A feasibility study of the HLA bridge

Juergen Dingel†
David Garlan‡
Craig A. Damon§

March 15, 2001
CMU-CS-01-103

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

†Queen's University
‡Carnegie Mellon University
§University of Vermont

This research was sponsored in part by the Defense Modelling Simulation Office (DMSO) and by the Defense Advanced Research Projects Agency (DARPA) under Contracts No. F30602-00-2-0616 and N66001-99-2-8918. The views, findings and conclusions or recommendations contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the DMSO or DARPA.

Keywords: Component integration standards, distributed modeling and simulation, High-Level Architecture, Run-Time Infrastructure, bridge federate

Contents

Abstract

The *High-Level Architecture* (HLA) provides a common architecture for distributed modeling and simulation. In its original form the HLA allows a number of simulations to be joined together into a federation using a single run time infrastructure. Recently there has been an interest in joining multiple such federations together using a mediating unit, called an HLA "bridge." This document presents an in-depth study of the feasibility of an HLA bridge in the context of the current HLA interface specification. The results are summarized on two levels. First, we identify general classes of problems and solutions. Second, we provide a detailed discussion of the desired behavior of selected service protocols in the presence of a bridge federate.

Chapter 1

Introduction

The High Level Architecture (HLA) was designed as a component integration standard for cooperating, distributed simulations [KWDJ99]. Specifically, it defines a simulation *Interface Specification* (IFSPEC) and *Run Time Infrastructure* (RTI) that permits a set of independently-developed simulations (called *Federates*), to be brought together into a single coordinated ensemble (called a *Federation*). The IFSPEC identifies a set of *Services* (sometimes also called *Messages*) that a participating federate may invoke on the RTI, or vice versa [Com99].

As standards go, the HLA is relatively complex. It contains facilities that allow federates to join and leave a federation; it defines services that federates can use to communicate simulated events to other federates and to receive events that they produce; it provides a timing model with varying levels of guarantees about temporal ordering, it supports object ownership migration; it allows federates to define synchronization points for checkpointing and saving state.

Given this complexity, reasoning about the specification (e.g., to guarantee properties such as absence of race conditions and deadlocks) is a non-trivial problem. Indeed, over the past five years the HLA has undergone considerable review and scrutiny, leading to numerous improvements to the standard as it has moved from a proposal of DMSO to an accredited IEEE standard.

In its original design the HLA assumed that a federation would be composed of a single RTI coordinating a single set of federates. More recently, however, there has been considerable interest in being able to define a federation as a set of linked RTIs each with their own sets of federates. Ideally, such a "composite" federation would permit separately-developed, and separately-specified federations to work together, without significant modification to any of the individual federations or to their RTIs. Moreover, with suitable glue mechanisms, it should be possible to provide certain kinds of visibility restriction across federates. For example one federation might not expose all of its objects or events to another.

In realizing such a scheme, an important question is how the various federations might be linked. One proposed solution is to provide the "glue" by using a special "bridge" federate to link two federations. Such a federate would act as a mediator, passing events between the two federations. The bridge federate would appear to be an ordinary federate to both federations, effectively encapsulating the federation substructure on each side.[1] Furthermore the bridge could handle the various filtering and event translation needs to "match impedances" of the joined federations or enforce security restrictions.

The use of a bridge federate is architecturally attractive for a number of reasons. Since it simply looks like any other federate, multiple federations could be joined transparently to the joined federations. Furthermore, in principal the use of a bridge would require no changes to the current HLA specification: by using existing services and event subscriptions a bridge should be able to update each side appropriately.

While attractive in principal, the notion of a bridge federate raises a number of questions. Does the introduction of a bridge introduce new sources of deadlock or inconsistency? Can a bridge federate obtain enough information via the current HLA API to keep both sides in synch? If not, what changes would need to be made to the API or RTI to allow sufficient visibility? Are there special protocols of interaction that a bridge developer would need to be aware of to make sure that the bridge is designed correctly?

In this report we provide answers to these questions. This is the result of an in-depth examination of the HLA specification. During this examination we attempted to identify in an exhaustive manner the potential sources of problems and itemize possible solutions to those problems.

Our approach to this investigation was based on two principles. First, rather than looking just at specific instances of problems (characterized in terms of specific service calls), we have attempted to characterize "problem classes" and "solution classes". In this way our insights and conclusions should remain valid even in the face of changes to the HLA specification. Moreover, the codification of problem classes helps to expose the underlying causes of the problems, rather than specific symptoms. In a similar way, by focusing on solution classes, we are arming the bridge designer with strategies that can be applied broadly, rather than pointwise solutions. Second, rather than looking at individual services, we have focused on collections of services and the mini- protocols that coordinate them. In this way, we are able to identify problems that are caused by sequences of calls – problems that don't show up in isolation when a single service is examined in isolation.

1.1 Description of HLA bridge

An HLA bridge should allow two federations that

- are physically/geographically separated, and
- employ different Federation Object Models (FOM)

to interact "seamlessly." [KWDJ99] That is, the presence of the bridge should be unobservable to other federates. Moreover, the bridge should be capable of carrying out various event processing actions, such as filtering and renaming.

[1] A bridge federate is, of course, not the only solution. For example, one might instead join the two RTIs via a lower-level RTI link.

To understand how a bridge federate might work, it is helpful to view it as consisting of three logical parts, as illustrated in Figure 1.1, where two federations *F* and *G* are joined by the bridge *B*.

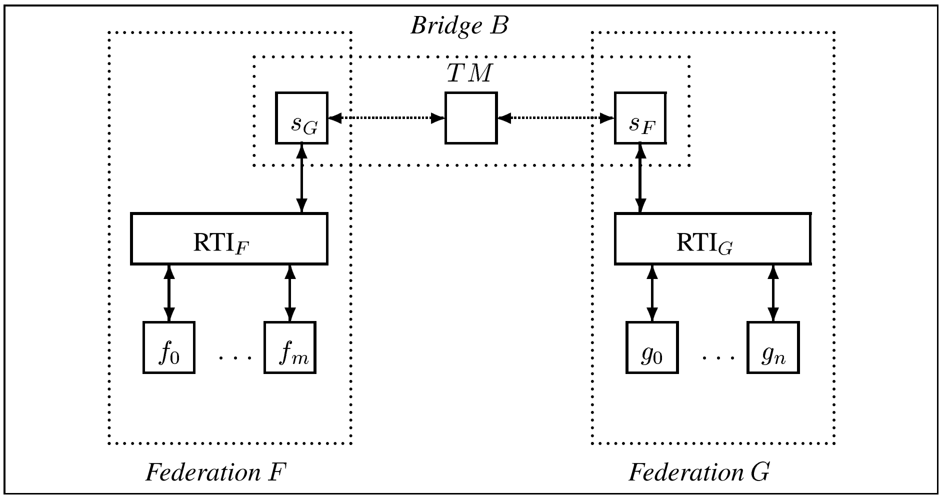

Figure 1.1: Federations *F* and *G* connected by a bridge

Surrogate s_F: Federate that interacts with the federation G on behalf of the federation F. We say that the surrogate s_F *represents* the federation F. s_F reflects relevant properties of the federation it represents to its federation, that is, the federation it is connected to. Note F may be connected to more federations through a second bridge. Intuitively, s_F represents all federations to the "left" of the bridge B.

Surrogate s_G: As above, except that every occurrence of s_F, F, and G is replaced by s_G, G, and F respectively.

Transformation manager *TM*: Module that translates between the two FOMs through a mapping that associates an entity (e.g., service, object, attribute, interaction) from one side of the bridge with corresponding entities on the other side of the bridge. Possibly carries out additional transformations, and thus may function as a guard.

We call the federates directly connected to an RTI *local* to that RTI. All other federates are called *remote*. For instance, federates $f_0, \ldots f_m$, and s_G in Figure 1.1 are local to RTI$_F$, while $g_0, \ldots g_n$, and s_F are remote to RTI$_F$.

1.2 Desired properties

The purpose of a bridge is to provide a transparent, loosely coupled, effective and efficient connection between federations. To this end, we would like a bridge to have at least the following properties.

1. The presence of the bridge should be unobservable to other federates.

2. The fact that some joined federate resides on the other side of the bridge and uses a different FOM should be unobservable to the federates.

3. Surrogates should be treated like "normal" federates: that is, neither the other federates nor the RTI should be aware of the special role of a surrogate. Moreover, as much as possible, surrogates should behave like all the other federates. Ideally, only the services that are available to any federate should be available to them.

4. The specification and implementation of the transformation manager should be modular with respect to the FOMs, the mapping, and the (security) transformations: that is, each of these should be easy to change.

5. The bridge should be as simple as possible. In particular, it should not behave as "another RTI" by, for instance, logging all communication.

6. We assume that a modeling federate cannot tell the precise number and identity of federates on the other side of a bridge.

1.3 Assumptions

We now list the assumptions that we will make in this document to simplify the analysis and presentation. The most important restriction is that bridges can connect federations only in a linear, list-like fashion.

1. We consider only *binary* bridge federates, that is, a bridge links at most two federations. For example, the left configuration in Figure 1.2 violates this condition.

2. A federation is connected to at most two bridges. For example, the right configuration in Figure 1.2 violates this condition.

3. Bridges are never used to connect federations in a circular fashion. In fact, a cycle of federations can give rise to a non-terminating sequence of service invocations. Consider, for instance, the *Request Federation Save (l,t)* service. Since a new save request replaces any outstanding save requests, the fact that a request reached a federate twice would not be discovered by that federate without special provisions.

4. We also assume that the surrogates do not engage in their own independent modeling behavior: that is, their sole purpose is to bridge federations. Consequently, the set of federates can be partitioned into surrogates and modeling federates.

5. Relevant capabilities of a federation that a surrogate has to provide include subscription, publication, and ownership of attributes; announcement of the current logical time; and participation in protocols for saving state and other synchronization.

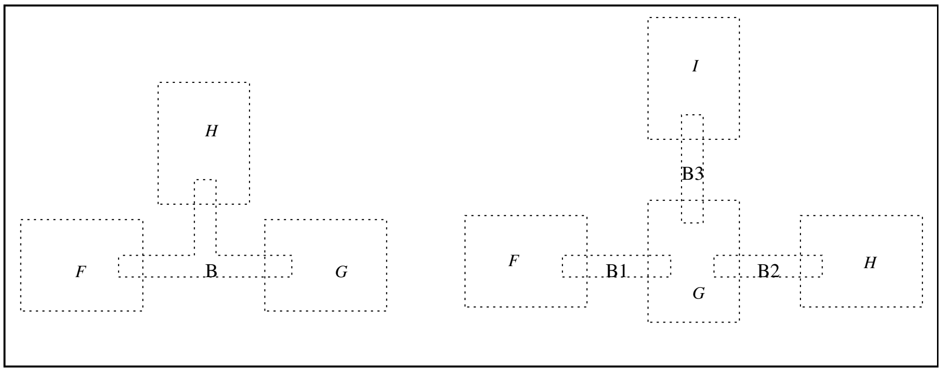

Figure 1.2: Two illegal configurations

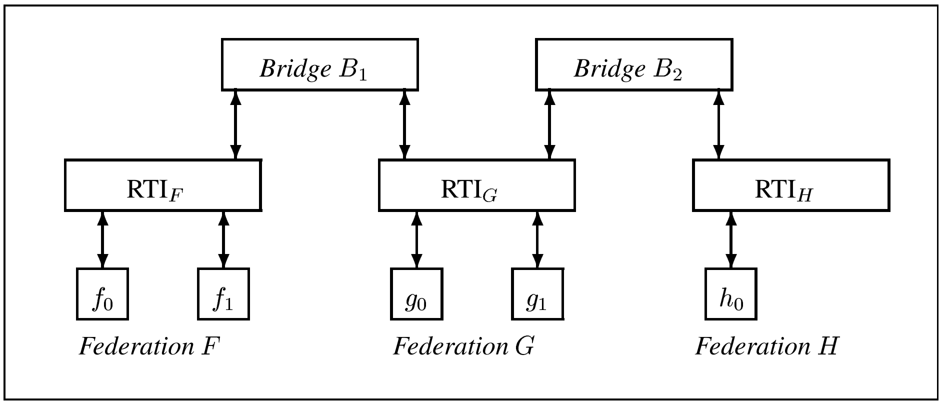

Figure 1.3: Three federations linked by two bridges

Suppose, for instance, three federations F, G, and H are connected via two bridges B_1 and B_2 as shown in Figure 1.3. Consider federate f_0. The connection should create the illusion to f_0 that its federation F is joined not only by f_1 but also by a federate whose properties are given by the sum of the properties of the federates g_0, g_1, and h_0. Similarly for the other federates. Figure 1.4 illustrates the effect of the bridges from the federate's point of view and attempts to capture the transparency of the bridges.

1.4 Approach

The goal of this paper is to understand the problems introduced into the HLA by bridges and consider possible solutions to these problems. The next three sections of the paper address these issues. The three sections are

Problems What kind of problems can plague a bridged HLA federation?

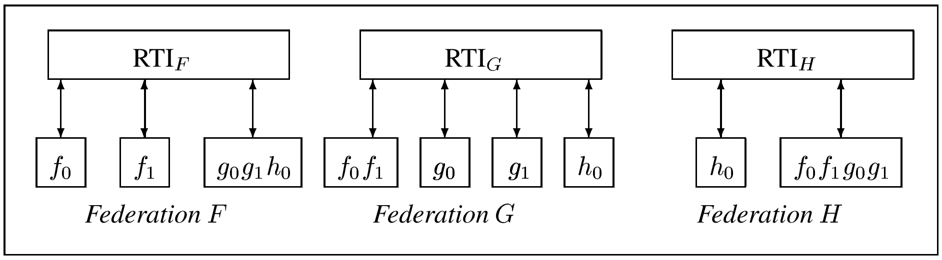

Figure 1.4: The effect of the bridges from the federates' perspective

Solutions What kinds of solutions can address these problems?

Protocol examples Detailed discussions of selected HLA protocols to illuminate the problems and solutions.

1.5 Action Categories

To help us abstract over specific behaviors it is helpful to distinguish between two kinds of services that cause various actions to take place. By action we mean some coordinated activity leading to a new overall state of a federation – such as a collective save of state, or the transfer of attribute ownership. Such actions are typically achieved by a sequence of service calls. For a given action the two kinds of services are:[2]

Action requests An action request, or request for short, is a service that requests one or more federates to carry out a specific action. An action request is called *universal* if it addresses all federates. Examples are *Register Federation Synchronization Point (l,t)* and *Request Federation Save (l)*. An action request is called *selective* if it addresses a specific subset of federates. An example is *Register Federation Synchronization Point (l,t,fds)* where *fds* is a non-empty set of federate designators. An action request is called *existential* if it addresses any one unknown federate. An example is *Negotiated Attribute Ownership Divestiture (oid,attrs,t)*. Let the services *Everybody Do A*, *Some Do A* and *Anyone Do A* denote universal, selective and existential federate-initiated action requests respectively.

Action request helper services An action request helper service, or request helper for short, is a service that is used to realize an action request. For instance, the set of synchronization request helpers is *Register Federation Synchronization Point*, *Confirm Synchronization Point Registration* †, *Announce Synchronization Point* †, *Synchronization Point Achieved*, and *Federation Synchronized* †. For all universal action requests *Everybody Do A* in the interface specification, the corresponding set of request helpers includes at least the services *Do A* †, *I've Done A*,

[2]When naming services, we follow HLA conventions, distinguishing services that are initiated by a federate from those that are initiated by the RTI by attaching a † to the latter.

and *Everybody's Done A* †. The semantics of these services is straightforward. If *Everybody Do A* is invoked on the RTI by some federate, then the RTI invokes *Do A* † on all federates. Once a federate has carried out action A successfully, it responds with *I've Done A*. Once the RTI has received an acknowledgement from all federates, it invokes *Everybody's Done A* † on all federates. Similarly for selective action requests. Note that for some action requests, the corresponding set of services may include additional services like *Everybody Do A Request Registration Status* † *(stat)* where $stat$ is a boolean value indicating the success or failure of the registration of the request. For instance, the request *Register Federation Synchronization Point* is answered by the RTI with *Confirm Synchronization Point Registration* † *(stat)*, before it notifies the other federates.

Tables 1.1 and 1.2 summarize the above classification. Note the subtle differences in the helper services between the three actions in Table 1.1.

Table 1.1: Classification of some federate management services

Action	**Action request**	**Address mode**	**Request helpers**
Synchronization	*Register Federation Synchronization Point (l,t)* *Register Federation Synchronization Point (l,t,fds)*	universal selective	*Confirm Synchronization Point Registration † (l,stat)* *Announce Synchronization Point †* *Synchronization Point Achieved* *Federation Synchronized †*
Save	*Request Federation Save (l,t)*	universal	*Initiate Federate Save † (l)* *Federate Save Begun* *Federate Save Complete (stat)* *Federation Saved (stat,expl)*
Restore	*Request Federation Restore (l,t)*	universal	*Confirm Synchronization Point Registration † (l,stat)* *Federate Restore Begun †* *Initiate Federate Restore † (l,fd)* *Federate Restore Complete (stat)* *Federation Restored † (stat)*
Advance Time	*Time Advance Request (lt)* *Time Advance Request Available (lt)*	universal universal	*Time Advance Grant † (lt)*
Acquire Attribute Ownership	*Unconditional Attribute Ownership Divestiture (oid,attrs)* *Negotiated Attribute Ownership Divestiture (oid,attrs,t)*	existential existential	*Request Attribute Ownership Assumption n(oid,attrs,t)* *Request Divestiture Confirmation (oid,attrs)* *Confirm Divestiture(oid,attrs,t)* *Attribute Ownership Acquisition Notification † (oid,attrs)*
Divest Attribute Ownership	*Attribute Ownership Acquisition (oid,attrs,t)*	existential	*Request Attribute Ownership Divestiture † (oid,attrs,t)* *Attribute Ownership Divestiture If Wanted (oid,attrs)*

Table 1.2: Classification schema generalizing Table 1.1

Action	**Action request**	**Address mode**	**Request helpers**
A	*Everybody Do A*	universal	*Do A* †
	Some Do A (fds)	selective	*I've Done A*
	Anyone Do A	existential	*Everybody's Done A* †

Chapter 2

Problems

We now turn to a presentation of the classes of problem that can arise when using a bridge federate. For each class of problem we will use the following template:

1. **Description.** Description of the problematic behavior in terms of an unbridged federation.

2. **Explanation.** Explanation why this behavior is problematic in the presence of a bridge.

3. **Example.** Illustration of the problem by means of an example.

4. **Occurrences.** Enumeration of the places in which the problem is known to occur.

5. **Related problems.** Brief discussion of related problems.

6. **Classification.** Each problem can be described in terms of the following three high-level problem categories. Some problems fall into more than one category. The next section describes these categories.

2.1 Problem categories

Before listing problem sources and solutions, we need first to be clear what we are aiming to achieve? We would assert that to realize the vision of a bridge federate, as outlined early, any solution must satisfy two general semantic requirements.

The first semantic requirement is *compositionality:* a set of bridged federations should in some sense be equivalent to a federation achieved with a single RTI and the union of the federates. In particular, all of the normal operations on federations, such as establishing synchronization points, transferring ownership, maintaining temporal orderings on events, etc., should continue to be possible.

The second semantic requirement is *uniformity:* a bridge should look no different from any other federate. Ideally there should be no special services or accommodations needed to design or use a bridge. This requirement can be violated in two ways: the bridge might do things that a normal federate cannot do, or it might not be able to do all of things that a normal federate can do.

This leads us to classify the problems into one of three categories:

1. **Violation of RTI semantics.** The behavior of the entire bridged system is different from behavior of corresponding unbridged system.
2. **Violation of federate semantics.** A surrogate exhibits behavior that a federate in the unbridged case would not exhibit.
3. **Weakening of federate semantics.** The behavior of a federate in the bridged system does not satisfy certain properties that the federate in the corresponding unbridged system would satisfy.

2.2 Selective addressing

Description

The HLA supports federate designators to allow for selective action requests. More precisely, some action requests are parametrized with a set of federate designators that specify which federates should be asked to carry out an action.

Explanation

In an unbridged federation F, a selective action request *Some Do A (fds)*, where *fds* is a set of federate designators, has the following, straightforward syntax and semantics.

Syntax An invocation of *Some Do A (fds)* is syntactically well-formed if all elements of *fds* are designators of federates in F.

Semantics An invocation of *Some Do A (fds)* triggers the following behavior:

- If the *fds* parameter is given and non-empty, then the RTI invokes the service *Do A* † on every federate that is designated by an element of *fds*.
- If the *fds* parameter is empty (or not supplied), then the RTI invokes the service *Do A* † on every federate in F.

When generalizing selective action requests to the bridged case, several decisions have to be made. For instance, should *fds* be allowed to contain designators of surrogates or remote federates? If so, what should the semantics of the invocation be? Can we ensure that the resulting generalized service respects the desired properties of a bridge listed in Section 1.2?

Example

Consider the service invocation

Register Federation Synchronization Point (l,t,fds)

in the unbridged case where *fds* is an optional set of federate designators. According to the interface specification [Com99], the elements of *fds* define the *synchronization set*, that is, the set of federates to be involved in the synchronization. The RTI will invoke *Announce Synchronization Point* † *(l,t)* on the federates specified by the elements of *fds*. If *fds* is empty or not given, the synchronization set consists of all federates. The addition of a bridge makes the service syntactically and semantically ambiguous. For instance, should the designators of remote federates be allowed to occur in *fds*? What is the semantics of the service when *fds* contains a surrogate?

Occurrences

In the presence of a bridge, all selective action requests pose the described problem. Fortunately, the interface specification contains only one selective action request: *Register Federation Synchronization Point (fds)*.

Related problems

One solution to the above problem is to allow an RTI to (directly or indirectly) invoke services on remote federates. The consequences of this kind of solution are investigated in Section 2.3.

Categorization

The possible inability of federates to address other, remote federates weakens the capabilities a federate normally has.

2.3 Consensus

Description

After the invocation of a universal action request *Everybody Do A* or a selective action request *Some Do A (fds)* by a federate, all the addressed federates have to achieve a common state by executing the action A.

Explanation

In the unbridged case, the RTI can invoke *Do A* † directly on the addressed federates. Assuming that the federates notify the RTI when they've done A, the RTI knows when the designated federates have done A and the desired common state is reached.

Two problems arise in the bridged case.

1. First, an RTI may not have direct access to the addressed federates.

2. Second, to be able to determine when all addressed federates have done A, an RTI requires a surrogate to know when all of the addressed federates that it represents have done A. This kind of information is currently unavailable to federates, which, in turn, creates a conflict with our desire to keep surrogates as indistinguishable from modeling federates as possible.

Occurrences

The HLA uses four actions that require consensus: synchronization, save, restore and time advance. The interface specification contains one related selective action request *Register Federation Synchronization Point (fds)* and five related universal action requests *Register Federation Synchronization Point (fds)*, *Request Federation Save*, *Request Federation Restore*, *Time Advance Request* and *Time Advance Request Available*.

Categorization

This problem is categorized as a violation of the federate semantics, because the desire to determine consensus forces a monitoring role on the surrogates that federates don't have.

2.4 Federate failure

Description

As illustrated above, action requests like *Register Federation Synchronization Point (l,t)* or *Request Federation Save (l,t)* cause a sequence of helper services being exchanged between federates and RTI. For some requests, once this sequence has passed a certain point, the interface specification does not allow any of the subsequent services to fail.

Explanation

In the unbridged case, the absence of a failure mode may be tolerable because the probability of failure is low and the RTI can handle them easily. However, the presence of a bridge may increase the failure probability substantially. In other words, it may frequently occur that a federate or the RTI is faced with the problem of having to communicate a failure without having the means to do it. In more abstract terms, the set of request helpers available to realize the request is incomplete in the sense that it does not accommodate failure adequately.

Example

Consider Figure 1.1 where $m = 1$ and $n > 0$. Suppose we interpret the inclusion of a surrogate in the federate designator set to indicate that all federates represented by the surrogate are implicitly included in the federate designator set; this is a solution described in Section 3.2. Furthermore, suppose that federate f_0 uses this ability and initiates a synchronization involving federates on both sides of the bridge. More precisely, assume the following sequence of service invocations:

1. Federate f_0 issues *Register Federation Synchronization Point (l, t)*. Remember that the missing set of federate designators indicates that all members in the federation are to engage in the synchronization. Consequently, the synchronization set consists of $f_0, f_1, g_0, \ldots, g_n$.

2. Suppose the registration succeeds and RTI$_F$ first sends a positive confirmation *Confirm Synchronization Point Registration* † *(l, "succ")* to f_0 and then uses *Announce Synchronization Point* † *(l,t)* to inform the federate f_1 and the surrogate s_G of the existence of the synchronization label.

3. Using the remote invocation mechanism to be described in Section 3.2 surrogate s_F issues *Register Federation Synchronization Point (l, t)*.

Suppose the registration of the label fails in federation G and RTI$_G$ sends a negative confirmation *Confirm Synchronization Point Registration* † *(l, "fail")* back to s_F. Two problems arise:

1. Surrogate s_G cannot communicate the failure of the synchronization to RTI$_F$, because the interface specification does not contain an appropriate service or parameter. In other words, the synchronization of a federate is not allowed to fail.

2. Similarly, RTI$_F$ has no means to tell its federates that the synchronization failed. In other words, once RTI$_F$ has issued *Confirm Synchronization Point Registration* † *(l, "succ")* it does not expect the entire synchronization to fail.

Occurrences

In the interface specification, only the synchronization action has an incomplete set of helpers and thus the failure problem occurs only there. The set of helpers for save and restore is complete, because the services *Federate Save Complete* and *Federate Restore Complete* have a success indicator as parameter.

Categorization

This problem has flavors of two categories. Failures may change the overall behavior (violation of RTI semantics) and force a surrogate to behave differently from a modeling federate (violation of federate semantics).

2.5 Service barriers

Description

Some services require certain designated federates to reach a certain state. Federates may assume that certain behaviors do not occur before that state is reached.

Explanation

In the unbridged case, there's only one federation that has to each the state. The RTI of that federation has all information to decide whether or not the state has been reached yet. Due to its central location it can adapt the processing of other invocations accordingly and thus enforce these barriers. Although service barriers are not mentioned in the interface specification, a particular RTI implementation may still enforce them and some federates of that RTI may implicitly or explicitly rely on them.

In the bridged case, each federation involved has to reach the state. One of them has to be the first to reach the state. The RTI of that federation will thus allow all invocations waiting for that state. However, other federations may not have reached that state yet and may still disallow these invocations. Consequently, the invocations can occur before all federates have reached the state leading to a violation of the service barrier. Two problems arise:

- First, the behavior of the entire bridged system does not match the behavior of the corresponding unbridged system.
- Second, federates relying on service barriers would have to be reimplemented.

Example

In an unbridged federation, for instance, the RTI implementation may be such that every synchronization request has to be completed before another one can be registered. More precisely, once a federate f has received *Announce Synchronization Point* † *(l,t)* f may assume that it will not receive another announcement of a different synchronization point until all federates have synchronized, that is, until f has received a *Federation Synchronized* † *(l)* from the RTI.

In the bridged case, however, it is very difficult for all federations involved in the synchronization to always simultaneously invoke *Federation Synchronized* † *(l)*. Some federates may receive the synchronization notification before other federates have synchronized. In that case, it would be possible for a federate in one federation to register another synchronization before the previous one is announced in other federations.

Occurrences

The service barrier problem described above is due to the inherently distributed nature of the consensus problem. Consequently, whenever federates assume a service barrier between action requests that require consensus, the above problem arises. Synchronization and ownership transfer are the primary examples in the HLA.

Categorization

The possible loss of service barriers implies a weakening of the RTI semantics.

2.6 State/behavior assumptions

Description

A federate can be in one of several states. Each state may enable different behaviors. The RTI knows the state of the federate and thus knows what behavior to expect from the federate.

Explanation

A surrogate may not always be in the state the RTI thinks it is in. Thus, a surrogate may exhibit "unexpected" behavior.

Example

Consider Figure 1.1. Suppose federate g_0 unconditionally divests the ownership of an attribute. Thus, the attribute becomes unowned in G. To avoid multiple ownership of this attribute, s_G, the surrogate representing G, acquires ownership. Thus, the attribute is owned by s_G, but not owned by any modeling federate. Consequently, if s_G is asked by RTI_F for the value of the attribute, it cannot provide it.

Occurrences

As described above, ownership divestiture is one example.

Categorization

This problem also points to a violation of the federate semantics.

2.7 Insufficient information

Description

At any given point, federates must have sufficient information to perform the behavior that is expected of them.

Explanation

Although surrogates and modeling federates serve largely different purposes, they should have the same interface, that is, their behaviors should be identical from the outside. This mismatch creates a conflict when the interface cannot provide a surrogate with all the information needed to perform some bridge-specific task.

Example

Suppose surrogate s_G of bridge (s_G, TM, s_F) joins a federation execution F. The bridge needs the relevant information of F so that s_F can adequately reflect it. However, there is no service to supply the bridge with that information. Note that the bridge could obtain the information from the FOM. This kind of reliance on the FOM may or may not be a problem.

Occurrences

As described in the example above, if a bridge joins a federation, the interface specification does not offer a way to communicate the relevant information.

Related problems

This problem is akin to the problem of federate failure described in Section 2.4.

Categorization

The problem is caused by the differences between federates and surrogates and thus implies a violation of the federate semantics.

Chapter 3

Solutions

Solutions will be presented for each of the problems listed in the previous sections.

3.1 Solution categories

Each solution falls into one of four categories. Not every problem will have solutions in all four categories.

1. **Add capability.** The capabilities of the RTI or a surrogate are enriched or the interface specification is enlarged.

   - **Strengthen surrogate.** A surrogate is allowed to query the MOM for the required information.
   - **Strengthen interface specification.** A service containing some required information is added to the interface specification. For instance, a service is added to communicate the failure of a request explicitly.
   - **Strengthen RTI.** The required information is communicated implicitly by adding a specific behavior to the RTI or federates. For instance, failure of a request is communicated implicitly through a timeout mechanism.

2. **Restrict use.** The use of problematic behaviors is avoided. For instance, a federate or the RTI could be prohibited from invoking certain services or using certain parameters.

3. **Ignore problem.** The problem and its consequences are ignored. For instance, a variety of possible failures appear to have been ignored in the current interface specification.

4. **Smart bridge.** The problem is solved by means of a clever implementation of the bridge.

3.2 Selective addressing

Add capability

Federates are given access to the designators of remote federates. The result is a fine-grain addressing scheme that allows each federate to address every subset of local and remote federates. However, this scheme would obviously allow federates to tell the number and identity of remote federates and thus violate one of the assumptions in Section 1.3.

Avoid use

There are two solutions in this category.

1. No selective addressing. If selective action requests are not used, the problem can be ignored. Since *Register Federation Synchronization Point (l,t,fds)* is the only such request, this might be a viable option.

2. Restricted selective addressing. We adopt a more coarse-grained addressing scheme that does not conflict with our initial assumptions. This solution is the most viable.

   Syntax As indicated in Figures 1.3 and 1.4, the surrogate representing some remote federates should appear to f as "owning" all of their relevant properties. An invocation of the service by a federate f in some possibly distributed federation F is syntactically well-formed if it is of the form *Some Do A (fds)* where all elements of *fds* are designators of federates that are local to the RTI f is connected to. Designators of remote federates must not occur in *fds*. Note, however, that local surrogates are allowed.

   Semantics The semantics of the invocation of a selective action request *Some Do A (fds)* in the bridged case is as follows.

   (a) If *fds* is not empty, then the service *Do A* † is invoked on a modeling federate f' in F if
      - the designator of f' is in *fds*, or
      - the designator of a surrogate representing f' is in *fds*.

   (b) If *fds* is empty (or not supplied), then the service *Do A* † is invoked on every modeling federate in F.

   In summary, using restricted selective addressing a bridged federation federate can address

   - each federate in its own RTI individually and
   - the remote federates represented by a surrogate in its own RTI collectively.

Ignore problem

If the problem is ignored, selective action requests may lead to unexpected results.

Smart bridge

Selective action requests are equipped with tags. The bridge maps each tag to a set of federates. In other words, tags are used to address subsets of federates implicitly.

3.3 Consensus

Add capability

Suppose the bridge (s_G, TM, s_F) connects two federations F and G as in Figure 1.1. Let f be a federate in F different from s_G. We discuss selective action requests of the form *Some Do A (fds)* only. The solution generalizes to universal requests of the form *Everybody Do A*. Suppose federate f has invoked *Some Do A (fds)* on RTI$_F$. The federates designated in *fds* need to be asked to perform A and subsequently, RTI$_F$ needs to be informed of completion. The protocol consists of two phases: A broadcast and a collection phase.

1. **Broadcast phase.** The designated federates are informed. First, RTI$_F$ uses the service *Do A* † to notify all designated, connected federates. If the surrogate s_G gets notified, the bridge invokes *Everybody Do A* on RTI$_G$. Now, RTI$_G$ can inform all its local federates except s_F through a *Do A* †. Note that if G is connected to a third federation H by another bridge, the request would also travel down that second bridge using the same mechanism.

2. **Collection phase.** Successful completion of the service is communicated back to RTI$_F$. We partition the set of federate descriptors *fds* in a single federation into in the set fds_{mf} of designators of modeling federates and the set fds_s of designators of surrogates. That is,

$$fds = fds_{mf} \cup fds_s$$

where

$$\begin{aligned} fds_{mf} &= \{fd \in fds \mid fd \textit{ designates a modeling federate}\} \\ fds_s &= \{fd \in fds \mid fd \textit{ designates a surrogate}\}. \end{aligned}$$

Note that due to the assumptions in Section 1.3, fds_s contains at most two elements.

The task of determining when all federates designated by *fds* are done can be divided into two subtasks:

(a) Determine when all federates in fds_{mf} are done. This is straightforward, because every federate in fds_{mf} will send an *I've Done A* message to RTI$_F$.

(b) Determine when all federates represented by the surrogates in fds_s are done. Let s_G be a surrogate in fds_s. We describe how the federates represented by s_G inform RTI$_F$ of completion of action A. We distinguish two cases:

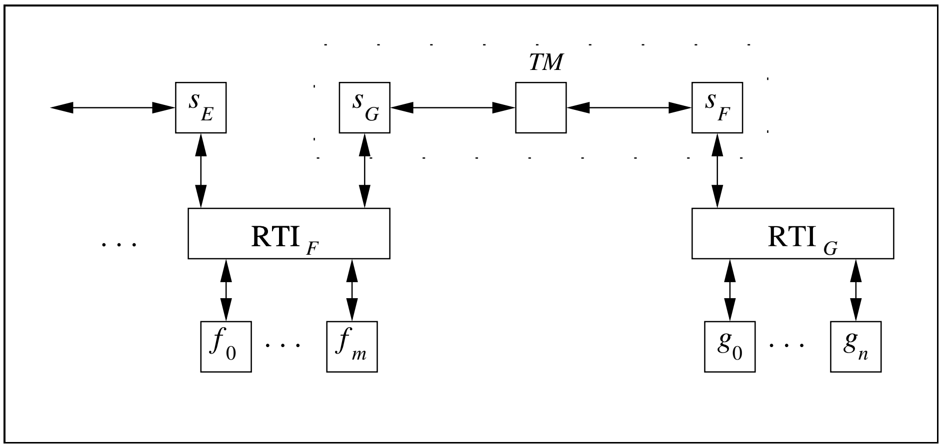

Figure 3.1: Illustration of the base case of the solution to the consensus problem

- Case 1 (Base case, Figure 3.1): All represented federates are only one bridge away from F, that is, they all are in federation G. In this case, surrogate s_F can determine whether all federates in G have done A in one of three ways.
  i. Autonomously: The surrogate s_F obtains this information directly from the MOM.
  ii. Without request: The surrogate s_F obtains this information from RTI$_G$ through a new service *Everybody But You Has Done A* †. Remember that we assume that an RTI cannot distinguish modeling federates from surrogates. Thus, not only surrogates but also modeling federates need to be prepared to receive this service.
  iii. Upon request: The surrogate s_G informs the RTI of its information needs through a new service *Let Me Know When Everybody But Me Has Done A*. RTI$_G$ then responds with *Everybody But You Has Done A* †. The shortcoming of the previous solution above is overcome at the expense of the introduction of a second new service.

  Once s_F has the information, it can send it across the bridge to s_G.
- Case 2 (Inductive case, Figure 3.2): At least one federate represented by s_G is more than one bridge away from F. Suppose that federate is reached via the bridge (s_H, TM, s'_G). By induction hypothesis, the completion of all federates represented by s_H can be communicated to s_H via s'_G. Surrogate s_H then communicates the information to RTI$_G$ using *I've Done A*. Surrogate s_F determines completion using one of the three ways described in the base case and communicates it to s_G.

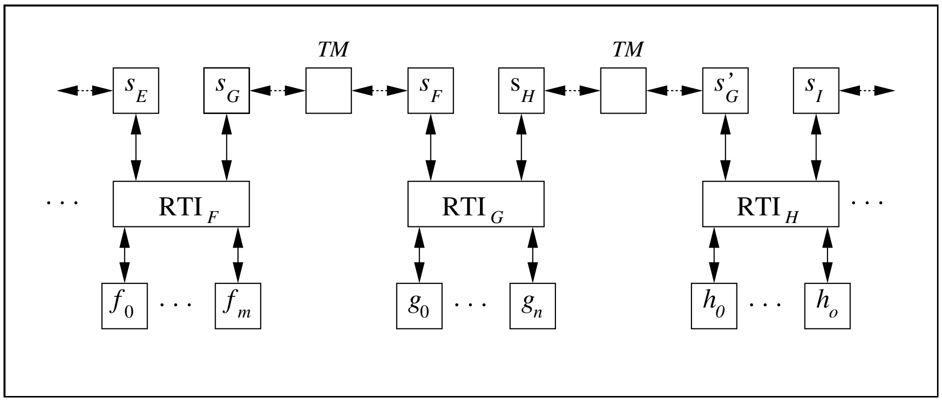

Figure 3.2: Illustration of the inductive case of the solution to the consensus problem

Avoid use

Since synchronization, save, restore and ownership transfer are central to the HLA, their use cannot be avoided.

Ignore problem

If the problem is ignored, action requests may lead to unexpected results.

Clever bridge

It is not clear how the bridge implementation could help solve this problem.

3.4 Federate failure

Add capability

Generally speaking, the cause of the federate failure problem is that the set of request helpers is not complete in the sense that they don't always allow the communication of a failure. The four solutions below attempt to remedy this by extending the capabilities of a surrogate and its RTI.

1. Add timeout. The surrogate s does not explicitly issue a service in the failure case. Instead, it withholds *I've Done A* forever. The RTI recognizes the failure of the request through timeout. For the second problem, the RTI withholds *Everybody's Done A* † forever. While timeouts are easy to implement, they also lead to inefficiency, especially for a long sequence of bridges.

2. Add a new service. The service *I Failed To Do A* is added to the interface specification. Surrogate s issues *I Failed To Do A* explicitly and the RTI behavior is

Table 3.1: Summary of a solution to the federate failure problem

Action request	**Helpers**
Everybody Do A	*Everybody Do A Request Registration Status † (stat)*
	Everybody Do A Request Status † (stat)
Do A †	*Do A Request Status (stat)*

augmented appropriately. To solve the second problem, the service *Somebody Failed To Do A †* is added and sent by the RTI when at least one federate failed to do A. This solution is costly, because RTI and federates must be able to send and receive the new service.

3. Add a parameter. The surrogate s issues *I've Done A ("fail")*. In other words, the service *I've Done A* is augmented with a success indicator. To solve the second problem, a success indicator is added to *Everybody's Done A †*. This solution is more efficient than timeouts and is easier to implement than a new service. However, since a negative success indicator negates the semantics conveyed in the name of the service, this solution can be regarded as unintuitive and error prone.

4. Rename service and add a parameter. The service *I've Done A* is renamed to *Do A Request Status ("succ")*. That is, *Do A Request Status ("succ")* and *Do A Request Status ("fail")* indicate successful and unsuccessful completion of A respectively. To solve the second problem, the service *Everybody's Done A †* is replaced by *Everybody Do A Request Status † (stat)*. While this solution is the conceptually cleanest, it is also costly. Table 3.1 summarizes this solution.

Avoid use

The avoidance of services that can fail appears to be too strong a restriction.

Ignore problem

Failures that cannot be communicated are ignored. Maybe an option if failure probability is low.

Smart bridge

A specialized bridge can alleviate the problem for at least some of the occurrences. For instance, for the case when synchronization fails in the bridged federation because the synchronization tag is already in use, the bridge can map the tag to a new unused tag.

3.5 Service barriers

Without a global, synchronizing entity it cannot be guaranteed that all federations reach a state at the same time. Since the purpose of the HLA-bridges is to connect distributed federations it seems unreasonable to assume the existence of such an entity. Consequently, the behavior deviation of the entire bridged system from the corresponding unbridged system described in Section 2.5 cannot be avoided. The possible reliance of federates on service barriers, however, can be dealt with.

Avoid use

We adapt the behavior of the federate. If possible, all dependencies on a service barrier assumption in the implementation of a federate are removed.

Smart bridge

The behavior of the bridge is adapted. By imposing additional constraints on the bridges it is possible to recreate the service barriers that federates rely on. For instance, the bridge could be asked to hold a particular service when a certain action request is pending.

3.6 State/behavior assumptions

Add capability

There does not appear to exist an added capability to remedy this problem.

Avoid use

Avoiding the use of ownership transfer is infeasible.

Ignore problem

Ignoring the problem introduces two issues to be handled. All federates must be prepared for the possibility of attribute updates becoming unavailable for indefinite periods of time unpredictably. Ignoring the problem also prevents any possibility of transferring ownership across the bridge, which may or may not be a problem.

Clever bridge

The bridge could stash the last value for any attribute that is orphaned. This solution, while fixing the first problem outlined above, introduces a new problem: the attribute value can never be updated, introducing new possible inconsistencies in the world being modeled.

3.7 Insufficient information

Add capability

Extending the capabilities of surrogates and RTI appears to be the most viable option.

1. Add new service: The surrogates ask the RTI for the information using a new service *What Is Value Of X?*. The RTI responds with *Value Of X Is* † *(v)* where v is the current value of x.

2. Add new parameter: It may be preferable to add a new parameter to an existing service. For instance, the value of x could be added to the existing service *Relevant Info* † *(info)* as an additional parameter. In other words, an invocation of *Relevant Info* † *(info)* could be replaced by *Relevant Info* † *(info,x)*.

Avoid use

The service *Join Federation Execution* appears to be too fundamental to be avoided.

Ignore problem

If the problem is ignored, the bridge may not behave as expected.

Clever bridge

Use the MOM to obtain any necessary information.

Chapter 4

HLA Protocol Examples

4.1 Overview

This chapter details how the proposed bridge definition impacts five different HLA protocols. Each protocol demonstrates distinct issues in how the bridge definition interacts with the base HLA definition.

Each section describing a protocol includes four subsections:

1. Review of the protocol.
2. Overview of how bridges work for this protocol.
3. Problems introduced by the bridge.
4. Potential solutions.

In all the scenarios, we assume a simple bridged world, with two RTI's, F and G, two modeling federates in each federation, f_0 and f_1 in F and g_0 and g_1 in G, and two surrogates for the bridge s_F and s_G. Figure 1.1 illustrates the general case of the simple bridged world.

4.2 Save

The protocol

Figure 4.1 illustrates the interesting services exchanged in the save protocol. In the protocol some modeling federate, f_0 in the example, requests a federation save by invoking the *Request Federation Save* service. The RTI responds by sending the *Initiate Federate Save* † to every federate in the federation. As soon as each federate has completed saving its own state, it invokes the *Federate Save Completed* service. Once every federate has informed the RTI that its save has completed, the RTI announces the completion of the save to each federate using the *Federation Saved* † service.

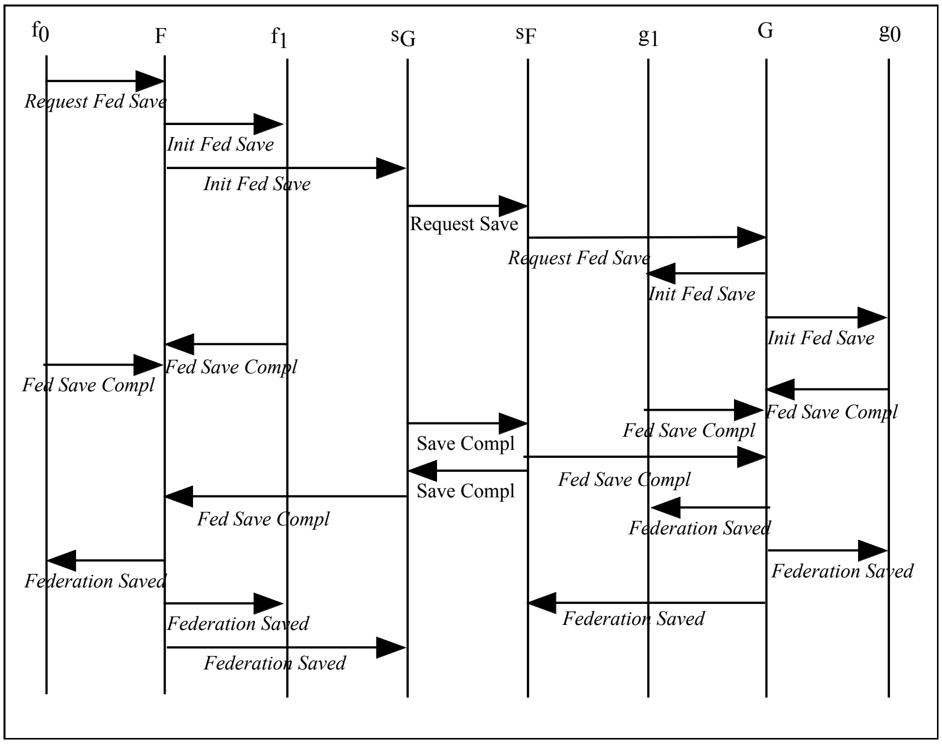

Figure 4.1: Example save protocol

Bridging the protocol

In a bridged environment, the bridge must propagate two kinds of information: the initial request for the save and the completion of the save in each federation being bridged. In Figure 4.1, we represent this communication as the Request Save and Save Completed messages sent across the bridge.

Problems

Although the implementation of the Request Save message is straightforward, the RTI provides no services to inform the surrogate when the Save Completed message should be sent.

Save is an example of the consensus problem. Neither surrogate can send the Save Completed message across the bridge until it knows that all other federates in its federation have completed saving. The RTI will not announce that the save is complete until it hears from the surrogate as well.

Solutions

Two solutions for this problem are obvious: polling the MOM and adding a new service. Each has its advantages and disadvantages.

The solution proposed in [SBBH98] requires the surrogate to poll the MOM to monitor when all other federates have announced the save. This polling allows the surrogate to become aware when the remaining federates have been saved and thus when it should send the Save Completed message across the bridge.

This polling is expensive if done frequently and leads to a possibly significant delay in the save completed notification if done infrequently. Adding a new service, itself a negative, obviates these problems. A pair of new services, *Request Notification of Last Saver* and *Last Saver Notification* † can be defined that allow the surrogate to be treated equivalently to any other federate and do not require any changes in the existing federate implementations.

4.3 Time advance

The protocol

Time advancement is another consensus-based protocol; logical time can advance for each federate only when it can advance for all federates. Figure 4.2 illustrates the interesting services exchanged during an example protocol.

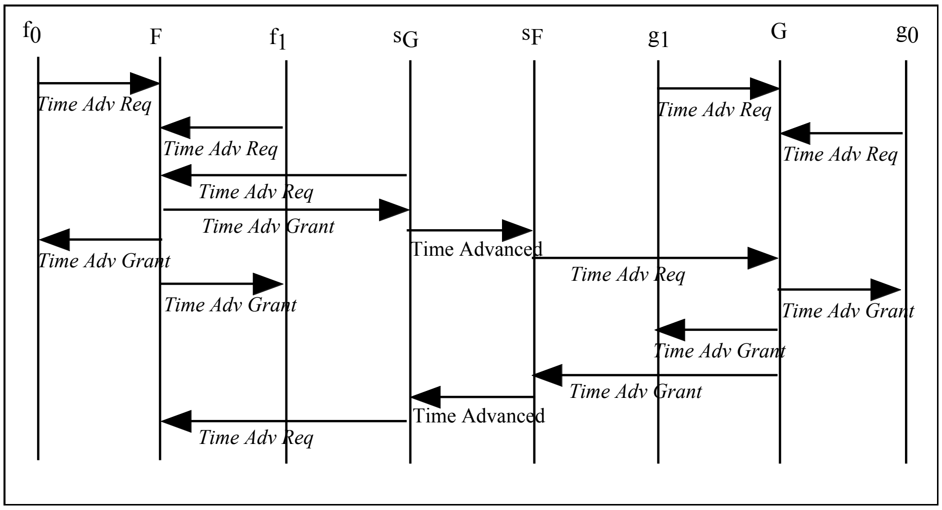

Figure 4.2: Example time advance protocol

In this discussion, we assume that all federates are both time-constrained and time-regulating. Furthermore, we assume that the lookahead is zero for all federates. None of these complications change the basic scenario considered here.

In the protocol, each federate announces how far it is prepared to advance time using the *Time Advance Request* or *Time Advance Request Available* service. In this example, we use the *Time Advance Request* service exclusively, but the two can be freely intermixed. Each of these services includes a parameter indicating how far the federate is willing to have time advance. By invoking these services, the federate is guaranteeing that it will not send any more time stamped messages with a time stamp prior to the indicated time (or prior or equal to the indicated time for *Time Advance Request Available*).

When every federate has agreed to advance beyond the requested time for any federate, the RTI responds by sending the *Time Advance Grant* † service to the federate. The federate may then advance its logical time to the time given in the *Time Advance Grant* † service.

Note that any one federate failing to request advancement of time will freeze time for all federates.

Bridging the protocol

In contrast with save requests, to bridge time advancement only one kind of information is transmitted across the bridge. Suppose the RTI of F has received time advancement requests from all federates (including the surrogates), a time advance has been determined, and the RTI has sent out the *Time Advance Grant* † services. The time advance grant received by surrogate s_G will be passed through the bridge in form of a *Time Advanced* message. Surrogate s_F on the other side of the bridge will then ask federation G for an advance contained in the message. Once a time advance has been negotiated in federation G, it flows across the bridge to surrogate s_G who uses it in the next time advance in F.

Problems

There appear to be three problems with the above protocol.

1. The protocol has the potential for the same consensus problem exhibited in the save protocol.

2. The bridge may also expose an existing flaw in a modeling federate. In the absence of a bridge, one federate in each federation may wait for the other federates to advance time before doing so itself. In a bridged federation, this behavior will freeze time for all federates in all federations that are connected.

3. The treatment of the two surrogates s_G and s_F is asymmetric. While s_F issues an advance request based on a Time Advanced message received from the bridge, s_G issues its first advance request independently. A surrogate must be able to determine which of the two behaviors is most appropriate.

Solutions

1. Because the time advancement protocol includes an *Everybody But You Has Done A* † service, the consensus problem is solved in the current implementation.

2. Disallowing aberrant federates solves the second problem above.

3. A mechanism based on time outs may offer a simple solution to the third problem.

4.4 Ownership transfer

The protocol

The ownership transfer protocol controls the transfer of ownership of attributes (the right to generate value updates for individual attributes). The primary requirement for the protocol is to prevent an attribute from being owned by two different federates at the same time. Ownership can be divested by the current owner conditionally (waiting for another federate to be willing to claim ownership) or unconditionally (divesting whether or not another federate is willing to claim ownership). The HLA does not require attributes to be owned at all times. Figure 4.3 shows an example protocol of conditional divestiture.

The conditional transfer protocol begins by the current owner, f_0 in the example above, invoking the *Negotiated Attribute Ownership Divestiture* service describing the attribute(s) the federate is seeking to divest. The RTI responds by requesting another federate to choose ownership by invoking the *Request Attribute Ownership Assumption* † service for each federate that can own the attribute. Any interested federate among those notified may respond with the *Request Attribute Ownership Acquisition If Available* service. Once the RTI has seen that another owner is available, it notifies the current owner using the *Request Divestiture Confirmation* † service. The original owner finally divests itself using the *Confirm Divestiture* service. The RTI now informs the new owner of its ownership using the *Attribute Ownership Acquisition Notification* †. The RTI may notify other potential owners of that they will not become the owner with the *Attribute Ownership Unavailable* † service.

Bridging the protocol

Transfer of control across the bridge is problematic. Several approaches are possible. The surrogate can actively seek ownership it cannot guarantee satisfying or the surrogate can hide the ownership transfer, preventing transfers across the bridge.

In the first approach, shown in Figure 4.3, the surrogate, s_G in the example, requests ownership of every attribute that is published across the bridge and then becomes unowned. For these attributes, the complementary surrogate in the other federation, s_F in the example, currently owns the attribute in the other federation. That complementary surrogate now begins a conditional divestiture protocol. If that protocol succeeds, the

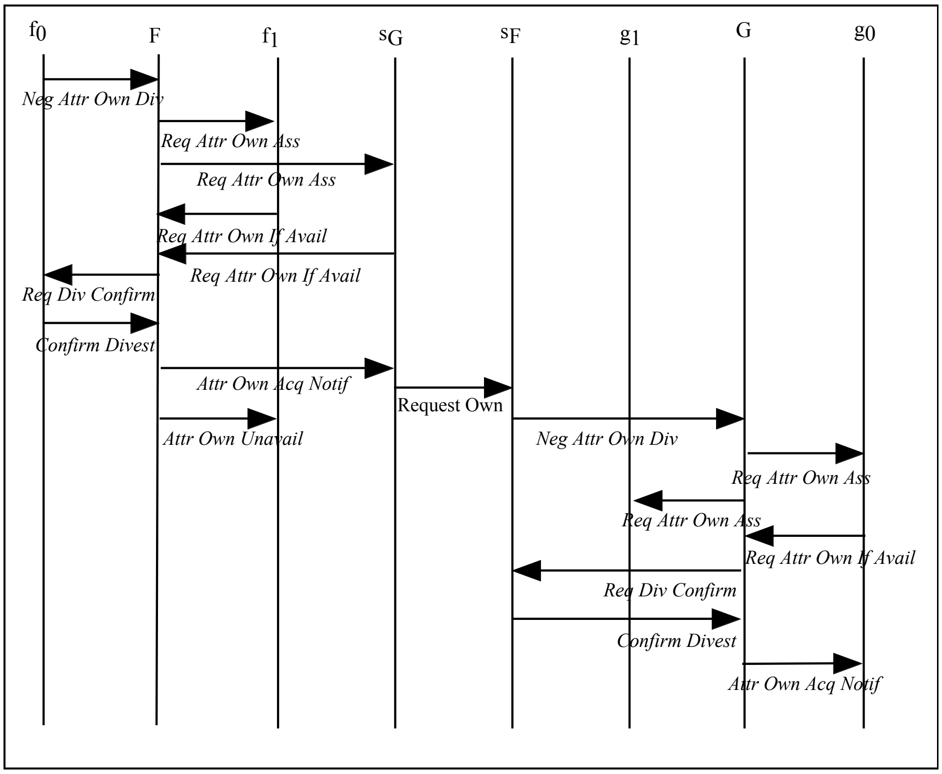

Figure 4.3: Example ownership transfer protocol

bridge is back to a consistent state. Otherwise, the original surrogate must initiate an unconditional divestiture protocol and the complementary surrogate must maintain the illusion of ownership.

The second approach is much simpler. The bridge hides all ownership transfers, with the complementary surrogate maintaining ownership of the attribute in the second federation.

Problems

The underlying problem with both of these approaches is that an allowable scenario, an unowned attribute, cannot be allowed in both federations simultaneously. If the attribute is unowned in both federations, the two RTI's could simultaneously award ownership to two distinct federates and violate the cardinal rule of attribute ownership.

To prevent this scenario, at least one of the surrogates must maintain ownership of the attribute, even if the attribute is currently unmodeled (unowned by any modeling federate). This unmodeled ownership introduces an inconsistency in the federation semantics.

Solutions

Three potential solutions present themselves for this problem; unfortunately, no solution is truly satisfactory.

The first solution is simply ignoring the inconsistency. This approach means that no value will be provided for the attribute for an indefinite period of time, even if an update is specifically requested. This lack of update means that a new federate can know about an object, but not be able to discover its attributes, for example.

The second approach is to add a small amount of modeling power to the surrogates, mainly maintaining the last value for an indefinite period of time. This approach also introduces anomalous behaviors, where objects no longer respond (within the simulation) in any way to external stimuli, thus significantly weakening the credibility of the simulation.

The third approach is to ban unconditional divestiture. Without unconditional divestiture, the surrogates can stand back and watch the ownership transfer, but an attribute can never become unowned, eliminating the concerns in the other two solutions. Unfortunately, unconditional divestiture may be an important consideration in some configurations.

The most realistic solution probably combines all three approaches, limiting the problem by limiting unconditional divestitures and supporting long lived final attributes where viable.

4.5 Synchronize

The synchronize protocol exhibits three issues: consensus, selective addressing and failure. The consensus problem is essentially identical to the one examined in the save protocol above; we will focus our attention for this protocol on the selective addressing and failure issues.

The protocol

The interesting portion of the protocol is given in Figure 4.4.

In the protocol, some modeling federate, f_0, requests a synchronization point by invoking the *Register Synchronization Point* service, including a tag argument that identifies the synchronization point. This service request may optionally also include a set of federates designators indicating what federates are expected to participate in the synchronization. If no set is indicated, all federates in the federation are assumed to be involved. The RTI responds with the *Confirm Synchronization Point Registration* † service indicating whether the synchronization point registration was valid. Currently, a synchronization point registration from a valid federate is invalid only if the tag is already in use.

If the synchronization point is valid, the RTI will request that each federate in the set synchronize with the *Announce Synchronization Point* † service. After a federate has performed the work required for synchronization, it responds to the RTI with the

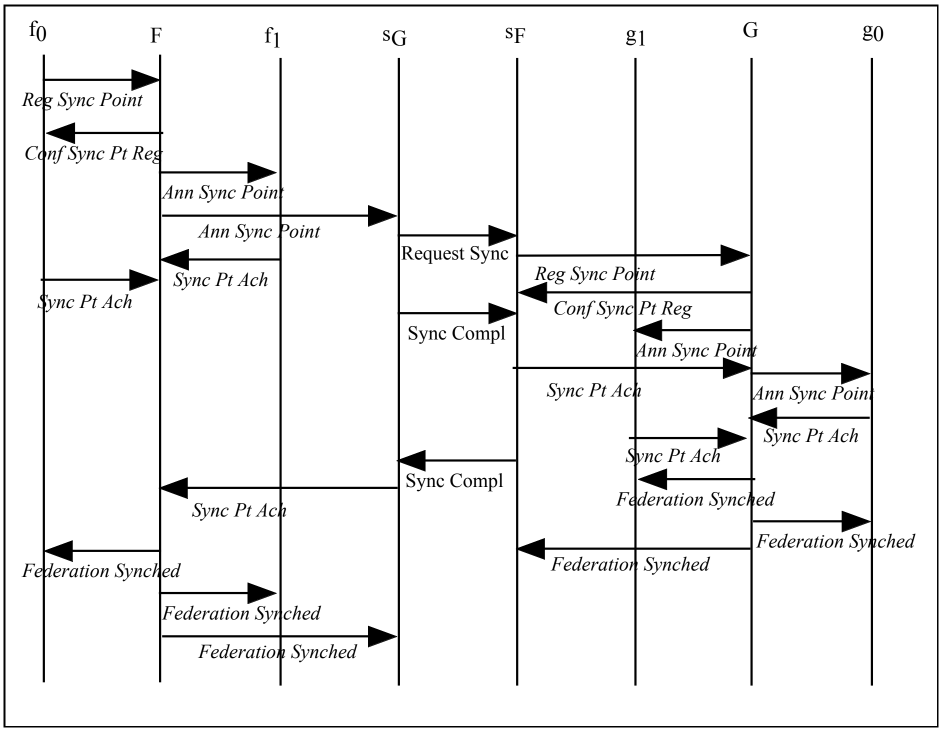

Figure 4.4: Example synchronization protocol

Synchronization Point Achieved service. When all federates in the set have responded, the RTI announces the synchronization with the *Federation Synchronized* † service.

Bridging the protocol

The bridge must propagate two pieces of information: a request for a synchronization point and an announcement of synchronization being achieved. These messages exactly parallel the two messages sent across the bridge in the save protocol.

Problems

Two problems can arise in the bridged scenario: the need to provide a set of federate designators and the synchronization request being invalidated by the second RTI.

The set of federate designators is itself problematic in two ways. In the current implementation, there is no mechanism for a federate to discover the complete set of federate designators involved in the synchronization. Furthermore, the RTI has no mechanism for designating federates in another federation.

If the second RTI (G in the example) disallows the synchronization point registration, the surrogate (s_G in this case) has no mechanism to report this failure to the original RTI (F in the example).

Solutions

We discuss two simple solutions for the designator problem in Section 3.2. The simplest solution simply disallows selective addressing across the bridge. In this scenario, the surrogate federate s_G is considered to be a full surrogate for all the federates in federation G (except the surrogate s_F). As such, inclusion of s_G implies inclusion of all federates in G except the surrogate s_F. Similarly, excluding s_G from the set of synchronizing federates implies the exclusion of all federates in G.

If selective addressing across the bridge is required, the bridge and all the federates involved in the synchronization need to be made more sophisticated. In this approach, the synchronization tag encodes the set of federates to be involved.

4.6 Initial publish

The initial publish protocol occurs when a bridge first registers the surrogates in the two federations. During this protocol, the bridge acquires the knowledge that it needs to represent the attributes modeled by the two federations.

The protocol

When a federate first joins the federation, it announces what attributes it will publish (invoking a *Publish Object Class Attributes* service for each object class supported across the bridge) and what attributes it will subscribe to (using a series of *Subscribe Object Class Attributes* service invocations). The RTI will inform the federate that it should start registering instances of classes of interest to other federates using the *Start Registration For Object Class* † service. The RTI will also invoke the *Discover Object Instance* † service to inform the federate of any interesting objects registered by other federates. At this point, the new federate may also begin announcing its own instances by invoking the *Register Object Instance* service.

Bridging the protocol

This protocol can be followed in a straightforward manner by the bridge. Each surrogate publishes and subscribes to all known (and allowable) attributes. As a surrogate becomes aware of an instance, it transfers that knowledge across the bridge and the complementary surrogate registers the instance in the second federation.

Problems

The problem here is inefficiency — the bridge may publish many attributes for which there is no interest. The interface specification provides no interface to discover which attributes may be of interest to another federate.

Solutions

Two solutions are possible: ignore the problem and use the MOM. Being an efficiency issue, ignoring the problem may be sufficient for many configurations. The alternative for more performance sensitive configurations is to add MOM access to the surrogates. These accesses could determine which attributes have an existing interest. By combining information the two surrogates can gain from the two MOMs, the bridge can limit its traffic to attributes that are published on one side and subscribed on the other.

Chapter 5

Conclusion

In this report we have presented the results of a comprehensive examination of the consequences of using a bridge federate as a the architectural glue for composing multiple federations. First we presented a general framework for understanding the nature of the problems that arise when two or more federations are bridged. Within this framework we then identified six kinds of problems, illustrating each with an example from the current HLA. Next we provided a framework for understanding possible solution strategies. For each problem class, we described how each solution strategy might be employed. Finally, we examined a number of key protocol sequences to show how these problems can arise in the more complex setting of a running HLA, and how the solution strategies can help.

The results indicate that while bridge federates are technically feasible, to use them correctly and efficiently will likely require certain changes to the IFSPEC, careful engineering of the bridges, and an understanding of the inherent limitations that are imposed by a bridge federate approach.

Acknowledgements

The research leading to this report was carried out under funding from the Defense Modeling and Simulation Office (DMSO). We would like to thank Reed Little and James Ivers for their help in understanding the HLA and bridge federate issues.

Bibliography

[Com99] Simulation Interoperability Standards Committee. *IEEE P1516.1/D4 Draft Standard for Modeling and Simulation, High Level Architecture — Federate Interface Specification*. IEEE Computer Society, 1999.

[KWDJ99] F. Kuhl, R. Weatherly, J. Dahmann, and A. Jones. *Creating Computer Simulation Systems: An Introduction to the High Level Architecture*. Prentice Hall, October 1999.

[SBBH98] David T. Shen, Christina Bouwens, Wesley Braudaway, and Daphne Hurrell. *Bridge Federate System Requirements Based on High Level Architecture Interface Specification Version 1.3 Draft 9*, July 1998. Document HLASEA-A001, Science Applications International Corporation.