# Service-oriented Reasoning Architecture for Resource-Task Assignment in Sensor Networks

Geeth de Mel[a] and Flavio Bergamaschi[b] and Tien Pham[c] and Wamberto Vasconcelos[a] and Tim Norman[a]

[a]Department of Computing Science, University of Aberdeen, Aberdeen AB24 3UE, Scotland
[b]Emerging Technologies, IBM Hursley Park, Winchester, SO21 2JN, United Kingdom
[c]US Army Research Laboratory, 2800 Powder Mill Road, Adelphi, MD 20783-1193 USA

## ABSTRACT

The net-centric ISR/ISTAR networks are expected to play a crucial role in the success of critical tasks such as base perimeter protection, border patrol and so on. To accomplish these tasks in an effective and expedient manner, it is important that these networks have the embedded capabilities to discover, delegate, and gather relevant information in a timely and robust manner. In this paper, we present a system architecture and an implementation that combines a service based reasoning mechanism with a sensor middleware infrastructure so that tasks can be executed efficiently and effectively. A knowledge base, utilising the Semantic Web technologies, provides the foundation for reasoning mechanism that assists users to discover, identify and allocate resources that are made available through the middleware, in order to satisfy the needs of tasks. Once resources are allocated to any given task, they can be accessed, controlled, shared, and their data feeds consumed through the Fabric middleware. We use the semantic descriptions from the knowledge base to annotate the resources (types, capabilities, etc.) in the sensor middleware so that they can be retrieved for reasoning during the discovery and identification phases. The reasoner is implemented as a HTTP web service, with the following characteristics:

1. Computational intensive operations are off-loaded to dedicated nodes, preserving the resources in the ISR/ISTAR networks.
2. HTTP services are accessible through a standard set of APIs irrespective of the reasoner technology used.
3. Support for seamless integration of different reasoners into the system.

**Keywords:** Knowledge Technologies, Semantic Web, Web Services, Reasoning as Services

## 1. INTRODUCTION

The net-centric ISR/ISTAR networks* are expected to play a crucial role in the success of critical tasks such as base perimeter protection, border patrol and so on.[1] This involves assigning appropriate sensing resources to tasks such that assigned resources cover the information needs of the individual tasks. However, effective and efficient assignment of sensing resources to such tasks is a computationally hard problem to solve.[2] This is because only a subset of available sensing resources are suitable to satisfy tasks due to the varying capabilities of sensing resources and requirements of tasks. The difficulty of this problem is amplified in ISR/ISTAR domain, and especially in a coalition context as highlighted in the example below.

Example 1. Consider a hypothetical country in which a civil war has been progressing for decades. Recently, a United Nations backed international peace keeping force is deployed to keep the peace between the authorities in place and militia. The peace keeping force backs the democratically elected current government but due to policy

Further author information: (Send correspondence to G. D. M.)
G. D. M. ⇒E-mail: g.demel@abdn.ac.uk, Telephone: 0044 1224 274174
F.B. ⇒E-mail: flavio@uk.ibm.com, Telephone: 0044 1962 815853

*An intelligence, surveillance, target acquisition, and reconnaissance network is a collection of sensing resources - a sensing resource is a platform which contains one or more sensors.

| Report Documentation Page | *Form Approved* *OMB No. 0704-0188* |
|---|---|

| | | |
|---|---|---|
| 1. REPORT DATE **2011** | 2. REPORT TYPE | 3. DATES COVERED **00-00-2011 to 00-00-2011** |

| | |
|---|---|
| 4. TITLE AND SUBTITLE **Service-oriented Reasoning Architecture For Resource-Task Assignment In Sensor Networks** | 5a. CONTRACT NUMBER |
| | 5b. GRANT NUMBER |
| | 5c. PROGRAM ELEMENT NUMBER |
| 6. AUTHOR(S) | 5d. PROJECT NUMBER |
| | 5e. TASK NUMBER |
| | 5f. WORK UNIT NUMBER |
| 7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **US Army Research Laboratory,2800 Powder Mill Road,Adelphi,MD,20783** | 8. PERFORMING ORGANIZATION REPORT NUMBER |
| 9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) | 10. SPONSOR/MONITOR’S ACRONYM(S) |
| | 11. SPONSOR/MONITOR’S REPORT NUMBER(S) |

12. DISTRIBUTION/AVAILABILITY STATEMENT
**Approved for public release; distribution unlimited**

13. SUPPLEMENTARY NOTES
**Presented during the SPIE Defense, Security, and Sensing Conference, 25-29 April, Orlando, FL**

14. ABSTRACT
**The net-centric ISR/ISTAR networks are expected to play a crucial role in the success of critical tasks such as base perimeter protection, border patrol and so on. To accomplish these tasks in an e ective and expedient manner, it is important that these networks have the embedded capabilities to discover, delegate, and gather relevant information in a timely and robust manner. In this paper, we present a system architecture and an implementation that combines a service based reasoning mechanism with a sensor middleware infrastructure so that tasks can be executed e ciently and e ectively. A knowledge base, utilising the Semantic Web technologies, provides the foundation for reasoning mechanism that assists users to discover, identify and allocate resources that are made available through the middleware, in order to satisfy the needs of tasks. Once resources are allocated to any given task, they can be accessed, controlled, shared, and their data feeds consumed through the Fabric middleware. We use the semantic descriptions from the knowledge base to annotate the resources (types capabilities, etc.) in the sensor middleware so that they can be retrieved for reasoning during the discovery and identi cation phases.**

15. SUBJECT TERMS

| 16. SECURITY CLASSIFICATION OF: | | | 17. LIMITATION OF ABSTRACT | 18. NUMBER OF PAGES | 19a. NAME OF RESPONSIBLE PERSON |
|---|---|---|---|---|---|
| a. REPORT **unclassified** | b. ABSTRACT **unclassified** | c. THIS PAGE **unclassified** | **Same as Report (SAR)** | **13** | |

differences, only a limited set of its resources and information are shared with the authorities. For example, the peace keeping force only provides low-resolution imagery over land to the authorities; this is mainly due to the fact that according to the policies of the countries involved in the peace keeping force, disclosure of high resolution imagery is classified. Therefore, while sharing resources with authorities, the peace keeping force has to respect these policies. Recently a major earthquake has struck the region and humanitarian organisations want to access the region to evacuate people and help the wounded. However, the humanitarian organisations lack means to detect people and transport people to a safe location. The humanitarian organisations want to collaborate with the peace keeping forces to acquire the vehicles but are not willing to use any vehicles with firepower in order to keep its neutral status within the region. Most of the vehicles owned by the peace keeping force are therefore not suitable for the task (i.e., they have got built in fire power capability) and only two helicopters and few trucks are available. The peace keeping force has prior knowledge of a landslide near the region where humanitarian organisations want access to; thus it recommends the use of helicopters to the humanitarian organisations. In order to keep a safe corridor for the humanitarian organisations, the peace keeping force must deploy drones with synthetic aperture radar cameras. This is because the drone can perform high altitude surveillance day and night in any weather conditions without arousing any suspicion.

Considering the above example we can observe the following for such situations as depicted in the Figure 1.

- Multiple tasks compete for a available resources (e.g., detect enemy activity, hostage rescue, boarder surveillance and so on).
- Environments in which these resources are deployed could rapidly change yielding new requirements (e.g., sand storms, thick cloud cover and so on).
- Demand placed on available ISR/ISTAR resources typically exceeds the inventory in coalition operations.[3]
- Coalition members may share resources to improve the total utility of achievable tasks but, there could be policies governing the resources which restrict partners of the coalition deploying some resources or accessing some particular information from resources.

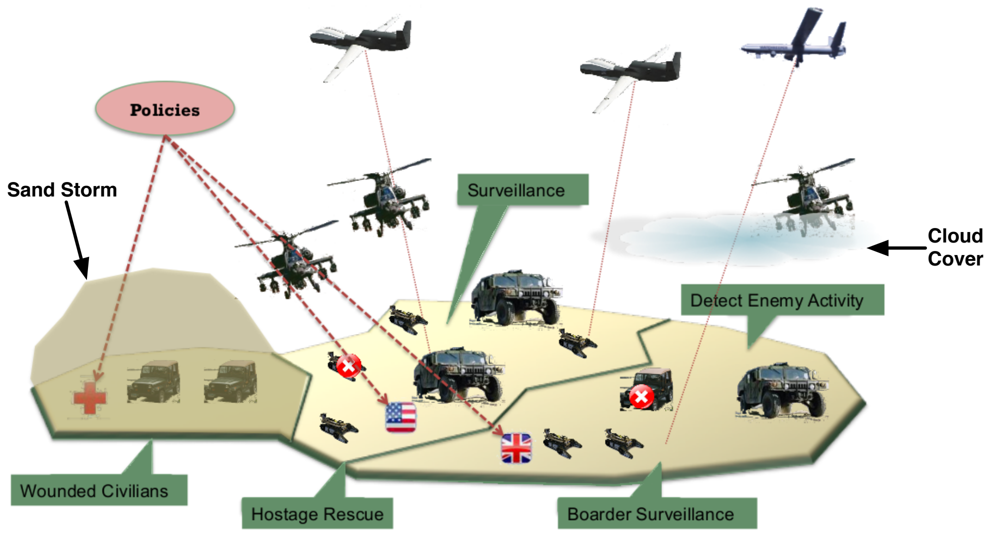

Figure 1. Requirements for Resources-Task Assignment

Therefore, it is important to assign resources for tasks such that the assigned resources are cost-effective (i.e., with regard to their capabilities, performance, etc.) as well as necessary and sufficient to cater for the needs of the tasks. Many communities have investigated different approaches to select resources for tasks and have proposed mechanisms that could be applied to solve the resource selection problem. Some of these approaches rely on having a "human in the loop" to decide which resources are appropriate to satisfy the requirements of tasks[4] whereas other approaches (especially from operations research) have tried to automate the assignment process.[2] It is very difficult for a human to have a bird's-eye view of the available resources for tasks. This is especially true in dynamic environments such as coalition operations. Therefore, having humans in the loop for resource selection in such environments could have a negative effect on the performance of tasks with respect to the duration of the solution discovery process. Operations research approaches tend to focus on the physical aspects of the sensor network such as range, power, bandwidth and so on to optimally select resources for tasks. Due to the lack of context-related information in such mechanisms, important domain specific information such as capabilities required to satisfy the needs of tasks, weather conditions, terrain, policies governing the resources and so on which make the assignment useful are ignored.

The International Technology Alliance in Network and Information Science (ITA)[†] project aims to create next generation net-centric technologies for the US and UK armed forces such that they are given the information advantage in challenging situations such as humanitarian relief, insurgent control, full combat operations and so on. A section of the ITA project has been working on addressing the issues highlighted in the previous section. *Sensor Assignment to Missions*[‡] (SAM) is a knowledge-based framework developed under the ITA to introduce context-related information into the solutions of the assignment problem. Another tool developed under ITA is *ITA Sensor Fabric*[§] (henceforth referred to as the 'Fabric') which allows users to discover, identify, access, control, share, and consume sensing resources robustly. In this document we propose an architecture which integrate these solutions to create a top-to-bottom system which assists users to intelligently specify their requirement, and select and task resources effectively. However, running computationally expensive reasoning processes directly within the sensor networks are not recommended nor practical. Therefore, the proposed architecture uses a service-oriented approach to decouple the core reasoning techniques from the application layer so that light versions of resource-task matching applications could be developed and execute with a remote set of reasoner services with fabric providing up-to-date informational about resources in the field.

The rest of the document is organised as follows. In Section 2 we discuss in detail the technologies we have used to derive our solution. Motivated by the limitations of the technologies described in Section 2, in Section 3 we introduce our integrated solution to address these concerns. We also present and discuss the architecture for the remote reasoning and its functionalities in this section. Section 4 describes an application built using the reasoning services introduced in Section 3 to implement a lightweight resource-task assignment algorithm. We also discuss possible configurations in running this application based on user intentions. We conclude the document in Section 5 with a discussion on the contributions of the proposed approach with respect to some existing work and highlighting the future work.

## 2. RELATED TECHNOLOGIES

Considering the Example 1, it is apparent that in order to address the resource selection for tasks, an intelligent (ideally, an autonomous mechanism) is needed. Such an approach should have the following characteristics:

- Effective knowledge representation (i.e., the language of choice should have efficient decision procedures to compute the results in a timely manner as well as should be based on standards so that the knowledge representation could be reused)
- Sufficient expressive power (i.e., ability to represent complex concepts, rules, and queries in a meaningful way)
- Flexibility in the resource discovery process (i.e., ability to find different solutions for tasks with varying needs and constraints)
- A mechanism to discover, identify, access, control, share, and consume sensing resources robustly.

Below we introduce some research works done under the ITA project to address these issues.

[†]http://wwww.usukita.org/

[‡]http://www.csd.abdn.ac.uk/research/ita/sam

[§]http://www.alphaworks.ibm.com/tech/fabric4sensors

## 2.1 A Knowledge-based Approach to Sensor-Task Assignment

SAM is a knowledge-based framework to compute resource types that are suitable to satisfy the needs of tasks. The core of the framework is a set of interlinking ontologies and rules describing sensors, platforms, capabilities, and constraints. The algorithms developed for SAM use this knowledge to infer a sound and complete set of resource types (i.e., all appropriate resource types) for tasks based on the requirements of those tasks and capabilities provided by the resources . The ontologies for the framework are developed using OWL-DL.[5] An important feature of Description Logic (DL)[6] based languages is that their ability to define concepts in terms of sufficient and necessary conditions. A new concept can be defined by specifying property restrictions and relations on the existing concepts. For example, consider the concept of 'Tactical Unmanned Aerial Vehicle' (TUAV) defined with respect to the existing ontology concepts.

$\Rightarrow$ $AerialVehicle \equiv Platform \sqcap \exists operatesIn.Air$

$\Rightarrow$ $UAV \equiv AerialVehicle \sqcap \forall hasNoCrew.Crew$

$\Rightarrow$ $TUAV \equiv UAV \sqcap \exists providesCapability.(Firepower \sqcup Surveillance)$

The formalism also allows us to use off-the-self reasoners to infer new classifications based on existing definitions. This is very important since it allows us to use same definitions in different contexts. For example consider the following definition of a sensing resource.

$\Rightarrow$ $SensingResource \equiv \exists!\ hasPlatform.Platform \sqcap \exists hasSensor.Sensor$

Using a reasoner we can show that any instance of the concept, sensing resource, be also classified as a platform concept. Furthermore, it allows us to select resources based on different criteria. For example, one might be interested in selecting resources based on the intelligence capabilities provided by the sensing resources while in another case, they might be interested finding resources based on the hybrid capabilities such as NIIRS.[7] Rules attached with SAM framework allows us the capture crucial domain knowledge which otherwise be impossible to represent. For example, to denote the sensing resources which can provide radar NIIRS 5 or above we use the following SWRL[8] rule.

$\Rightarrow$ $needsCapability(?sr,?n) \leftarrow SensingResource(?sr) \wedge hasCapability(?sr,?n) \wedge RadarNIIRS5(?n)$

As the result of the above rule, based on the knowledge we have, we can infer that the following resource set can provide radar NIIRS 5 or above with respect to the ontology: $\{\{GlobalHawk, SAR\}, \{Predator, SAR\}, \{Reaper, SAR\}\}$.

We have defined a query language in de Mel *et al.*[9] which can be used to represent a multitude of sensing requirements and be executed efficiently and effectively through SAM. Consider the example 1 highlighted in Section 1. Imagine a situation in which the local authorities need constant imagery intelligence (IMINT) from the peacekeeping force. The request query would be as follows: $IMINT \wedge ConstantSurveillance$. Upon receiving the request, peacekeeping force sends data back to the authorities while respecting its policies. Remember, they are not allowed to share high resolution imagery with other parties. This notion is captured by the following rule; this again highlights the importance of having rules.

$\Rightarrow$ $requestIntel(?c,?int) \leftarrow Not(CoalitionMember(?c)) \wedge IMINT(?int) \wedge hasQuality(?int,?q) \wedge Low(?q)$

The new version of the SAM uses the Ontology Logic Programming (OLP)[¶][10] to implement the matching algorithms. Developed under the ITA project, OLP allows us to present crucial domain knowledge using standard Semantic Web technology languages and use logic-based programming to reason about this captured knowledge. Current Semantic Web languages such as OWL[5] and SWRL[8] and the reasoners such as Pellet[11] operates under certain restrictions that make them impossible to be used in some cases. For example, due to the open-world assumption in OWL-DL, it is impossible to implement negation by failure which is very important in matching resources to tasks (e.g., find all sensing resources without any firepower capability). The importance of a knowledge-based approach to resource-task assignment (and the accompanying SAM tool) is well published and demonstrated in many internationally renowned conferences.[9, 12, 13]

[¶]http://sourceforge.net/projects/olp-api/

## 2.2 ITA Sensor Fabric

Fabric provides a middleware layer that encapsulates network and security management for resource-constrained networks. The technology mitigates the complexity of managing message flows between coalition partners and across disparate networks. The distributed stream-oriented SOA solutions provided by the fabric simplifies the development and management of sensor network solutions. It addresses the challenges of sensing recourse identification and discovery, access and control, classification and interoperability, information and data sharing, dissemination and consummation, and policy-based interoperability and trust by providing unified access to, and management of, sensor networks. It employs a security model based upon authorisation and obligation policy enforcement leveraging the Policy Management Toolkit[‖].

The Fabric spans the operational network from the command centre to the deployed resources (e.g., sensing resources, services, personnel and so on). It tracks the sensors, nodes, services and the users of a network, and facilitates universal access to resources from any point in the network while maximising its availability and utility to applications, services and users. The Fabric is designed as an extensible platform with a plug-in architecture that allows new functions including services, policies, security, filters, transformations, and event detection algorithms to be deployed directly into the sensor network and selectively applied to the message (information, data, control commands, etc.) flows between the resources and the users.

The Fabric implements a two-way messaging bus and a set of middleware services providing connectivity between all of the networks resources to each other and to users. A typical Fabric node consists of three basic elements: (1) an instance of a message broker[**], (2) an instance of the Fabric Manager and (3) instance of the Fabric Registry. The Fabric Manager is the main service on any given node and leverages a publish/subscribe messaging model with multi-hop capabilities, and ensures that messages propagate efficiently in the network, without duplication, and with the optimal use of bandwidth. It manages all the communication channels between nodes, the routing of messages between nodes, sensing resources and users, and the plug-in container; it also tracks the status of all connected resources , services and users. Information about all nodes, the routes and other ISR/ISTAR resources is recorded in the Fabric Registry. This database leverages the Dynamic Distributed Federated Database[14] and is distributed across each of the Fabric nodes, with each node responsible for managing the information about itself, and the resources and users attached to it. The information recorded in the Registry includes: resource types, physical locations, operational characteristics, neighbours, task commitments, and current operational status/availability. As new resources are added to the Fabric, they are automatically included in the Fabric Registry, making them available to all Fabric users (subject to applicable policies). The Registry also tracks users, tasks, and Fabric extensions (pluggable Fabric functionality).

The technologies highlighted above help us to address two main issues identified for the resource-task assignment problem in sensor networks: (1) effective resource selection for tasks (2) tasking of sensing resources and retrieving sensor data robustly. However, one limitation of the current version of the SAM framework is that, it assumes every resource known to it is available for the assignment (i.e., all resources described in the ontology are already deployed). This is clearly not practical nor feasible; for example, consider a situation where SAM has information about a 'Global Hawk' in its knowledge base, but there exists no 'Global Hawks' in the field. On the other hand, fabric requires users to have prior knowledge about which sensing resources could satisfy their needs. However, in an environment where available resources could rapidly change, it is not prudent to assume that users would always know the best possible resources to use in their tasks. Motivated by these limitations, in the next section, we propose a technology integration to capitalise on the strengths both technologies in order to eliminate limitations of the other technology - i.e., SAM can bring context of operation into the fabric and in turn fabric could provide SAM with a current snapshot of the sensing environment to make the selection appropriate.

## 3. SAM - FABRIC INTEGRATION

The overview of the integration is depicted in Figure 2. In the figure, **R1**. . .**R5** represents sensing resources, and **N1**. . .**N4** are fabric nodes. SAM is proposed to be deployed as an application platform in the fabric so that

[‖]http://www.alphaworks.ibm.com/tech/wpml
[**]http://www.alphaworks.ibm.com/tech/rsmb

it would become an entry point to users to perform intelligent resource selection and tasking. This integration would allow SAM to become more resource aware (i.e., fabric could provide an up-to-date snapshot of the field) and fabric could compensate its lack of context related issues with the functionalities provided by the SAM.

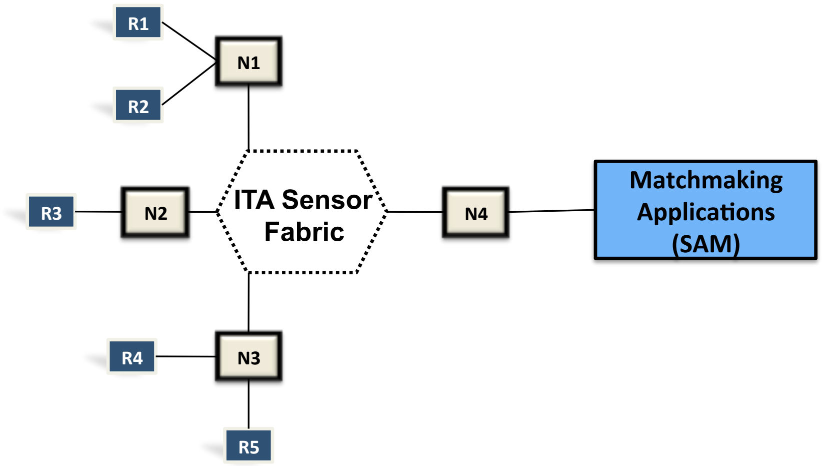

Figure 2. SAM - Fabric Integration

However, as described in Subsection 2.1, SAM uses DL reasoning to compute the appropriate resources for tasks. DL reasoning is a computationally intensive operation; running such expensive reasoning processes directly within low powered sensor networks are not recommended nor practical. Motivated by this issue, we propose an architecture and an implementation which expose the core reasoning capabilities of a reasoner as a set of services in the next subsection.

## 3.1 Reasoning as Services

Exposing the core functionalities of a reasoner as services is an important problem to be solved for many domains. This is due to a variety of reasons:

- Though knowledge-based reasoning has become very efficient with knowledge representation techniques such as OWL-DL, it is still a very computationally intensive operation (CPU, memory, etc.). This makes it impossible to create application which can utilise the power of reasoning in low powered devices such as handhelds.
- There are many reasoners for the Semantic Web implemented in many different languages. For example Pelletis implemented in java whereas Fact++ is implemented using C++. Thus, interacting with such reasoners would require the use of specific languages (java,c++,python) or a great deal of integration efforts (e.g., writing native code).
- Applications want to seamlessly integrate with different reasoners (i.e., reasoning mechanisms are loosely coupled with the applications) based on their needs.

Influenced by the above requirements, we propose the architecture depicted in Figure 3 as means to expose the core functionalities of an arbitrary reasoner as a set of services.

The reasoning as services (henceforth referred to as 'RasS') architecture has two main parts: Server side and client side. They are both implemented as an application programming interface (API) so that they could be easily integrated with other system and extended in an interoperable manner. At the RasS server end, each knowledge base (i.e., $R_iKB_j$, i=1...N, j=1...M) attached with a particular reasoner engine (i.e., $RE_i$, i=1...N) is exposed by a unique URI of the form 'http://host:port/$RE_i$/$R_iKB_j$'. Externally, all these knowledge bases are accessed and manipulated through *RasS Server Interface* using RESTFul[15] requests. We have used a RESTFul approach in this framework due to a variety of reasons:

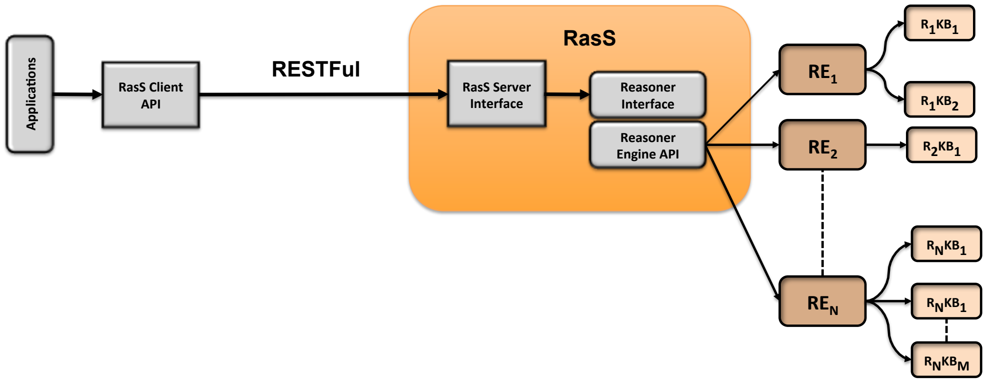

Figure 3. RasS Architecture

- It is architecturally well understood; the whole web is based on it.
- Complements the latest and well researched knowledge representation techniques supported under Semantic Web vision (i.e., each and every resource is uniquely identified by a URI).
- RESTFul services are lightweight and easy to use and implement compared to SOAP-based[16] services.

Internally, these requests are processed by the appropriate *Reasoner Engine API* that implements the *Reasoner Interface*. This interface-based approach allows seamless integration of new reasoners the architecture; this is because, one would only need to implement the interface to add a new reasoner at the back-end. At the moment the reasoning service framework support two reasoners (Pellet[11] and OLP[10]) through the uniform interface. The *Reasoner Engine API* utilisers APIs such as OLP API,[10] OWL API[17] and so on to manipulate the knowledge bases.

When the RasS server receives RESTFul requests, it maps each and every request to an appropriate reasoner resource. The appropriate reasoner resource is picked based on the unique reasoner ID (i.e., RE$_i$, i=1…N) and knowledge based ID (i.e., R$_i$KB$_j$, i=1…N, j=1…M) attached with the request. These reasoning resources support all the core reasoning techniques (e.g., classification of concepts, consistency checking of a knowledge base, subsumption between concepts, identifying specific instances of a concept, details description of an individual, etc.,) supported by the exposed reasoner. Due to the uniform interface approach we have taken in this architecture, it allows users to formulate similar requests to multiple knowledge bases by simply changing the ID of the knowledge-base. This allows users to easily compare and contrast the results of different reasoners. Once the request is executed, RasS server responses to the requestor with a packet that contains information such as success or failure of the request, matching values to the requests, instantiated values for the query variables and so on.

The *RasS Client API* helps users to formulate requests based on a set of APIs that are easy to use. All the core reasoning techniques of a desired reasoner service could be accessed using a higher-level programming language. Furthermore, we have made provision to users to create and modify knowledge bases using the same API so that if the required knowledge base does not exists, users could create it easily.

The architecture proposed here does not allow modification of instances, concepts, or properties of existing knowledge bases; it is targeted purely to be a knowledge inferring system (i.e., reasoning, querying, etc.). If the knowledge needs changing, users need to update the knowledge externally, and refresh the appropriate knowledge-bases identified by the reasoner resource ID. Though the knowledge management techniques could easily be introduced into the architecture, authors believe that by doing so would unnecessarily increase the complexity of the system. By targeting the architecture to be reasoning only, we could examine interesting aspects of reasoning such as query answering, different reasoning paradigms, optimised searching of knowledge-bases and so on. In the next subsection we discuss the functionalities of RasS with examples.

## 3.2 Functionality

RasS supports ontology-based reasoning, first-order-logic reasoning, and query answering in its current form. Query answering is limited to SPARQL[18] and first-order-logic queries. Below we discuss the core functionalities of RasS with examples. In order to generate requests to the RasS server, we use cURL[††] in this section.

**Create and update knowledge bases:** A RasS knowledge base can be created by import a number of ontologies or other knowledge bases (e.g., Prolog[19] knolewdge bases) into a unique resource ID under a particular reasoner. For example, the following cURL request creates a reasoner resource 'host:port/RE$_i$/R$_i$KB$_j$' using the knowledge from files $f_1 \ldots f_o$ and reasoning engine RE$_i$.

```
curl -sv -X POST -HContent-type:application/xml --data "<reasoner><id>R_iKB_j</id><uri>f1,...,f_o
</uri></reasoner>" host:port/RE_i
```

One can update this reasoning resource by substituting the POST directive of the above cURL request with PUT directive with appropriate references to the updated knowledge. The consistency checking of knowledge bases are done while creating them. Therefore, in the case of an inconsistent knowledge, the response to a request could end up being `false`.

**Concept checking:** The notions such as subclasses and superclasses of a concept, subsumption and equivalence between concepts are performed easily. For example, in order to check whether an ISTAR 'Global Hawk' is included under the concept 'High Altitude Long Endurance UAV' (HALE), one could send a request to the ISTAR knowledge base as follows.

```
curl -sv -X GET host:port/RE_i/ISTAR/classes/HALE/subsumes/Global_Hawk
```

In order to check equivalence of the two concepts, one simply substitute the `subsumes` in the above request with `eqv`.

**Instance checking:** The direct and indirect instances of concepts are easily discovered. For example, non-direct individuals of the ISTAR concept UAV is discovered from the response to the request

```
curl -sv -X GET host:port/RE_i/ISTAR/individuals/UAV/nondirect
```

Also, the properties of a particular individual could be found by sending request as follows; it request for all the capabilities provided a particular 'Global Hawk' instance. This is very important in identifying appropriate resources for a task.

```
curl -sv -X GET host:port/RE_i/ISTAR/individuals/Global_Hawk_PI/property/providesCapability
```

**Query answering:** RasS in its current form supports SPARQL[18] and OLP[10] query formalisms. For example, an ISTAR sensing resource that provides 'High Altitude' capability could be queried from an OLP knowledge base using the following request with the query `Q`.

```
curl -sv -X POST -HContent-type:application/xml --data "<queries><query>Q</query></queries>"
host:port/olp/istar/query
Q = istar:'S_Resource'(X),istar:'providesCapability'(X,istar:'HighAltitudeCapability').
```

In the next section we describe an application that uses these reasoning service, SAM matchmaking algorithm, and fabric to perform resource-task assignment in an intelligent and resource aware manner.

[††]cURL is a suit of libraries and tools to transfer data using various protocols - http://curl.haxx.se/

| Sensing Resource | Platform | Sensors | Node Affiliation |
|---|---|---|---|
| SR1 | Harrier_GR9 | EOCamera,IRCamera | node1 |
| SR2 | Global_Hawk | EOCamera, IRCamera, SARCamera | node2 |
| SR3 | Predator_A | LDRFCamera, SARCamera, TVCamera | node3 |

Table 1. Sensing resources and their node affiliations

## 4. CASE-STUDY

In order to demonstrate the use of RasS in a real world situation, we have have created an application platform on the fabric which helps users to perform intelligent recourse selection for their tasks. The whole system is simulated in a MacBook with 1.8 GHz Intel Core 2 Duo processor and 2GB of RAM.

As shown in Figure 4, a US owned sensor network is deployed using the fabric. This particular instance of the fabric manages three sensing resources which in turn report data to the fabric through relevant nodes. The Table 1 shows the sensing recourses, the platforms and sensors attached with those sensing resources, and the ID of the node to which the sensing resource reports. Platform and sensor descriptions in the fabric's tables are semantically annotated the using the ISTAR ontology*. This enables us to retrieve sensing resources information from the fabric and reason about them with an ISTAR reasoning resource, since each and every resource is identified with an ISTAR concept. We have deployed a RasS server and registered that with the fabric through node 4. We could deploy multiple instances of the RasS server with different capabilities and utilise them in a multitude of applications. Discovery of these reasoner services is straight forward using the fabric's discovery capabilities.

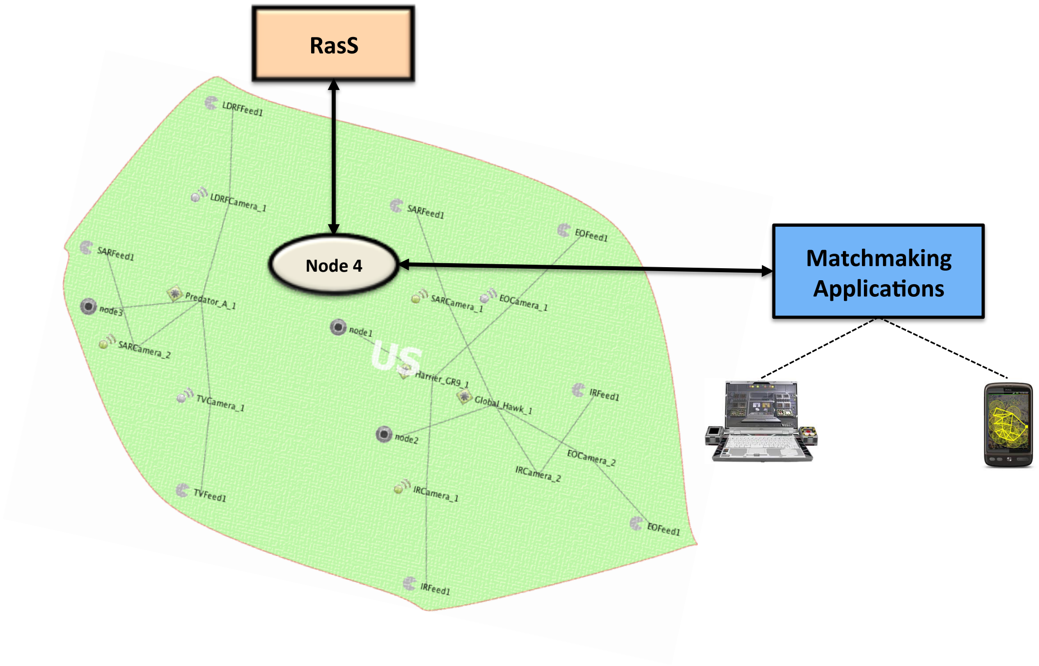

Figure 4. Resource-Task Assignment Application based on RasS

A lightweight application has been created where users could specify their requirements adhering to the query language defined in.[9] We execute this application in two different settings.

**Selection based on deployed resources:** In this setting, matchmaking application first retrieved the deployed sensing resources (i.e., SR1,SR2, and SR3) from the fabric. We have implemented a set of APIs to

*http://www.csd.abdn.ac.uk/research/ita/sam/downloads/ontology/ISTAR.owl

| Sensing Resource | Platform | Sensors |
|---|---|---|
| SR4 | Nimrod_MR2 | LDRFCamera, SARCamera, TVCamera |
| SR5 | WASP | EOCamera |
| SR6 | WASP | IRCamera |
| SR7 | Hunter | IRCamera, EOCamera |
| SR5 | Shadow_200 | LDRFCamera, IRCamera, TVCamera |

Table 2. Sensing resources available in the theatre

assists this process so that multitude of applications could easily access the fabric. The retrieved records contain information such as unique IDs of the resources, their types, and references to the ISTAR ontology concepts. The application then used these ontology references to check whether each and every resources deployed in the fabric is capable of satisfying the user query. This was done by accessing an ISTAR reasoner service and using subsumption and property checking capabilities of the service. The types of the resources that satisfied the user query were recorded and proposed as solutions to the user. Below we provide two example queries we ran.

**Find available resources with firepower capability:** The proposed solutions were SR1 and SR2. This is because, both of them have built-in firepower whereas 'Global Hawk' has no firepower (i.e., consistent with the ISTAR knowledge in the reasoner)

**Find available resources with firepower and high altitude capability:** The proposed solution was SR1. This is because SR3 is a 'Predator' resource and only provides medium altitude capability.

**Selection based on available resources:** In this case, not all resources were deployed in the field, i.e., some are in theatre. However, these resources are registered with the fabric. The Table 2 shows these resources. These resources could play an important role while planning future missions. This is because, even though these resources were not available be on the field, they are ready to be deployed. In this setting, matchmaking application retrieved deployed resources as well as resources that were available in the theatre using the same API mentioned above. Below we provide two example queries we ran and the solutions obtained.

**Gather intelligence discreetly:** This implied that we would need high altitude surveillance. For this query, the matching application proposed SR1 and SR3 from the deployed set as well as SR4 from the resources in the theatre. This is because according to the knowledge available to the ISTAR reasoner service, a 'Nimrod' could perform high altitude surveillance.

**Gather intelligence discreetly in cloudy conditions:** The matching application only proposed SR3 and SR4. This is because only they provide the 'synthetic aperture radar' capability which is needed for cloud penetration.

## 5. RELATED WORK & CONCLUSION

Web services help in creating interoperable systems and assist in code reuse. However, they are rarely described formally, which hinders automatic discovery of services. In order to address this issue, semantic web services are proposed.[20] Semantic web services are web services that are annotated with semantic descriptions so that services could be matched against requests, compose new services, and so on.[21] There is also a sizeable amount of literature on automating the service discovery based of the user quires (i.e., matching queries to services).[22, 23]

Despite the ever-growing number of available reasoners for the Semantic Web, there is very little work done to expose the core reasoning mechanisms (i.e., classification of concepts, subsumptions, etc) of these reasoners as services. DIG specification[24] is the first of its kind to examine the feasibility to opening knowledge management and reasoning as remote services. However, this is now been deprecated, thus, does no support any new OWL constructs or technologies (e..g, SPARQL,[18] rules, etc.,). The successor to the DIG specification is OWLLink protocol.[25] Though this specification looks very promising, it is still very much at the specification stage and does not support many technologies such as SWRL in its implementation. Also, it aims at providing a protocol for OWL knowledge and reasoner management, thus the implementations could become complex and bulky to execute. PelletServer[26] is an alpha version of the popular OWL-DL reasoner Pellet as a RESTFul interface. Even though it addresses most of the concerns identified for other remote reasoner services approaches, it lacks in terms of the reasoner support. It only supports Pellet, thus, makes it impossible to tryout other reasoners and techniques (i.e., closed world reasoning).

Currently we are working on automating the reasoner services to query matching based on many criteria: capabilities of the services, ownership, policies governing the services, user intensions. We use abstract interpretation techniques to capture user intentions.[27] Further extension to the query-service matching could be done by incorporating physical properties of the reasoner services and their environments such as CPU, Memory use and so on but is out of the scope of this work. In this paper we have highlighted the need for a knowledge-based approach for resource-task selection to make to assignment effective. We have also shown the importance of a framework that supports discovery, identification, and tasking of sensing resources in a robust manner. In order to increase the interoperability, reuse, and to reduce the strain of expensive reasoning processes, we have proposed an architecture and an implementation that exposes the core reasoning techniques as RESTFul services. An application is built based on the reasoner services framework and we have discussed with example results the possible use-cases of this application.

## ACKNOWLEDGMENTS

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

[1] Pearson, G., "A Vision of Network-Centric ISTAR and the Resulting Challenges," in [*Unattended Ground, Sea, and Air Sensor Technologies and Applications X*], Carapezza, E. M., ed., **6963**(1) (2008). Available from: http://link.aip.org/link/?PSI/6963/696302/1.

[2] Johnson, M. P., Rowaihy, H., Pizzocaroz, D., Bar-Noy, A., Chalmers, S., Porta, T. L., and Preece, A., "Frugal Sensor Assignment," in [*4th IEEE International Conference on Distributed Computing in Sensor Systems*], (June 2008).

[3] Joint Publication, "JP 2-01: Joint and National Intelligence Support to Military Operations," *Intelligence, Series 2-0 Publications* **2** (2004). Available from: http://www.dtic.mil/doctrine/jpintelligenceseriespubs.htm.

[4] Botts, M., Percivall, G., Reed, C., and Davidson, J., "OGC Sensor Web Enablement: Overview And High Level Architecture," tech. rep., Open Geospatial Consortium Inc (December 2007).

[5] Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P. F., and Rudolph, S., eds., [*OWL 2 Web Ontology Language: Primer*], W3C Recommendation (27 October 2009). Available from: http://www.w3.org/TR/owl2-primer.

[6] Baader, F., Horrocks, I., and Sattler, U., "Description Logics," in [*Handbook of Knowledge Representation*], van Harmelen, F., Lifschitz, V., and Porter, B., eds., ch. 3, Elsevier (2007).

[7] Irvine, J. M., [*National Imagery Interpretability Rating Scales (NIIRS)*], 1442–1456, Marcel Dekker (Oct. 2003).

[8] O'Connor, M., Knublauch, H., Tu, S., Grosof, B., Dean, M., Grosso, W., , and Musen, M., "Supporting Rule System Interoperability on the Semantic Web with SWRL," in [*The Semantic Web ISWC 2005*], *Lecture Notes in Computer Science* **3729/2005**, 974–986, Springer Berlin / Heidelberg (Oct. 2005).

[9] G de Mel and W Vasconcelos and T Norman, "Intelligent Resource Selection For Sensor-Task Assignment: A Knowledge Based Approach," in [*In Proceedings of International Conference on Advanced Topics in Artificial Intelligence (ATAI 2010)*], Global Science and Technology Forum (2010). Best Research Student Paper.

[10] Sensoy, M., de Mel, G., Vasconcelos, W., and Norman, T., "Ontological Logic Programming," in [*In Proceedings of International Conference on Web Intelligence, Mining and Semantics (WIMS11)*], ACM, Sogndal, Norway (May 2011).

[11] Sirin, E., Parsia, B., Grau, B., Kalyanpur, A., and Katz, Y., "Pellet: A practical OWL-DL reasoner," *Web Semantics: Science, Services and Agents on the World Wide Web* **5**(2), 51–53 (2007).

[12] de Mel, G., Sensoy, M., Vasconcelos, W., and Preece, A., "Flexible Resource Assignment in Sensor Networks: A Hybrid Reasoning Approach," in [*Proceedings of the 1st International Workshop on the Semantic Sensor Web*], (2009).

[13] Preece, A., Gomez, M., de Mel, G., Vasconcelos, W., Sleeman, D., Colley, S., Pearson, G., Pham, T., and Porta, T. L., "Matching Sensors to Missions using a Knowledge-Based Approach," in [*Proceedings of SPIE Defense Transformation and Net-Centric Systems 2008*], (to appear, mar 2008).

[14] Bent, G., Dantressangle, P., Vyvyan, D., and Mitsou, V., "A Dynamic Distributed Federated Database," in [*Proceedings of 2nd Annual Conference of ITA*], (September 2008).

[15] Richardson, L. and Ruby, S., [*Restful Web Services*], O'Reilly Media, 1 ed. (May 2007).

[16] Weerawarana, S., Curbera, F., Leymann, F., Storey, T., and Ferguson, D. F., [*Web Services Platform Architecture : SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging, and More*], Prentice Hall PTR (Mar. 2005).

[17] Horridge, M. and Bechhofer, S., "The OWL API: A Java API for Working with OWL 2 Ontologies," in [*OWLED*], Hoekstra, R. and Patel-Schneider, P. F., eds., *CEUR Workshop Proceedings* **529**, CEUR-WS.org (2008).

[18] Prud'hommeaux, E., , and Seaborne, A., "SPARQL Query Language for RDF," tech. rep., W3C (2006).

[19] Clocksin, W. F. and Mellish, C. S., [*Programming in Prolog: Using the ISO Standard*], Springer, 5 ed. (2003).

[20] McIlraith, S., Son, T. C., , and Zeng, H., "Semantic Web Services," in [*IEEE Intelligent Systems*], D Fensel and M Musen, ed., *Special Issue on the Semantic Web* **16**, 46–53, IEEE (March/April 2001).

[21] Paolucci, M., Kawamura, T., Payne, T. R., and Sycara, K. P., "Semantic Matching of Web Services Capabilities," in [*ISWC '02: Proceedings of the First International Semantic Web Conference on The Semantic Web*], 333–347, Springer-Verlag, London, UK (2002).

[22] Wang, H. and Zhai, S., "Query for Semantic Web Services Using SPARQL-DL. ," in [*In 2nd International Symposium on Knowledge Acquisition and Modeling*], (2009).

[23] Bellur, U. and Kulkarni, R., "Improved Matchmaking Algorithm for Semantic Web Services Based on Bipartite Graph Matching," in [*In IEEE International Conference on Web Services (ICWS 2007)*], (2007).

[24] Bechhofer, S., "The DIG Description Logic Interface: DIG/1.1," in [*In Proceedings of the 2003 Description Logic Workshop*], (2003).

[25] Liebig, T., Luther, M., and Noppens, O., "The OWLlink protocol: Infrastructure for interfacing and managing OWL 2 reasoning systems," in [*In 6th OWL Experienced and Directions Workshop (OWLED 2009)*], **529** (October 2009).

[26] Bulka, B. and Sirin, E., "PelletServer: HTTP & OWL2 Reasoning," in [*In 7th OWL Experienced and Directions Workshop (OWLED 2010)*], (2010).

[27] Hentenryck, P. V., Degimbe, O., Charlier, B. L., and Michel, L., "The Impact of Granularity in Abstract Interpretation of Prolog," in [*International Workshop on Static Analysis (WSA-93)*], (September 1993). (Invited Paper).