

A SOFTWARE ARCHITECTURE FOR FEDERATING INFORMATION SPACES FOR COALITION OPERATIONS

Gail Mitchell, Joseph Loyall, Jonathan Webb, Matthew Gillen, Andrew Gronosky, Michael Atighetchi
BBN Technologies, Cambridge, MA 02138
{gmitchell, jloyall, jwebb, mgillen, agronosky, matighet}@bbn.com

and

Asher Sinclair
Air Force Research Laboratory, Rome, NY 13441
Asher.Sinclair@rl.af.mil

ABSTRACT

Modern warfare relies on dynamic, coalition operations supported by small, agile teams in time sensitive missions. While each member of a coalition may maintain a local information space supporting the activities of its own teams, coalition members must be able to share information to cooperate effectively in dynamic environments and succeed in their missions.

We present an extensible, layered architecture for federating individual information spaces into an interoperating information federation in which coalition partners can, as members of the federation, choose to share their information with other members of the federation. Local information spaces employ federate services to become federated information spaces (i.e., federates) who can share information with other federates in a federation. Federation services provide the capabilities needed to dynamically form and manage a federation of information spaces. Both federate and federation services are designed to maintain the autonomy of local information spaces while still allowing secure and efficient collaboration between them.

We describe the services we are developing to demonstrate the architecture, present our current prototype federation, explain the analysis, design and implementation decisions made during the development of this prototype, and review an evaluation of the current implementation.

I. INTRODUCTION

Modern warfare relies on dynamic, coalition operations supported by small, agile teams in time sensitive missions. While each member of a coalition may maintain a local information space supporting the activities of its own teams, coalition members must be able to share information to cooperate effectively in dynamic environments and succeed in their missions.

The information space concept we are targeting with this architecture has grown out of the Joint Battlespace Infosphere (JBI) [1][2], a US Air Force initiative supporting network centric warfare concepts. JBI is related to other

network centric warfare initiatives, including Net-Centric Enterprise Systems (NCES) [3], a set of services enabling access to and use of the Global Information Grid (GIG) [11] in warfighting operations. An information space provides managed exchange of information between members of a Community of Interest (COI), including information brokering and dissemination by publish-subscribe-query middleware. In the information space model, *clients* are information publishers and consumers, communicating anonymously with other clients via a shared *information management system (IMS)* [2]. Information published into the information space is in the form of typed *managed information objects (MIOs)* consisting of payload and metadata describing the object and its payload. Consumers make requests for future (*subscription*) or past (*query*) information using predicates over MIO types and metadata values.

If all participants in a mission are clients of the same information space, they will be able to share information through that space's IMS. However, in a coalition the participants will each have their own information space and it is possible that information needed by one participant may exist in the information space of another. Even when information is available locally, other sources may be more suitable, current, or of higher quality, and therefore offer greater probability of mission success. This motivates the need for federating information spaces to provide the information critical to a mission no matter where it exists or

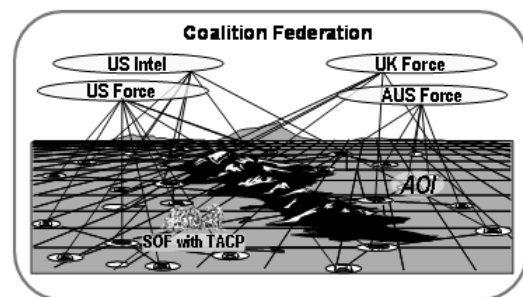


Figure 1. An example of federated information spaces. The SOF makes a request for imagery of the AOI, which can be serviced by other, coalition federates.

when it becomes available. This is illustrated in Figure 1, in which the special operations force (SOF) served by one information space (US Force) inquires about an area of interest (AOI) observed by clients in a different information space (AUS Force). When the information spaces are federated, the SOF makes a request of its local (US Force) information space and that request is fulfilled by the AUS Force information space.

In this paper we present an extensible, layered architecture for federating individual information spaces into an interoperating federation. Coalition partners can, as members of the federation, choose to share their information with other members of the federation. In Section II we discuss some of the requirements for federation, and in Section III we present our architecture for federation. In Section IV we describe a prototype federation that illustrates the architecture and in Section V we analyze some of the design decisions made in developing this prototype. In Section VI we discuss an evaluation of the current prototype. We conclude in Section VII with a discussion of our next steps in research and design.

II. BACKGROUND

Federation technologies must support information sharing between possibly heterogeneous information spaces while still enabling individual information spaces to maintain governance of their own information. Information space federation must also be flexible to enable many different patterns of interaction and to avoid motivating clients to bypass the IMS and create unmanaged, *ad hoc* communication paths.

The architecture and services aspects of our solution are driven by general requirements for federation in coalition environments. The most basic of these requirements are those that *enable information exchange between information spaces*. These include

- Creating a federation from existing information spaces, i.e., establishing information exchange between heterogeneous information spaces.
- Processing an information space's requests (i.e., subscriptions or queries) to determine how the information needs can be met.
- Propagating published information in response to requests and in anticipation of them.
- Discovering information and federated information spaces.

The technologies for establishing information exchange cannot subsume or assume to know military doctrine or command authority. Therefore, federation technologies must maintain at least the capabilities for control that are currently available in local information spaces, leaving the decisions of how much information to share, and how to

share it, to the policy makers and managers of each local information space. Specifically,

- Clients of an information space should be only as aware of federation as they want or need to be (transparency).
 - There should be support for *policies* provided by command authorities and for *information service agreements* that can be negotiated between federates to control their interaction.
 - Information spaces should control access to their own information, i.e., there should be services for federates that provide access control and information protection.
- Another set of requirements comes from the likelihood that two information spaces that join a federation, and the clients within them, might need additional processing capabilities to handle each other's information. This motivates requirements for federation capabilities such as:
- Providing efficient information exchange and sufficient information quality to meet changing conditions.
 - Managing differences in syntax and semantics between vocabularies used in different information spaces.
 - Capturing, maintaining, and propagating provenance metadata, such as origin and pedigree, or access policies.
 - Fusing and filtering information.

Finally, since it is impossible to anticipate all the services needed by current and future federations, a federation architecture must be extensible to support new services being produced, variations of service designs and implementations, and service updates.

III. THE ELSIF ARCHITECTURE

We are developing an Extensible, Layered Services Architecture for Information Space Federation (ELSIF) to provide the framework for managed information sharing in coalition environments. As illustrated in Figure 2, the major components of this architecture are the Federation and its Federates, the services provided by the federation, the services provided by the different federates, the federation-enabled Information Management Systems (FE-IMS) supporting each federate, and Federated Managed Information Objects (FMIOs) that move between federates in the federation.

A *Federation* is a system of cooperating, independent information spaces. To foster and support cooperation, the federation has rules and procedures for how federates can behave when participating in the federation, and provides services to its federates that guide and assist their behavior within the federation. The services shown at the top of Figure 2 are examples of what might be provided by a federation. Some of these services are required for all federations. For example, Lifecycle Services provide capabilities needed to form and manage the membership of federations, Data Discovery Services provide a means for federates to

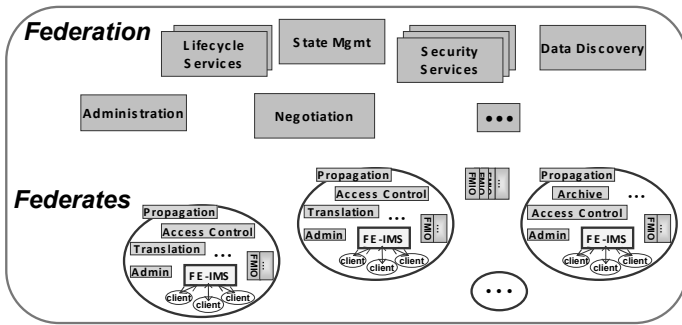


Figure 2. Extensible, Layered Services Architecture for Information Space Federation.

find possible locations for information within the federation, and State Management Services manage information about the federation’s membership, members, and active policies.

Other federation services provide valuable capabilities that may or may not be required by a particular federation. For example, Security Services could be provided to authorize the participation of individual federates, to authenticate the origin of information and requests, or to define and authorize specific interactions between federates.

Note that any of these services could be provided as centralized services or could be distributed among the federation members.

Each *Federate* is an information space with an Information Management System (IMS) that can communicate with federate services. Clients do not need to be aware that their information space is participating in the federation, although the space could choose to let them be aware. We would expect an administrative client, for example, to be aware of federation since it would be managing the configuration of the federated information space and thus the aspects of configuration that are federation-aware.

These information spaces, and their IMSs, are expected to be heterogeneous systems with varying capabilities. Federate services provide those capabilities needed by a federate (i.e., the federated information space) to participate in the federation. For example, each federate uses a Propagation Service to handle interactions between the federate and other components of the federation (such as other federates’ Propagation Services or specific federation services). A federate could include additional services for local or federation use. For example, a federate might have a Translation Service to allow it to send or receive information in a different format than that understood by its local IMS. The federate might also provide Translation as a service to the federation – i.e., it could provide through the federation a particular type of translation for other federates who do not have their own translation capability. Similarly, a federate could provide an Archiving Service that could be used internally to the federate or could be used by the federation for additional storage.

It is interesting to note that the services themselves can be layered and that the order in which they apply in a specific federate or in any particular interaction might be defined differently depending on the interaction and on the policies that are in place in the federation and in individual federates. For example, the publication of an MIO might be processed first by a federate’s Access Control Service to ensure that the federate permits the MIO to pass into the federation, then by the Propagation Service to interact with the federation’s Data Discovery Service to determine where to send the MIO, then by a federation Security Service to determine whether the federation access control policies allow such interaction between federates. Alternatively, the publication might first be processed by the Propagation Service to determine where it could be sent, then by the federate’s Access Control Service to make sure the federate agrees to send the MIO to those places. In Section V we discuss further the layering of services and analyses of service alternatives.

Federate and federation services are designed to maintain, at minimum, the capabilities and control currently available in a local information space. Therefore, they are designed to be transparent to existing clients, maintain the publish/subscribe/query paradigm of information exchange, and enable each information space to maintain controlled access to its own information.

A third component of the ELSIF architecture is the *Federation-Enabled Information Management System (FE-IMS)*. A federation-enabled IMS is aware that it is part of a federated information space and thus exposes an interface for interacting with its federate services. The FE-IMS interface helps maintain the autonomy of the information space. An IMS could implement this interface directly, or a compatibility layer could be added to an existing IMS to allow it to participate as an FE-IMS.

The final component of the architecture is the information that can be passed between federates – the *Federated Managed Information Objects (FMIOs)*. We want to maintain the integrity of local MIOs provided by each information space for federation use. In particular, an information space needs to be able to distinguish between its own information and MIOs it received from elsewhere in the federation. It may also want the federation to maintain provenance of the MIO, i.e., information about who has handled or changed the information (perhaps for security purposes). In ELSIF, local MIOs, such as described in Section I, are extended (or wrapped) with metadata used for federation purposes. This includes metadata such as a federate ID of the information source and a federation-assigned unique identifier for the object. This information allows the object to be managed within the federation. For example, the federation could prohibit movement of FMIOs that do not have authentic source IDs. The federated metadata does not, however,

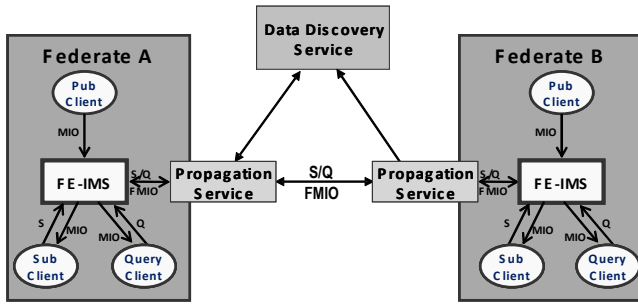


Figure 3. Basic interaction between any two federates in a federation.

modify the local MIO, so Translation services may be necessary for one information space to make use of an MIO from another space.

Our service-based approach to federating information systems is consistent with other service-based approaches including service-oriented architectures (SOA) and NCES. Unlike some architectures, we do not limit a federation to use a specific Enterprise-Service Bus (ESB) and service registration/discovery implementations. For example, NCES registration and discovery services could be used to register and discover our federation services. In addition, our approach allows information spaces to choose which federate services they wish to expose to federation. That is, a federate can broker information for clients of other information spaces without the other clients being able to discover and access the federate’s brokering, security, translation or other services. Likewise, our federate and federation services can be used to federate ESBs, enabling such systems as federations of NCES domains. Development and prototyping of these concepts is future work we plan to undertake.

IV. A PROTOTYPE FEDERATION

We illustrate the ELSIF architecture with a prototype federation with baseline capabilities for access-controlled data movement via subscription, query and publication between federates. In this baseline system, all federates are based on the same FE-IMS (see Section IV.B) and all federates understand the same set of types for their information objects.¹ In this way we examine the basic services developed to support federate interaction and the capabilities needed by an FE-IMS to support these services. We discuss security capabilities we added to the federation, and configuration options for services for lifecycle management of the federation.

¹ The problems of semantic interoperability [4] are beyond the scope of this prototype although, as discussed in Section III, ELSIF’s extensible, service-based architecture supports incorporating services that can mediate semantic differences in a federation.

A. Basic Federate Interaction

The basic requirements for publish/subscribe/query interaction in a federation are illustrated in Figure 3. This interaction involves a *Propagation Service* that can move requests and information provided by a federation’s *Data Discovery Service*. In the figure, if the Propagation Service for Federate A wants to publish an FMIO produced by its FE-IMS into the federation, it contacts the Data Discovery Service to determine which other federates might have clients that are interested in this FMIO, obtains the addresses of those federates, then interacts directly with the Propagation Service of each other federate to provide the FMIO. Conversely, if one federate’s Propagation Service receives an FMIO from another federate’s Propagation Service, it pushes the FMIO into its FE-IMS for local processing.

In the current prototype, the Data Discovery Service has no restrictions defined for access control or known types so will always route queries and publications to all other federates.² Indeed, in this implementation all access control resides with the local FE-IMS which determines whether to send a local client’s subscription, query or publication to its Propagation Service. The FE-IMS also determines (based on its own access control policies) whether to process queries from the federation and to which local clients to send publications from the federation.

B. Federation-Enabled IMS

Federation-enabling an IMS provides it with the ability to push subscriptions, queries and MIOs to its federate services, and to accept and process the same kinds of actions from its federate component on behalf of the federation. Our prototype federation currently has multiple instances of the same FE-IMS, each with its own set of clients. We use the Apollo 1.0 IMS [5] in this prototype. Apollo provides services that allow the registration of subscription predicates (specified using XQuery [6]), matching of metadata for published MIOs (specified using XML [7]), and delivering matched MIOs to clients using the Java Message Service [10]. Client-side distribution middleware exposes publication, subscription, and query interfaces conforming to the Joint Battlespace Infosphere *Common API (CAPI)* [8] using SOAP messages over HTTP or HTTPS.

We built each FE-IMS by wrapping Apollo with components that interact with federate services. Figure 4 shows the design of each prototype FE-IMS. Federation capabilities are provided using:

- *Proxy clients that use the CAPI [5] and are portable across IMSs.* These include a *proxy subscriber* client that subscribes to locally published MIOs and pushes

² Section V describes possible efficiency improvements if type and subscription information is available to the Discovery Service.

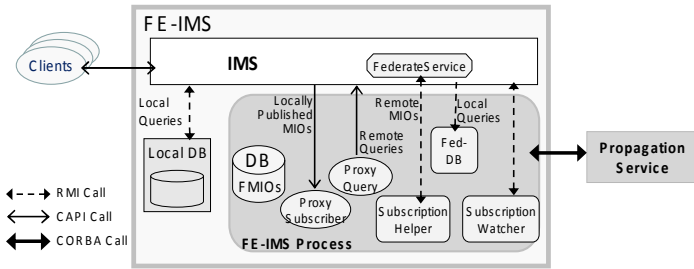


Figure 4. Design of the prototype FE-IMS

them to the Propagation Service for possible delivery to the federation, and a *proxy query* client that executes queries on the local IMS on behalf of remote federates.

- *Components that use IMS interfaces.* These include a *Subscription Helper* that delivers remotely published federated MIOs to local subscribers; a *Subscription Watcher* that keeps track of subscription status in the local IMS and pushes this information to the federation; and a *Federated DB* that transforms local queries into federated queries and processes the federated results.

The FE-IMS process also includes an embedded DB for caching FMIOs.

C. Federation Security

The ELSIF prototype implements services that address the following three aspects of security:

- **Authentication** – Federates establish trustworthy identities within a federation.
- **Authorization** – Resources are used by consumers who have been granted authority to use them.
- **Data protection** – Information integrity and confidentiality are maintained.

Specifically, we have prototyped the following application level security capabilities to work in the context of a single security domain³:

Authentication. We provide unique identifiers (stored as Strings) for each federate’s Propagation Service instance and for the Data Discovery Service (corresponding to a federation). Each of these “subjects” has a local keystore holding a private key and an X509 certificate signed by a Certificate Authority. All message exchanges between subjects are authenticated using X509 signatures. New federates entering the federation go through a certificate exchange protocol prior to any information exchange. We use crypto-strong algorithms and the certificate exchange protocol to prevent man-in-the-middle violations.

Authorization. We include a centralized XACML [6] policy decision point (PDP), deployed in the Discovery Service, and write XACML policies for inter-federate behavior based on federate identities, MIO types and/or request

³ Multiple security domains are beyond scope of the current project.

types. Each Propagation Service instance implements a policy enforcement point (PEP). Outgoing requests or MIOs cause the PEP to access the PDP to “request permission” to forward the request or MIO to a set of remote Propagation Service IDs. The PDP checks the XACML policy and provides an answer in the form of a modified set of allowed Propagation Service IDs. Since a receiving federate cannot be sure that the sending federate acted in accordance with the PDP’s decision, access control can optionally also be enforced by the receiver’s PEP. In this case, the receipt of a federated request or MIO triggers the PEP to ask the PDP for permission to receive the forwarded request or MIO. If permission is not granted, the receiving Propagation Service can drop it.

Data Protection. To ensure information integrity, we can attach PKI signatures of message hashes to each forwarded request or FMIO. The signatures also include cryptographic nonces to prevent replay attacks. The use of signatures is configured on a per-federate basis. The prototype does not yet include support for information confidentiality.

D. Federation Lifecycle

One of the keys to our federation approach is that a federation is a first class entity, with its own identity and life cycle. Accordingly, a federation defines an administrative domain for managing its configuration and membership, which collectively define the *state* of the federation. Federation state includes the following information, which can change throughout the federation’s lifecycle:

- Its membership list and information about each member.
- The available federation services and their configurations.
- Policies defined for the federation (e.g., for access control, information routing).

In addition, each federate defines its own administrative domain and has its own state. A federate’s configuration and policies determine what subscriptions, publications and queries can be federated and which archived MIOs can be exposed to the federation.

Federation configuration includes choices of how to handle federates that join, leave or re-join the federation. For example, one configuration choice is how to handle cached MIOs that include provenance information about a federate that leaves the federation. For one approach to this situation, consider that in a non-federated IMS a published MIO can persist in the IMS’s store long after its publishing client has left the information space. For example, a sensor such as an unmanned vehicle (UAV) can collect surveillance imagery and publish it in an IMS. After the UAV has completed its mission and left the information space, the information it collected remains useful and available to querying clients.

This approach does not necessarily carry over to the federated case because of differences in the *ownership* model for MIOs in a federation. In an IMS, a publishing client in essence transfers ownership of the MIO to the IMS, while in a federation the ownership of an MIO typically resides with the originating federate. However, in a federation, copies of an MIO may be cached in other federates to reduce the network traffic between federates or to maintain replicas for fault tolerance. For example, our prototyped design for querying includes caching of MIOs at remote federates to minimize the number of MIO transfers in response to queries. A federate responding to a federated query initially sends only MIO identifying information; if the querying federate finds a copy of the MIO in its cache then the MIO itself does not have to be transferred. Note though that if the publishing federate has left the federation it can't service queries (i.e., it can't identify the MIOs) so we don't know which locally cached MIOs satisfy a query. In this situation the federation could be configured to remove such cached MIOs. On the other hand, throwing away all cached MIOs "owned" by a federate when it disappears could be inefficient if that federate later re-joins the federation. A better solution might simply be to consider adaptive quality of service approaches and keep the cache intact as long as sufficient memory exists. If space constraints require removing MIOs from cache, removal could start with MIOs owned by federates that are no longer members of the federation.

Supporting persistence of cached MIOs and operations such as re-joining a federation requires federate identities, and federation names for those, with lifecycles at least as long as the federation's. Without such a notion of identity, provenance information for cached objects could be meaningless and all joins would have to be treated as new memberships.

Deployed federations will need to consider cases in which federates can leave and subsequently rejoin a federation, yet need to be treated consistently across the multiple memberships. For example, federations will need to support federates that have temporary communication disruption whether it is planned (e.g., temporarily maintaining radio silence) or unplanned (e.g., unreliable communications). We have identified a number of issues in managing intermittent membership, including:

- Dealing with federation and federate operations that occur during federate outage.
- Interaction of federate state with federation state during outage.
- Synchronizing federation and federate states (e.g., subscriptions, policies, archived MIOs).
- Dealing with interrupted operations (e.g., a query in process when a federate leaves).

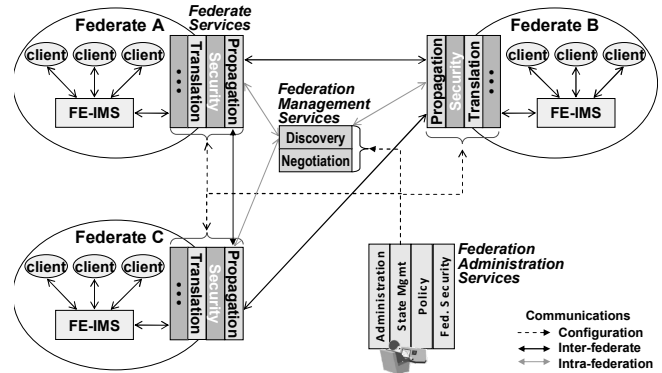


Figure 5. Services can be composed to match the needs and configurations of particular federates or the federation.

- The effect of the length of absence on the federation state and actions within the federation.

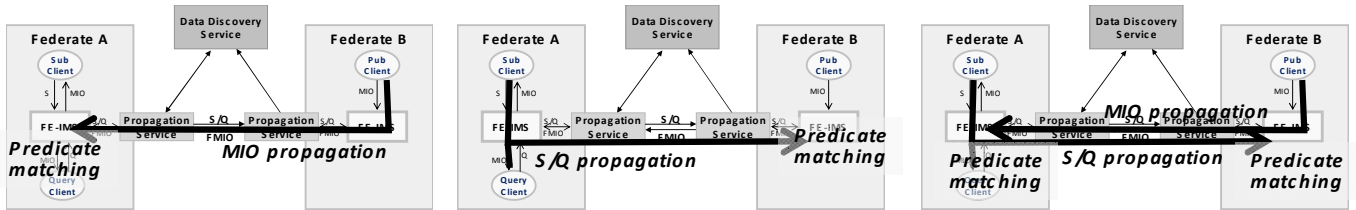
V. ALTERNATE DESIGNS

One of the keys to our layered services-based architecture is the ability for services to be composed in different configurations, as needed, and for federates to use different services or different implementations of similar services. As illustrated in Figure 5, each federate can compose its own set of services (such as security, translation, propagation, etc.). The Propagation Service must expose an interface consistent with other Propagation Services to communicate with other federates, and must preserve consistency with the federation services to interact with those.

This flexibility enables a wide variety of options. For example, security can exist in the IMS (providing client and MIO level security), at the federate layer (providing access control and information protection for the federate), at the federation layer (enforcing security policies across a set of federates), or at several of these layers.

As another example of the flexibility enabled by our ELSIF approach, the matching of request predicates (subscriptions or queries) to MIO metadata can be configured to happen at any of multiple places in a federation. The choice of where predicate matching occurs comes with tradeoffs in functionality and performance, as illustrated in Figure 6.

If matching occurs in the recipient federate only (Figure 6a) then published MIOs would propagate from the federate containing the publishing client to the federate where they would be matched. An advantage to this configuration is that the local access policies of the receiving IMS and requesting clients would be enforced. A disadvantage is that MIOs would be moved whether or not they match a subscription or query predicate in the receiving federate. Another disadvantage is that MIOs would need to be archived in the source and recipient IMSs; the former to ser-



(a) Predicate matching at requestor

(b) Predicate matching at publisher

(c) Predicate matching at both

Figure 6. Configuration options for predicate matching.

vice local, non-federated requests and the latter to service federated queries.

If matching occurs in the source federate only (Figure 6b), then subscriptions (and queries) must be propagated from the requesting client’s federate to federates containing the publishers (or archives). This could reduce the number of MIOs traversing the network since only MIOs that match requests would be moved. Likewise, MIOs would only need to be stored in the source federate’s IMS, where they could service both local and federated queries. However, the recipient IMS’s access policy would not be enforced, active subscription information would need to be maintained outside the subscriber’s federate, MIOs would have to be re-sent when queried more than once, and client identity would have to be attached to requests and responses to support delivery in the recipient federate.

If matching occurs both in the recipient and source federates (Figure 6c), local IMS access control is enforced on both sides and more existing IMS infrastructure is reused since MIOs are delivered by the local IMS. However, the additional matching for federated operations could impact performance.

VI. EVALUATION

We conducted experiments to evaluate the performance of federated operations compared to local non-federated operations. As a baseline, we used the Air Force Research La-

boratory’s Apollo version 1.0 [5]. We compared this baseline to two experimental cases. The first is a single-federate federation, i.e., a single Apollo instance, wrapped as an FE-IMS with Propagation and Data Discovery Services, but no other federates and thus no federated operations. This case tests the overhead introduced by the federated services on local, non-federated operations. The second experimental case is a two-federate federation with only federated operations, i.e., all publications are in one federate and all subscriptions in the other.

We ran the experiments on three identical rack-mounted machines, each with a single Intel Pentium 4 2.66 GHz processor and 1 GB RAM, connected via a private network (i.e., there was no competing traffic). Each federate had one machine dedicated to its IMS, FE-IMS, and Propagation Service and the third machine held all information space clients and a Data Discovery Service. We configured the services with no security, i.e., message traffic was not authenticated, there were no hashes attached to methods, no signature traffic, and no signature checking. The Discovery Service was configured to send all publications to every federate.

To conduct the experiments, one client published as many MIOs as possible for 30 seconds, each having an XML-formatted header under 1024 characters long, followed by a 1024-byte payload. We varied the number of subscribing clients, each with a single active subscription that would

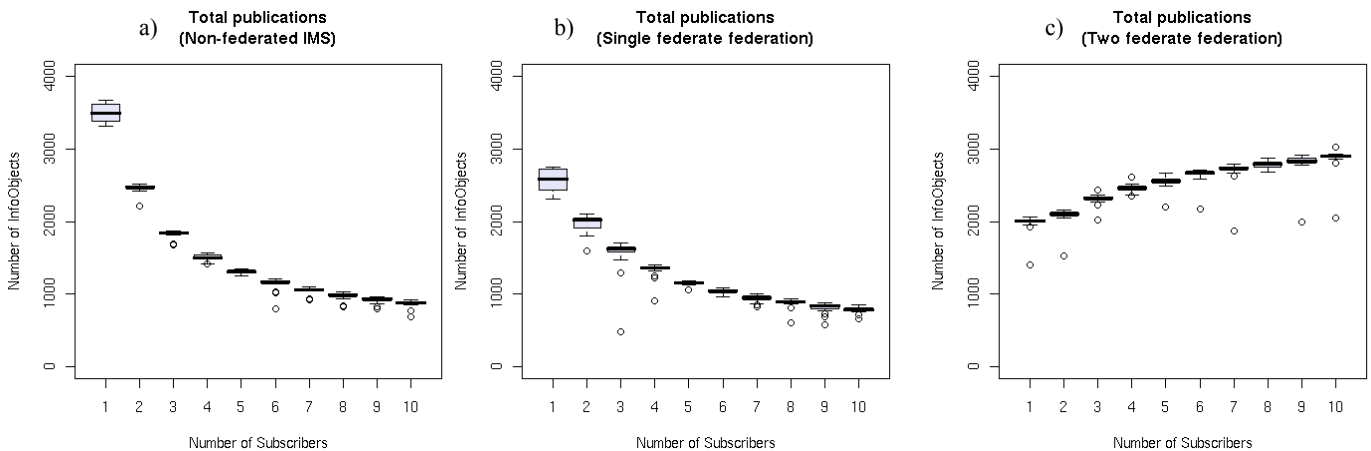


Figure 7. Total publications in 30 seconds in each of the three configurations.

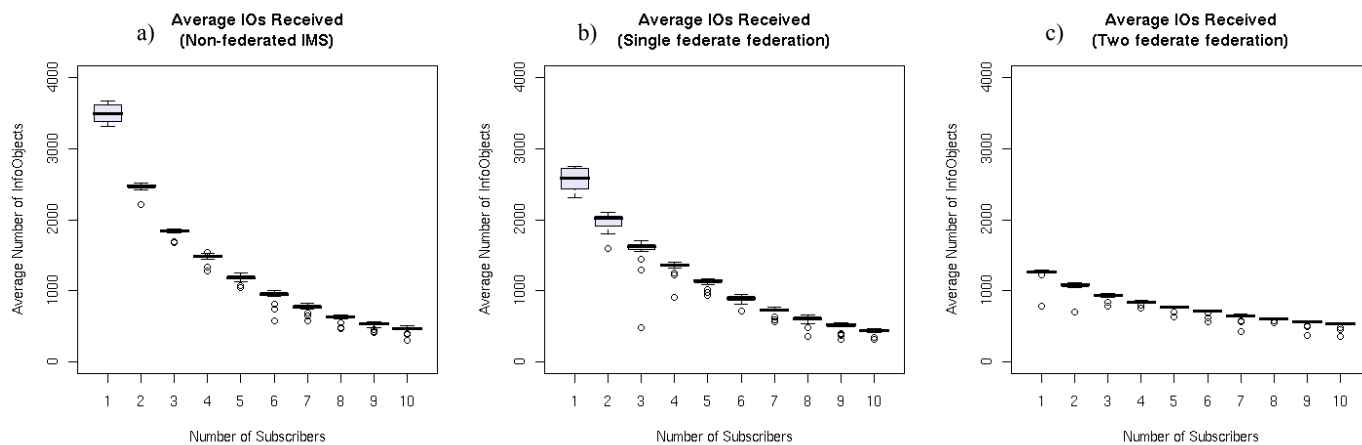


Figure 8. Maximum throughput of each configuration over a 30 second period.

match every published MIO.⁴ We ran twenty iterations for each number of subscribers and graphed the results as box plots [12].

Figure 7 shows the effect of federation on the maximum possible publication rate. Graphs a and b show that the federate and federation services result in a significant reduction in publication rate when there are only a few local subscribers; as much as a 28% reduction with only one local subscriber. As the number of subscribers increases, however, the relative overhead introduced by federation decreases, indicating that as the information space scales the limitations of the IMS operations dominate those of the local federate and federation services.

Figure 7c shows that delivering the MIOs across the federation, rather than locally, increases the maximum publication rate. This suggests the need for further evaluation to examine effects such as thread contention (inside an IMS vs. between federates) on the time to move objects between federates.

Figure 8 (a – c) shows the average number of MIOs received by a subscriber in each of the three configurations during the 30 second time frame – a relative measure of the throughput possible in the IMS with and without federation.⁵ As expected, with only a few subscribers the federation configurations show lower throughput than the baseline configuration. However, again, as the number of subscribers increases, the throughput of the federation cases is essentially equal to that of the non-federated IMS.

⁴ Benchmarking tests indicated that varying the number of publishers does not affect the rate at which the Apollo IMS will accept publications, while varying the number of subscribers does.

⁵ Differences between Figures 7 and 8 indicate that some MIOs are still in delivery queues at the end of the measurement period.

VII. SUMMARY AND NEXT STEPS

In this paper we presented the results of our research into applying a services-based approach to the federation of information spaces. This work builds upon ongoing research in publish-subscribe-query information management and addresses the current and future need to support coalition operations and communities of interest organized around political, doctrinal, or mission boundaries. We are providing solutions to address the technical aspects of federated interoperation, including the exchange of information across federate boundaries, information protection and security, and configuration and dynamic state management.

There is a tension between trying to define a comprehensive set of services for federation and providing architecture with no (or few) implemented services. The former scenario is unlikely because it is impossible to anticipate all the services that could ever be used in federating information spaces; the latter provides too little support for current federation needs. Our approach strikes a practical balance by defining both a baseline set of key services and an extensible architecture. The baseline service implementations ensure that the results of our research prove the concept and provide the key enabling capabilities needed in federations; and the extensible architecture serves as a solid foundation for the development of future federation capabilities.

Our future research will continue to build upon this foundation. We plan to conduct more experiments to determine the causes of some of the scalability results we observed and to evaluate additional scalability, configuration, and tradeoff variations. We plan to investigate more aspects of service composition, especially building upon trends in service-oriented architectures, and additional needed services. Finally, we plan to investigate additional aspects of heterogeneity, including how to configure our services to

support both tactical and enterprise environments, support for different IMS implementations, and operation across multiple security domains.

ACKNOWLEDGMENT

The authors would like to acknowledge the support and collaboration of the USAF Air Force Research Laboratory (AFRL) Information Directorate. This work was sponsored by AFRL under contract number FA8750-07-C-0168.

REFERENCES

- [1] M. Linderman, B. Siegel, D. Ouellet, J. Brichacek, S. Haines, G. Chase, J. O'May. A Reference Model for Information Management to Support Coalition Information Sharing Needs. Tenth International Command and Control Technology Symposium (ICCRTS), 2005.
- [2] V. Combs, R. Hillman, M. Muccio, R. McKeel. Joint Battlespace Infosphere: Information Management within a C2 Enterprise. Tenth International Command and Control Technology Symposium (ICCRTS), 2005.
- [3] Defense Information Systems Agency, Net-Centric Enterprise Services. <http://www.disa.mil/nces/>.
- [4] A. Sheth, J. Larson. Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases. ACM Computing Surveys. 22(3): 183-236 (1990).
- [5] Air Force Research Laboratory, Apollo 1.0 User Guide.
- [6] W3C. XQuery 1.0: An XML Query Language, W3C Recommendation, 23 January 2007, <http://www.w3.org/TR/xquery/>.
- [7] W3C, Extensible Markup Language (XML) 1.0, W3C Recommendation 16 August 2006, <http://www.w3.org/TR/xml/>.
- [8] The Infospherics Group. Common Application Programming Interface (CAPI): <http://www.infospherics.org>.
- [9] OASIS eXtensible Access Control Markup Language (XACML) Version 2.0. OASIS Standard, 1 Feb 2005. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml.
- [10] Sun Microsystems, Java Message Service, V1.1. April 12, 2002. <http://java.sun.com/products/jms/docs.html>.
- [11] DoD CIO, Department of Defense Global Information Grid Architectural Vision, Vision for a Net-Centric, Service-Oriented DoD Enterprise, Version 1.0, June 2007. <http://www.defenselink.mil/cio-nii/docs/GIGArchVision.pdf>.
- [12] Boxplots: www.wikipedia.org/Boxplots.