ITM Web of Conferences **1**, 01003 (2013)
DOI: 10.1051/itmconf/ 20130101003

On the use of Dynamic Topologies for Representing Spatially Moving Entities

Fernando Barros[1,a]

[1] *Departamento de Engenharia Informática*
Universidade de Coimbra
Portugal

Abstract. In many psychical systems, entities can only communicate when within the range of their sensors. When entities are static, the topology of the communication infrastructure can be known in advance and kept unchanged during simulation. However, representing moving entities becomes more complex since interconnections change with time as the relative distances between entities evolve. Given the dynamic nature of the problem, a common solution involves the use of publish/subscribe communication and the corresponding multicast message passing. In this paper we exploit the use of a peer-to-peer infrastructure (p2p) exhibiting a dynamic topology as an alternative representation for spatially moving entities. We use the Heterogeneous Flow Systems Specification (HFSS), a modular modeling formalism designed to represent hybrid systems with time-variant topologies. We present a simplified model of a defense system comprising airborne radars, drone detection and cancellation. We demonstrate the benefits of p2p communication over publish subscribe interaction.

1 Introduction

In many psychical systems, entities can only communicate when within the range of their sensors. For static entities, the topology of the communication can be computed in advance and kept constant during the simulation. However, representing moving entities is more challenging since their interconnection change with time as the distances between entities evolve. In air traffic systems, for example, airplanes only interact when they are in some physical vicinity. Radars can only detect targets when they are within some range. From a model perspective, these kinds of systems have a dynamic topology since the interconnection between the entities evolves over time, being dependent on their locations. Since locations obey to arbitrary trajectories it becomes virtually impossible to know in advance what topologies are required during the lifetime of an entity.

A common approach to represent moving entities involves the use of publish/subscribe (pub/sub) communication [1], [2], [3], [4]. Pub/sub uses broadcast communication to connect entities interested in a common topic, enabling the decoupling of entities and their reuse. Pub/sub however, have performance and modeling limitations. Message broadcast is costly and it requires an intensive use of message filtering since not all entities are within each others range. Moreover, pub/sub imposes an

[a] e-mail: barros@dei.uc.pt

awkward modeling style, since it relies on message push, making it difficult to represent information pulling. If we take, for example, a defense system, pub/sub requires aircrafts to push information position to radars. However, in real systems, radars actually pull information location from the aircrafts. The former representation allow the modeling of radars with different sampling rates. On the contrary, pub/sub imposes the sampling rates on radars, which is an intolerable simplification in detailed models. Pub/sub exhibits another modeling limitation related to their inability to represent tightly coupled entities. While aircraft position can be obtained by message broadcast (with the limitations already pointed out), the connection between, for example, a missile and a target can be more efficiently/naturally represented using point-to-point communication. The use of this kind of hybrid communication can hardly be obtained using pub/sub systems like the High Level Architecture (HLA) [3].

In this paper we propose the representation of moving entities using the Heterogeneous Flow System Specification (HFSS), a formalism aimed to represent hierarchical and modular hybrid systems [5]. HFSS treats sampling as a first-order construct, enabling a simple specification of pull-communication as a complement to push-communication, typical of discrete event-based systems. HFSS models exhibit a dynamic topology, making it possible to make arbitrary changes in model composition and coupling. We demonstrate HFSS flexibility by presenting a sample defense system composed by fixed radars, aerial targets and dynamically created pursuers.

2 The HFSS Formalism

The Heterogeneous Flow System Specification (HFSS) is a formalism aimed to represent piecewise constant partial state systems. This characteristic enables its implementation on digital computers. HFSS achieves the representation of continuous variables using the concept of generalized sampling [5], while the representation of discrete events is based on the Discrete Event System Specification (DEVS) [6]. HFSS has two types of models: basic and network. Basic models provide state representation and state transition functions. Network models are a composition of basic models and/or other network models. Given its definition, a network provides an abstraction for representing hierarchical systems.

2.1 HFSS Basic Model

We consider $\widehat{B}$ as the set of names corresponding to basic HFSS models. A HFSS basic model associated with name $B \in \widehat{B}$ is defined by

$$M_B = (X, Y, P, \rho, \omega, s_0, \delta, \bar{\Lambda}, \lambda)$$

where

$X = \bar{X} \times \ddot{X}$ is the set of input flow values

$\bar{X}$ is the set of continuous input flow values
$\ddot{X}$ is the set of discrete input flow values

$Y = \bar{Y} \times \ddot{Y}$ is the set of output flow values

$\bar{Y}$ is the set of continuous output flow values
$\ddot{Y}$ is the set of discrete output flow values

P is the set of partial states (p-states)

$\rho : P \longrightarrow \mathbf{H}^0_{+\infty}$ is the time-to-input function

$\omega : P \longrightarrow \mathbf{H}^0_{+\infty}$ is the time-to-output function

$S = \{(p, e) | p \in P, 0 \leq e \leq \nu(p)\}$ is the state set

with $\nu(p) = \min\{\rho(p), \omega(p)\}$, representing the time to transition function

$s_0 = (p_0, e_0) \in S$ is the initial state

$\delta : S \times X^{\varnothing} \longrightarrow P$ is the transition function

where $X^{\varnothing} = \bar{X} \times \ddot{X}^{\varnothing}$ and $\ddot{X}^{\varnothing} = \ddot{X} \cup \{\varnothing\}$
and $\varnothing$ represents the null value (absence of value)

$\bar{\Lambda} : S \longrightarrow \bar{Y}$ is the continuous output function

$\lambda : P \longrightarrow \ddot{Y}$ is the partial discrete output function

HFSS components have their behavior ruled by HFSS models and their semantics are presented in [5].

Example: Continuous Flow Generator

Function generation plays an important role in the representation of mobile entities. Entity trajectory can be described with the help of a HFSS function generator. A moving entity following the 2-D trajectory given by the function $f : \mathbf{R} \longrightarrow \mathbf{R}^2$ is described by

$$M_f = (X, Y, P, \rho, \omega, s_0, \delta, \bar{\Lambda}, \lambda)$$

where

$X = \{\varnothing\} \times \{\varnothing\}$

$Y = \mathbf{R}^2 \times \{\varnothing\}$

$P = \{\varnothing\}$

$\rho(\varnothing) = \omega(\varnothing) = \infty$

$s_0 = (\varnothing, 0)$

$\delta((\varnothing, 0), (\varnothing, \varnothing)) = (\varnothing, 0)$

$\bar{\Lambda}((\varnothing, e)) = f(e_{std})$, where e_{std} is the standard part of the hyperreal e

$\lambda(\varnothing) = \varnothing$

This model can represent arbitrary continuous functions. For example, a circular trajectory of radius R, angular velocity ϖ and initial phase φ, can be described by the function $\odot(t)$ defined by: $\odot(t) = R(\cos(\varpi t + \varphi), \sin(\varpi t + \varphi))$. This generator uses only continuous flows to describe a trajectory. The generator is a passive component that has no autonomous behavior, and where all information is retrieved through sampling. Trajectory discontinuities can be modeled with the help of discrete flows, that can be used, for example, for signaling the start/end of a maneuver.

2.2 HFSS Network Model

As mentioned before, HFSS networks models are compositions of HFSS models (basic or other HFSS networks models). Let $\widehat{N}$ be the set of names corresponding to HFSS network models, with $\widehat{N} \cap \widehat{B} = \{\}$. Formally, a HFSS network model associated with name $N \in \widehat{N}$ is defined by

$$M_N = (X, Y, \eta)$$

where

N is the network name

$X = \bar{X} \times \ddot{X}$ is the set of network input flows

$\bar{X}$ is the set of network continuous input flows

$\ddot{X}$ is the set of network discrete input flows

$Y = \bar{Y} \times \ddot{Y}$ is the set of network output flows

$\bar{Y}$ is the set of network continuous output flows

$\ddot{Y}$ is the set of network discrete output flows

$\eta \in \widehat{\eta}$ is the name of the dynamic topology network executive

with

$\eta \in \widehat{\eta}$ representing the set of all names associated with HFSS executive models, constrained to $\widehat{\eta} \cap \widehat{B} = \widehat{\eta} \cap \widehat{N} = \{\}$

Executives are uniquely assigned to network models, i.e.,

$\forall_{i,j \in \widehat{N}, i \neq j} \eta_i \neq \eta_j$ with $M_k = (X_k, Y_k, \eta_k), \forall_{k \in \widehat{N}}$

The model of the executive is a modified HFSS model, defined by

$$M_\eta = (X_\eta, Y_\eta, P, \rho, \omega, s_0, \delta, \bar{\Lambda}, \lambda, \widehat{\Sigma}, \gamma), \text{ for } \eta \in \widehat{\eta}$$

where

$\widehat{\Sigma}$ is the set of network topologies

$\gamma : P \longrightarrow \widehat{\Sigma}$ is the topology function

The network topology $\Sigma_\alpha \in \widehat{\Sigma}$, corresponding to the p-state $p_\alpha \in P$, is given by the 3-tuple

$$\Sigma_\alpha = \gamma(p_\alpha) = (C_\alpha, \{I_{i,\alpha}\} \cup \{I_{\eta,\alpha}, I_{N,\alpha}\}, F_{i,\alpha} \cup \{F_{\eta,\alpha}, F_{N,\alpha}\})$$

where

C_α is the set of names associated with the executive state p_α

for all $i \in C_\alpha \cup \{\eta\}$

$I_{i,\alpha}$ is the sequence of asynchronous influencers of i

$F_{i,\alpha}$ is the input function of i

$I_{N,\alpha}$ is the sequence of network influencers

$F_{N,\alpha}$ is the network output function

For all $i \in C_\alpha$

$M_i = (X_i, Y_i, P_i, \rho_i, \omega_i, s_{0,i}, \delta_i, \bar{\Lambda}_i, \lambda_i)$ if $i \in \widehat{B}$

$M_i = (X_i, Y_i, \eta_i)$ if $i \in \widehat{N}$

Variables are subjected to the following constraints for every $p_\alpha \in P_\alpha$

$N \notin C_\alpha, N \notin I_{N,\alpha}, \eta \notin C_\alpha$

$N \notin E_{i,\alpha}$ for all $i \in C_\alpha \cup \{\eta, N\}$

$F_{N,\alpha} : \times_{k \in I_{N,\alpha}} Y_k \longrightarrow Y^\varnothing$

$F_{i,\alpha} : \times_{k \in I_{i,\alpha}} V_k \longrightarrow X_i^\varnothing$

where

$$V_k = \begin{cases} Y_k^\varnothing & \text{if } k \neq N \\ X^\varnothing & \text{if } k = N \end{cases}$$

$F_{N,\alpha}((\bar{v}_{k_1}, \varnothing), (\bar{v}_{k_2}, \varnothing), ...) = (\bar{y}_N, \varnothing)$

$F_{i,\alpha}((\bar{v}_{k_1}, \varnothing), (\bar{v}_{k_2}, \varnothing), ...) = (\bar{x}_i, \varnothing)$

Since the network topology is a function of executive p-sate it can be modified by the executive transition function. Applications of the HFSS formalism to the representation of dynamic topology models can be found in [7].

Example: Two-Tank System (Chattering)

To illustrate an application that can be modeled with a dynamic topology, we consider a simplification of a 2-tank system described in [8]. Our representation fixes a flaw in the original tank design that exhibits, under certain conditions, a chattering behavior that can lead to a very inefficient simulation execution. We show that a simple solution to chattering requires the use of two models and the ability to switch between these models. In this system there are two tanks, T1 and T2, and two pumps, P1 and P2, as depicted in Figure 1. For simplicity, we consider tanks with unbounded capacities since we are only interested in signaling when tank T1 becomes empty. We assume that pumps P1 and P2 have (variable) rates p_1 and p_2, respectively.

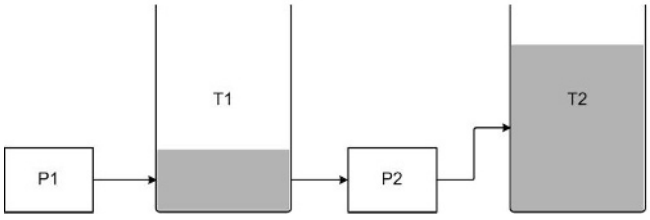


Figure 1. Two-tank system.

The model of this system is represented in Figure 2 where components are represented by rectangles and functions by circles. The influencers of a component are give by the arrows pointing to component input function. Tank T2 is fed by pump P2 that removes liquid from tank T1. Tank T1 is filled by pump P1 and is drained by pump P2. The liquid volume in tank T1 is thus described by: $dV_1/dt = p_1 - p_2$, subjected to the constraint: $V_1 \geqslant 0$.

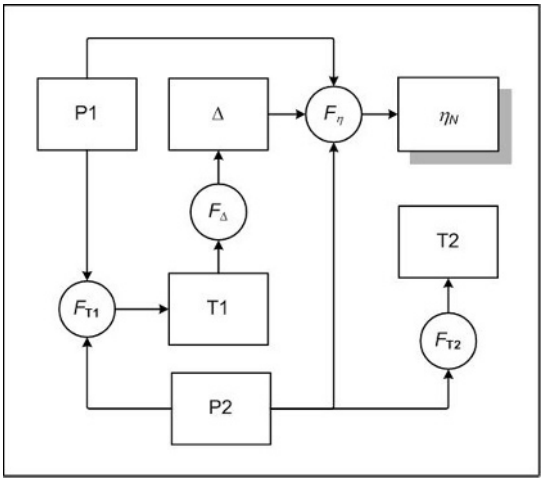


Figure 2. 2-Tank system in the regular mode ($V_1 > 0$).

However, this model is no longer valid when $V_1 = 0$ and $p_1 < p_2$. In this case pump P2 cannot drain liquid at rate p_2 since there is no liquid storage and liquid arrives at rate $p_1 < p_2$ to tank T2. Under these conditions the model enters a chattering mode and it becomes controlled by the sampling rates of the numerical solvers. This mode, however, has no relationship with the physical reality, which does not exhibit a chattering behavior. In reality, in this mode, pump P2 pumps the liquid at rate p_1, and most importantly, tank T1 becomes effectively a simple pipe. The model that eliminates chattering can be represented by the network of Figure 3, where tank T1 and detector Δ were removed from the initial model of Figure 2. HFSS semantics guarantees that, after a topology reconfiguration, the remaining components keep their state, eliminating the need for complex state transfer, used in representations based in mode switching.

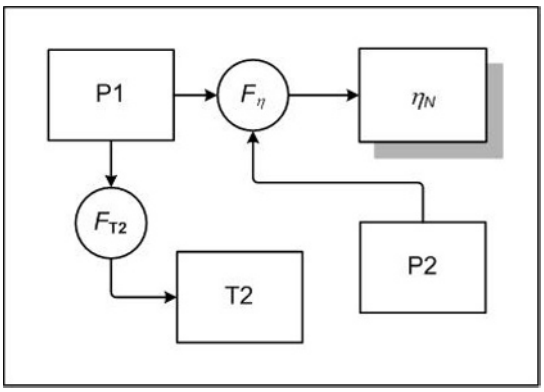


Figure 3. 2-Tank system in the non-chattering mode ($V_1 = 0, p_1 < p_2$).

When $p1 > p2$, the volume in T1 starts to increase and the T1 model is given back again by Figure 2. The system switches between these two modes, thus eliminating the flawed chattering behavior of a single tank model [8]. In the next sections we exploit the ability to modify model topology, described in this section, to represent moving entities.

To finish the description of this system, we sketch its formalization in HFSS. The executive has two p-states, q_0 and q_1, corresponding to the two modes of operation. The partial HFSS description of the network executive is given by:

$P_\eta = \{p_{\eta,0}, p_{\eta,1}\}$

$\gamma(p_{\eta,0}) = (C_0, \{I_0\}, \{F_0\})$, with

$C_0 = \{P_1, P_2, T_1, T_2, \Delta\}$

$I_{P_1,0} = I_{P_2,0} = \{\}$

$I_{T_1,0} = \{P_1, P_2\},\ I_{T_2,0} = \{P_2\}$

$I_{\Delta,0} = \{T_1\},\ I_{\eta,0} = \{P_1, P_2\}$

$F_{P_1,0}(x) = F_{P_2,0}(x) = F_{T_2,0}(x) = F_{\Delta,0}(x) = x$

$F_{T_1,0}((\bar{x}_1, \ddot{x}_1), (\bar{x}_2, \ddot{x}_2)) = (\bar{x}_1 - \bar{x}_2, \oplus(\ddot{x}_1, \ddot{x}_2))$

$F_{\eta,0}((\bar{x}_1, \ddot{x}_1), (\bar{x}_2, \ddot{x}_2)) = (\bar{x}_1 - \bar{x}_2, \oplus(\ddot{x}_1, \ddot{x}_2))$

where

$$\oplus(a, b) = \begin{cases} \varnothing & \text{if } a = b = \varnothing \\ \texttt{not} - \texttt{null} & \text{otherwise} \end{cases}$$

$\gamma(p_{\eta,1}) = (C_1, \{I_1\}, \{F_1\})$, with

$C_1 = \{P_1, P_2, T_2\},\ \ldots$

The two tanks T1 and T2 can be described by the integrator model described in the last section. Pumps P1 and P2 can be described by the function generator also presented in the last section. For brevity we omit the remaining description of the network model.

3 Modeling Spatial Entities

The efficient representation of spatially moving entities is a challenging problem. The interaction of non-moving entities is static and it can be computed in advance, posing, in general, no major representation problem. On the contrary, the interaction of moving entities is dynamic and requires a continuous update according to the current positions of the entities. Examples of spatial moving entities include aircrafts that can communicate with other devices when they are in a certain range of other objects, like radars and antennas. Usually, these entities can only communicate when sensed signals are above a certain threshold so they can be detected. In general, the communication will depend on the power of the emitted signal, the distance between emitter and receiver, and on the sensibility of the receiver. A possible solution for describing the interaction of two moving entities is depicted in Figure 4. In this example, we consider the drone D and the radar R. A detector Δ is responsible for signaling when the two entities can start communicating. The detector will signal the time instant when the drone enters radar range.

The general problem of establishing the communication of moving entities has $O(n^2)$ complexity since, in the worst case, any two entities can possibly interact, requiring a pair of detectors for each possible interaction. In practice, the general solution sketched in Figure 4 becomes unfeasible and more efficient representations need to be sought. Particular solutions often take advantage of some

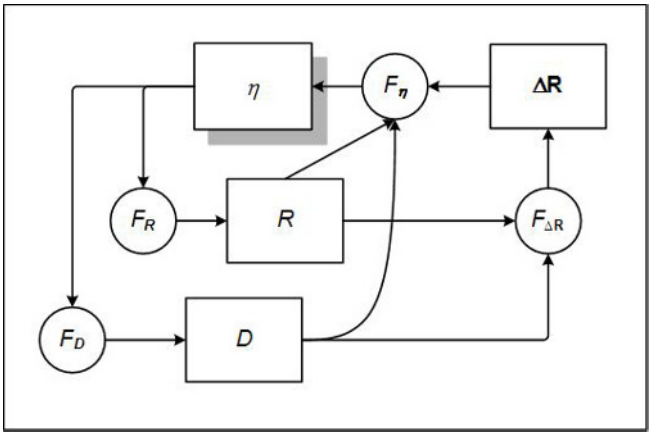


Figure 4. D and R communication range checked by detector ΔR.

domain information about the system being modeled and can offer good performance without incurring in the high cost of a general solution. Concrete examples include the identification of entities that can never communicate, since they are static and out of range of each other. Another example is a system where entities are grouped into clusters, requiring inter-cluster communicating, but where clusters are too far apart for exhibiting inter-cluster communication.

A common strategy is to use a virtual grid for partitioning space, introducing a pre-detection phase [9]. In this representation, a connection needs only to be maintained when two entities can *possibly* communicate. This solution requires that when an entity crosses a cell boundary it signals an event so new interactions can be found. Figure 5 represents drone D crossing several grid boundaries. In this solution, D sends a signal whenever it changes from cell in the virtual grid. In the example, only within cells C1 and C2 can drone D and radar R interact. Thus, communication links need only to be established when D is inside these two cells. Actually, radar detection occurs only between points **a** and **a** of Figure 5, and a more detailed model is needed, as described below.

This solution requires the dynamic management of links between the entities that can possibly interact. Dynamic topologies are supported by HFSS networks where grid management can be assigned to the network executive. Each entity has a connection to the executive that becomes responsible to establish the dynamic communication topology. In the drone/radar system, when the entities do not communicate, the model can be represented by the simplified HFSS network of Figure 6. This model keeps drone and radar apart since the drone is not in radar vicinity.

When entities become closer, a range detector is created to signal the exact instant when interaction begins. In the drone/radar pair this event signals the instant when the radar starts sampling drone position. This situation is depicted in Figure 7 corresponding to the HFSS topology already presented in Figure 4. With this solution we postpone the introduction of a (computationally) expensive detector until there is some possibility of communication. In scenarios with a large number of entities this strategy can dramatically improve performance. The degree of *closeness* between entities is a function of the grid size, and it needs to be chosen to minimize false positive communication messages.

When the drone enters radar range we need to generate drone positions as sensed by the (rotating) radar beam. For obtaining this data we need to detect drone and radar-beam interaction/collision, as depicted in Figure 8.

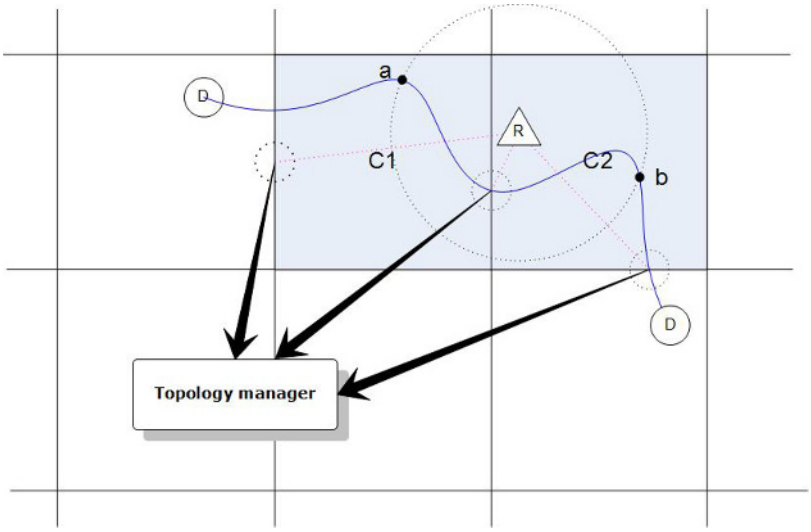


Figure 5. Space tessellation with a virtual grid.

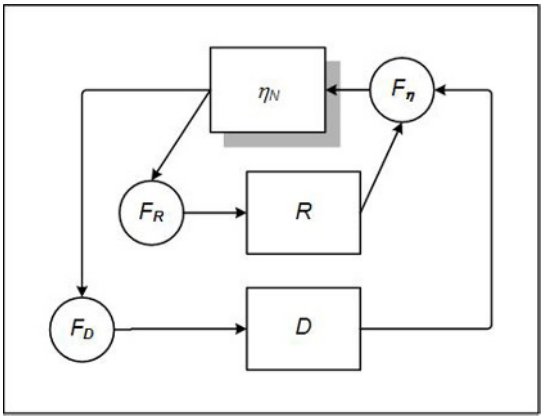


Figure 6. D and R not communicating.

The production of radar data involves the replacement of ΔR by a more complex detector ΔB that signals the exact detection of the drone by the radar beam. Figure 9 represents the topology of the new model. Additionally, new links were also created to represent drone/radar interaction.

As pointed by this example, modifications in topology required for representing mobile entities appears to be application dependent and we foresee no one-fits-all solution to the problem. In the next section we present a more complex scenario involving mobile entities that corroborates this conjecture.

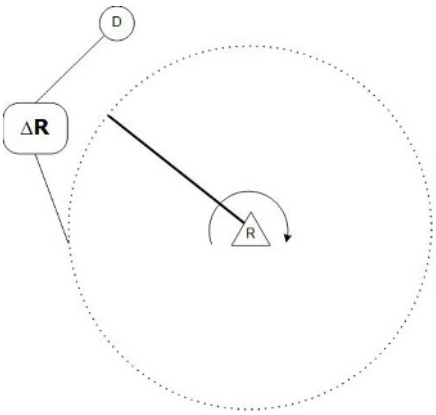


Figure 7. D and R in proximity.

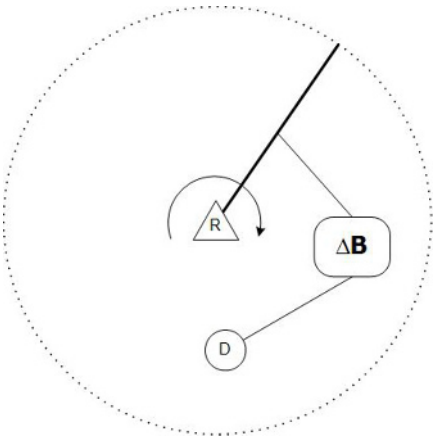


Figure 8. Drone in radar range.

4 Defense System

We consider now a scenario involving a defense system able to launch pursues to cancel drones. Figure 10 represents a model with radar R1, drone D1 and Pursuer P1. Detector Δ1 is responsible for tracking the collision between the beam of radar R1 and drone D1, signaling the corresponding detections to R1. Detector Δ2 checks for the proximity between D1 and P1, signaling D1 cancellation. This model is dynamic since detectors Δ1 and Δ2 are only created when/if certain distance conditions occur.

To show the effectiveness of HFSS in representing more complex systems we have developed a scenario with 2 airborne radars. Radars are modeled by their rotating beam and they are themselves mobile entities. Simulation results are presented in Figure 11, where results were obtained using the implementation of HFSS based in JUSE [10], a programming language supporting software components. The two drones D1 and D2 are modeled with a HFSS function generator. D1 has a circular

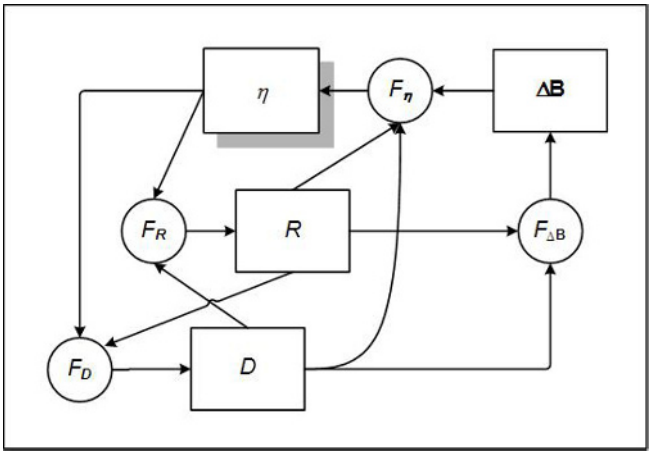


Figure 9. Model topology for drone within radar range.

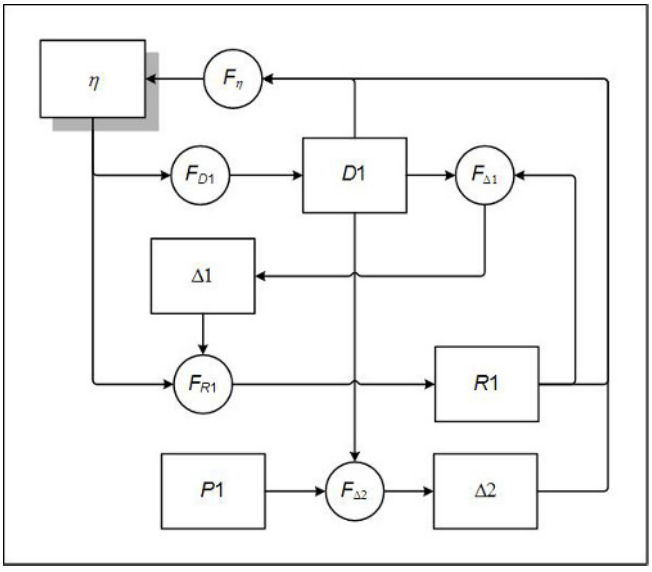


Figure 10. HFSS topology for pair drone/pursuer.

trajectory while D2 performs a sequence of arched trajectories. The linear velocity of all entities are kept constant during the simulation run. When drones are detected, a pursuer linked to the detecting radar is launched. Drone detection is based on a virtual grid algorithm run by the executive. As shown in the last section, this approach requires the emission of a discrete flow each time a drone/radar changes its position in the virtual grid. Only a single pursuer is assigned to each drone, requiring the cooperation between radars. For a better description, the graphical representation of Figure 11 depicts

radar range, radar beam position, and the trail of each mobile entity. Radars detection points (relative to radar current position), are represented in the small left frames.

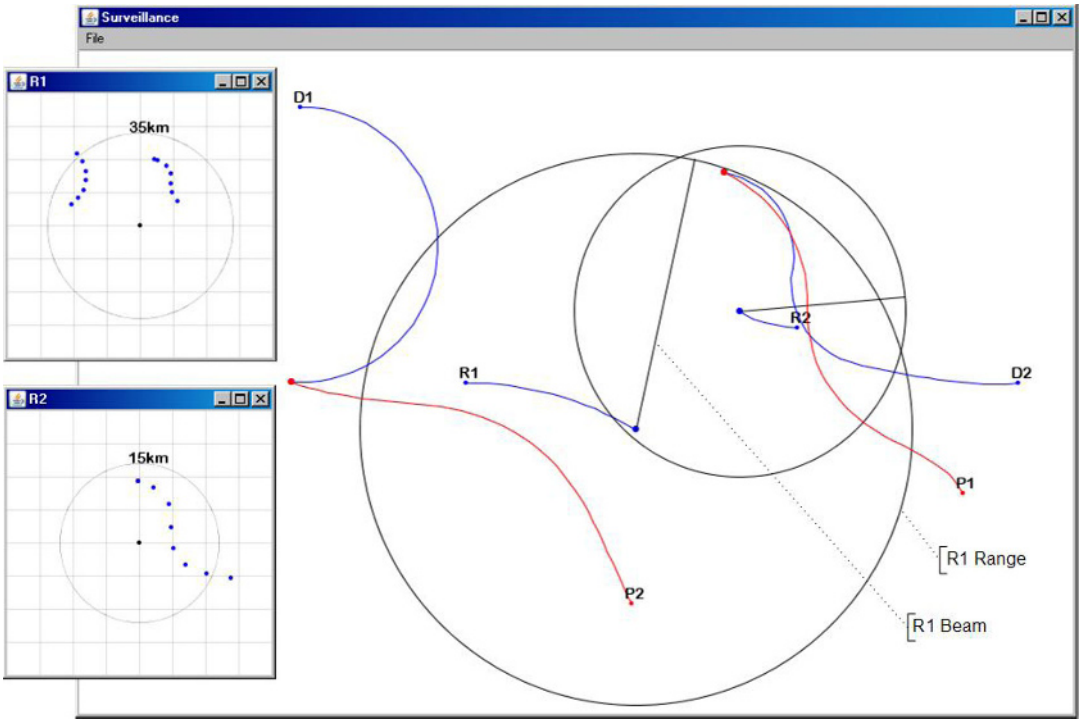


Figure 11. Simulation results for two moving/rotating radars and two-pairs drone/pursuer.

This adaptable topology used in the defense system proved to be scalable, accommodating easily a variable number of drones/radars. The solution becomes also very effective since it frees simulations from unnecessary components. Domain based optimizations were also enabled by HFSS general representation expressiveness and flexibility. This was the case of pursuers that were kept disconnected from radars, improving simulation efficiency by avoiding the creation of non-required pursuer/radar detectors.

Pub/sub communication as supported by the HLA [3], can be used to represent low accuracy detectors based, usually, on fixed sampling rate. However, high accuracy detectors, like the ones used here to represent radar beams, require adaptive sampling rate and p2p communication. This kind of communication also simplifies the interaction drone-pursuer, whereas multicast would become non-effective. A p2p formalism supporting dynamic topologies, like HFSS, can easily represent multicast communication, as shown in this paper. However, the representation of p2p using multicast seems not an easy task.

5 Perspectives for Activity Integration

The concept of activity has been used to track the computational effort in quantized systems [11]. A promising research perspective is to apply this concept to other formalisms, like HFSS, that provide a representation of continuous signals based on discrete automata.

The activity of a signal is defined by [11]:

$$A(T) = \sum_i |m_{i+1} - m_i|$$

where m_i and m_{i+1} are adjacent maxima and minima.

The number of threshold crossings of a Q-DEVS signal is given by [11]:

$$C(T, D) = \frac{A(T)}{D}$$

where D is the quantum size.

A typical trajectory of a Q-DEVS integrator is depicted in Figure 12. The integrator receives values $x_0, ..., x_3$ at times $t_0, ..., t_3$. Whenever the value y' crosses a threshold level the integrator produces a discrete event value and for the case in figure the integrator generates values $y_0, ..., y_4$. Q-DEVS has been show to perform better than the corresponding fixed step-size solvers (time-driven simulation) [11].

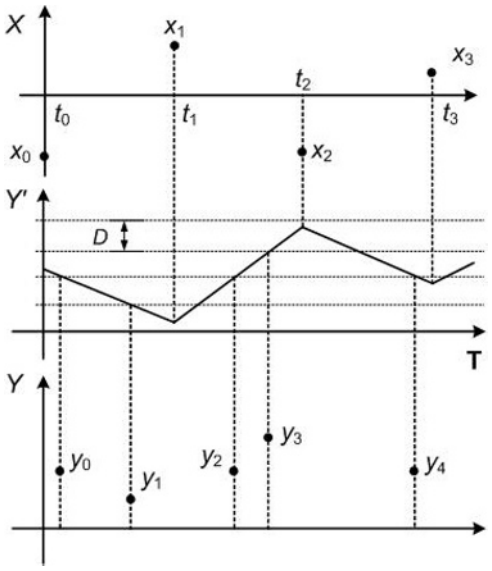


Figure 12. 1st order Q-DEVS integrator trajectories.

HFSS uses a representation of continuous signals based on sampling. HFSS models have independent and self-adjusting sampling rate, departing, like Q-DEVS, from the conventional time-driven simulation. However, sampling rate is not directly connected to threshold crossing but rather to the estimated error in signal representation. In a HFSS integrator, for example, a new sample is taken whenever the estimated error goes beyond a certain value. Figure 13 depicts typical trajectories of a n-order HFSS integrator ($n > 1$), where a sample is taken whenever the estimated error reaches a maximum value. Likewise Q-DEVS, HFSS also enables a better performance than fixed-step solvers [12]. An interesting research perspective is to adapt the concept of activity to represent higher order solvers based on error information.

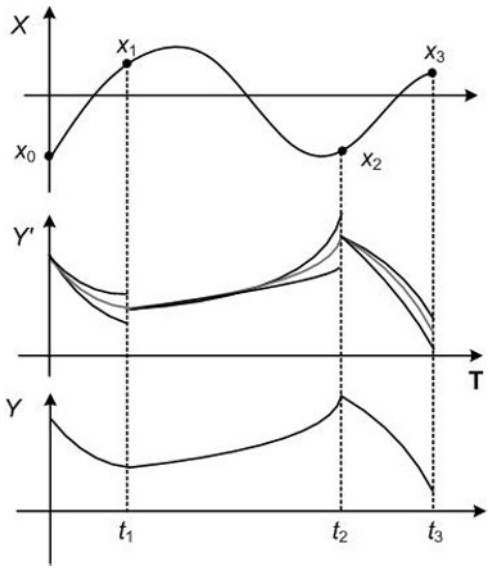
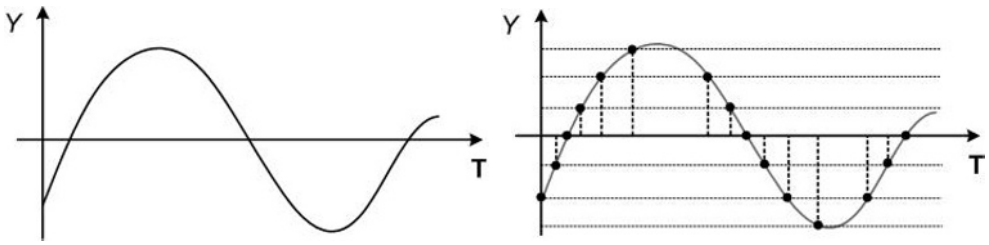


Figure 13. HFSS integrator trajectories.

The use of a continuous representation to describe continuous signals is another distinctive feature between HFSS and Q-DEVS. When modeling generators -like the one defined in Section 2.1 and used in Sections 3 to describe the trajectories of mobile entities- a HFSS generator does not undergo in any activity. For example, the generator of Section 2.1 requires no computation effort to represent the signal of Figure 14(a). On the contrary, a Q-DEVS representation requires the generation of values to represent the continuous signal with discrete events as depicted in Figure 14(b).



(a) HFSS generator. (b) Q-DEVS generator.

Figure 14. Signal generators.

Using activity to estimate the computation effort in sampled-based systems looks another promising topic for future research.

6 Conclusion

Peer-to-peer (p2p) dynamic topologies offer an effective representation of spatially moving entities. Under this paradigm models can capture the current system configurations that is dependent on the physical location of entities. Given that position evolves over time, dynamic topologies enable to keep an up to date model that mimics reality. This representation contrasts with a more common solution based on publish/subscribe communication that imposes less clear models and make it difficult to represent the peer-to-peer communication that is commonly imposed by some entities, like for example, pursuers and drones in a defense system. The ability to represent all interactions as p2p provides a general framework for described arbitrary systems requiring different kind of interactions. HFSS provides a flexible framework for describing spatial entities by enabling the necessary changes in model topology to capture entities dynamic interactions.

Acknowledgments

This work was partially supported by the Portuguese Foundation for Science and Technology under project PTDC/EIA-EIA/100752/2008.

References

[1] X. Chen, Y. Chen, F. Rao, *An Efficient Spatial Publish/Subscribe System for Intelligent Location-Based Services*, in *Distributed Event-Based Systems* (2003), pp. 1–6

[2] S.Y. Hu, *Spatial Publish Subscribe*, in *2nd International Workshop on Massively Multiuser Virtual Environments* (2009), pp. 252–258

[3] F. Kuhl, R. Weatherly, J. Dahmann, *Creating Computer Simulation Systems: An Introduction to the High Level Architecture* (Prentice Hall, 1999)

[4] USDoD, Tech. rep., US Defense Modeling and Simulation Office (2002)

[5] F. Barros, *Semantics of Discrete Event Systems*, in *Distributed Event-Based Systems* (2008), pp. 252–258

[6] B. Zeigler, *Theory of Modelling and Simulation* (Wiley, 1976)

[7] F. Barros, *Dynamic Structure Multi-Paradigm Modeling and Simulation*, ACM Transactions on Modeling and Computer Simulation **13(3)**, 259:275 (2003)

[8] G. Ciardo, D. Nicol, K. Trivedi, *Discrete-Event Simulation of Fluid Stochastic Petri Nets*, IEEE Transactions on Software Engineering **25(2)**, 207:217 (1999)

[9] C. Ericson, *Real-Time Collision Detection* (Elsevier, 2005)

[10] F. Barros, *Increasing Software Quality through Design Reuse*, in *7th International Conference on the Quality of Information and Communications Technology* (2010), pp. 236–241

[11] A. Muzy, R. Jammalamadaka, B. Zeigler, J. Nutaro, *The Activity tracking paradigm in discrete-event modeling and simulation: The case of spatially continuous distributed systems*, SIMULATION: Transactions on the SCS **87(5)**, 449:464 (2011)

[12] F. Barros, *Comparing Synchronous and Asynchronous Variable Step Size Explicit ODE Solvers: A Simulation Study*, in *PADS* (2007), pp. 32–37