Technologies for Federation and Interoperation of Coalition Networks

Seraphin Calo†, David Wood†, Petros Zerfos†, David Vyvyan§, Patrick Dantressangle§, Graham Bent§

Abstract— When data from two or more coalition members is used in a joint mission, the flow of information over the respective networks must be tightly controlled. Data sent over the networks would be routed through nodes that should enforce policies concerning: what information about each coalition member’s assets will be exchanged with other coalition members; who, with what authentication, and under what conditions can access the information, and in what form it can be provided. We discuss the requirements of such a policy-enabled federated network, in which any node could be responsible for enforcement of all policies applicable to it and the data that flows through it. We describe the high-level architecture of the system currently under development, and the design trade-offs that were made to balance ease of deployability and flexibility in policy enforcement. We also present our initial implementation with performance evaluation results that showcase its applicability to the scenario considered.

Index Terms— Data federation, policies, controlled information sharing, autonomic resource management

I. INTRODUCTION

A COALITION operation usually entails an ad hoc arrangement between two or more organizations that act together to pursue a common objective. In such a coalition, each organization will have its own inherent restrictions on how it will interact with the others. These are usually stated as a set of policies, including security and privacy policies. Within a coalition environment, ad hoc Communities of Interest (CoI’s) come together, perhaps for only a short time, with different sensors, sensor platforms, data fusion elements, and networks to conduct a task (or set of tasks) with different coalition members taking different roles.

To abide with the inherent restrictions of each of the coalition members for information sharing, a key capability required for a coalition operation is the ability to deliver the right information to the right coalition partner in conformance with the policies of all. This capability can be provided by having the flow of information from the organizations’ networks be controlled by a set of coalition policies and all information exchanged amongst the networks be routed through a network of secure gateways, which federate data within the distributed ad hoc network and enforce the policies. The gateway network would enforce policies concerning: what information about each coalition member’s assets will be exchanged with other coalition members; and, who, with what authentication, and under what conditions can access sensor information, and in what form it can be provided.

This controlled information flow can be accomplished by event driven, policy-enabled replication of data between data stores in the separate networks, or directly in response to received requests that are forwarded to the appropriate data sources. We have begun to explore policy-managed sharing of data in a distributed database network in which locally-defined policies are enforced globally. More specifically, this paper makes the following contributions:

• It discusses the challenges and the requirements for controlled information sharing among coalition forces.
• It describes the design and implementation of a novel system that provides policy-based filtering for federated data access, and constitutes a flexible and easily deployable solution for controlled information sharing.
• It provides preliminary performance evaluation results of the system that showcase its feasibility and applicability, along with system optimizations implemented to increase its performance.

The rest of this paper is organized as follows. Section II describes a typical military scenario for controlled information sharing, along with the approaches for providing such functionality and the challenges that are involved. The design and implementation of the system that we developed for policy-enabled federated data access are provided in Section III. Section IV presents preliminary performance results from real experiments that evaluate various aspects of the system. A discussion of the limitations of our proposed system along with avenues for future work is presented in Section V. Section VI discusses related work, and the paper concludes in Section VII.

1Seraphin Calo, David Wood and Petros Zerfos are with the IBM T.J. Watson Research Center, Hawthorne, NY 10532 USA (Petros Zerfos is corresponding author, phone: 914-784-7124; fax: 914-784-6205; e-mail: {scalo, dawood, pzerfos}@us.ibm.com).

2David Vyvyan, Patrick Dantressangle, Graham Bent, are with the IBM UK Emerging Technology Services, Hursley Park, UK (e-mail: {dvyvyan, dantress, grahambent}@uk.ibm.com)
II. FEDERATION AND INTEROPERATION IN COALITION NETWORKS

To provide some context for the description of the challenges involved in controlled information dissemination and the possible approaches for satisfying such need, we first describe a use case scenario of a typical support operation involving multiple coalition forces.

A. Use Case Scenario

Consider a typical Peace Support Operation [18] in which UK and US Coalition forces have been deployed into a region to assist the indigenous Government forces in encounters with an active insurgency and reassuring/supporting the local population. In such a scenario, the Coalition (for example of UK, US and possibly the indigenous Government) forces must operate together to protect the forces in the region and control the region (to protect and support the local population, and deter/defeat the insurgency). This requires the use of ISR (Intelligence, Surveillance and Reconnaissance) networks to provide suitable and actionable intelligence information.

We assume that each coalition partner operates and maintains an ISR network which consists of the various assets belonging to the coalition partner. Each of the ISR networks that are operated by the coalition partners would need the ability to support access from other members of the coalition who may not all be equally trusted. This requires that policy and access control restrictions be put in place to control the flow of ISR data and control operations throughout the network environment.

In the context of such scenario, different policy constraints may need to be incorporated to limit access to asset information. For example, consider the following policies that dictate how information is shared amongst the forces:

- **US can only view UK assets assigned to the mission.**
- **UK can only view US assets that are located in the mission’s perimeter.**
- **The munitions state of UK tanks assigned to the mission are not exposed to non-UK coalition members.**
- **Only share the location of US or UK assets with trusted parties.**
- **US asset information may not be routed through non-US network elements.**

In this use case, it is desirable to define policies locally (i.e. near the asset or by the asset owner), but have them be applied globally, as needed, within the coalition ISR network.

B. Network Setting

ISR networks of the future will consist of a multitude of devices with heterogeneous energy, processing, storage and communication capabilities [11]. Nodes will be anything from sensor platforms to unmanned aerial and ground vehicles equipped with wireless communication capabilities. Center to the Future Combat Systems vision will be the formation of mobile ad hoc networks (MANETs) that offer unprecedented capabilities over conventional networks in terms of deployment efficiency and flexibility as they require no infrastructure support in the battlefield [10]. For this reason, any solution for federated, controlled data access for network-centric coalition operations should be flexible enough for deployment in such networks.

C. Approaches and Challenges

The traditional approach for enforcing policies is in a centralized fashion, by requiring that all data flowing between the coalition networks be routed through a well known gateway. This may be the simplest solution, and it decreases processing overhead at the other nodes of the network, but if only one gateway exists, it becomes a single point-of-failure and a potential traffic bottleneck. Due to the dynamic nature of the networks under consideration, connectivity to the gateway may be lost and then no traffic could be securely transmitted between affected nodes.

Alternatively, every node in the network could be responsible for enforcement of all policies applicable to it and the data that flows through it. The system becomes more robust, but the awareness of every node to more global conditions needs to be increased, with large sets of redundant policies being deployed throughout the coalition networks. This also leads to an increased processing load at every node, for executing all policy decisions locally.

A more flexible and adaptive approach is to have network traffic flow through a network of federation gateways and have all relevant policies be accessible from a distributed database system. Policies can then be deployed selectively, with more global federation-related controls being applied at the gateways. In this work, we explore this approach and report our experiences with designing and implementing such system.

III. SYSTEM DESIGN

In this section, we first describe the requirements and the design rationale for a system that enables interoperability and federation of information. Following, we introduce the system architecture of our proposed policy-enabled network of federated gateways and elaborate on its component technologies.

A. Requirements

The design process of our system was guided by a small set of key requirements that were derived based on the unique characteristics and functional needs of the target (MANET) operational environment:

* **Distributed, ad hoc policy deployment:** the ability to define policies locally and yet have them be enforced globally in the ISR network is highly desirable. Network elements will join and leave, together with the policies they would like to have be enforced. As they join, their policies...
need to be visible and applied across the network.

Data-centric policies for information sharing: agile control of information flow between two or more coalition members of a joint mission should apply at the application-layer protocol of the data that is shared. This allows for a finer grained approach to information sharing than simple allow/deny rules, which typically consider only the origin and destination of information requests. For example, if limited access on data regarding artillery capacity is allowed to some members of the coalition, data sharing policies might specify transformation rules that generate only aggregate summaries, instead of detailed reports on deployed artillery.

Transparent deployment: as coalition networks are managed by disparate administrative domains, it would be unrealistic to expect requestors of information as well as information sources to modify their heterogeneous systems to accommodate controlled access by allied members. Instead, it is desirable to interpose federation technology that is easy to deploy and requires no modifications to existing end-systems, their schemas and data models. This federation technology is deployed upon negotiation and agreement of the coalition members on the policies regarding interoperation; however its deployment does not impose any implementation burden.

Context-aware interoperability policies: coalition sensor networks consist of heterogeneous nodes with varying degrees of processing, communication, and energy capabilities, operating in highly dynamic environments due to increased mobility and under a variety of unpredictable and adversarial situations in the battlefield. For this reason, information sharing has to take into account the context in which information sharing takes place, such as spatio-temporal constraints, device capabilities, level of security clearance, etc.

The above list of key requirements is by no means exhaustive, but instead highlights key aspects of federation and interoperability considered in a MANET coalition setting.

B. Design Rationale

Interoperation among coalition networks is typically conducted using a standards-based language for information interchange such as the structured query language (SQL), the extensible markup language (XML), or publish/subscribe messaging. Consequently, placing filters driven by policies on the application-layer protocol that encodes and transports data described in the aforementioned language formats enables controlled federation of these networks. We begin our exploration of the spectrum of federation and interoperation technologies by developing an application-layer firewall that filters information exchanged using the SQL language over JDBC [1], the de facto protocol used for database-independent connectivity.

Besides establishing connectivity to databases and other tabular data sources, the JDBC protocol is also used to send SQL queries to the sources and process the returned results. Based on this mode of operation, there are two main approaches for performing filtering and controlling of the information flow: a) re-writing the SQL query [3] that is submitted according to filtering policies, so as to limit at the source the results that are retrieved from the databases, and b) filtering the returned results on a row-by-row basis, before returning them to the requesting entity. The former approach has several advantages such as that it minimizes the results that are transmitted over the network. Also, it exhibits performance benefits, as the query optimizer at the data source identifies the most efficient execution plan for the filter-enhanced query. However, the latter method, namely filtering the returned result-set, provides the finest granularity of control as obligation and authorization policies are directly applied to the raw data at the individual cell level. Moreover, it allows for easy incorporation of contextual information that filtering policies can take into account, without requiring any extensions in the information sources [3]. In this paper, we implement and evaluate the latter scheme in our system prototype.

C. System Architecture

Our policy-enabled network of federated gateways brings together two key technologies: a software library for policy management (called Watson Policy Management Library –WPML) and a data federation technology through the use of a distributed federated database system called GaianDB [5]. In this section, we describe these two key systems followed by the integration effort into the final system.

1) WPML: Watson Policy Management Library

Policy is an expression of management intent in a computer readable format, which enables a system to automatically react to some changes in its environment. The goal of policy based self-management is to enable development of software systems that can implement a subset of the reasoning processes of a human administrator in an automated manner.

The policy architecture standardized by the Distributed Management Task Force (DMTF) [2] provides a comprehensive framework for using policies in computer communication networks. This policy architecture consists of four components: a policy management tool, a policy repository, a policy decision point (PDP) and a policy enforcement point (PEP), as depicted in Figure 1. The policy management tool provides a user interface for the creation and definition of policies. The policy repository provides mechanisms for storing policies and retrieving them as needed by the decision points. The policy decision points are modules in the system responsible for selecting and evaluating policies stored in the repository. The policy
enforcement points are elements that are responsible for enforcing the outcome of those policy evaluations.

The Watson Policy Management Library (WPML) provides a set of Java APIs and tools that implement the policy architecture. A generalized policy model able to support arbitrary policy languages sits at the core of the library. A WPML policy supports metadata including name, type, and optionally a description, along with zero or more application-defined attributes (name-value pairs). Policies may be one of two types: authorization policies that provide true/false results and obligation that enable conditional execution of some action. Finally, the language-specific policy string is used to determine the set of required instance data for the policy to be evaluated successfully.

Our initial policy implementation uses the Apache Imperius [12] Java binding for the CIM-SPL policy language [2]. SPL uses Import statements to define required instance data and symbol names onto which policies will be evaluated, Condition statements to define a logical expression, Decision statements to define actions to be taken when the condition evaluates to true, and a Strategy statement to control how multiple policies should be evaluated. For example,

```
Import java.util.List:assets;
Strategy Execute All Applicable;
Condition { assets.contains('UAV') };
Decision { assets.remove('UAV') }
```

The policies and metadata are stored in a Policy Repository. A policy stored in a repository may be marked as active or inactive. WPML implements a number of policy repositories including non-persistent in-memory, and a persistent relational database.

Before requesting policy decisions, a Policy Enforcement Point (PEP) registers with a PDP and provides information including its type (i.e. authorization or obligation), a description of the instance data that will be provided on decision requests, and an optional set of attributes. When a PDP receives an evaluation request from PEP, it performs a matching of the PEP metadata with the policy metadata, using type, instance, attributes and policy activation state.

This matching process must be done on every evaluation request and so needs to be as efficient as possible. WPML caches the results of the matching process, which requires that the repository provide a timestamp of the last policy modification. Since the policy repository is updated infrequently, this generally reduces a full read of the policy repository together with the matching process down to a single read of the timestamp from the repository and an in-memory cache look up. Our relational database implementation uses triggers to maintain the repository timestamps.

While the WPML component can be used for fine-grained filtering of the information that is exchanged among coalition members, a method for enabling such interoperation is needed. Following, we provide more details on how data federation is achieved in our system.

2) GaianDB: Data Federation

To provide data federation capability in our system, we employ a Dynamic Distributed Federated Database (DDFD) named GaianDB, which combines the principles of large distributed databases [6], database federation [5], network topology [7] and the semantics of data [8]. The resulting database architecture is ideally suited to a range of applications that are required in support of networked operations. To implement a network of federated gateways, each node acts as a policy controlled gateway to the locally federated heterogeneous data sources. These can include databases, sensor message streams etc. which are exposed as virtual tables to the rest of the network. Policy controlled access to data is then performed from any gateway node using SQL-like queries and distributed stored procedure-like processing. Using the same ‘Store-Locally-Query-Anywhere’ (SLQA) paradigm, access to a distributed policy repository is also maintained such that policies can be stored anywhere in the network and accessed from any gateway node. Policies relevant to local execution would be stored on the local node, so they would not go missing without the node being disconnected and unavailable itself.

We have investigated various strategies for how such ad hoc networks of gateway nodes can be configured. In this paper we have adopted a strategy which uses "preferential attachment" [5], such that as new nodes join the network, all nodes remain within a limited number of network hops from each other and no single node becomes over-connected, thus avoiding vulnerabilities and bottlenecks. The average number of ‘hops’ between any two nodes in the network is a measure of network diameter, and typically this rises logarithmically with the number of nodes (n) [5]. Using controlled flood queries, the query execution time across the network is minimized through parallel processing to a linear function of the diameter [19].

The GaianDB also provides several other features that help to support a distributed policy solution. One of these is its capability to define contextual meta-data for data on a
database node using its constant columns feature. For example, a database node may declare constant columns to describe its location and its processing capability. Values for these columns can be updated any time to reflect varying contextual conditions. They are only constant in that they are assigned at a meta-data level, hence minimizing data storage requirements and optimizing any filtering that may be based on them.

The GaianDB provides the capability for a node to cache the rows of its federated data sources in-memory, and additionally to index these rows. This can dramatically improve performance for data retrieval and is best suited to reasonably static data (with infrequent data updates) to minimize the need for synchronization with the underlying data sources.

The current implementation of the GaianDB is based on the Derby database engine [9]. The choice of Derby was dictated by its availability as an ‘open source’ Java relational database that implements a Virtual Table Interface (VTI) [20]. The VTI facilitates connections to data sources other than the Derby tables and has been extended to enable database nodes to self configure into the type of distributed database architecture illustrated in Figure 2.

3) Distributed Policy Implementation

Enforcing global data access policies across this type of distributed database can be achieved using a distributed version of the type of policy architecture described above.

In this case the PEP and PDP components are deployed to all database nodes where policy is to be enforced. Each PDP then has access to a distributed policy repository which uses a local database connection for writes (Store Locally) and a Gaian database connection for reads (Query Anywhere). A Policy Management Tool can be used at any node in the database to create, check and store policies locally, which will then be enforced at all PEPs.

The GaianDB incorporates a ‘plug-in’ mechanism that facilitates the implementation of various types of filters on the JDBC data that flows through the system, including the SQL query that is submitted and the ResultSet structure that is returned as the result to that query [1]. To realize the distributed policy enforcement architecture, we implement the policy management module in the GaianDB as a ‘plug in’ to the VTI, as shown in Figure 3. The PDP and PEP components of the policy module are instantiated in the policy plug-in for each new query that arrives in the node. Then, for each row of the ResultSet that is returned for that query, a policy evaluation is requested based on the contents of that row and the context in which the query has been requested. The result of the policy evaluation might be either to return/drop the row as a partial result to the requestor (in case of an authorization policy), or even modify the contents of some (or all) of the columns of the row (in case of an obligation policy).

To support evaluations and decisions based on the requested data, an object model that describes the elements of the query and its results has been developed. The model provides anchor classes for representing tables, rows and columns, which are fundamental elements of the relational schema. Columns can be referenced in the policy by their name in the query, and the values of their data can be changed on the fly before being returned, according to policy decisions. A generic QueryContext anchor class represents the context in which the query is submitted, which can include arbitrary information such as the name of the user that requested it, her country affiliation and security clearance, or any other type of information that might be considered by the policies.

The policy repository is a Derby database table which is accessed through the VTI as a virtual table and the same query is propagated to all other nodes in the network. Local policies can be accessed from the virtual table by limiting the query propagation to the local database only (i.e. query depth of zero) whilst global policies can be accessed from any (or all) other policy repositories by performing a query of the same virtual table but with an unlimited query depth. The payload of query requests is rather small, so scalability is not compromised due to the flooding of requests across the network. However, the propagation of the returned data

Fig. 2: A typical network of federated gateways (N1,N2…) to heterogeneous data sources.

Fig. 3: Expanded single node view with PEP.
needs to be controlled as the payload that it carries might be significantly larger. Currently, the returned data is routed through the optimal paths as established by the outgoing query and the use of broadcast messages is being further investigated for very low bandwidth networks, to avoid their unicast propagation from node to node. More details on how the flooding of queries is performed can be found in [5].

IV. FEASIBILITY STUDY

The goal of our feasibility study is twofold: first, to assess the processing overhead of applying policy evaluations onto a single federated gateway node, and second, to investigate the scalability properties of a network of gateway nodes performing distributed policy evaluation for networks of various sizes.

A. Evaluation of the Processing Overhead

The performance of WPML is evaluated using the Gaian DB running against a single local database. The workload is a simple loop over policy evaluation against a policy repository containing a single policy that matches the evaluation request and N additional policies which will not be evaluated, but must be examined for matches against the evaluation request. Policy evaluation only requires reading the policies in the policy repository. We compared a memory-based repository with a direct connection to Derby and a GaianDB connection. Figure 4 shows the performance of these configurations, with the processing overhead measured as time required per policy evaluation on the y-axis and the number of policies over which evaluation is required on the x-axis.

WPML provides two levels of caching to enable better performance. The first (denoted as “DB cache”) is an in-memory database cache of policy results. The second (referred to as “PDP cache”) is an in-memory cache of policies matched with evaluation requests. Both require a single read of a timestamp table which is maintained by triggers on the policy table. As we can see in Figure 5, the lower bound (memory) is about 50 µsec/evaluation, and 550 µsec/evaluation and 1500 µsec/evaluation for Derby and the GaianDB, respectively.

B. Evaluation of Scalability

We investigated the query performance of a distributed policy repository across typical GaianDB networks of varying size, in an emulated experiment. GaianDB networks were created on a 16 node blade centre with each blade comprising two dual core processors with up to eight database nodes per core. The blades are connected over a high speed Ethernet adapter with no processor or network contention. The results confirmed that from any single node in the database network, the query time to access and evaluate a single policy is proportional to the number of

![Fig. 4: Expanded Processing overhead (in ms/evaluation) as a function of policies in the repository.](image)

![Fig. 5: Processing overhead for the two different optimizations for policy evaluation implemented.](image)

![Fig. 6: A gateway network of 250 nodes showing the logical topology.](image)
hops to the furthest node.

Figure 6 shows a network configuration of 250 gateway nodes, created using preferential attachment with the maximum number of connections at any single node limited to 10 connections. The average graph diameter of this configuration is 5.9 with a corresponding average query time of 2.8 ms (i.e. 480 usec per hop).

Figure 7 shows the average query time for a simple SQL query as it propagates in networks of varying size (up to 512 emulated nodes on the 16 physical machines of the blade center) using a preferential attachment strategy [5]. The goal is to assess the propagation time of a simple query in the network irrespective of the returned results and the policies that might be enforced on the returned data. The results show the expected logarithmic scaling. The results indicate that, under conditions where network bandwidth is not the bottleneck, GaianDB is highly scalable and with a predictable query time.

In summary, our initial results identified the performance characteristics of both our policy framework and the federated database network. Policies in the network can be evaluated at an extra cost of 80 usec/policy and incur only a constant overhead from the queries to the policy repository. The GaianDB provides single query performance of 1 msec/query on a single node and up to about 3 msec/query on a 256 node network. Based on these results, we can expect an upper bound on performance of 3 msec + 50 usec/row of results, which for a 1000 row result set would require about 53 msec. As the size of the results sets grows, the policy evaluation dominates. As future work we will consider strategies for optimizing policy evaluation.

V. DISCUSSION

Our system for federated, policy-controlled data access showed promising results regarding the flexibility of deployment and performance of data filtering based on expressive policies written in SPL. Certain areas though deem further investigation: for example, in the current incarnation of our system, policies are evaluated against the data model that is described from the query that is submitted as the request for data. This could be further expanded to include the data schema of the sources as well, so as to allow conditional statements for data sharing that might depend on tables, columns and rows that might not be directly included in the SQL of the query. Logical tables provided by the GaianDB and augmentation of the query would provide this extra functionality.

While the information filtering using policies imposes only a small overhead on the delivery of data as we showed in Section IV, a hybrid system that re-writes the query first so as to allow pruning of data at the source [3] would improve the performance of filtering as it would decrease the number of policies that would need to be evaluated on the returned ResultSet. We intend to implement and evaluate this extension in future work.

The system presented in this paper has been designed for enforcement of local policies, which are independent of each other on every node. This does not address the important issue of global policy resolution. Methods for obtaining consistency between network-level policies are the subject of our further research.

An important assumption that was made during the design of the system is that the gateways that participate in the federated network are trusted to collect and enforce the coalition policies. This can be achieved by putting in place authentication mechanisms and encryption of the queries and the returned results that guarantee that rogue nodes cannot be part of the network of federated gateways.

Filtering on data that is exchanged using the JDBC protocol is not the only method for sharing information among coalition forces. Two other popular application-layer protocols, namely XML/SOAP and publish/subscribe systems, could similarly be used for federation and interoperability of coalition members.

Finally, while our preliminary feasibility study showed promising results for our system, we plan to conduct more experiments for gaining further insights into the performance of distributed policy enforcement.

VI. RELATED WORK

As database management systems have become more interconnected and able to seamlessly exchange information using standardized interfaces, privacy concerns have led to the development of various techniques for limiting the amount of information that is being shared. These techniques primarily focus on the modification of the original query that is submitted when data is requested, which is augmented with additional filtering constraints.
before being answered at the data source [3] [16]. While this approach exhibits good performance, it is not as expressive when fine-grained control regarding the filtering of data is required, and requires extensions to the schema of the data source, in order to perform filtering based on the contextual information of the query. It is the latter reason for which we chose to implement an application-layer filtering mechanism through a transparent JDBC proxy.

Data federation and information integration has been extensively studied (e.g. [4] [5] [15]) over the past several years. To the best of our knowledge, only rudimentary operations for data filtering are provided, which do not possess the expressiveness of a full-fledged policy specification language such as SPL [2]. Recently, clustered JDBC proxies [14] have also been proposed for transparent data federation (although without any filtering capabilities). Our work builds and extends this effort.

Recent research efforts in the field of medical informatics and federated healthcare databases [13] as well as XML firewalls [17] explore filtering of data that is shared using the XML language for web services. Although very similar in terms of the end goal with our approach, the emphasis there is more on access control (allow/deny rules) rather than transformation of data based on policies, as enabled by the WPML library of our system. Moreover, they require the use of XML language for information exchange, as policies are also XML-based (XACML or X-GTRBAC).

VII. CONCLUSION

As ISR networks of coalition forces deployed in joint missions need to interoperate and share information amongst them, controlling the flow of this information becomes a task of paramount importance. Control of information is typically expressed through policies that filter and transform the data that is being exchanged. Policies provide an expressive and flexible way for specifying complex conditions that dictate the sharing of data and are therefore amenable to analysis (e.g. for conflict, coverage, and dominance checks), which is a distinguishing feature from other alternatives such as network firewalls.

In this paper we described a novel system that we developed for controlled data sharing through distributed policy enforcement. It features a distributed, federated database that employs a policy management library for evaluating and enforcing policies written in the Simple Policy Language (SPL). Policies can be deployed and enforced on any node of the network of federated gateway nodes, which offers maximum flexibility regarding deployment scenarios. Preliminary experiments of our prototype showed promising results regarding the performance and scalability of our design.

ACKNOWLEDGMENT

This research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the author(s) and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

REFERENCES