $Fed_4^1 : <0, 5, 50, 100>$;
24. $Fed_4^2 : <0, 5, 100, 100>$.



- Schedulability conditions are satisfied by ***CHEDDAR*** Open Source Tool (GPL):
  - *http://beru.univ-brest.fr/~singhoff/cheddar/*

SUMMARY (1/1)

- <u>We propose an analysis to validate a hard real time simulation</u>
  - High granularity model with strong initial assumptions;
  - We use of monoprocessor schedulability analysis:
    - *Deadline Monotonic* techniques;
    - *Simple precedence constraint* techniques.
  - We show the feasibility of the formal validation for an HLA simulation.

- <u>Implementation of the model on Linux operating system</u>
  - We use ***Posix Real time*** compliant system call API:
    - High resolution timers;
    - Processor affinity;
    - SCHED_FIFO Linux scheduler;
    - Paging memory management.
  - We extend CERTI API ***to allow priority handling*** of federates processes and both RTIA and RTIG processes during the simulation runtime.

FUTURE TRENDS (1/2)

- Studies on low level granularity model (work in progress)
  - Similar techniques;
  - Each simulation process becomes a real time task:
    - Federate processes, RTIA processes and RTIG process.
  - Closer to what's really happening during the run:
    - The model has *exactly the number* of concurrent processes;
    - Better evaluation of *communication impact* on each socket.
  - Experiments and validation.

- Formal model extension for distributed case
  - In distributed case, determinism is guaranteed only if the underlying network supports *timely delivery* of messages;
  - We should investigate a novel approach to take into account a formal model for the tasks (*execution units*) and also for the messages (*communication units*);
  - We hope this new approach will help for *validation of a distributed simulation* using CERTI.

FUTURE TRENDS (2/2)

- To add deterministic mechanism to CERTI
  - C++ has some ***gap for real time*** like memory allocation:
    - *Unbounded time* to compute in its original version algorithm.
  - To use ***predictable allocation techniques*** and algorithms:
    - For example, the Open Source *TLSF library* (GPL);
    - Available at : *http://rtportal.upv.es/rtmalloc/*
  - To analyze ***RTIA and RTIG computational complexity*** to get a better ***WCET*** estimate.

- To increase the CERTI performance
  - To evaluate the use of ***multi-threadings*** for RTIG central process;
  - To implement ***shared memory transportation*** (work in progress):
    - For communication between *each CERTI process on the same processor*;
    - Federate / RTIA communication by *Ring Buffer shared memory*;
    - RTIA / RTIG communication *usual shared memory segment*.