A Dynamic and Policy-Controlled Approach to Federating Information Systems

Niranjan Suri, Massimiliano Marcon,
Andrzej Uszok, Maggie Breedy,
Jeffrey M. Bradshaw, Marco Carvalho
Florida Institute for Human and Machine Cognition
Pensacola, FL

James Hanna, Robert Hillman,
Asher Sinclair, Vaughn Combs
Air Force Research Laboratory
Rome, NY

Abstract— **Timely access to relevant data and information is critical to successful mission execution in network centric warfare. Often, the data required to support a mission is not always resident within a single system, but is distributed among multiple systems that must be dynamically interconnected to support the data and information needs. While proprietary and stove-piped information systems have slowly given way to standardized information management architectures (such as the Joint Battlespace Infosphere (JBI) architecture developed by the US Air Force Research Laboratory), each independent organization and/or mission is normally associated with a separate instance of a managed information space that operates in an independent manner. This is necessary given the different stakeholders and administrative domains responsible for the information. However, the demands for coordination and cooperation require interoperability and information exchange between these independently operating information spaces. This paper describes a federated approach to interconnecting multiple information spaces to enable data interchange. We propose a set of interfaces to facilitate dynamic, runtime discovery and federation of information spaces. We also integrate with the KAoS policy and domain services framework to realize policy-based control over the federation and exchange of information. Our approach allows clients to transparently perform publish, subscribe, and query operations across all the federated information spaces. We have integrated with three existing JBI implementations – Apollo from the Air Force Research Laboratory, Mercury from General Dynamics and AIMS from Northrop Grumman. Most recently, we have integrated with Phoenix, a fully SoA (Service-oriented Architecture) based approach to information management.**

Keywords- Architectures; Coalition Operations; Interoperability; Policy-based Information Sharing; Network-centric systems and technologies; System of systems.

I. INTRODUCTION

Information systems are a key component of any military mission and are essential to ensuring their successful execution. Traditionally, information management was supported by stove-piped systems that were difficult to update, modify, and integrate. In order to address this problem, the US Air Force Research Laboratory developed the Joint Battlespace Infosphere (JBI) architecture [1] first and started working on the Phoenix specification [2] afterwards. Both JBI and Phoenix try to define a standard for the implementation of information management architectures that support a publish/subscribe/query model. In addition to that, the JBI architecture standardizes the interfaces for client applications (CAPI – the Client API) to facilitate client integration into any JBI implementation.

This standardization enables the implementation of information management architectures that are based on a common information management model. However, the interconnection and information sharing between information spaces (infospaces) that may belong to different administrative domains still remains an open issue.

Federation solves this problem by supporting the interconnection of multiple, independently managed infospaces for information sharing. This paper describes our approach to federation. We propose a set of interfaces to facilitate dynamic, runtime discovery and federation of infospaces. We also integrate with KAoS - policy and domain services framework, to realize policy-based control over the federation and the exchange of information. Our approach allows clients to transparently perform publish, subscribe, and query operations across all the federated information spaces.

For the purpose of this paper we only consider the JBI architecture when explaining the key components of our federation approach. In terms of implementation we have also generalized our approach in order to fit into the Phoenix specification as well.

II. OVERVIEW OF JBI

The architecture and motivations for JBI are described in detail in [3], which presents a reference model for information management. The elements of the JBI architecture essential for the scope of federation are highlighted in Figure 1.

An Information Space is defined as one instance of a JBI based system, which facilitates exchange of information between clients. A number of clients connect to the system, behaving as producers and/or consumers of information.

The system includes both an Information Catalogue, that is a directory of information types known to the system, as well as an Information Repository, which handles the actual data. The Information Repository may optionally archive information for later retrieval using queries. Different JBI based implementations are free to use any approach as long as

they comply with the syntax and semantics of the CAPI - the Client API. In the case of Apollo, one of AFRL's reference implementations of the JBI information management concepts, the Information Catalogue is called the Metadata Repository (MDR), while the Information Repository is called the Information Object Repository (IOR). Published data is represented as a Managed Information Object (MIO). Each MIO has a corresponding data type that is registered in the MDR, metadata in the form of an XML document, and a payload. Clients may have standing subscriptions based on the type, with an optional predicate to match against published metadata. If a predicate is specified, it is in the form of an XPATH expression, which can filter out unnecessary MIOs that a client is not interested in receiving. Clients may also execute queries that result in matching MIOs being retrieved from the IOR and returned to the client.

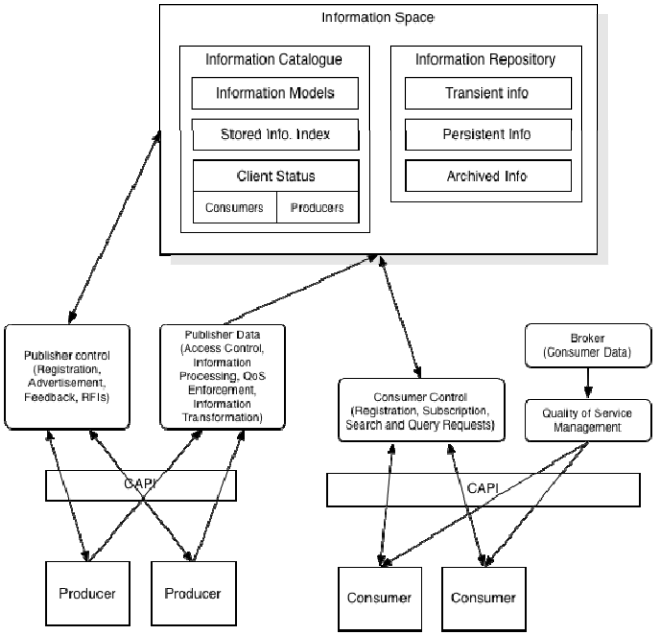

Figure 1: Architecture of a JBI-Oriented Information Management System

A client typically connects to one (and only one) Information Space. While it is possible to connect to multiple information spaces, doing so places the onus on the client to discover the information spaces and connect to each one. The client would also need to be authenticated with multiple information spaces, which implies that all of them must have accounts for the client (difficult when there are multiple administrative domains involved). One of the benefits of Federation is to hide the presence of multiple information spaces from the clients. Each client continues to connect to one information space, but has access to all allowed information (controlled by policy) across multiple information spaces.

III. Federation Architecture

The federation architecture supports seamless and secure integration of multiple information spaces, that are called federates. Seamless implies that the architecture supports automatic discovery of and interconnection between federates. The process of federation is transparent to clients, which still connect to their home federate as normal. Secure implies that the federation process is not arbitrary and open. The establishment of federation and exchange of information is controlled via policies. Section VI describes the role of policies in greater detail.

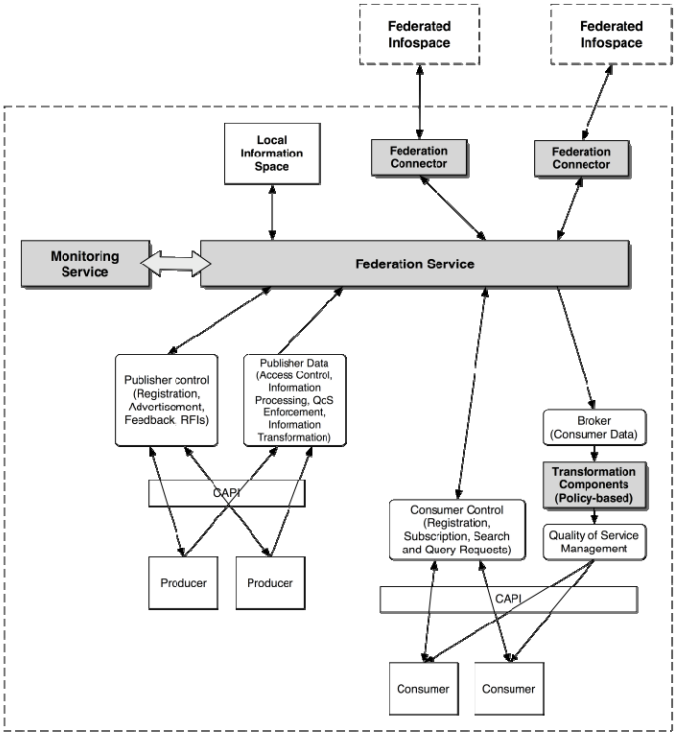

Figure 2: Architecture for Federation

One aspect of our federation architecture is that all federates are peers. Each federate independently manages its connection with other federates. Each federate has its own set of policies that govern the exchange of information with other federates. This is a logical approach considering that federates may sit on different administrative domains. Policies can however be established from a single administrator point using the KAoS Policy Administration Tool (KPAT).

Figure 2 shows our architecture for federation. The components we added to the original JBI architecture are represented with shaded boxes. The Federation Service (FS) is the major component. It handles the exchange of data between the local infospace and its federates. The interaction between federates can be controlled thereby restricting the behavior of FS through a set of policies specifying obligations and constraints. The Monitoring Service (MS) provides statistics about host and network performances as well as about the FS itself. FS may use this information to perform different types of adaptation. The communication with other information spaces that are part of the federation is handled by the Federation Connectors (FC). Each federate instantiates a number of FCs, one for each of the other federates it is connected with.

IV. Federation Interfaces and Implementation

One of the main goals of this research effort was the development of a generic set of interfaces supporting federation in order to obtain a flexible architecture easily adaptable to

different IMS implementations. After examining the JBI CAPI specification and a number of implementations (i.e. Apollo from AFRL, Mercury from General Dynamics and AIMS from Northrop Grumman), we developed the following main interfaces:

- IMDService,
- InfoObjectReceptor,
- QueryReceptor,
- IMDServiceMonitor,
- AdaptationOracle.

Figure 3 shows in detail how all the interfaces are inserted in the Federation Service Architecture.

IMDService supports all the basic operations that each federate may want to perform on another federate. This set of operations is implemented in the FS and is invoked by the local Information Management System (e.g., Apollo). FS interacts with the remote federates using the Remote Federation Service Proxy (RFSP) that exposes the same interface. For the local FS, each RFSP represents a hook to the related remote federate. A proxy contains an instance of the Federation Connector (FC) and an instance of a Remote Request Handler (RRH). FC manages the communication with the related remote federate across the network and RRH handles the requests received from the remote federate executing them on the local IMS.

The InfoObjectReceptor interface is used by RRH for the delivery of information objects to any subscribed client. When a query is issued from a remote federate, the query is executed in the local IMS by invoking the QueryReceptor interface. In our federation architecture, both these interface are implemented by a direct modification in the IMS.

The Discovery Manager (DM) component provides the discovery functionalities that are necessary to automatically find other potential federates in the network. The discovery process relies either on the capabilities of the Group Manager [4] or on the Cross-layer communication substrate (XLayer) [5] for discovery and grouping support.

The Federation Manager (FM) takes the necessary actions when new potential federates are discovered by the DM and when connections with remote federates are terminated by the FCs.

The Federation Monitoring Component (FMC) implements the IMDServiceMonitor interface that exposes all the functionalities for monitoring the behavior of FS. FMC registers with the underlying Monitoring Service [6] as a provider for application-level statistics about the performance of the local IMS (e.g., number of info objects published per second, predicate matching rate per subscription, etc.). The Adaptation Manager (AM) provides a concrete implementation of the AdaptationOracle interface. AM takes advantage of the application-level statistics along with information about system and network behavior to dynamically adapt the behavior of federation.

AM currently incorporates two specific adaptation mechanisms: a CPU-overload adaptation and a low-bandwidth adaptation. The CPU-overload adaptation is triggered when CPU utilization on the local federate becomes higher than a predefined threshold. In such cases, remote subscriptions are sorted based on the hit-rate of their predicates and the predicate evaluation is then temporarily disabled, starting with the predicates that match the most, until the CPU is no longer overloaded. Turning off local evaluation of remote predicates implies that all publications that match the type are sent to the remote federates. Predicates with high hit-rates are selected first because their predicate matching would likely succeed; therefore disabling their evaluation increases the bandwidth utilization by the minimum amount possible. The low-bandwidth adaptation handles network overload situations in the connection with a remote federate. In this case, the adaptation mechanism entirely disables remote subscriptions from that federate. The subscriptions are chosen based on their priorities, which can be specified by the remote clients.

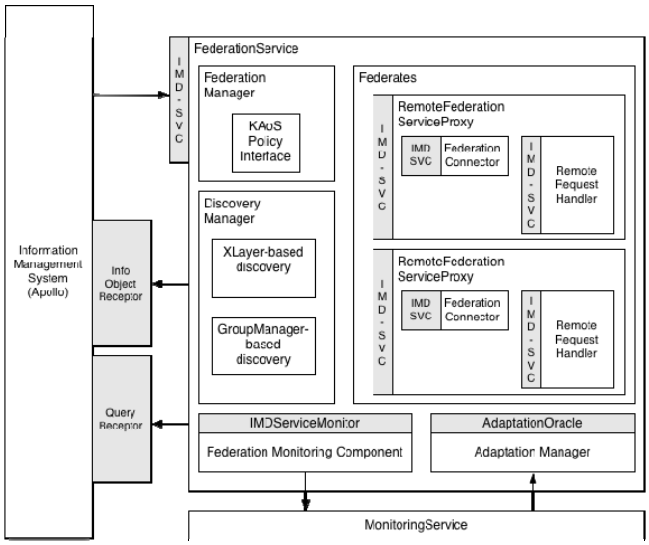

Figure 3: Federation Interfaces and Federation Service Architecture

V. FEDERATION IN ACTION

This section describes in detail the process of federation from the discovery phase to the disconnection of a federate. The publish and subscribe mechanism implemented by the Federation Service are described as well.

For simplicity, we will consider a scenario where the federation happens only between two instances of an Information Management System (IMS), which we will refer to as Federate A and Federate B. We will also assume that the nodes where the IMSs run are discovered with a lower level discovery-enabled communication substrate, such as XLayer or Group Manager mentioned in the previous section. This discovery process provides the endpoint address (IP address and port) for each federate to the other.

A. Federation Establishment

When the Federation Service is instantiated along with the other services that are part of the IMS architecture, the first step is the registration by the DM with the discovery and grouping API provided by the sub-layer with such capabilities. By

registering and joining a predefined group, the IMS manifests its intention of being part of the federation. Once that happens, each IMS instance is notified about the presence of the other. At this point, a handshake phase starts. During the handshake, each potential federate opens a connection to the other and eventually a contract negotiation occurs (described in Section VI). Upon contract acceptance by both nodes, the federation is officially established, and each federate creates an instance of a RFSP.

B. Subscription forwarding

When a client connected to Federate A issues a subscription with its local IMS, the request is captured by the Federation Service (FS). FS retrieves the RFSP for Federate B and uses it to remotely forward the received request for subscription. Once Federate B obtains it, the subscription is stored in a remote subscriptions table, ready to be matched against local publications.

C. Publication handling

When a client publishes information to the local IMS (Federate A), such publication is intercepted by the FS. In normal conditions (i.e., with no adaptations in effect), Federate A attempts to execute the predicate matching locally, by comparing the publication type and metadata with the remote subscriptions it may have previously stored in its remote subscription table. Publications for which the local matching succeeds are marked as matched, and sent to Federate B via the RFSP. Federate B receives the publication, verifies if it was already matched (and if it was not it matches it with the local subscriptions), and forwards it to the IMS. Finally the IMS takes care of the delivery to the correct subscriber clients.

D. Federation Termination

Federation lasts until at least one of the nodes dies or leaves the federation group. When the other is notified about one of these events it cleans up any references to the former remote federate, including any cached remote subscriptions.

E. Policies

All the federation operational behavior detailed above is entirely governed by policies. Before performing any step in its execution flow, FS verifies with the policy framework whether the current operation is allowed, and whether there are any restrictions to be imposed. Section VI explains in detail contracts and policies to dynamically control the behavior of federation.

VI. Federation Service Contracts

An important aspect for the coordinated operation of federated infospheres is a comprehensive, semantically-rich, and enforceable service agreements. The privileges and obligations of each infosphere within the federation must be established and monitored for compliance at all times. The service agreements bind all parties to act according to the constraints accepted when the federation was formed. This approach is necessary to ensure the proper flow of information through the federation. The KAoS Policy Service [7] with federation specific extensions is used by the FS to create and enforce federation contracts. KAoS allows for the specification, management, conflict resolution, and enforcement of policies within domains. The use of ontologies, encoded in OWL [8], to represent policies enables reasoning about the controlled environment, about policy relations and disclosure, policy conflict resolution, as well as about domain structure and concepts. The behavior of all the components in the FS is dynamically controllable at runtime via policies. The FS on each federate is integrated with the KAoS Guard software component, which stores policies controlling establishment, lifecycle, information exchange, and adaptation of the federations established by this federate. When a node discovers a new possible federation partner and the initial connection is established, the two potential federates exchange information about their current configuration. Based on this information as well as its own local policies, each federate independently decides:

- Whether to establish a federation with the remote federate,
- What priority to assign to the remote federate,
- Based on the current resource usage for the federation operations and the assigned federate priority, how to estimate the quantity of resources it can devote to server requests from the federate,
- What metadata type subscriptions or queries it would be able to support for a given federate.

During subsequent subscription exchanges, queries, and publication with federates, each operation is analyzed with respect to current policies. Policies may allow or prevent a given operation. They may also modify the operation by changing the subscription or query predicate, or by removing metadata from the published information object being forwarded to the remote federate. Moreover, policies may enforce or waive obligations (e.g., logging) relevant to certain types of operations. In addition, policies and the agreed adaptation matrix control how and when a given adaptation mechanism is activated when the share of resources used by the given federate exceeds the agreed-upon limit.

KAoS is controlled using the KAoS Policy Administration Tool (KPAT), a graphical policy management tool. KPAT configuration for the control of federation consists of sets of predefined policy templates and policies associated with them. Each policy can be easily activated and deactivated. The policy templates are grouped into four categories:

- Federation Acceptance Polices,
- Gatekeeping Policies,
- Adaptation Policies,
- Contract Policies.

VII. Experimental Evaluation

The Federation Service has been experimentally evaluated in terms of measuring overhead from adding federation capabilities to the base Information Management System (IMS). For this experimentation we considered both Apollo as well as Phoenix. We measured the performance of publish and subscribe operations considering a baseline installation of the

evaluated IMS versus two installations of the same IMS sitting on two different nodes collaborating together through federation.

This experimental evaluation was conducted on virtual machines running on VMWare Server. The host machines have a dual core 3.06 GHz Intel Xeon processor and 4GB of memory. We deployed one virtual machine per physical machine. All the virtual machines were running Ubuntu Server 8.04, and were provided with 1GB of RAM.

In order to understand the overhead that may be caused by the Federation Service, we measured the throughput in terms of time spent to send and receive information by the clients (execution time) and the maximum number of Information Objects per second that clients were able to send and receive (throughput).

All the tests involved one publisher client and two subscriber clients. In the first set, all are connected to the same instance of the IMS. In the second set of experiments, the publisher was connected to the first instance of the IMS and the subscribers were both connected to the second one, so the Information Objects were sent to the other side across the federation. The performance evaluation was executed using the benchmark suite provided with Apollo and adding clients that would support the information exchange protocols defined by the Phoenix architecture. We chose to run the clients with 55 iterations. With this configuration, publisher and subscriber clients exchange 1275 Information Objects. Figure 4 shows the experimental scenario.

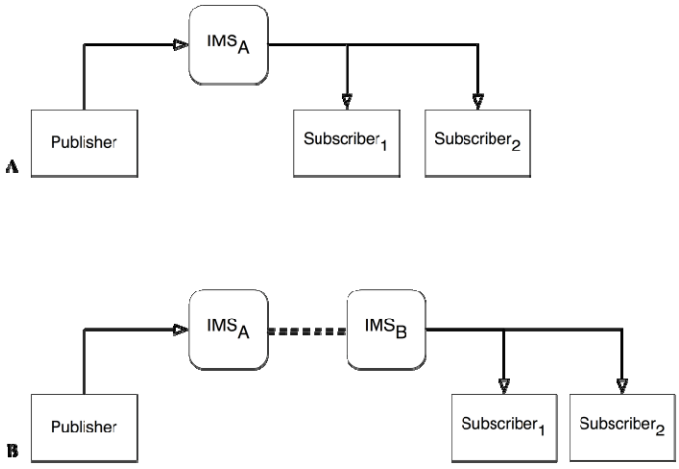

Figure 4: Experimental Scenario for the Performance Evaluation: The Baseline Version of the Tested IMS is shown in A. B shows the Configuration for the Tests with the Federation Service

From the results presented in Table 1 and Table 2, we make two different observations. In the case of Apollo, we can see how the presence of the Federation Service improves the overall performance of the IMS instead of creating overhead. That actually makes sense: by adding federation capabilities, we split the load between the two federates (which are on separate physical nodes). In particular the publications are handled by the first instance of the IMS while the subscriptions are managed by the second instance.

This becomes even clearer when considering the numbers obtained in Phoenix tests. Phoenix uses asynchronous channels that rely on the Netty framework [9] for exchanging information from and to clients. On the publisher side, this means that the publication time and rate are not affected by the computation that is necessary to manage every single piece of information being published. The publisher keeps putting information into the channel as fast as it can. The underlying layer will then manage the delivery to the IMS. This explains the very small difference in terms of performance of the publishing information to Phoenix with or without federation.

TABLE 1: TIME SPENT TO PUBLISH AND RECEIVE THE INFORMATION OBJECTS BY CLIENTS

Configuration	Publisher Time	Subscriber 1 Time	Subscriber 2 Time
Apollo Baseline	46.62 sec	41.16 sec	40.87 sec
Apollo with Federation	29.00 sec	28.14 sec	28.19 sec
Phoenix Baseline	3.81 sec	9.27 sec	9.26 sec
Phoenix with Federation	3.88 sec	4.81 sec	4.81 sec

TABLE 2: NUMBER OF INFORMATION OBJECTS PER SECOND PUBLISHED AND RECEIVED BY CLIENTS

Configuration	Publisher	Subscriber 1	Subscriber 2
Apollo Baseline	29.2 IO/s	30.97 IO/s	31.19 IO/s
Apollo with Federation	42.10 IO/s	45.31 IO/s	45.22 IO/s
Phoenix Baseline	327.80 IO/s	137.55 IO/s	137.63 IO/s
Phoenix with Federation	322.09 IO/s	265.29 IO/s	265.29 IO/s

On the other hand, the subscribers' performance is affected by the computation the IMS needs to accomplish in order to manage the Information Objects it receives from the publisher and then dispatch them to the right subscriber clients. Time and reception rate are calculated from when the first piece of information is received to when the last one is delivered, which occurs concurrently with incoming information from publishers that needs to be handled. Having the load divided between two IMSs interconnected with federation shows its benefits also in the case of Phoenix.

The throughput results presented above show that from the client perspective, there is no performance degradation in terms of time and rate caused by adding federation capabilities to an IMS. The overhead of federation does manifest itself in terms of increased latency in information delivery. Latency of the information, i.e. the difference in time between when the information is produced by the publisher and when the same information is received by the subscriber, is crucial for certain types of applications, particularly in the tactical environment. The delay in the delivery of Information Objects to the subscribers clearly increases when such Information Objects have to be transmitted through the network to remote federates. Preliminary tests show that when the Federation Service is involved in the publish-delivery process, the latency of a single Information Object increases by 20% on average. This increase in latency is highly dependent on the network latency. As

shown in Figure 4, there is an extra network hop involved with federation, which is the primary factor contributing to the latency.

One more noteworthy aspect with the results is the comparison between Apollo and Phoenix. The results show a slight improvement in the performance of federation between Apollo and Phoenix. When adapting our architecture for the Phoenix environment we started moving towards a lighter-weight services approach, and that seems to have produced benefits in terms of efficiency of the federation implementation. If we evaluate the subscriber side, Apollo with federation was about 1.4 times faster than Apollo baseline. Phoenix with federation instead is almost 2 times faster than Phoenix baseline.

VIII. Future Work

The Air Force and the Department of Defense, in the process of moving toward network-centric operations, have embraced Service Oriented Architecture (SOA) based systems as a necessary means to help implementing the conceived overarching Global Information Grid (GIG). The JBI information management client-server concepts are now being morphed and extended into Phoenix, a flexible SOA-based approach to information management. The SOA characteristics of the services in Phoenix allow applications to exchange data as they perform their individual or collaborative processing. These SOA services provide the perfect blend of rigidness and flexibility that is needed for effective information management operations. The federation approach discussed above is also being evaluated against the Phoenix architecture to enable the federated capabilities as a set of services, as mentioned in Section VII. The federation services will work in harmony with the information services and other service deployments in other domains. The effort will result in the architecture enhancements to assure that the necessary set of federation SOA services are designed in a manner that is consistent with the Phoenix architecture. Following the specification and architecture design phase, service implementations will be developed to provide federation capabilities for multiple SOA service deployments.

Acknowledgment

This work is supported by the U.S. Air Force Research Laboratory under Cooperative Agreement FA8750-07-2-0174.

References

[1] Infospherics Web Site. Online reference: http://www.infospherics.org.

[2] Grant, R., Combs, C., Hanna, J., Lipa, B., Reilly, J. "Phoenix: SOA based information management services," Proceedings of the 2009 SPIE Defense Transformation and Net-Centric Systems Conference, Orlando, Fl, April 2009.

[3] Linderman, M., et. al., "A Reference Model for Information Management to Support Coalition Information Sharing Needs", In Proceedings of 10th International Command and Control Research and Technology Symposium, 2005.

[4] Suri, N, Marcon, M., Quitadamo, R., Rebeschini, M., Arguedas, M., Stabellini, S., Tortonesi, M., Stefanelli, C. An Adaptive and Efficient Peer-to-Peer Service-oriented Architecture for MANET Environments with Agile Computing. In Proceedings of the Second IEEE Workshop on Autonomic Computing and Network Management (ACNM'08).

[5] Carvalho, M., Suri, N., Arguedas, M., Rebeschini, M., and Breedy, M. A Cross-Layer Communications Framework for Tactical Environments. In Proceedings of the 2006 IEEE Military Communications Conference (MILCOM 2006), October 2006, Washington, D.C.

[6] Loyal J. P., Carvalho, M., Martignoni III A., Schmidth, D., Sinclair, A., Gillen, M., Edmonson J., Bunch, L., Corman, D. QoS Enabled Dissemination of Managed Information Objects in a Publish – Subscribe – Query Information Broker. In Proceedings of the SPIE Conference on Defense Transformation and Net-Centric Systems 2009.

[7] Uszok, A., Bradshaw, J., Lott, J. Breedy, M., Bunch, L., Feltovich, P., Johnson, M. and Jung, H., (2008). New Developments in Ontology-Based Policy Management: Increasing the Practicality and Comprehensiveness of KAoS. In Proceedings of the IEEE Workshop on Policy 2008, IEEE Press.

[8] Web Ontology Language, online reference: http://www.w3.org/TR/owl-features.

[9] Netty Framework Web Site: http://www.jboss.org/netty.