Sixth IEEE International Workshop on Distributed Simulation and Real Time Applications (DS-RT '02)
Fort Worth, Texas, USA, October 11-13, 2002.

Bridging the HLA: Problems and Solutions

Juergen Dingel
School of Computing
Queen's University
Kingston, Ontario, Canada
dingel@cs.queensu.ca

David Garlan
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA
garlan@cs.cmu.edu

Craig Damon
Computer Science Department
University of Vermont
Burlington, VT
cdamon@cs.uvm.edu

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 paper presents the results of an in-depth study of the feasibility of an HLA bridge in the context of the current HLA interface specification. Problems and solutions are discussed and illustrated using particular HLA services.

1. Introduction

The High Level Architecture (HLA) was designed as a component integration standard for cooperating, distributed simulations [13]. Specifically, it defines a simulation *interface specification* (IFSPEC) and a *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 [4].

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 check pointing 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 the Defense Modeling and Simulation Office (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 [2]. 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 to enforce security restrictions.

The use of a bridge federate is architecturally attractive for two reasons. A bridge looks like any other federate, allowing multiple federations to be joined transparently. Bridge should, in principle at least, allow the HLA specifi-

[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.

cation itself to remain unchanged: a bridge should be able to update each side using existing services and event subscriptions.

While attractive in principle, 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 synchronized? 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 our study, we provide answers to these questions by presenting the main results of an in-depth study of the feasibility of an HLA bridge in the context of the current IF-SPEC [7]. In our study we have 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 only point wise. 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 interaction problems which do not appear when a single service is examined in isolation.

2. Related Work

Three general areas of design work relate to this paper: the HLA, publish-subscribe systems, and distributed systems.

The HLA itself has generated considerable attention from the practitioner community that uses it (e.g., [12, 10, 11]). The concept of an HLA bridge appears to be mentioned first in [2]. An initial investigation into time management over bridges was conducted in [14]. As Section 3 shows, the realization of a bridge requires a federate to be part of two federations simulanenously and to "represent" each of the federations to the other. In [17], this idea is developed further to allow for the hierarchical composition of federates. That paper also contains a taxonomy of systems in which a single federate participates in multiple federations.

Many researchers have investigated publish-subscribe (pub-sub) systems directly, either from an engineering perspective (e.g., [18, 19, 3]), or from a foundational perspective (e.g., [9, 1, 8, 21]). Most of these treatments have been concerned with the problems of building or reasoning about single pub-sub federations. In contrast, our research looks at composing *multiple* federations. One exception is the C2 system [20], which has an architecture consisting of multiple pub-sub connectors, arranged hierarchically. In that work, pub-sub connectors may be joined together directly or via a component, thereby providing the potential for lightweight pub-sub bridging mechanisms similar to the one we investigate here. However, C2 research has not focused as much on algorithms and protocols for maintaining global forms of consistency, or on the impact of a bridge combinator on such algorithms.

In the area of distributed systems, researchers have investigated the problems and solutions associated with coordinating distributed processes. For instance, the problem of achieving consensus about various system properties, such as membership, time, and topology is addressed (e.g., [16, 5]). While this research is relevant for identifying the kind of scenarios that can lead to inconsistency or potential deadlock, it does not directly address the problem of bridging collections of loosely-coupled components with lightweight mechanisms such as the HLA bridge.

3. The HLA and the HLA bridge

The High Level Architecture (HLA) for distributed simulation defines a framework for the integration of cooperating, distributed simulations, possibly built by many vendors [13]. Initially proposed by the Defense Modeling and Simulation Office, it is now an IEEE Standard and widely used in practice [4].

The HLA defines a distributed simulation as a collection (called a *federation*) of semi-independent simulations (each called a *federate*) that communicate using the services provided by a run-time communications infrastructure (called an *RTI*). Simulation events are communicated using a pub-sub model: new values of simulated entities announced by one federate will be received by the federates who subscribe to those events. Events are characterized in terms of updates to attributes of objects that are defined by a shared object model (called the *Federation Object Model*, or *FOM*). In addition, the RTI provides a large number of services to handle other mechanisms for coordinating simulations. These services are grouped into five different categories and include support for starting and stopping a federation, synchronizing federates, and saving and restoring federation state (Federation Management Services), declaring potential ownership of or interest in certain objects (Declaration Management Services), updating and reflecting object attributes, creating and deleting objects (Object Management

Services), transferring object ownership from one federate to another (Ownership Management Services), and global clock management (Time Management Services).

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. Such a composite federation allows two or more independently-developed federations to work together, without requiring significant modification to the individual federations or to their RTIs. Additionally, suitable glue mechanisms could act as a filter, transforming or hiding certain events between federations.

One proposed solution to realize such a composition is to use a bridge federate to link two federations [2, 14]. The bridge would appear as an ordinary federate to each federation, ideally requiring no changes to the RTI. The bridge might implement various filtering and transformation policies on events, but it should encapsulate those policies, making them easy to change. Moreover, the intention was for the bridge to be a relatively simple component: it would be inappropriate for a bridge to maintain an amount of state similar that maintained by the RTI. To achieve this transparency, seamlessness and simplicity, the bridge was conceptualized as consisting of three logical parts, shown in Figure 1. As illustrated, bridge B joins the two federations F and G. The bridge B itself consists of three parts:

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. Intuitively, s_G represents all federations to the "right" of the bridge.

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. TM may possibly carry out additional transformations, including serving as a security guard.

Using several bridges, federations theoretically could be linked to form linear, hierarchical, or graph-like topologies. However, earlier work has already identified serious problems with circular structures [6]. For instance, if bridges connect federations in a circular fashion and no special provisions are taken, the invocation of a service may give rise to an infinite number of invocations of the same service with each new invocation overriding the old one.

From a semantic point of view a bridge should have two key properties:

1. It should respect RTI semantics: The behavior of a bridged system should be the same as the behavior of a corresponding unbridged system with all of the federates linked by a single RTI, modulo naming and filtering issues. Consider, for instance, federate f_0 in the bridged system in Figure 2. The bridge B_1 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. In other words, to each of the federates f in federation F, the existence of the bridge should be noticeable to f only in so far it makes the attributes of the federates g_0, g_1, and h_0, available to f. Note that according to the current standard [4], a federate cannot determine the number or identity of federates that participate in its federation. Figure 3 attempts to illustrate the effect of the bridges in Figure 2 from the federates' perspective.

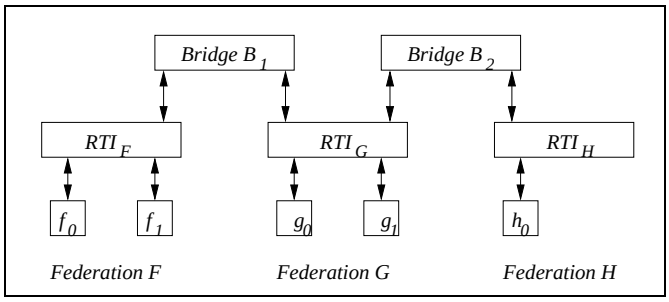

Figure 2. Three federations linked by two bridges

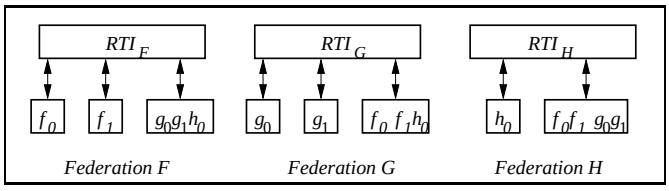

Figure 3. The effect of the bridges in Figure 2 from the federates' perspective

2. It should respect federate semantics: the behavior of a bridge should be identical to any other federate, that

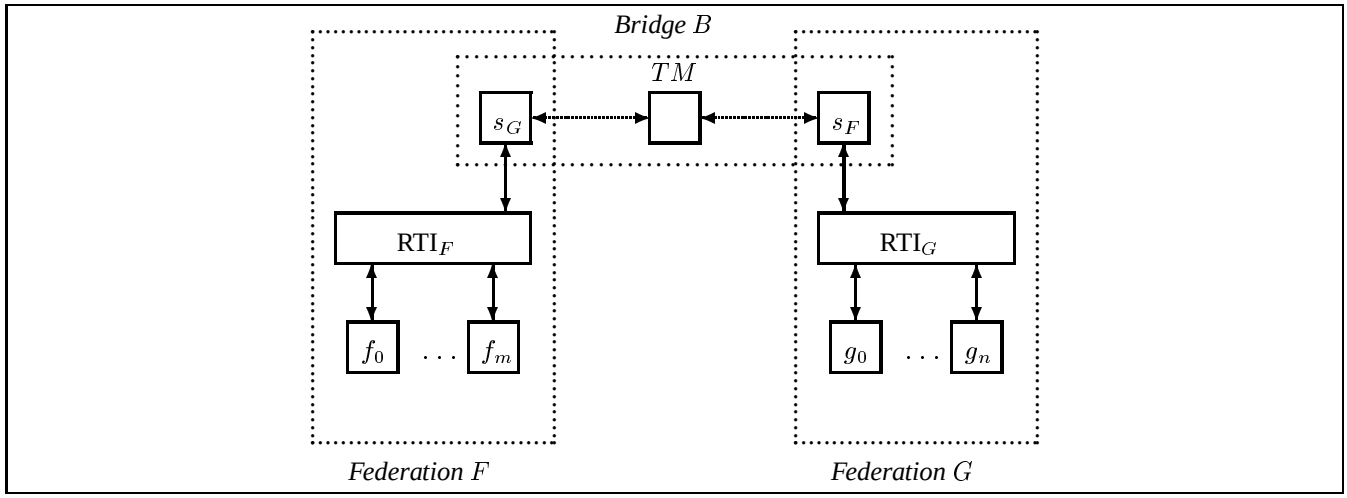

Figure 1. Federations F and G connected by a bridge B

is, its behavior should be a subset of the behavior that a normal federate might exhibit. This implies, for instance, that the bridge surrogates cannot exhibit behaviour that a normal federate could not exhibit.

However, as noted earlier, it is not apparent that such a bridge can be built to satisfy these properties without significant modification to the existing HLA standard given in [4]. In the remainder of this paper, we describe the results of our investigation of problems that arise when using a bridge. For the purposes of our analysis we consider only the case of linear topologies of federations.

In carrying out this work, our approach was to use the HLA standard specification [4] (and to some extent a previously formalized model [6]) to look for anomalous situations. We then categorized those situations in the form of more general problem classes. In this way, we are able to identify problem areas that both capture a large number of parts of the HLA and extend beyond the specific protocols of the HLA. Having identified problem classes, we then attempt to classify possible solution paths, and to understand the mapping between problems and solutions. This paper is based on [7] which gives a more detailed account of our work.

The next section presents a specific problem experienced in bridging the HLA save protocol. Discussion of this problem presents our first problem category. Section 5 presents the other problem categories while Section 6 discusses the solution categories.

4. An Example Problem

We now provide an example to illustrate the kind of problems that arise when trying to extend an HLA service to the bridged case.

4.1. Federation Save

In the HLA, any federate may make a request to the RTI that all federates in the federation checkpoint their state. Such checkpoints can be used to recover from federation failures, by restoring a federation to a previously saved state.

In the unbridged case, invocation of federation save causes the following protocol to be executed:

1. To request a save, a federate invokes the *Request Federation Save* service on the RTI.

2. The RTI responds by sending the *Initiate Federate Save* †[2] message to every federate in the federation.

3. As soon as an individual federate has completed saving its own state, it invokes the *Federate Save Completed* service.

4. Once each federate has informed the RTI that its save has completed, the RTI announces the completion of the save by sending the *Federation Saved* † message to every federate.

Figure 4 shows the message sequence chart corresponding to the above protocol for a federation F containing three federates f_0, f_1, and f_2.

[2] In the HLA, events announced by the RTI are decorated with a dagger †.

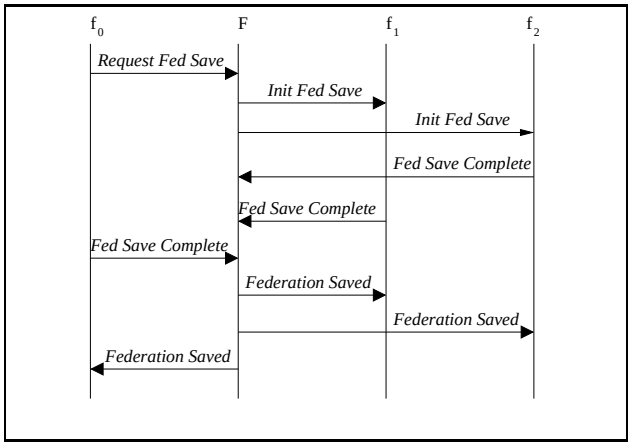

Figure 4. Example of the save protocol for unbridged case

4.2. Bridging the Protocol

To extend the above protocol to the bridged case, the bridge must ensure that a surrogate

- requests a save on behalf of the federation that it represents. That is, whenever a save request was issued in the federation that it represents, the surrogate must issue a save request in the federation to which it is connected to.
- reflects the successful completion of the request. That is, whenever the federation that it represents has saved successfully, the surrogate must announce the completion in the federation to which it is connected to.

In other words, the bridge must propagate two kinds of information: the initial request for the save and the completion of the save. We represent this communication as, respectively, *Request Save* and *Save Completed* messages sent across the bridge.

Assuming that the bridge simply transmits initiation and completion of saves, we obtain the following protocol:

1. To request a save, a federate invokes the *Request Federation Save* service on the RTI.
2. The RTI responds by sending the *Initiate Federate Save* † message to every federate and surrogate in the federation.
3. When a surrogate receives the notification to initiate a save from the RTI, it sends a *Request Save* message to the surrogate at the other end of the bridge.
4. When a surrogate receives a *Request Save*, it invokes the *Request Federation Save* service on its RTI.
5. As soon as a federate has completed saving its own state, it invokes the *Federate Save Completed* service.
6. When all federates that a surrogate represents have completed the save, the surrogate issues a *Federate Save Completed*.
7. Once each federate has informed the RTI that its save has completed, the RTI announces the completion of the save by sending the *Federation Saved* † message to every federate.

Figure 5 shows the message sequence chart corresponding to the above protocol for a bridge that connects two federations F (containing f_0, f_1, and s_G) and G (containing g_0, g_1, and s_F).

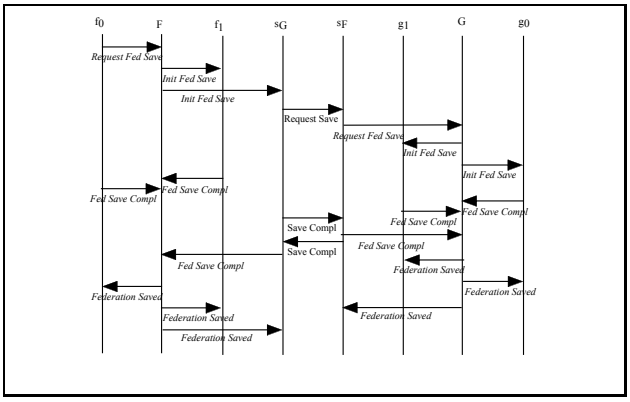

Figure 5. Example of the save protocol for bridged case

4.3. Deadlock!

On the surface, the protocol above seems like a natural extension to the bridged case: the bridge simply triggers saves on the other side, and notifies the initiating RTI when it is saved. This behavior has the desirable properties indicated in Section 3 of making surrogate behavior indistinguishable from ordinary federates, and of keeping the bridge simple. It also requires no new additions to HLA protocol, as the only new activity is encapsulated within the bridge itself (i.e., between its two surrogates).

Unfortunately, this protocol deadlocks. Consider Figure 1 and suppose federates f_0 to f_m and g_0 to g_n have issued *Federate Save Completed*. RTI F cannot signal a *Federation Saved* †, because it is waiting on s_G. s_G is waiting on s_F, which in turn is waiting on G. RTI G, however, is waiting on F.

One might handle the situation in a number of different ways. One is to break the circularity by allowing each surrogate to send *Save Completed* as soon as *all federates in its*

federation except itself have saved. However, according to the HLA standard federates do not have the capability to obtain this information. We thus must give the surrogates the additional capability to determine when all federates in its federation except itself have saved successfully. Adding this capability requires changing the set of services and events of the HLA itself — a change we would prefer not to make.

4.4. The Consensus Problem Category

Although couched in terms of a particular protocol for federation save, the problem exposed by the above analysis is not unique to that protocol. Within the HLA, this consensus problem also occurs with the protocols for synchronization and time advance. As with save, synchronization and time advancement require consensus between federates. Therefore, the naive adaptations of these two protocols to the bridged case suffer from the same problem as the save protocol described above and will therefore deadlock. As with save, enriching the capability of the surrogate breaks the circularity, but stands in conflict with the HLA standard.

In order to capture the general case, we can generalize the problem by considering it as one that is triggered whenever a federation of pub-sub components must reach consensus. For the HLA, if surrogates are restricted to federate capabilities, the consensus decisions of each surrogate of a bridge are mutually dependent. Circularity is broken by giving the surrogate the capability to make the consensus decision by considering the state of the federates in its federation alone. But this change requires making changes to the protocols for the non-bridged case.

5. Other Problem Categories

Consensus is not the only category of problems introduced by bridging federations. Through our investigation of the HLA, we have have identified five other problem categories, which we detail in this section.

5.1. Service Barriers

The consensus problem induces a second category of problems that may not, at first, be obvious. The problem arises from the fact that one federation must always announce consensus first. Federates within that federation may choose to perform actions only allowable, by protocol or by policy, after consensus has been achieved. The bridge may forward these actions to a second federation that has yet to announce consensus. A critical race thus ensues: if that action is propagated before consensus is announced, a disallowed action will be taken.

To illustrate the problem in the HLA, we consider the synchronization service and its underlying protocol. Apart from the consensus problem, we run into a second problem when trying to bridge the synchronization protocol. Suppose, for instance, that a federate requests a synchronization by registering a synchronization point with its RTI. In an unbridged federation, 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 been informed of the synchronization point by the RTI, f may assume that it will not receive another announcement of a different synchronization point until this request has been serviced.

In the bridged case, however, it is unreasonable to assume that all federates notify their RTIs of successful synchronization at exactly the same time. Consequently, 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 completed in other federations. Thus, federate f may (unexpectedly) receive another synchronization request while a previous one is still pending. We refer to problems of this kind as *service barrier* problems.

5.2. Unexpected Failure

Another source of problems is associated with the handling of failures that arise in the course of a service protocol.

For example, in the case of synchronization outlined above, the RTI may reject the registration of a synchronization point by a federate. If, for instance, the name chosen by the federate to identify the synchronization point is already used, the RTI will refuse registration.

The possibility of failure introduces yet another problem into the synchronization protocol. Consider the bridged federation in Figure 2. After a federate in federation F has registered a synchronization point, the surrogate s_F representing F must also register the synchronization point in federation G. The RTI for federation G may reject the synchronization point. In this situation, the surrogate s_G has no means to report the problem to the original RTI. More precisely, the HLA standard [4] is devoid of any mechanism that would allow s_G to communicate the registration failure to its RTI, since it assumes that all such failures would originate from the RTI itself.

In general, this problem is caused by the following mismatch: On the one hand, the behaviour of a surrogate is restricted to that of a federate. But on the other hand, a surrogate must also represent an entire federation adequately. If the entire federation can fail to complete an action that the HLA standard expects each individual federate to complete successfully, the standard will not offer the surrogate any mechanism to report the failure. We refer to this situation as the *unexpected failure* problem.

5.3. Selective Addressing

In early versions of the HLA [4], the federate registering a synchronization point was allowed to indicate a set of federates. If this parameter was supplied, only those federates indicated were informed of the synchronization point and only they were required to respond to the request. This mechanism removed the need for all federates to know how to respond to every synchronization.

Another class of problems arises from this kind of selective addressing, leading to conflicts with the desired properties of a bridge expressed in Section 3. According to the first property of that section, the bridge is supposed to hide the number and identities of the federates on the other side of a bridge. With the identity of the federates on the other side of a bridge hidden, a federate cannot address any of those federates individually. Indeed, the only options are for a federate to include none or all of the federates on the other side of the bridge. The first option is achieved by excluding the surrogate (of the federates on the other side of the bridge) from the synchronization set. If, on the other hand, the surrogate is included, all of the represented federates will be notified of the synchronization request and must be able to respond appropriately. Hence a bridged federation cannot have the full capabilities of a single federation.

Any HLA service that allows selective addressing exhibits the problem described above in the presence of a bridge. We refer to this problem as the *selective addressing* problem.

5.4. State/Behavior Problems

The HLA, like other pub-sub systems, makes assumptions about the behavior of the federates based on shared state. One instance of shared state relates to ownership of attributes.[3] Within the HLA, a particular attribute for a particular object being modeled can be owned by a single federate, or be unowned altogether. Ownership by more than one federate is never allowed; this restriction is crucial to the proper functioning of the HLA. While any number of federates may *subscribe* to that attribute (meaning they will be notified of attribute update events), the owning federate is the only federate that may *publish* updates for that attribute.

An owning federate is free to divest ownership in that attribute at almost any time. The HLA includes two divestiture protocols: conditional and unconditional. With conditional divestiture, the federate retains ownership until another federate is willing to claim ownership. A federate executing unconditional divestiture loses ownership even if no other federate is willing to take ownership, thus introducing an unowned attribute.

In a bridged environment, a surrogate acts as the owner of all attributes modeled by any federate in the federation it represents. The bridge could choose two possible policies for handling ownership transfer: allowing or disallowing ownership to transfer across the bridge.

Either policy introduces difficulties during or after the divestiture protocol. If the bridge allows transfer, the surrogate must conditionally divest its ownership when the true owner conditionally divests. This divestiture allows a federate on the other side of the bridge to claim ownership. If two federates attempt to claim ownership, a critical race condition arises. Using standard HLA protocols, if both federates are on the same side of the bridge, the HLA protocols allow the RTI to arbitrate the winner. However, when federates on different sides of the bridge request ownership, no single entity can arbitrate the race condition. Consequently it is possible that two federates can each believe they own the same attribute, violating a critical state invariant.

If the alternative policy is implemented (that is, ownership is never transferred across the bridge), the surrogate will never divest ownership of an attribute. If the true owner unconditionally divests, the attribute becomes unowned. But the surrogate still claims to be the owner and will be expected to generate new values for the attribute when so requested by the RTI. Thus, again, we have a serious inconsistency.

These problems are introduced because the actual state of the surrogate reflects an intermediate state not considered in the original protocol. As a result, the surrogate must declare itself to be in some closely-related state. For any incorrect state, the system may make assumptions about legal behaviors that conflict with the actual state of surrogate. We use the name *state/behavior* problems to describe situations such as these.

5.5. Unavailable Information

Although federates and surrogates serve largely different purposes, they should have the same interface: that is, their external behaviors should be identical. As we saw earlier, the unexpected failure problem violates this property. There is also a conflict when the federate interface specification cannot provide a surrogate with all the information needed to perform some bridge-specific task.

More precisely, the state of a surrogate must reflect the state of all federates in the federation being represented. In addition, the surrogate must also sometimes reflect the state of the RTI in that federation. Because no federate has a need for that information in a traditional, unbridged federation, no services are provided by the HLA to provide the surrogate access to that information. We refer to a problem

[3] In the case of the HLA, shared state is expressed as a shared simulation object model, the FOM described earlier. In other pub-sub systems shared state might be represented by things like shared files in a file system, or the contents of documents pointed to by URLs.

as an *unavailable information* problem when no service is defined that can supply the information required by a surrogate.

For an instance of the unavailable information problem, suppose that a surrogate s_G of a bridge (s_G, TM, s_F) joins a federation F (Figure 2). The bridge needs the relevant information of F so that surrogate s_F can adequately reflect it. This information includes, for instance, the current ownership state of the attributes defined in the FOM. However, there is no service to supply the surrogate with that information.

6. Solution Categories

For problems in each of these categories, solutions, or at least partial solutions, do exist. As with the problems, we divide the solutions into general categories. For most problem categories, many, if not all, of the solution categories are meaningful. Each solution offers different advantages and disadvantages. By providing the designer with an array of approaches to solving the problems we have outlined, we hope that the designer can find a solution that fits the needs of the specific problem in the context of a specific system.

6.1. Add Services

An obvious solution to many of the problems is to add additional capabilities or services to the infrastructure.

In the case of an unavailable information problem or an unexpected failure problem it is relatively clear how additional services can help solve the problem. Selective addressing problems can be addressed with a new capability, a set whose exact membership is determined by the federation within which it is considered. Federates could register to belong to certain named sets or could be selected by some form of query over RTI-understood properties of them.

As indicated in Section 4.4, consensus problems can be resolved by a service that will notify a federate when it becomes the only non-consenting federate for some issue. As long as no cycles are allowed in the topology of the bridges, this service would resolve all consensus deadlock problems. In the context of the save protocol discussed in Section 4.4, for instance, the RTI could issue a message *Only One Not Yet Saved* † to inform a federate when it has become the only federate that has not yet reported a successful save. A receiving surrogate could then safely send a *Save Complete* across the bridge to the representing surrogate. In [7], the resulting protocol is shown to work for binary bridges that connect federations in a linear, non-cyclic fashion.

If the RTI cannot tell surrogates from federates, the message *Only One Not Yet Saved* † could potentially be sent to any federate in the federation. If, however, the identity of the surrogates is disclosed to the RTI, only surrogates would need to receive the message.

6.2. Smart Bridge

Adding new services and capabilities is not without cost, however. Increasing the set of messages that federates must understand is expensive, both for upgrading existing federate implementations and for future development of new federate implementations. This expense may lead to significant (and understandable) opposition from the user community to such changes in the protocols. In the context of the save protocol, for instance, federates may have to be told to ignore *Only One Not Yet Saved* † messages.

Placing additional burden on the bridge can alleviate the cost of bridging to the federates and the RTI. Many problems vanish with a sufficiently clever bridge implementation. For the unowned attribute case of the state/behavior problem category, the bridge could simply remember the last value for every attribute on every object. The surrogate can therefore provide the expected modeling behavior assumed by its ownership of the attribute.

As a second example of bridge cleverness, consider solving the service barrier problem with the bridge. If the bridge encodes the legal sequences of actions, it can buffer illegal activities until after the required consensus has been announced. To solve problems such as the consensus problem, the bridge also needs the ability to query the state of the RTI to check on the status of all other federates. Once the surrogate realizes that all other federates have achieved consensus, the bridge can indicate that the other surrogate should announce its consent.

In combination with other new, but simple, mechanisms, an arbitrarily complicated bridge can replicate the entire behavior of an RTI, allowing it to solve almost any problem that might arise in a bridged environment. One of our initial goals, however, was to build a simple bridge. If the bridge takes on the complexity of a full RTI, many advantages of the bridged approach vanish.

6.3. Restrict Usage

The designer of a specific HLA application may be able to resolve some of these problems by imposing a policy on how the system uses the infrastructure provided. For example, the use of selective addressing could be limited to sets of federates within one federation.

Not all problems are amenable to this approach, however. For example, no policy will prevent the critical race potentially allowing an illegal action to take place before consensus has been achieved.

6.4. Ignore Problem

The final approach to solving these problems is the simplest to implement: simply ignore the problem. For some service barrier problems, the risk of the critical race being won by the illegal action may be negligible. For other service barrier problems, the outcome of an "illegal" action may not be a severe problem. For unexpected failure problems, the chance of failure or significance of the failure may not be worth the cost of extending the model to support the failure. For a state/behavior problem such as the unowned attribute problem addressed earlier, the cost of never responding to a request may be minimal.

Of course, ignoring the problem should not be done lightly. For some problems, ignoring the problem could have disastrous effects. For example, if the result introduces two owners of a given attribute or a deadlock in seeking consensus, ignoring the problem could invalidate the on-going simulation.

7. Conclusion and Future Work

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 have identified six problem categories and illustrated them by discussing possible realizations of particular HLA services in the presence of a bridge. Next, we have categorized possible solution strategies. For each problem class, we have described how each solution strategy might be employed.

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.

In our work to date, we have not focused on timing issues that may arise from the introduction of bridges. We have restricted our consideration to new critical race issues that may invalidate the protocols. Beyond the obvious performance issues, the timing changes introduced by bridges may change the order that federates observe events and thereby introduce otherwise unknown semantic conditions. While these conditions will be legal in an absolute sense, they may expose flaws in the federates that would not be exposed in an unbridged system. Additional investigations in the timing issues may uncover other types of problems.

Of course, using a bridge is not the only way of combining multiple pub-sub federations. One might, for example join the RTIs directly using some form of RTI-to-RTI communication link. That approach has the advantage that RTI state could be communicated directly without going through the component (federate) interface. But it also has the disadvantage that it requires considerable cooperation at the RTI implementation level. This, in turn, would make it hard to integrate federations that use RTI implementations created by different vendors. It also would make it harder to localize policies of filtering and transformation. Nonetheless, investigating such lower-level bridging mechanisms would be an interesting direction for future work.

Other avenues of future work include the use of formal models and analysis techniques to demonstrate the existence of the problems discussed more concretely, to validate our proposed solutions, and to look for other problem categories. We are currently working on a formal model of the HLA bridge using Magee and Kramer's Finite State Processes (FSP) notation and their Labeled Transition System Analyzer (LTSA) [15].

In this paper, we focused on the case in which bridges connect federations in a linear, acyclic fashion. Other topologies would also be worth exploring. We have done some work on investigating problems that arise when one allows cyclic topologies and identified some problems that do not arise in the simple case [6]. A more complete characterization of topologies and problem classes would be a valuable extension of our work.

Finally, investigating the use of bridges for other architectural styles would be worth pursuing. For example, one might investigate lightweight compositional mechanisms for architectures based on asynchronous (point-to-point) messages, or shared data approaches.

8. Acknowledgments

This research was sponsored in part by the Defense Modeling Simulation Office (DMSO), by the Department of Defense (DOD), the Army Research Office (ARO) under Contract No. DAAD19-01-1-0485, by the Defense Advanced Research Projects Agency (DARPA) under Contracts No. F30602-00-2-0616 and N66001-99-2-8918, by the Air Force Research Laboratory under Contract No. F30602-00-C-0000, and by the National Sciences and Engineering Research Council of Canada (NSERC) under Contract No. 228103-00. 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, DOD, ARO, DARPA, the United States Air Force, or NSERC.

References

[1] R. Allen, D. Garlan, and J. Ivers. Formal modeling and analysis of the HLA component integration standard. In *Sixth International Symposium on the Foundation of Software Engineering (FSE 6)*, pages 209–221, Lake Buena Vista, Florida, November 1998. ACM Press.

[2] W. Braudaway and R. Little. The High Level Architecture's bridge federate. In *Simulation Interoperability Workshop*, September 1997.

[3] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf. Design and evaluation of a wide-area event notification service. *ACM Transactions on Computer Systems*, 19(3):332–383, Aug. 2001.

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

[5] G. Coulouris, J. Dollimore, and T. Kindberg. *Distributed Systems: Concepts and Designs (Third Edition)*. Addison Wesley, Pearson Education, 2001.

[6] C. Damon, R. Melton, R. Allen, E. Bigelow, J. Ivers, and D. Garlan. Formalizing a specification for analysis: The HLA ownership properties. Technical Report CMU-CS-98-149, Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 1998.

[7] J. Dingel, D. Garlan, and C. Damon. A feasibility study of the HLA bridge. Technical Report CMU-CS-01-103, Department of Computer Science, Carnegie Mellon University, March 2001.

[8] J. Dingel, D. Garlan, S. Jha, and D. Notkin. Reasoning about implicit invocation. In *Sixth International Symposium on the Foundation of Software Engineering (FSE 6)*, pages 209–221, Lake Buena Vista, Florida, November 1998. ACM Press.

[9] J. Dingel, D. Garlan, S. Jha, and D. Notkin. Towards a formal treatment of implicit invocation using rely/guarantee reasoning. *Formal Aspects of Computing*, 10:193–213, 1998.

[10] IEEE. *Proceedings of Winter Simulation Conferences (WSC'95 - WSC'01)*. IEEE Press, 1995-2001.

[11] IEEE. *Proceedings of IEEE International Workshop on Distributed Simulation and Real Time Applications (DS-RT'97 – DS-RT'01)*. IEEE Press, 1997-2001.

[12] IEEE. *Proceedings of Simulation Interoperability Workshop (SIW'97 - SIW'02)*. IEEE Press, 1997-2002.

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

[14] T. Lake. Time management over inter-federation bridges. In *Simulation Interoperability Workshop*, September 1998.

[15] J. Magee and J. Kramer. *Concurrency: state models & Java programs*. Wiley & Sons, 1999.

[16] S. Mullender. *Distributed Systems (Second Edition)*. Addison Wesley, 1993.

[17] M. Myjak, R. Carter, D. Wood, and M. Petty. A taxonomy of multiple federation executions. In *20th Interservice/Industry Training Systems and Education Conference*, pages 179–189, November 1998.

[18] S. Reiss. Connecting tools using message passing in the FIELD program development environment. *IEEE Software*, July 1990.

[19] B. Segall and D. Arnold. Elvin has left the building: A publish/subscribe notification service with quenching. In *Proceedings AUUG97*, September 1997.

[20] R. Taylor, N. Medvidovic, K. Anderson, E. W. Jr., J. Robbins, K. Nies, P. Oreizy, and D. Dubrow. A component- and message-based architectural style for GUI software. *IEEE Transactions on Software Engineering*, June 1996.

[21] C. Wang, A. Carzaniga, D. Evans, and A. Wolf. Security issues and requirements for internet-scale publish-subscribe systems. In *Hawaii International Conference on System Sciences*, January 2002.