How to solve ODEs in real-time HLA distributed simulation

Jean-Baptiste Chaudron, David Saussié, Pierre Siron, Martin Adelantado

How to solve ODEs in real-time HLA distributed simulation

Jean-Baptiste Chaudron
Institut Supérieur de l'Aéronautique et de l'Espace
10 av. Edouard Belin, BP 54032, 31055 Toulouse, France
jean-baptiste.chaudron@isae.fr

David Saussié
Ecole Polytechnique de Montréal
C.P. 6079, Montréal (QC), H3C 3A7, Canada
d.saussie@polymtl.ca

Pierre Siron
Institut Supérieur de l'Aéronautique et de l'Espace
10 av. Edouard Belin, BP 54032, 31055 Toulouse, France
pierre.siron@isae.fr

Martin Adelantado
Office National d'Etudes et de Recherches Aérospatiales
2 av. Edouard Belin, BP 4025, 31055 Toulouse, France
martin.adelantado@onera.fr

Keywords:
Simulation, ODEs, HLA, RTI, real-time, scheduling.

ABSTRACT: *In the context of the Research Platform for Embedded Systems Engineering (PRISE) Project, we are developing and maintaining a complete aircraft flight simulation using the High Level Architecture (HLA), an IEEE standard for distributed simulation. This complex distributed simulation is composed of different distributed HLA simulators (e.g., Flight Dynamics, Sensors), whose dynamic behaviors are implemented as Ordinary Differential Equations (ODEs). The resolution of these equations is done, locally for each simulator, by numerical integration with methods like Euler or Adams-Bashforth. The global behavior of this distributed simulation, where each component runs its own local resolution, is a key challenge. The main problem is to ensure the global simulation consistency and, in particular, the specific data flows between components with the correct temporal real-time behavior. This paper specifically addresses the problem of solving ODEs over an HLA distributed architecture and offers a complete study (specifications, implementation and validation) where several theoretical concepts and methods are discussed.*

1. Introduction

The main goal of the Research Platform for Embedded Systems Engineering (PRISE) project is to provide students and researchers with a platform for the study, evaluation and validation of new embedded system concepts, architectures and techniques through a dedicated hardware and software environment. Modern embedded systems become more and more complex with an increasing number of both components and interactions between them and the use of simulation is essential for relevant studies. In this context, we implemented from scratch and we are currently maintaining an aircraft component-based simulation composed of several distributed simulators [1], [2]. Each simulator represents a specific part of the aircraft (e.g., engines, actuators, control devices, flight control laws), its dynamic in the environment (flight dynamics) or the environment itself. An overview of this distributed simulation is depicted in Figure 1.

This flight simulation, called SDSE (French acronym for Distributed Simulation of Embedded Systems) has been implemented by using the HLA IEEE standard [3], [4], [5] which provides a well-known simulation framework for composability, maintenance and reusability for this complex simulation. Moreover, HLA provides time management mechanisms [6], guaranteeing a consistent global logical time throughout the whole simulation, which is one of the main benefits of this simulation standard. Previous works showed that these time management mechanisms could be used to ensure the good behavior during run time,

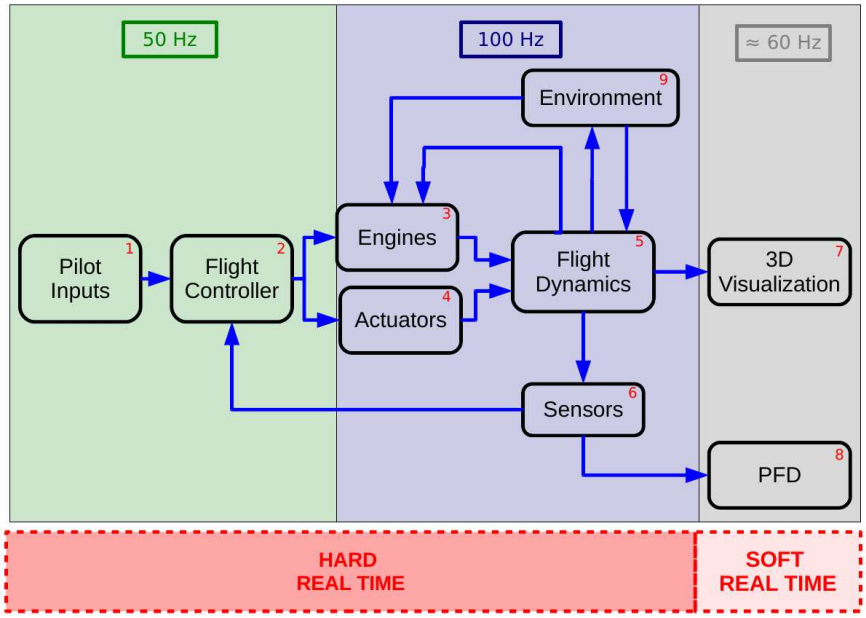

Figure 1. Overview of the SDSE aircraft simulation

especially in real-time simulations [7], [8], which is typically an aspect of the SDSE simulation. From the software point of view, we are using the open source CERTI [9] Run Time Infrastructure (HLA middleware implementation) because we have a complete control on its implementation. We have shown that it is suitable for interconnection of simulators (i.e., HLA federates) with short computation and communication cycles [10], [11]. Additionally, we have also investigated the usage of CERTI time management techniques and implementations for real-time simulations [12] and especially for SDSE.

Scheduling a real-time federation (i.e., an HLA simulation) consists of two complementary facets that must be distinguished, namely, the *temporal scheduling* and the *functional scheduling*.

1) The *temporal scheduling* belongs to scheduling theory. Here, the objective is to formally verify that the distributed simulation will run while ensuring compliance with the real-time constraints according to its specifications. This aspect has been addressed in [2] and is not the purpose of the present study.
2) The *functional scheduling* ensures that the sequence of the communications and computations between each simulator will provide global simulation results that will always be fair and relevant according to simulation model semantics. HLA time management mechanisms are providing a good baseline for this purpose. In the SDSE simulation, each federate implements a dedicated specific algorithm and some of these models are using Ordinary Differential Equations (ODEs) [13]. Therefore, the distributed resolution of these ODEs is quite complex and needs to be clearly analyzed; this is the purpose of the paper.

Also, the notion of "*time*" within a real-time simulation is an important notion which needs to be clearly stated for consistent temporal and functional scheduling study. Three different concepts can be associated to the notion of time, each one describing a specific aspect [14]:

1) The *physical time*, denoted by t_{PHY}, is the reference time of the physical system under study (the source system that we want to model and to simulate).
2) The *simulated time*, denoted by t_{SIM}, is a representation of the physical time within the simulation. It is a set of ordered values representing different time instants for the modeled physical system.
3) *The absolute time* or *wall-clock time*, denoted by t_{WCT}, is the time elapsed while executing the simulation. It can be measured by a hardware clock, like the CPU clock.

In our real-time simulations, with the usage of conservative time management methods, these times might be equal (with respect to a certain unit and a certain precision). Therefore, during the run of a simulation, we have:

$$t_{PHY} = t_{SIM} = t_{WCT} \tag{1}$$

For example, if the *millisecond* is considered as the reference unit, 1 millisecond of the real physical system is equal to 1 unit of logical HLA simulation time, which is also equal to 1 millisecond of execution time.

To resume, this paper focuses on the study of functional scheduling for HLA simulations (i.e., federations) using HLA time management semantics where some simulators (i.e., federates) are modeled by ODEs. Based on our experience with the SDSE aircraft simulation development, we present here a detailed analysis from initial ODEs characterization to distributed resolution description. The paper is organized as follows:

- Section 2 describes ODEs characterization for our real-time simulation;
- Section 3 outlines the resolution of ODEs for federation with a communication sequence;
- Section 4 presents the resolution of ODEs for federation with a communication loop;
- Section 5 summarizes these concepts and shows an illustration for the SDSE aircraft simulation.

2. ODEs Characterization

2.1 State-Space Description

The dynamic behavior of a system can usually be modeled by a set of Ordinary Differential Equations (ODEs) and/or Partial Derivative Equations (PDEs) derived from different laws of physics. Under the action of external excitation (*inputs*) and/or initial conditions, the system evolves accordingly, and one can then monitor signals of interest (*outputs*), representative of the system behavior. In this paper, we suppose that the federates simulate systems governed by ODEs and described by state-space models of the form:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t) \tag{2}$$

$$\mathbf{y}(t) = \mathbf{g}(\mathbf{x}(t), \mathbf{u}(t), t) \tag{3}$$

$$\mathbf{x}(0) = \mathbf{x}_0 \tag{4}$$

where $\mathbf{x} = [x_1, \ldots, x_n]^\top \in \mathbb{R}^n$ denotes the state vector composed of n state variables x_i, $\mathbf{u} = [u_1, \ldots, u_m]^\top \in \mathbb{R}^m$ the input vector, $\mathbf{y} = [y_1, \ldots, y_p]^\top \in \mathbb{R}^p$ the output vector and $\dot{\mathbf{x}}(t) := \mathrm{d}\mathbf{x}(t)/\mathrm{d}t$. The *state equation* (2) is a set of n coupled first-order ODEs, in which each time derivative $\dot{x}_i$ is expressed in terms of the state variables $x_1, \ldots, x_n$ and the system inputs $u_1, \ldots, u_m$. The state equation represents the dynamic behavior of the system under the action of the inputs and the initial conditions defined by equation (4). The *output equation* (3) yields any signal of interest depending on state variables and/or input variables. In the general case, $\mathbf{f}(.)$ et $\mathbf{g}(.)$ are non-linear vector functions.

From an HLA point of view, a federate subscribes to object attributes that will constitute the input vector that drives the state equation, and publishes object attributes yielded by the output equation.

2.2 Numerical methods for ODEs

Numerical solutions of ODEs can be performed by a large variety of methods depending on the nature of the problem and the desired accuracy [13]. Starting from an initial point, a numerical method takes a short step forward in time to find the next point, and so on. Among all the available numerical methods, one can find single-step methods (e.g., Euler method), Runge-Kutta methods, and multistep methods (e.g., Adams-Bashforth and Adams-Moulton methods). These methods imply the choice of an adequate time step Δt, which can have a significant impact on the accuracy of the solution; they all lead to recurrence equations. For instance, the forward Euler method applied to original state equation (2) leads to the recurrence equation (5):

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \Delta t\, \mathbf{f}(\mathbf{x}_n, \mathbf{u}_n, t_n) \tag{5}$$

where n denotes here the iteration, $t_n = n\Delta t$ the corresponding time, and $\mathbf{x}_n = \mathbf{x}(t_n)$, $\mathbf{u}_n = \mathbf{u}(t_n)$ the values of $\mathbf{x}$ and $\mathbf{u}$ at t_n. The Euler method is an explicit method in the sense that it calculates the next state value $\mathbf{x}_{n+1}$ from previous values $\mathbf{x}_n$ and $\mathbf{u}_n$. By extension, explicit Adams-Bashforth methods, can be written using s previous values as in equation (6):

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \Delta t \sum_{k=0}^{s} b_k \mathbf{f}(\mathbf{x}_{n-k}, \mathbf{u}_{n-k}, t_{n-k}) \tag{6}$$

where the b_k are chosen such that the method has order s. As an example, the second-order and third-order Adams-Bashforth methods are respectively given by equations (7)-(8):

$$\text{(AB2)} \qquad \mathbf{x}_{n+1} = \mathbf{x}_n + \Delta t \left(\frac{3}{2}\mathbf{f}(\mathbf{x}_n, \mathbf{u}_n, t_n) - \frac{1}{2}\mathbf{f}(\mathbf{x}_{n-1}, \mathbf{u}_{n-1}, t_{n-1}) \right) \tag{7}$$

$$\text{(AB3)} \qquad \mathbf{x}_{n+1} = \mathbf{x}_n + \Delta t \left(\frac{23}{12}\mathbf{f}(\mathbf{x}_n, \mathbf{u}_n, t_n) - \frac{4}{3}\mathbf{f}(\mathbf{x}_{n-1}, \mathbf{u}_{n-1}, t_{n-1}) + \frac{5}{12}\mathbf{f}(\mathbf{x}_{n-2}, \mathbf{u}_{n-2}, t_{n-2}) \right) \tag{8}$$

On the opposite, there are implicit methods such as the backward Euler method described here in equation (9):

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \Delta t\, \mathbf{f}(\mathbf{x}_{n+1}, \mathbf{u}_{n+1}, t_{n+1}) \tag{9}$$

where one needs to solve a non-linear algebraic equation at every time step in order to produce the next value $\mathbf{x}_{n+1}$; this is usually achieved by using fixed point iteration or Newton-Raphson method. The computational cost is obviously higher, but implicit methods are usually more stable than their explicit counterparts. Nevertheless, the implicit methods are rejected in this HLA simulation context as they necessitate the value of the input $\mathbf{u}_{n+1}$ at the next step $n+1$, i.e., in the future.

2.3 Direct feedthrough

The output equation (3) is an algebraic equation that produces the output vector $\mathbf{y}(t)$ containing the signals of interest. Generally, the output vector $\mathbf{y}(t)$ may depend on both the state vector $\mathbf{x}(t)$ and the input vector $\mathbf{u}(t)$, and eventually, explicitly on the time t, as in equation (3). Contrary to the state equation (2) which requires numeric solvers as shown above, the discretization of the output equation, described in (10), is straightforward:

$$\forall t, \mathbf{y}(t) = \mathbf{g}\left(\mathbf{x}(t), \mathbf{u}(t), t\right) \Rightarrow \mathbf{y}_n = \mathbf{g}\left(\mathbf{x}_n, \mathbf{u}_n, t_n\right) \tag{10}$$

Nevertheless, special attention must be paid to the presence, or not, of the input vector $\mathbf{u}(t)$ (or $\mathbf{u}_n$) in the output equation. When the system output $\mathbf{y}(t)$ depends on the input vector $\mathbf{u}(t)$ (or on some of the inputs u_i), it is called *direct feedthrough*. As it will be presented in Sections 3 and 4, this can reveal particularly intricate when simulating a distributed system. In the case where there is no direct feedthrough, the output equation (11) is:

$$\forall t, \mathbf{y}(t) = \mathbf{g}\left(\mathbf{x}(t), t\right) \Rightarrow \mathbf{y}_n = \mathbf{g}\left(\mathbf{x}_n, t_n\right) \tag{11}$$

2.4 Typical execution within a federate

The choice of the numerical method and the time step Δt depends on the characteristics of the system but is not discussed here. The discretization of a state-space model with an explicit linear multi step method leads to the following discreet system (12):

$$\left\{ \begin{array}{rcl} \dot{\mathbf{x}}(t) &=& \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t) \\ \mathbf{y}(t) &=& \mathbf{g}(\mathbf{x}(t), \mathbf{u}(t), t) \\ \mathbf{x}(0) &=& \mathbf{x}_0 \end{array} \right. \Rightarrow \left\{ \begin{array}{rcl} \mathbf{x}_{n+1} &=& \mathbf{x}_n + \Delta t \sum_{k=0}^{s} b_k \mathbf{f}(\mathbf{x}_{n-k}, \mathbf{u}_{n-k}, t_{n-k}) \\ \mathbf{y}_n &=& \mathbf{g}(\mathbf{x}_n, \mathbf{u}_n, t_n) \\ \mathbf{x}(0) &=& \mathbf{x}_0 \end{array} \right. \tag{12}$$

At iteration n, knowing $\mathbf{x}_n$ and $\mathbf{u}_n$, the output equation is first executed to provide $\mathbf{y}_n$, then the state equation yields the state vector update $\mathbf{x}_{n+1}$ for the next iteration. Note that depending on the numerical methods, previous values $\mathbf{x}_{n-k}$ and $\mathbf{u}_{n-k}$ must be stored for the recurrence equation.

For the sake of simplicity but without loss of generality, we will consider in the following linear time-invariant (LTI) systems, i.e, systems described by linear differential and algebraic equations with constant coefficients. The original equations (2)-(3) can be rewritten as linear equations (13)-(14):

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \tag{13}$$

$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t) \tag{14}$$

where $\mathbf{A}, \mathbf{B}, \mathbf{C}$ and $\mathbf{D}$ are respectively $n \times n$, $n \times m$, $p \times n$ and $p \times m$ constant matrices. For example, the application of the forward Euler method yields equations (15)-(16):

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \Delta t \left(\mathbf{A}\mathbf{x}_n + \mathbf{B}\mathbf{u}_n\right) \tag{15}$$

$$\mathbf{y}_n = \mathbf{C}\mathbf{x}_n + \mathbf{D}\mathbf{u}_n \tag{16}$$

In the following, the influence of direct feedthrough on simulation, is illustrated on a first example with a federation involving federates with a communication sequence. Thereafter, we pursue the study on a federation that presents communication loop between federates. These studies will lead to the establishment of simple rules for the automatic implementation of consistent functional scheduling for these types of federation.

3. Federation with a Communication Sequence

Let an HLA federation composed of 4 federates which are communicating in series as illustrated in Figure 2. We study the impact of direct feedthrough in some of the federates.

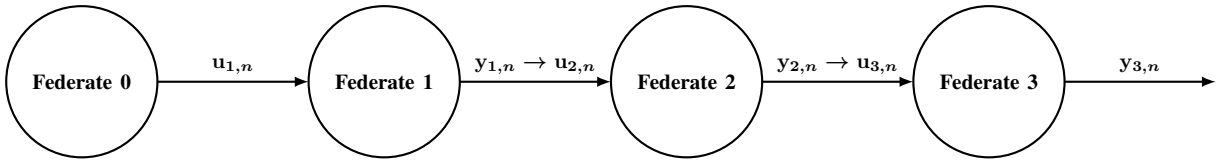

Figure 2. Federation with federates communicating in series

3.1 Standard case

Federate 0 is a federate which provides a signal $\mathbf{u}_{1,n}$ (every cycle n). For the sake of simplicity but without loss of generality, it is assumed that the numerical solution of the state-space models representing the dynamic behavior of the next 3 federates is described in equations (17)-(18):

$$\mathbf{x}_{i,n+1} = \mathbf{A}_i \mathbf{x}_{i,n} + \mathbf{B}_i \mathbf{u}_{i,n} \tag{17}$$

$$\mathbf{y}_{i,n} = \mathbf{C}_i \mathbf{x}_{i,n} \tag{18}$$

where $i \in 1, 2, 3$ denotes the federate index. The federates are linked in series such that the output of Federate i is entirely the input of Federate $i+1$, i.e., $\forall i \in 2, 3$, $\mathbf{u}_{i,n} = \mathbf{y}_{i-1,n}$. We can see that these federates do not exhibit direct feedthrough in their respective output equation, and therefore no special care is needed for the execution of the federates. This gives the following global system with equations (19)-(20):

$$\begin{bmatrix} \mathbf{x}_{1,n+1} \\ \mathbf{x}_{2,n+1} \\ \mathbf{x}_{3,n+1} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_1 & \mathbf{0} & \mathbf{0} \\ \mathbf{B}_2\mathbf{C}_1 & \mathbf{A}_2 & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_3\mathbf{C}_2 & \mathbf{A}_3 \end{bmatrix} \begin{bmatrix} \mathbf{x}_{1,n} \\ \mathbf{x}_{2,n} \\ \mathbf{x}_{3,n} \end{bmatrix} + \begin{bmatrix} \mathbf{B}_1 \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix} \mathbf{u}_{1,n} \tag{19}$$

$$\begin{bmatrix} \mathbf{y}_{1,n} \\ \mathbf{y}_{2,n} \\ \mathbf{y}_{3,n} \end{bmatrix} = \begin{bmatrix} \mathbf{C}_1 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{C}_2 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{C}_3 \end{bmatrix} \begin{bmatrix} \mathbf{x}_{1,n} \\ \mathbf{x}_{2,n} \\ \mathbf{x}_{3,n} \end{bmatrix} \tag{20}$$

From the initial conditions $[\mathbf{x}_{1,0}\, \mathbf{x}_{2,0}\, \mathbf{x}_{3,0}]^\top$, we can calculate the state and output for every iteration step $[\mathbf{y}_{1,n}\, \mathbf{y}_{2,n}\, \mathbf{y}_{3,n}]^\top$. We assume that, from initial moment $t = 0$, all states from Federates 1, 2 and 3 are $\mathbf{x}_{i,0}$, Δt is the time step between cycle and $lk \leq \Delta t$ is the *lookahead*. Federate 0 can be implemented according the pseudo-code described in Table 1 and the other Federates as in Table 2.

In the remainder of the paper, for clarity sake, acronyms will be used to denote well-known HLA services and callbacks:

- UAV stands for `updateAttributesValues(...);`
- RAV for `reflectAttributesValues(...);`
- TAR for `timeAdvanceRequest(...);`
- NER for `nextEventRequest(...);`
- and TAG for `timeAdvanceGrant(...).`

Also, we consider the case where each federate is regulator and constrained for HLA time management mechanisms and we do not consider the *zero-lookahead* case (i.e., $lk = 0$).

Figure 3 illustrates the temporal behavior of this federation with $\Delta t = 10$ and $lk = 1$. Letters *O* and *S* describe respectively output equation calculation and state equation calculation in every federate of interest.

1) *Initialization phase*
   a) $n \leftarrow 0$
2) *Simulation phase (while loop)*
   a) Update attributes: UAV($n\Delta t + lk$) for $\mathbf{u}_{1,n}$
   b) Ask for HLA time advance to cycle $n+1$ for local time $(n+1)\Delta t$: TAR($(n+1)\Delta t$)
   c) HLA time advance is granted to cycle $n+1$ for local time $(n+1)\Delta t$: TAG($(n+1)\Delta t$)
   d) $n \leftarrow n+1$

Table 1. Federate 0 pseudo-code

1) *Initialization phase*
   a) $n \leftarrow 0$
   b) Initial choice for $\mathbf{x}_{i,0}$ value
2) *Simulation phase (while loop)*
   a) Output calculation $\mathbf{y}_{i,n}$ from state value $\mathbf{x}_{i,n}$
   b) Publish attributes: UAV ($n\Delta t + lk$) for $\mathbf{y}_{i,n}$
   c) Request for HLA time advance to cycle $n+1$ for local time $(n+1)\Delta t$: TAR($(n+1)\Delta t$)
   d) Receive attributes $\mathbf{u}_{i,n}$: RAV($n\Delta t + lk$)
   e) HLA time advance is granted to cycle $n+1$ for local time $(n+1)\Delta t$: TAG($(n+1)\Delta t$)
   f) Local state calculation $\mathbf{x}_{i,n+1}$ from $\mathbf{x}_{i,n}$ and $\mathbf{u}_{i,n}$
   g) $n \leftarrow n+1$

Table 2. Federate 1-2-3 pseudo-code

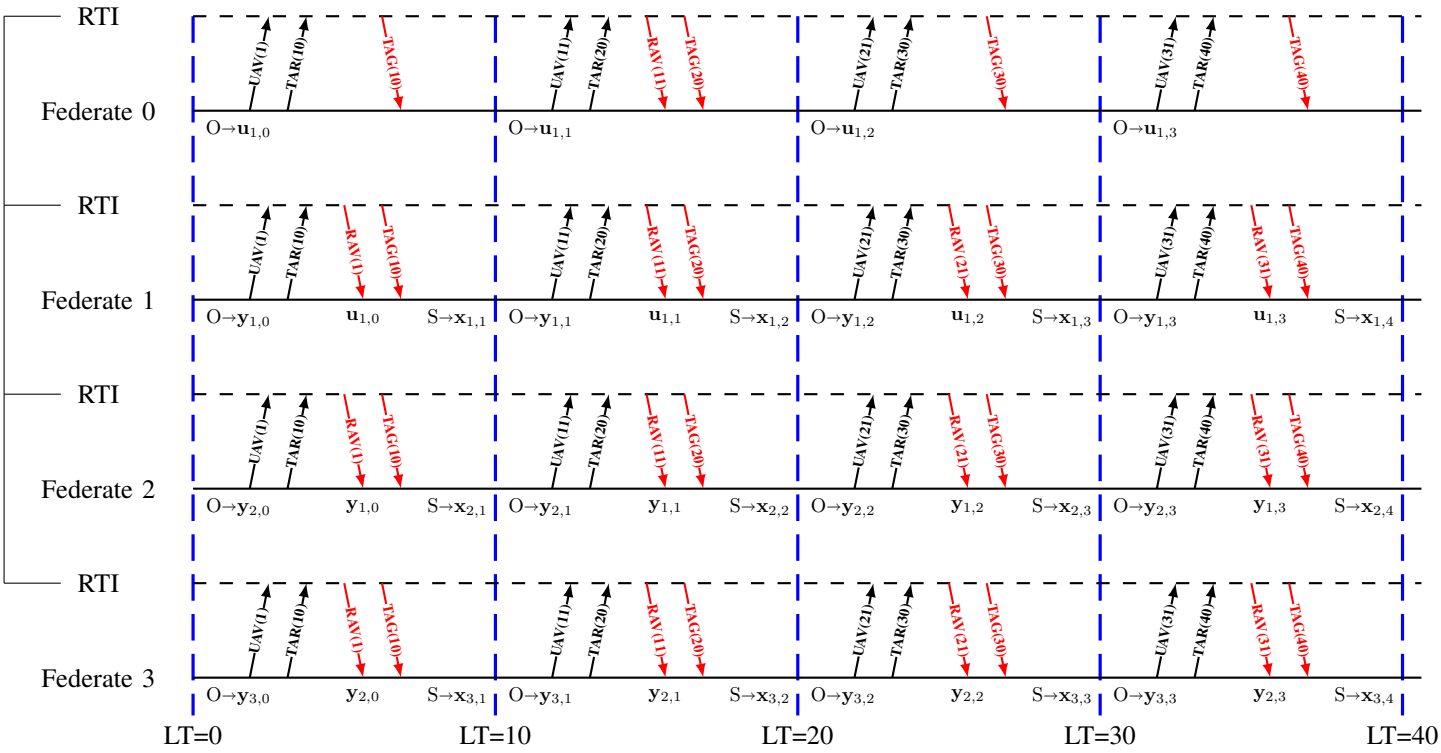

Figure 3. Execution sequence for standard case

3.2 Direct feedthrough case

We still consider the test case described above but we are assuming now that Federates 2 and 3 contain direct feedthrough relation. The equation system for Federates 1, 2 and 3 is depicted in (21)-(22)-(23)-(24):

$$\mathbf{x}_{1,n+1} = \mathbf{A}_1\mathbf{x}_{1,n} + \mathbf{B}_1\mathbf{u}_{1,n} \qquad \mathbf{x}_{2,n+1} = \mathbf{A}_2\mathbf{x}_{2,n} + \mathbf{B}_2\mathbf{u}_{2,n} \qquad \mathbf{x}_{3,n+1} = \mathbf{A}_3\mathbf{x}_{3,n} + \mathbf{B}_3\mathbf{u}_{3,n} \tag{21}$$

$$\mathbf{y}_{1,n} = \mathbf{C}_1\mathbf{x}_{1,n} \qquad \mathbf{y}_{2,n} = \mathbf{C}_2\mathbf{x}_{2,n} + \mathbf{D}_2\mathbf{u}_{2,n} \qquad \mathbf{y}_{3,n} = \mathbf{C}_3\mathbf{x}_{3,n} + \mathbf{D}_3\mathbf{u}_{3,n} \tag{22}$$

$$\begin{bmatrix}\mathbf{x}_{1,n+1}\\ \mathbf{x}_{2,n+1}\\ \mathbf{x}_{3,n+1}\end{bmatrix} = \begin{bmatrix}\mathbf{A}_1 & \mathbf{0} & \mathbf{0}\\ \mathbf{B}_2\mathbf{C}_1 & \mathbf{A}_2 & \mathbf{0}\\ \mathbf{B}_3\mathbf{D}_2\mathbf{C}_1 & \mathbf{B}_3\mathbf{C}_2 & \mathbf{A}_3\end{bmatrix}\begin{bmatrix}\mathbf{x}_{1,n}\\ \mathbf{x}_{2,n}\\ \mathbf{x}_{3,n}\end{bmatrix} + \begin{bmatrix}\mathbf{B}_1\\ \mathbf{0}\\ \mathbf{0}\end{bmatrix}\mathbf{u}_{1,n} \tag{23}$$

$$\begin{bmatrix}\mathbf{y}_{1,n}\\ \mathbf{y}_{2,n}\\ \mathbf{y}_{3,n}\end{bmatrix} = \begin{bmatrix}\mathbf{C}_1 & \mathbf{0} & \mathbf{0}\\ \mathbf{D}_2\mathbf{C}_1 & \mathbf{C}_2 & \mathbf{0}\\ \mathbf{D}_3\mathbf{D}_2\mathbf{C}_1 & \mathbf{D}_3\mathbf{C}_3 & \mathbf{C}_3\end{bmatrix}\begin{bmatrix}\mathbf{x}_{1,n}\\ \mathbf{x}_{2,n}\\ \mathbf{x}_{3,n}\end{bmatrix} \tag{24}$$

Reminding that $\forall i \in 1,2,3, \mathbf{u}_{i,n} = \mathbf{y}_{i-1,n}$ we intuitively feel that direct feedthrough will cause problems. Let's consider Federate 2 implemented as described in Table 2. At phase 2.(a), the federate should calculate its output value $\mathbf{y}_{2,n}$ from $\mathbf{x}_{2,n}$ and $\mathbf{u}_{2,n} = \mathbf{y}_{1,n}$. However for $n > 0$, according to this pseudo-code implementation, Federate 1 doesn't know about $\mathbf{y}_{1,n}$ but only about $\mathbf{y}_{1,n-1}$. This means that it knows values from the previous step but not the one from the same step which has just been calculated per federate 1. Therefore for the same cycle, Federate 2 needs to know its input at instant n to calculate $\mathbf{y}_{2,n}$. As Federates are communicating in sequence, this problem is even worth for Federate 3 which must wait for the update of the output from Federate 2, which is itself depending on Federate 1. So there is a functional schedule principle to build in order to ensure consistency between publications and acquisitions.

We solve this problem by combining the usage of NER with the usage of TAR and we also define a dedicated and correct value for the *lookahead*. NER service allows to request an advance in the HLA logical time but unlike the TAR service, this time advance will be granted either for the required time or for the time of the first available RAV if it is smaller than the requested time. Here the choice of a correct *lookahead* is a key issue to ensure a correct number of UAV/RAV for the same simulation cycle. Implementation strategies For federate 0 and Federate 1 don't change but, as described in Tables 3 and 4, Federates 2 and 3 contain NER usage in their implementation strategies. Figure 4 illustrates the temporal behavior of this new strategy with $\Delta t = 10$ et $lk = 1$.

1) *Initialization phase*
   a) $n \leftarrow 0$
   b) Initial choice for $\mathbf{x}_{i,0}$ value
2) *Simulation phase (while loop)*
   a) Request for HLA time advance to cycle $n+1$ de temps local $(n+1)\Delta t$: NER($(n+1)\Delta t$)
   b) Receive attributes $\mathbf{y}_{1,n} = \mathbf{u}_{2,n}$: RAV($n\Delta t + lk$)
   c) HLA time advance is granted to HLA simulation time $n\Delta t + lk$: TAG($n\Delta t + lk$)
   d) Output calculation $\mathbf{y}_{2,n}$ partir de $\mathbf{x}_{2,n}$ et $\mathbf{u}_{2,n}$
   e) Publish attributes: UAV($n\Delta t + 2lk$) de $\mathbf{y}_{2,n}$
   f) Request for HLA time advance to cycle $n+1$ for local time $(n+1)\Delta t$: TAR($(n+1)\Delta t$)
   g) HLA time advance is granted to cycle $n+1$ for local time $(n+1)\Delta t$: TAG($(n+1)\Delta t$)
   h) Local state calculation $\mathbf{x}_{2,n+1}$ from $\mathbf{x}_{2,n}$ and $\mathbf{u}_{2,n}$
   i) $n \leftarrow n+1$

Table 3. Federate 2 pseudo-code

4. Federation with a Communication Loop

We consider now a federation containing a communication loop between Federates as depicted in Figure 5 where $\mathbf{u}_{2,n} = \mathbf{y}_{1,n}$ and $\mathbf{u}'_{1,n} = \mathbf{y}_{2,n}$.

4.1 Standard case

As before, Federate 0 delivers signal $\mathbf{u}_{1,n}$ at every cycle n. We assume that the numerical solutions of state-space models representing the dynamic behaviors of Federate 1 and Federate 2 are given by equations (25)-(26)-(27)-(28):

$$\mathbf{x}_{1,n+1} = \mathbf{A}_1\mathbf{x}_{1,n} + \mathbf{B}_1\mathbf{u}_{1,n} + \mathbf{B}'_1\mathbf{u}'_{1,n} \tag{25}$$

$$\mathbf{y}_{1,n} = \mathbf{C}_1\mathbf{x}_{1,n} \tag{26}$$

$$\mathbf{x}_{2,n+1} = \mathbf{A}_2\mathbf{x}_{2,n} + \mathbf{B}_2\mathbf{u}_{2,n} \tag{27}$$

$$\mathbf{y}_{2,n} = \mathbf{C}_2\mathbf{x}_{2,n} \tag{28}$$

1) *Initialization phase*
   a) $n \leftarrow 0$
   b) Initial choice for $\mathbf{x}_{i,0}$ value
2) *Simulation phase (while loop)*
   a) Receive attributes $\mathbf{y}_{3,n} = \mathbf{u}_{2,n}$: RAV($n\Delta t + 2lk$)
   b) HLA time advance is granted to HLA simulation time $n\Delta t + lk$: TAG($n\Delta t + 2lk$)
   c) Output calculation $\mathbf{y}_{3,n}$ partir de $\mathbf{x}_{3,n}$ et $\mathbf{u}_{3,n}$
   d) Publish attributes: UAV($n\Delta t + 2lk$) de $\mathbf{y}_{3,n}$
   e) Request for HLA time advance to cycle $n+1$ for local time $(n+1)\Delta t$: TAR($(n+1)\Delta t$)
   f) HLA time advance is granted to cycle $n+1$ for local time $(n+1)\Delta t$: TAG($(n+1)\Delta t$)
   g) Local state calculation $\mathbf{x}_{3,n+1}$ from $\mathbf{x}_{3,n}$ and $\mathbf{u}_{3,n}$
   h) $n \leftarrow n+1$

Table 4. Federate 3 pseudo-code

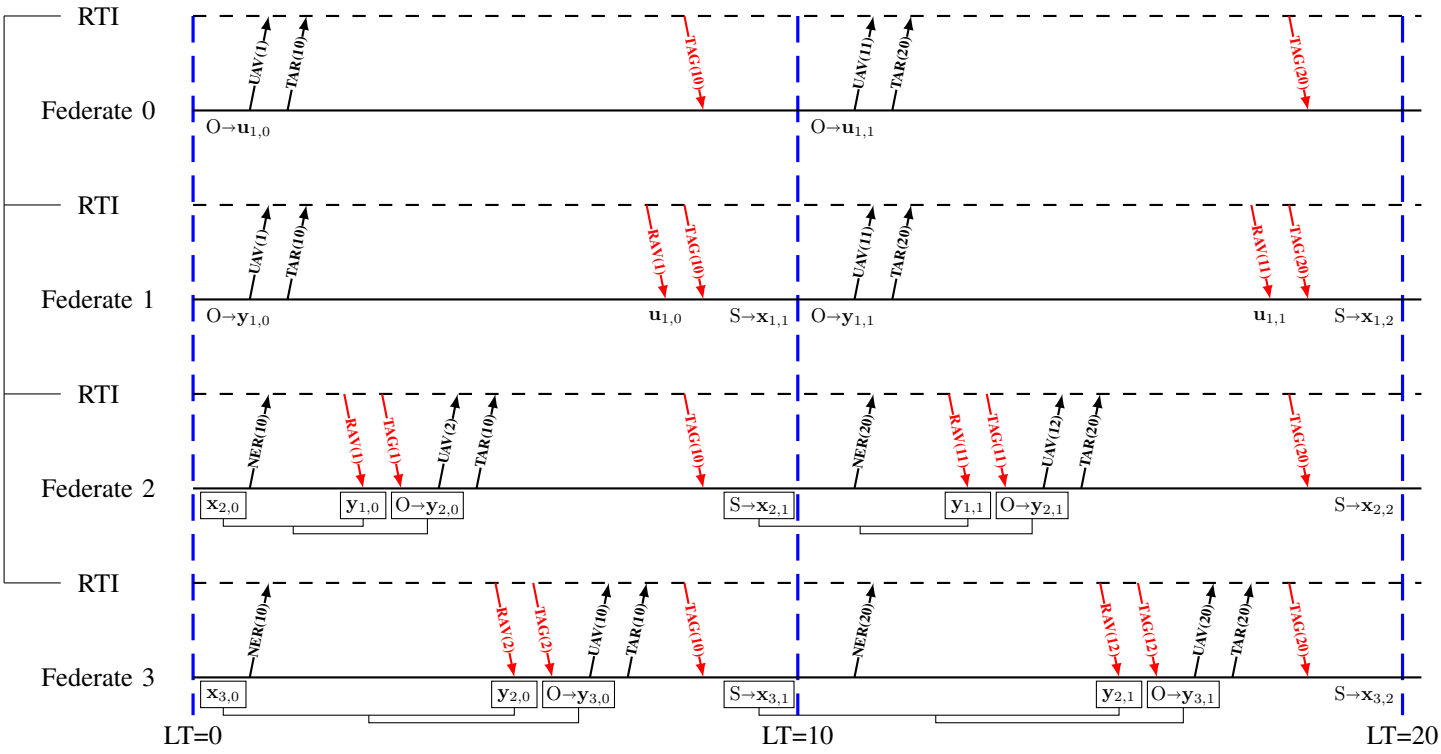

Figure 4. Execution sequence direct feedthrough case

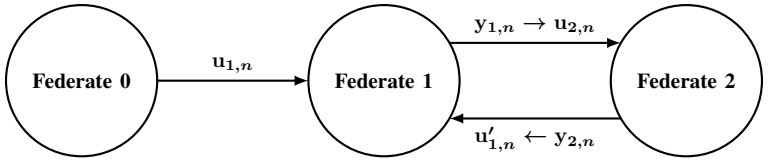

Figure 5. Illustration of a federation with a communication loop

This brings the global state equations (29)-(30) system:

$$\begin{bmatrix}\mathbf{x}_{1,n+1}\\ \mathbf{x}_{2,n+1}\end{bmatrix} = \begin{bmatrix}\mathbf{A}_1 & \mathbf{B}'_1\mathbf{C}_2\\ \mathbf{B}_2\mathbf{C}_1 & \mathbf{A}_2\end{bmatrix}\begin{bmatrix}\mathbf{x}_{1,n}\\ \mathbf{x}_{2,n}\end{bmatrix} + \begin{bmatrix}\mathbf{B}_1\\ 0\end{bmatrix}\mathbf{u}_{1,n} \tag{29}$$

$$\begin{bmatrix}\mathbf{y}_{1,n}\\ \mathbf{y}_{2,n}\end{bmatrix} = \begin{bmatrix}\mathbf{C}_1 & 0\\ 0 & \mathbf{C}_2\end{bmatrix}\begin{bmatrix}\mathbf{x}_{1,n}\\ \mathbf{x}_{2,n}\end{bmatrix} \tag{30}$$

Without direct feedthrough and despite the presence of a loop, the resolution takes place as it is done in the context of a federation with communication sequence. Thus, we can apply the implementation strategies described in Tables 1 and 2.

4.2 Direct feedthrough case

Now, let's consider a direct feedthrough relation for federate 1 depending on its second input $u'_{1,n}$, then the model of federate 1 can be described as in equations (31)-(32):

$$\mathbf{x}_{1,n+1} = \mathbf{A}_1\mathbf{x}_{1,n} + \mathbf{B}_1\mathbf{u}_{1,n} + \mathbf{B}'_1\mathbf{u}'_{1,n} \tag{31}$$

$$\mathbf{y}_{1,n} = \mathbf{C}_1\mathbf{x}_{1,n} + \mathbf{D}'_1\mathbf{u}'_{1,n} \tag{32}$$

This brings the global state equations (33)-(34) system to:

$$\begin{bmatrix}\mathbf{x}_{1,n+1}\\ \mathbf{x}_{2,n+1}\end{bmatrix} = \begin{bmatrix}\mathbf{A}_1 & \mathbf{B}'_1\mathbf{C}_2\\ \mathbf{B}_2\mathbf{C}_1 & \mathbf{A}_2 + \mathbf{B}_2\mathbf{D}'_1\mathbf{C}_2\end{bmatrix}\begin{bmatrix}\mathbf{x}_{1,n}\\ \mathbf{x}_{2,n}\end{bmatrix} + \begin{bmatrix}\mathbf{B}_1\\ 0\end{bmatrix}\mathbf{u}_{1,n} \tag{33}$$

$$\begin{bmatrix}\mathbf{y}_{1,n}\\ \mathbf{y}_{2,n}\end{bmatrix} = \begin{bmatrix}\mathbf{C}_1 & \mathbf{D}'_1\mathbf{C}_2\\ 0 & \mathbf{C}_2\end{bmatrix}\begin{bmatrix}\mathbf{x}_{1,n}\\ \mathbf{x}_{2,n}\end{bmatrix} \tag{34}$$

As for the federation with communication sequence, the presence of the direct feedthrough requires therefore that Federate 1 receives the output from Federate 2 for the same cycle. Therefore Federate 1 has to behave according to the implementation strategy described above in Table 3.

4.3 Algebraic loop case

MATLAB/SIMULINK users are usually familiar with the concept of algebraic loops, which occur when a signal loop exists with only direct feedthrough blocks within the loop. More details on algebraic loops and how they are handled in the SIMULINK tool can be found in the corresponding user's guide [15]. To illustrate this concept, from the direct feedthrough case, let's consider a second direct feedthrough relation for Federate 2 depending on its second input $\mathbf{u}_{2,n}$. The model of federate 2 should now be described with equations (35)-(36):

$$\mathbf{x}_{2,n+1} = \mathbf{A}_2\mathbf{x}_{2,n} + \mathbf{B}_2\mathbf{u}_{2,n} \tag{35}$$

$$\mathbf{y}_{2,n} = \mathbf{C}_2\mathbf{x}_{2,n} + \mathbf{D}_2\mathbf{u}_{2,n} \tag{36}$$

Obtaining the global system equation is much more complicated. Indeed, by replacing $\mathbf{u}'_{1,n}$ et $\mathbf{u}_{2,n}$ per their respective values $\mathbf{y}_{2,n}$ et $\mathbf{y}_{1,n}$, we obtain the following output equations (37)-(38):

$$\mathbf{y}_{1,n} = \mathbf{C}_1\mathbf{x}_{1,n} + \mathbf{D}'_1\mathbf{y}_{2,n} \tag{37}$$

$$\mathbf{y}_{2,n} = \mathbf{C}_2\mathbf{x}_{2,n} + \mathbf{D}_2\mathbf{y}_{1,n} \tag{38}$$

In this specific case, to calculate the output for step n, every federate needs the output from the other federates at the same instant n; this is the so-called *algebraic loop* concept mentionned above. This loop shows an implicit equation requiring the resolution of a system at each step of integration and the solution, if existing, can be described per equations (39)-(40):

$$\mathbf{y}_{1,n} = \left(\mathbf{I} - \mathbf{D}'_1\mathbf{D}_2\right)^{-1}\mathbf{C}_1\mathbf{x}_{1,n} + \left(\mathbf{I} - \mathbf{D}'_1\mathbf{D}_2\right)^{-1}\mathbf{D}'_1\mathbf{C}_2\mathbf{x}_{2,n} \tag{39}$$

$$\mathbf{y}_{2,n} = \left(\mathbf{I} - \mathbf{D}_2\mathbf{D}_1\right)^{-1}\mathbf{D}_2\mathbf{C}_1\mathbf{x}_{1,n} + \left(\mathbf{I} - \mathbf{D}_2\mathbf{D}'_1\right)^{-1}\mathbf{C}_2\mathbf{x}_{2,n} \tag{40}$$

where $\mathbf{I}$ denotes the identity matrix with the correct dimension. Therefore, in the context of a linear description, if matrices $\mathbf{I} - \mathbf{D}'_1\mathbf{D}_2$ and $\mathbf{I} - \mathbf{D}_2\mathbf{D}'_1$ are non singular, a solution exists and the model for the global state is given per (41)-(42):

$$\begin{bmatrix}\mathbf{x}_{1,n+1}\\ \mathbf{x}_{2,n+1}\end{bmatrix} = \begin{bmatrix}\mathbf{A}_1 + \mathbf{B}_1' \left(\mathbf{I} - \mathbf{D}_2\mathbf{D}_1\right)^{-1} \mathbf{D}_2\mathbf{C}_1 & \mathbf{B}_1' \left(\mathbf{I} - \mathbf{D}_2\mathbf{D}_1'\right)^{-1} \mathbf{C}_2\\ \mathbf{B}_2 \left(\mathbf{I} - \mathbf{D}_1'\mathbf{D}_2\right)^{-1} \mathbf{C}_1 & \mathbf{A}_2 + \mathbf{B}_2 \left(\mathbf{I} - \mathbf{D}_1'\mathbf{D}_2\right)^{-1} \mathbf{D}_1'\mathbf{C}_2\end{bmatrix} \begin{bmatrix}\mathbf{x}_{1,n}\\ \mathbf{x}_{2,n}\end{bmatrix} + \begin{bmatrix}\mathbf{B}_1\\ \mathbf{0}\end{bmatrix} \mathbf{u}_{1,n} \tag{41}$$

$$\begin{bmatrix}\mathbf{y}_{1,n}\\ \mathbf{y}_{2,n}\end{bmatrix} = \begin{bmatrix}\left(\mathbf{I} - \mathbf{D}_1'\mathbf{D}_2\right)^{-1} \mathbf{C}_1 & \left(\mathbf{I} - \mathbf{D}_1'\mathbf{D}_2\right)^{-1} \mathbf{D}_1'\mathbf{C}_2\\ \left(\mathbf{I} - \mathbf{D}_2\mathbf{D}_1\right)^{-1} \mathbf{D}_2\mathbf{C}_1 & \left(\mathbf{I} - \mathbf{D}_2\mathbf{D}_1'\right)^{-1} \mathbf{C}_2\end{bmatrix} \begin{bmatrix}\mathbf{x}_{1,n}\\ \mathbf{x}_{2,n}\end{bmatrix} \tag{42}$$

However, from the HLA point of view, this problem has simply no solution since it would imply an endless exchange of messages during the same iteration n. The distributed nature of the federation prevents solving the global equations (41)-(42). An approximate solution would be that, at iteration n, one of the federates involved in the algebraic loop takes the previous value of the input, e.g., $\mathbf{u_{1,n-1}}$. This would actually break the loop by introducing an artificial delay. This is also one of the solutions used by SIMULINK when a global solving is not possible.

5. Formalization and Application

5.1 Discussion

Through this non exhaustive list of simple examples, we raised the issue of direct feedthrough within HLA federations. If this problem is not taken into account, it usually introduces delays between federate models (state and output equation calculations). Obviously, this put into question the validity of the global simulation semantics and, depending on the size of the simulation step, this can have a significant impact on the overall system response. For example, suppose 50 federates involved in a communication sequence (as depicted in Figure 2) where each one is related with the previous by a direct feedthrough relation. If we do not care of direct feedthrough, the 50^{th} federate will receive its input value with a delay of 49 time units, whereas it should have received it during its first cycle. For sure, this example is a little exaggerated but it illustrates quite well the issue introduced by the presence of direct feedthrough, and why, one should apply the proper functional scheduling strategy.

To resume, there are three main cases to consider:

1) Federation does not have direct feedthrough constraint $\Rightarrow$ each federate can be executed following the classical schedule (see Figure 3).
2) Federation has direct feedthrough constraint without any algebraic loop $\Rightarrow$ a scheduling can be found according to the strategy described in Subsection 5.2 below.
3) Federation has direct feedthrough constraint with algebraic loops $\Rightarrow$ no exact solution can be found. If possible, federate implementations must be modified in order to break these algebraic loop; then you might be in the case 1) or 2). Or, one of the federates involved in the loop should take the previous input value, as it would break the loop.

5.2 Strategy for direct feedthrough

A generic strategy is presented to schedule executions in case 2). We first define a sequence of direct feedthrough, as a sequence of output equations, beginning with an output equation without direct feedthrough and followed exclusively by output equations with direct feedthrough, Furthermore, the output value of an output equation is completely or partly the input value of the next equation.

A correct strategy to properly ensure the functional scheduling of a HLA federation implying distributed ODEs could be:

1) Identify offline all sequences of direct feedthrough;
2) Set the lookahead lk of each federate depending on the size of the sequences of direct feedthrough;
3) Decide the adequate strategy for NER and TAR usage.

5.3 SDSE example

Taking the example of the SDSE federation, the offline analysis of the ODEs describing each federate shows some direct feedthrough (bold arrows) and a possible algebraic loop. This is illustrated in figure 6 where direct feedthrough relations are illustrated using bold arrows.

- Communications between *Pilots Inputs* and *Flight Controller* federates: the output equation for *Flight Controller* directly depends on the pilot commands.
- Communications between *Sensors* and *Flight Controller* federates: the output equation for *Flight Controller* directly depends on the measured values from sensors.
- Communications between *Actuators* (i.e. control surfaces) and *Flight Dynamics* federates: the output equation for *Flight Dynamics* directly depends on the control surfaces deflections.

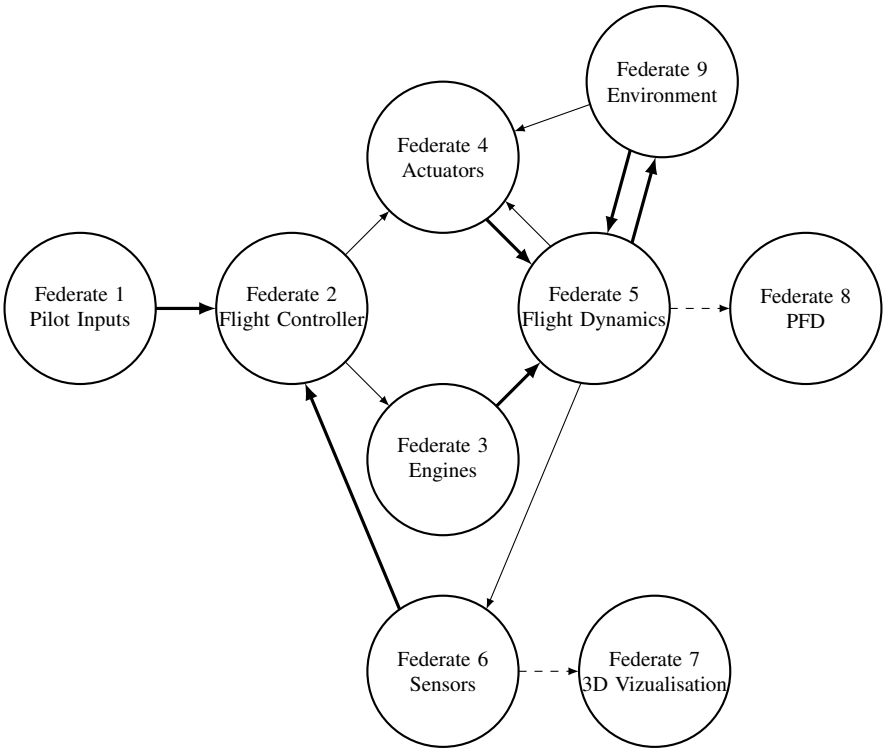

Figure 6. Illustration of direct feedthrough within SDSE communications

- Communications between *Engines* (i.e. control surfaces) and *Flight Dynamics* federates: the output equation for *Flight Dynamics* directly depends on the thrust from each engine.
- Communications between *Flight Dynamics* and *Environment* federates: *Environment* federate needs the aircraft position on Earth (latitude, longitude and altitude) to compute the corresponding atmosphere relevant parameters (temperature, pressure, air density, speed of sound) for the current time step. In addition, *Flight Dynamics* needs these quantities for the same time step to calculate some other outputs (but not the latitude, longitude and altitude). Strictly speaking, it is not an algebraic loop. The situation is solved by executing the output equation of Federate 5 in two steps: first, the sub vector of $\mathbf{y}$ that yields the aircraft position is calculated, so it can be sent to Federate 9 to obtain the atmosphere characteristics; secondly the rest of the output equation of Federate 5 is executed once the atmosphere characteristics are acquired.

6. Conclusion

We have presented, in this paper, a complete analysis for distributed resolution of ODEs in real-time HLA simulations. Based on our experience with the SDSE simulation test case, we offered a theoretical study on functional scheduling and deduced a set of rules in order to ensure consistent simulations. Also, we applied these rules ourselves to ensure the functional scheduling of our SDSE simulator and we compared the results of this simulation with the similar Simulink implementation to verify the correct semantic behavior. In addition, some performance metrics and tests (e.g., worst case execution time, worst case transit time, time management algorithm performances) have been conducted to guarantee the correct execution of these simulations with respect to specified real-time constraints. Interested readers may find more details and descriptions about the SDSE aircraft simulation in references [1], [2].

References

[1] C. Gervais, J.-B. Chaudron, P. Siron, R. Leconte and D. Saussié: “Real-Time Distributed Aircraft Simulation through HLA”, DS-RT ’12: Proceedings of the 2012 IEEE/ACM 16th International Symposium on Distributed Simulation and Real Time Applications, October 2012.

[2] J.-B. Chaudron, D. Saussié, P. Siron and M. Adelantado: “Real-time distributed simulations in an HLA framework: Application to aircraft simulation”, Simulation, Vol.90, Issue 6, pp 627-643, June 2014.

[3] IEEE Std 1516™-2010, IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Framework and Rules

[4] IEEE Std 1516.1™-2010, IEEE Standard for Modeling and Simulation High Level Architecture - Federate Interface Specification

[5] IEEE Std 1516.2™-2010, IEEE Standard for Modeling and Simulation High Level Architecture - Object Model Template Specification

[6] R.M. Fujimoto:"Time Management in the High Level Architecture", Simulation, Vol.71, pp 388-400. December 1998.

[7] R.G. Phillips and E.Z. Crues:"Time Management Issues and Approaches for Real Time HLA based simulations", Proceedings of the Fall Simulation Interoperability Workshop, September 2005.

[8] R.M. Fujimoto and T. McLean:"Repeatability in real-time distributed simulation executions", Proceedings of the fourteenth workshop on Parallel and Distributed Simulation, 2000.

[9] P. Siron, E. Noulard and J.-Y. Rousselot:"CERTI : an open Source RTI, why and how", Proceedings of the Fall Simulation Interoperability Workshop, 2009.

[10] E. Noulard, B. D'Ausbourg and P. Siron:"Running Real Time Distributed Simulations under Linux and CERTI", Proceedings of the European Simulation Interoperability Workshop, 2007.

[11] J.-B. Chaudron, E. Noulard and P. Siron:"HLA High-Performance and Real-Time Simulation Studies with CERTI", Proceedings of the European Simulation and Modelling Conference, October 2011.

[12] J.-B. Chaudron, P. Siron and E. Noulard:"Design and model-checking techniques applied to real-time RTI time management", Proceedings of the Spring Simulation Interoperability Workshop, April 2011.

[13] R. Cooke and V.I. Arnold:"Ordinary Differential Equations", Springer Berlin Heidelberg, 1992.

[14] , R.M. Fujimoto :"Parallel Discrete Event Simulation", Communications of the ACM - Special issue on Simulation, Vol.33, Issue 10, pp 30-53, October 1990.

[15] The MathWorks, Inc. :"SIMULINK User's guide (r2016a) ", pp 252-288, Retrieved June 29, 2011 from www.mathworks.com/help/pdf_doc/simulink/sl_using.pdf, 2016.

Author Biographies

JEAN-BAPTISTE CHAUDRON received his doctorate in 2012 from ISAE-Supaero (Toulouse, France). He works at ISAE as a research engineer in the Department of Complex Systems Engineering (DISC). His fields of interest include parallel and distributed simulation, time-triggered and real-time systems and scheduling theory.

DAVID SAUSSIE received his B. Sc. degree from ISAE-Supaero in 2003 and his M. Sc. degree from Polytechnique Montréal (Canada) in 2004. He received a joint Ph.D. degree from both institutions in 2010. He is currently an assistant professor in automation and control in the Department of Electrical Engineering, at Polytechnique Montréal. His research interests are mainly simulation and control for aerospace systems.

PIERRE SIRON graduated from a French engineering school in computer science (ENSEEIHT) in 1980, and received his doctorate in 1984. He works in parallel and distributed systems and he is the leader of the CERTI project. He is a professor at ISAE-Supaero, and the head of the computer science program of the ISAE training.

MARTIN ADELANTADO graduated from a French engineering school in computer science (ENSEEIHT) in 1979, and received his doctorate in 1981. He is an ONERA (the French Aerospace Lab) research engineer and works in the Information Processing and Modelling Department (DTIM). His fields of interest include simulation, real-time systems and distributed systems.

Acknowledgments

The authors would like to thank **ERIC NOULARD** from ONERA for his help with CERTI and **REGINE LECONTE** from ISAE for her help on the evolution and the maintenance of the SDSE simulation.