# Extending Portico HLA to Federations of Federations with Transport Layer Security

Thomas Roth and Martin Burns
Smart Grid and Cyber-Physical Systems Program Office
National Institute of Standards and Technology
Gaithersburg, MD
thomas.roth@nist.gov, martin.burns@nist.gov

Tim Pokorny
Calytrix Technologies
Perth, Western Australia
tim.pokorny@calytrix.com

Keywords:
cyber-physical systems, internet of things, high level architecture, hierarchical federations

***Abstract*—The Internet of Things (IoT) promises to connect and manage an unprecedented number of heterogeneous devices that are not designed to interoperate as a system. Modeling and Simulation of IoT will face new challenges in the representation of this scale because an accurate model must consider the characteristics of each unique IoT device in the system. In addition, hardware-in-the-loop simulation of IoT will produce massive quantities of information that can overwhelm the processing power of the low-power devices often deployed in these systems. This paper introduces a new infrastructural component called a Forwarder to the Portico implementation of the High-Level Architecture (HLA). It transforms the flat structure of HLA into a hierarchical structure where federates are partitioned into different clusters that communicate through their respective Forwarders. A Forwarder acts as both a data router between the different clusters in the federation, and a firewall that limits the amount of information that traverses the boundaries of its local cluster. Using this approach, Portico now has a scalable architecture suited to IoT that simulates a federation of hierarchical federations. A novel aspect of the Forwarder design is that the scalability of HLA traffic flow is amplified by the inherent function of traditional Internet Switch fabric technology. This technology resulted from a collaboration between the United States National Institute of Standards and Technology and Portico's core development team at Calytrix Technologies. The new Portico 2.2.0 will be used in the Universal CPS Environment for Federation (UCEF) and in other projects worldwide.**

## 1 Introduction

As the internet of things (IoT) becomes more prominent, better methods are needed to assure the trustworthiness of IoT applications that involve interactions between a myriad of distributed devices. One such powerful method is co-simulation, or the integration of multiple simulators into a joint simulation. Hardwired testbeds are always needed for ground truth. However, especially regarding networking effects and behavior of large systems, real equipment testbeds fail to provide deterministic and reproducible results. This is where simulation becomes extremely valuable as a first step before live trials.

The US National Institute of Standards and Technology (NIST) operates a cyber-physical systems (CPS) testbed to study behavior of CPS. As part of its repertoire of capabilities, a robust co-simulation environment has been developed in collaboration with colleagues at Vanderbilt University and using open-source software developed by Calytrix. This co-simulation environment has been mounted in a virtual machine and released as the Universal CPS Environment for Federation (UCEF) [1]. Through UCEF, the NIST CPS testbed can join various simulators, emulators, and hardware-in-the-loop across multiple platforms that span large geographic distances.

At the core of UCEF is the standard IEEE 1516-2010 High Level Architecture (HLA) that defines the services involved in co-simulation of distributed processes [2][3]. The HLA defines the simulators that participate in a co-simulation as *federates*, and the collection of federates that comprise the joint simulation as a *federation*. The federates communicate and coordinate using software called the *runtime infrastructure* (RTI) that implements the common set of services described in the HLA federate interface specification. UCEF utilizes the Portico RTI.

The deployment of large scale IoT simulations presents some subtle challenges for services build around the HLA model. Although the HLA provides some elegant conceptual constructs for dealing with these problems, the realization of these services is difficult to scale within the type of platforms used in IoT applications. Core to this problem is that the conceptual HLA model natively assumes a flat, equal performance network environment. For the initial development and local testing of models this presents no problems. However, as the size of experiments grows, and geographical separation is brought into play, the deployment topology starts to shift and becomes more clustered than flat as pockets of high performance environments are tethered to either low-bandwidth links or resource constrained environments.

This paper discusses some of the scalability issues associated with HLA federations that involve devices and networks with different performance characteristics potentially distributed across large geographical distances. Part of this discussion includes the security concerns that arise when federations involve multiple organizations who want to protect their data and federate designs. The discussion is framed in the context of the solutions implemented in the latest 2.2.0 release of the Portico RTI to address these scalability issues for UCEF. Section 2 provides an overview of common approaches to HLA scalability. Section 3 discusses the performance concerns related to large scale IoT federations, and Section 4 discusses the changes made for the release of Portico 2.2.0 to address these concerns. Section 5 discusses the security concerns related to federations that include federates from multiple organizations, and Section 6 concludes the work.

## 2 Federation Communities

When the original draft of the High-Level Architecture was working its way towards becoming an IEEE standard in the late 1990s, there were significant efforts within the community related to the reuse and interoperability of federations. This is different from the discussion of whether an individual federate built for a specific RTI could be reused in a separate federation implemented using the same RTI. It's easy to imagine how a model in one federation might be brought into another federation, although this might require an adapter to translate between minor differences in the two federations' object models. The concern at the time was whether a federation whose aggregate behavior represented, say, an individual house could be composed into a super-federation that could simulate an entire power grid. A primary motivator for this discussion was that the HLA as a standard defines nothing more than an *application programming interface* (API); it does not provide a reference implementation and provides no guidance on appropriate architectures for an RTI implementation. And so, in the 1990s, there was a worry that federations built using one RTI would never be able to communicate with federations built using a second RTI. This led to this idea of a super-federation; a federation of federations that might bridge multiple RTI implementations.

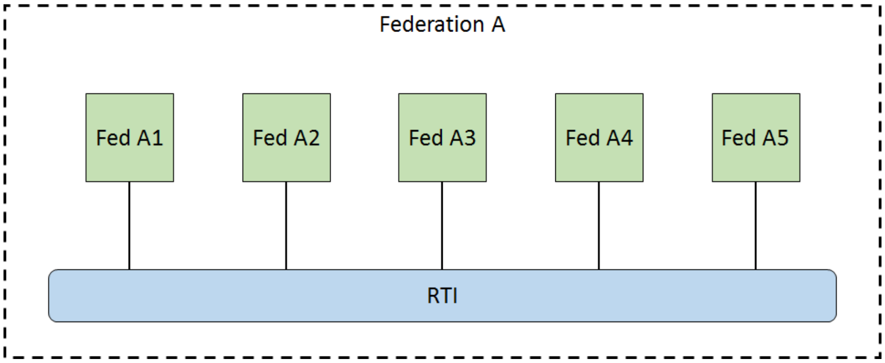

**Figure 1 A flat federation in the high-level architecture**

Figure 1 depicts a normal HLA federation. There is a single RTI that serves as the mechanism for communication and coordination between multiple federates. The architecture is flat, and the activity of federate A1 will be seen by all the remaining 4 federates without distinction. It is hard to imagine how to incorporate multiple interacting RTIs in this architecture without a very complicated implementation.

The RTI Interoperability Study Group proposed a new term called a *federation community* that is a combination of federations and RTIs working together to achieve a common goal [4]. Alternative terms for a federation community are a *hierarchical federation*, or a *federation of federations*. In a federation community, multiple flat federations, like the one shown in Figure 1, are bridged together to reuse the functionality of an entire federation in a larger system. This achieves the goal of having a federation that models the behavior of a single house, and being able to reuse that federation as if it were a federate in a higher-level federation for a power grid.

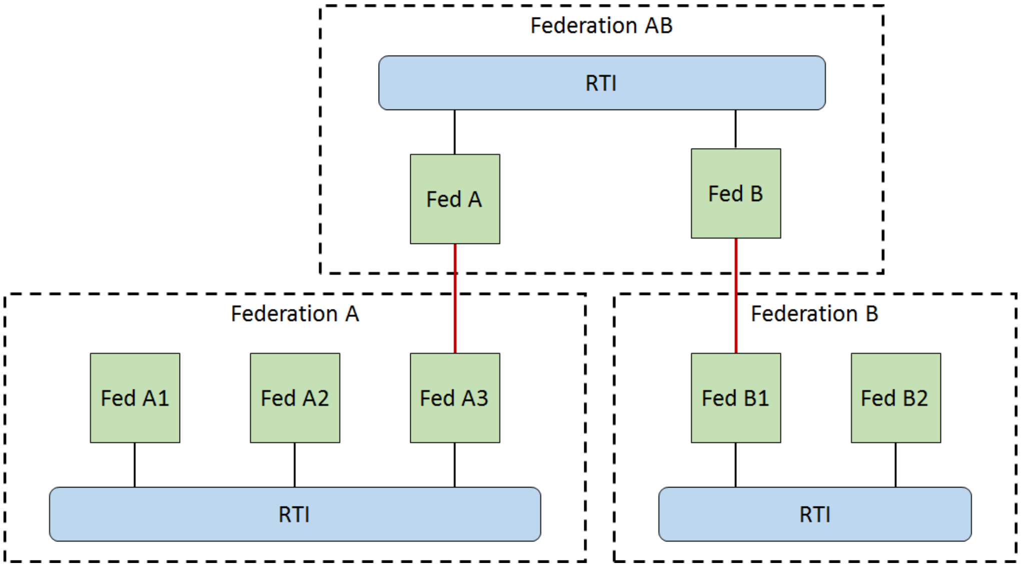

**Figure 2: A federation community that combines two separate federations into a hierarchy**

There are several possible architectures that can be used to implement a federation community that differ in whether the bridge between federations is a responsibility of the RTI or a designated federate [5]. Figure 2 shows an example of one of these architectures. Federate A3 serves as a proxy for its Federation A, while Federate B1 serves as the proxy for Federation B. There is an indirect communication channel between Federation A and Federation B that uses the proxies in the higher-level Federation AB.

Suppose, in this figure, Federate A1 needs to send an interaction to Federate B2. When Federate A1 broadcasts its interaction to Federation A, Federate A3 will receive the message as a member of the same federation. Because it is the designated proxy from Federation A, Federate A3 will use some mechanism to relay the interaction to Federate A using an out-of-bounds communication channel. The easiest mechanism would be that Federate A3 and Federate A are the same process joined to two separate federation executions. Federate A can then send a duplicate interaction in Federation AB, which Federate B will relay using out-of-bounds communication to Federate B1. And at last, the interaction can be sent from Federate B1 to the final destination at Federate B2.

A federation community has powerful applications. Let's suppose you have a group of federates that simulate the internals of a military application. These federates exchange top secret information that is required for the group to perform its intended tasks. You now want to take this military application, and reuse it in a public scenario that involves one or more civilian actors. None of the top-secret information should flow from the military federates into the civilian federates, but the top-secret information must be sent at runtime for the military federates to achieve their stated purpose. The solution to this problem is to use a federation community where one federation consists of all the federates with top secret clearance, while the other federation contains the rest of the federation. This is a known application of federation communities called the information hiding problem [6].

Another powerful application of federation communities is as a mechanism to effectively navigate corporate firewalls. Suppose there are two organizations that want to perform a joint simulation. However, each organization wants to run its federates on its own internal network to maintain control of its own data. Some mechanism is needed to bridge the two isolated clusters of federates running on separate networks. The MAK RTI introduced a Forwarder component to bridge networks for this purpose [7]. Their implementation can also bridge to other RTI implementations.

In the context of IoT applications, the main benefit of federation communities is likely to be performance. Most IoT devices are low performance to minimize the cost of each device. When these devices participate in a co-simulation, they must be shielded from as much network traffic as possible to reduce their amount of computation and prevent them from becoming bottlenecks. A concept like a federation community can be used to cluster the federates into high-performance and low-performance groups, and minimize the amount of network traffic that crosses the groups using a technique like information hiding. The remainder of this paper presents an implementation of this idea in the Portico RTI.

# 3 The Problem of Scale

This section introduces some of the problems that result from a flat federation model, and then look at how a more clustered approach can help reduce the scope of those issues, as well as the costs to standards compliance of doing so.

## 3.1 The IoT Environment

The deployment of large scale IoT simulations presents two primary problems. Firstly, federations can have a large number of federates. The IoT devices or simulations of such devices can range into the thousands very quickly. When considering a smart grid example, even simulating the impact of a single device per household at the scale of a suburb could introduce more than 15 000 federates to a federation. Secondly, the computational power available on physical IoT devices is far less than those in traditional environments. Although capable, they do impose serious limitations around CPU, memory and network connectivity. When combined, these two traits present a challenging large scale, low power environment.

To further complicate the matter, environments such as UCEF aim to create a live testbed where participants can interact either across the country or globe. Simulations that bring together models and devices from a range of government and commercial sources allow for trial and assessment of different approaches to a shared problem. Where the technology involved may be commercially sensitive, or where groups simply cannot be co-located for various other non-technical reasons, an interconnected environment that spans remote sites is needed. This results in a large-scale simulation environment consisting of potentially low powered and resource constrained devices spread across several different networks with varying levels of bandwidth between them.

## 3.2 Practical Problems When Scaling

One of the most challenging aspects of large-scale deployments is the way they affect network communications. In our smart grid example, an individual federate or device focused on a sub-system of a single house exchanges high-bandwidth data with federates representing other sub-systems in the house, but it does not have any use for the information generated in a parallel federate for its neighbor’s dwelling. However, the parts of the object model structure that it is interested in overlap with all other instances that fulfil the same purpose.

The flat nature of standard HLA publication and subscription services mean that once a federate is subscribed to a set of attributes, it hears about changes to those for every registered object instance in a federation. For our small IoT device, responsible for a sub-system within a single logical unit of a house, this requires that it contend with traffic generated by every other house within the simulation.

From an HLA standards perspective, Data Distribution Management (DDM) is cited as the solution to this problem. It allows a federate to publish and subscribe within an arbitrarily defined logical sub-space, which in this example could be set up to represent each house. Only federates subscribed within an overlapping space would receive the updates, effectively cutting them out. At a logical level, this works. However, at a practical level, it is difficult to scale to the level motivating this paper.

For DDM to work, it requires that filtering data be calculated for each federate. The location at which that filtering takes place has large ramifications on overall communications model for the federation that can impact performance and feasibility of the federation.

As it relates to network communication, RTI implementations typically fall into one of two molds [8]. They are either centralized, running all data back through a single aggregation point (the RTI itself) where the filtering can take place; or they are distributed, exchanging messages in a peer-to-peer fashion and performing local filtering at each federate to ensure that only relevant information is delivered to the client code.

Centralized implementations can push this computational load to the RTI, which from a resource capacity perspective can be better managed. However, the routing of messages through a central location presents two further problems. Firstly, message multiplication can be substantial. For each update sent by one federate, there will be *n* additional messages generated by the RTI as it reflects the message to each of the *n* subscribers. DDM will reduce the impact of multiplication, but the RTI must still perform calculation for every subscriber, and this must be performed on every reflection sent. This increases latency between message send time and its receipt by all interested subscribers.

Secondly, additional latency is introduced between co-located federates as they are now communicating through the RTI as a third party. When there are two federates existing on the same network segment, but they are connected via a slower long-haul connection to the central RTI, the round-trip path for a message must cross that boundary twice, despite the close location of the two devices.

Distributed implementations, such as the Portico RTI, use group communications like multicast to allow for efficient use of the network. Every message is sent just once, with the network hardware managing duplication and ensuring it is delivered to all registered participants. Filtering is performed at the receiver side to ensure that client code only receives updates for attributes it has subscribed to, and within the appropriate DDM scope. Multicast typically only works on a local network, so a point-to-point long-haul connection is still required to connect remote sites, however the multiplexing of a single gateway for multiple federates can offset some of the losses and it does not impact local neighbors.

This approach supports high-performance communications because it both delegates delivery to the networking hardware and effectively parallelizes the filtering computational work. However, it is also effectively nullifying the advantages of DDM as local processing must still be performed on all messages, even if they are not passed through to client code. For resource constrained devices this can further impose an undue burden on each receiver that is beyond its computational power profile.

As an API standard, the HLA naturally separates itself from implementation issues. However, this separation can lead to difficulties realizing the conceptual framework within the limitations of certain usage scenarios. The target domain for this work will contain large numbers of federate clusters within which there will potentially be high-volume of messaging, but outside of which the data exchange is much more reserved.

The key tension between this deployment structure and the HLA standard revolves around the inability to adequately specify locality as a distribution constraint. The DDM part of the HLA standard requires that any federate, no matter its location, be able to subscribe to and receive data within any space. If it remains standards compliant, an RTI implementation must provide ways to route data to anywhere on the deployment graph, irrespective of the performance impacts of that.

Regardless of the conceptual elegance of DDM, when it is mapped into the deployment domain at the scale of federations this work is interested in supporting, the practical burdens can quickly become overwhelming. The next section looks at ways to minimize that burden while also discussing the standards compliance trade-offs that come with it. Additionally, the changes made to the Portico RTI to support operation at small and large scale are described.

# 4 Clustered Federations in Portico

UCEF presents some broad goals in terms of its ability to act as both a development and test environment at the low level while also being able to support experimentation at an extremely large scale. Distributed experiments will stretch typical performance patterns, while the need to manage low-power devices on the same fundamental architecture presents challenges.

A flexible solution that reduces the long-haul and latency issues for co-located federates within a local cluster and is also able to scale to large numbers of federates is needed. Taking this into consideration, there are three main objectives:

1. Support scaling to large numbers of federates
2. Support efficient communications between local clusters of co-located federates
3. Support the shielding or throttling of data for low-power devices

To address these problems, some fundamental changes to the way the Portico RTI operates and communicates have been made.

## 4.1 Scaling Federation Start

The first major architectural change made to Portico was the re-introduction of a central RTI process. As a fully distributed RTI, previous implementations had no central RTI process. Rather, the support library for every federate would track the status of every other federate by snooping on the messages it emitted. This allowed it to make unilateral decisions based on its knowledge of the full federation state.

As the size of federations grow, this approach presents a memory consumption problem for IoT devices. More importantly however it triggers instability in the start-up process. Upon joining a federation, every existing federate must send the new arrival information about its current state. For moderate federation sizes this presents no concern. For large federations, the volume of data this generates and the time it takes to process renders it unviable.

The only true way to manage coordinated activity over large volumes of federates is through a central arbiter. As such, an RTI Server component was reintroduced to the Portico architecture. This lifts the management burden from federates, allowing them to remain light weight. It also reduces the message exchange paradigm

### 4.2 Separating Data and Control Messaging

The HLA specification defines more than 100 service calls that can be made to an RTI. They can be broken into two types: those that are intended for consumption by the RTI ("control" messages), and those that are intended for consumption by other federates ("data" messages). The latter group are the data-exchange calls; Update Attributes and Send Interaction.

Although they represent only a small fraction of the available RTI calls, when profiling a federate it quickly becomes apparent that data messages are both the most frequently sent and the most bandwidth intensive. Intuitively this makes sense, as data exchange is the key purpose of the HLA. When optimizing, anything relating to these calls is of keen interest.

Given that these two types of messages both have different targets and different performance profiles, the second change to the Portico implementation was to split their processing paths.

To support this, the message wire format was updated to include a header that signals what type of message is contained (Data or Control), and if it is a data message, what the object or interaction class handle contained is. With this information data messages are readily identified and they can be processed accordingly.

### 4.3 Forwarders and Filters

The final key change made to the Portico RTI as part of this work was the introduction of a transparent component able to both connect clusters of federates together over distance and able it to act as a filtering device to help manage the traffic flows of extremely large federations.

A wide-area network (WAN) bridge is a common component for RTI implementations [8]. While Portico has always had a WAN bridge that could connect clusters of federates over a TCP/Unicast connection, this new "Forwarder" component provides several key differences:

1. Communications between federates and the Forwarder, and between the Forwarder and the RTI can use any communication mechanism available to Portico (multicast, unicast or shared memory)

2. The Forwarder can perform explicit filtering on ingress and egress traffic for data messages

3. The Forwarder can connect through other Forwarders, allowing hierarchical structures to be built

Much like Portico's original WAN bridge, the Forwarder is capable of supporting WAN links. However, it is important not to think of the Forwarder purely in the context of site-to-site connections. It may also be used to partition logical clusters of federates on a single network.

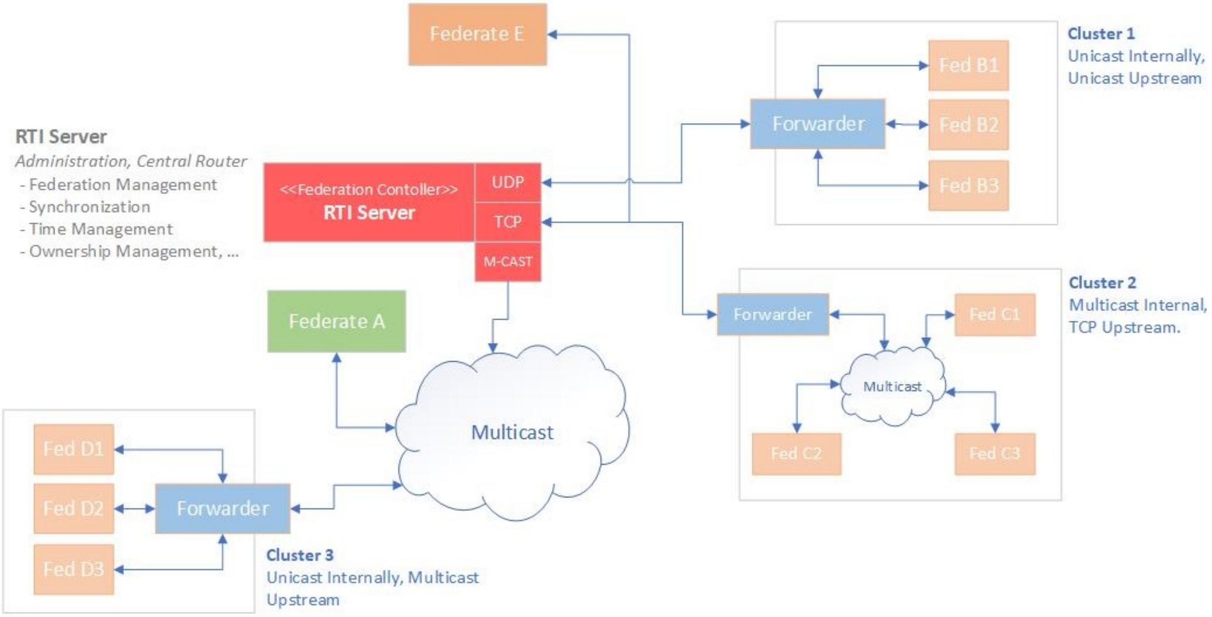

**Figure 3 Topology of Forwarder**

The use of DDM as a partitioning mechanism in the types of environments UCEF wants to support make it difficult and inefficient to implement. In its place the Forwarder adds explicit ingress and egress filtering. In its configuration file, lists of object and interaction classes that should be allowed through these filters can be defined. Egress filters effectively stop data exiting a local cluster, while ingress filters can be used to stop unnecessary traffic entering the cluster.

Using information contained in the packet header, the Forwarder identifies which messages should be allowed to pass, and which should be dropped. It will also snoop on all calls to create and join a federation, and uses the information within them to get access to the federation object model (FOM) structure that allow it to convert between class names specified in the filters and the class handles used as identifiers in the Portico RTI.

Control messages are not subject to this filtering as they *must* go to the RTI. Data messages retain the old Portico trait of “receiver side filtering”. This allows multicast/group communications to be used natively where possible for efficiency, however the filtering can substantially reduce the flow of traffic to federates, increasing the signal-to-noise ratio and reducing the burden on computationally limited IoT devices.

To more aggressively constrain the data fed to a limited device, a Forwarder can be used in front of it as a firewall, allowing only information it is known to have an interest in both into the cluster-of-one and out of it.

Finally, the Forwarder itself has no true notion of a Federate or an RTI. It is configured in terms of an upstream and downstream connection. This can be exploited to allow hierarchical clusters to be built up through a network of forwarders by connecting the upstream pipe of one forwarder to the downstream side of another.

# 5 Authentication and Authorization

Developers of IoT devices, systems, and systems of systems have proprietary interests. Yet, they are often willing to share access to their developments to obtain benefits to their design process from integration testing with others. However, in co-simulation environments, such interface sharing is made more difficult by the typical boundary-less interfaces of many collaboration technologies. To facilitate the exposure of both proprietary and public designs for experimentation, some form of information hiding is required.

Several common mechanisms to limit undesirable information flow are:

- *Sender-side filtering* – in a federation community, an organization might own all the federates connected to one of the sub-federations that participates in the community. This organization might, rightfully so, want to own all the data produced by their federates and limit which information is available and to which parties. The sender, rather than the receiver, should determine what information is allowed onto the network in order to maintain this data ownership. An RTI that implements sender-side filtering allows federates to handle information hiding.

- *Message encryption* – by encrypting all messages, data exchanged between sub-federations in a community over wide area networks (WAN) are hidden from prying eyes. Only federates authorized to know the decryption key can decode the message contents.
- *Message authorization* – encryption alone only protects a message channel. It does not say anything about the rights to this data and its use by the recipient. Authorization allows the recipient to maintain a list of permissions that it recognizes from the token offered by the sender asserting the rights that accompany the data.
- *Sender Authentication* – Public Key Infrastructure (PKI) can be used to establish authentication of the federates in a federation. Based on the acceptance of certificates, a session key can be provided by the RTI that can be used in encrypting and decrypting messages on a per-federate basis assuming the RTI is trusted. To make routing decisions across a WAN, the messages must utilize an unencrypted header that contains opaque federate IDs and message identifications. The header must also contain an access-token that can be used by the recipient to enforce authorization permissions on what can be done with the data once encrypted.

The present version of Portico has been extended to support authentication and authorization using PKI, a routing header, and an embedded access-token. It takes advantage of the presence of the RTI Server as a trusted component in a key management scheme to facilitate confidentiality in messaging.

Figure 4 shows the basic mechanism devised. Specifically, when a federate joins a federation, it performs a key exchange with the RTI Server. Typically, this is a certificate based authentication that allows the RTI Server to verify the identity of the sender. Through this registration process, a session key is created to be used with confidential communications with the federate. The RTI Server caches these keys and makes them available to those federates authorized to interact with the federate. Although the authorization management and configuration is not specifically part of the current development, the unencrypted message header provides an access token that represents the authorization.

Implementations of the RTI Server can constrain the use of these tokens and key caches to provide from lightweight to austere cyber-security implementations for messaging. The software implementation emulates the Transport Layer Security (TLS) [9] standard used throughout conventional web services on the Internet today, but tailored to the messaging in HLA. Subsequent papers will describe scenarios where this capability is used and relied upon.

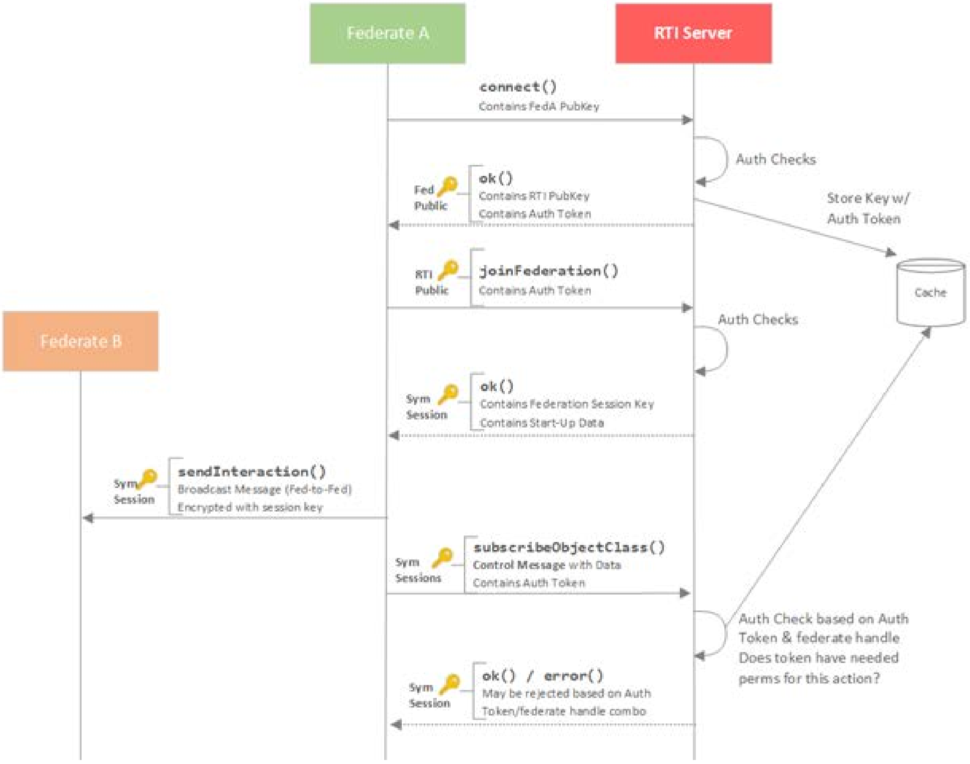

Figure 4 Encryption and Key Management for Forwarder

## 6 Conclusion

This paper discussed several issues related to large-scale deployment of HLA federations for IoT applications. It introduces a new Forwarder component implemented in the next release of the Portico RTI that addresses some of these issues. The Forwarder gives large UCEF federation participants the required structure and deployment tools they need to build networking topologies that make the best use of network resources they can. However, this approach does differ from the concept of a federation community in that the filtering and information hiding capabilities of the Forwarder are performed within a single federation. Rather than a federation of federations, the Forwarder approach constructs clusters within a single federation that can define public versus private information exchange across the cluster boundaries. This does impose a key configuration constraint on the overall federation to ensure that a federate will hear about all the data it was meant to.

In return for this trade-off benefits are gained. The implementation of a component that acts more like traditional networking tools such as routers and firewalls provides more flexibility in deployment while also keeping the architecture of the RTI simple. From the perspective of RTIs and federates, there is no difference between a flat federation and a hierarchical one that contains multiple clusters, allowing the implementation to remain free from such complexities.

The memory burden on small devices is lowered; filtering adds support for further traffic reduction to reduce computational load. Clusters of federates can run side-by-side without any undue impact on one another even if their subscription interests overlap. Throughout the federation the reflection rate and burden on the infrastructure to support it is reduced to levels close to the minimum of what they can be.

Finally, the insertion of a homogeneous confidentiality and authorization mechanism was achieved that will allow fine-grained protection of messaging and hiding of proprietary information.

The Forwarder and the other mentioned changes to the RTI architecture will be part of the next major release of the Portico RTI [10].

# 7 Acknowledgment

Official contribution of the National Institute of Standards and Technology; not subject to copyright in the United States. Certain commercial products are identified in order to adequately specify the procedure; this does not imply endorsement or recommendation by NIST, nor does it imply that such products are necessarily the best available for the purpose.

# 8 References

[1] T. Roth, E. Song, M. Burns, H. Neema, W. Emfinger and J. Sztipanovits, "Cyber-physical system development environment for energy applications," in ASME 11th International Conference on Energy Sustainability, Charlotte, NC, 2017. doi 10.1115/ES2017-3589

[2] IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)-- Framework and Rules, IEEE Std. 1516-2010, Aug. 18 2010. doi: 10.1109/IEEESTD.2010.5553440

[3] IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)-- Federate Interface Specification, IEEE Std. 1516.1-2010, Aug. 18 2010. doi: 10.1109/IEEESTD.2010.5557728

[4] M. Myjak, D. Clark and T. Lake, "RTI interoperability study group final report," in 1999 Fall Simulation Interoperability Workshop (SIW), 1999.

[5] M. Myjak and S. Sharp, "Implementations of hierarchical federations," in 1999 Fall Simulation Interoperability Workshop (SIW), 1999

[6] W. Cai, S. Turner and B. P. Gan, "Hierarchical federations: an architecture for information hiding," in Proceedings 15th Workshop on Parallel and Distributed Simulation, Lake Arrowhead, CA, 2001. doi: 10.1109/PADS.2001.924622

[7] L. Granowetter and B. Watrous, "Challenges and solutions for large-scale hla federations," Available at: https://www.edstechnologies.com/VR-Link.pdf [Accessed 28 June 2018]

[8] P. Ross, "Comparison of high level architecture run-time infrastructure wire protocols - part two," in 2014 Fall Simulation Interoperability Workshop (SIW), Orlando, FL, 2014.

[9] T. Dierks, IETF (2008) The Transport Layer Security (TLS) Protocol Version 1.2 (RFC5246), Available at: https://tools.ietf.org/html/rfc5246 [Accessed 3 July 2018]

[10] Portico Project. https://github.com/openlvc/portico [Accessed 29 June 2018].

# Author Biographies

**Thomas Roth** is working on development of the technology behind the cyber-physical testbed at the National Institute of Standards and Technology as a member of its Smart Grid and Cyber-Physical Systems program office. His research interests are in formal methods for the composition of cyber-physical systems, and the detection of compromised cyber-physical devices through comparison of their reported behavior against the constraints of the physical system.

**Tim Pokorny** is Calytrix's Chief Technology Officer and responsible for the oversight and management of our suite of COTS products. Tim has over 15 years' experience with distributed simulation technologies such asHLA and DIS. He is the co-creator of the Portico RTI and its lead developer. Tim holds a PhD in Distributed Simulation from the University of Ballarat (now Federation University) andhas worked as an active member of simulation industry groups both nationally and internationally viaparticipation with the Simulation Industry Association of Australasia (SIAA) and the SimulationInteroperability and Standards Organisation (SISO), previously as an appointed member on the SISO Standards Activity Committee (SAC).

**Martin Burns** is the Associate Director for Testbed Science in the Engineering Laboratory, Smart Grid and Cyber-Physical Systems Program Office at NIST. With his background in IEC and ANSI standards development for semantic models and data exchange, he has facilitated the development of the underlying Green Button technologies which define energy usage information and APIs in the Smart Grid in the US and internationally. He co-chairs the data interoperability working group for the NIST led Framework for Cyber-Physical Systems (CPS) and is a key contributor to NISTs architecture for UCEF – federated testbeds for investigating the behaviors of CPS/IoT.