Resource Brokering Service: Timely and Efficient Information Resource Allocation[*]

Daniel J. Van Hook[†a], Magnus Ljungberg[a], Robert Shaw[a], Mark Ford[a], Ethan Aubin[a],
Eric Konieczny[b], Daniel H. Lee[b], Samuel T. Brown IV[b]
[a]MIT Lincoln Laboratory, 244 Wood Street, Lexington, MA 02420
[b]Booz Allen Hamilton, 8283 Greensboro Drive, McLean, VA 22124

ABSTRACT

We address supporting unanticipated users and uses of limited information resources (sensors, databases, weapons - any resource intrinsically tied to digital information) in a timely and efficient fashion. Platform-centric systems often preclude users and uses not identified when the system was developed and deployed. Net-centric approaches, however, can address these problems by allowing services and information to be discovered and accessed at run-time. We have developed a resource brokering service that uses net-centric principles and semantic metadata to enable multi-domain information and resource sharing and support for unanticipated users and uses. The resource brokering service uses federated brokering agents and a modular software component framework for dynamically composing and tasking heterogeneous resources including sensors, data feeds, processors, archived data, networks, and even analysts into resilient, mission-oriented workflows. The resource brokering service is applicable to multiple sense-decide-act military domains including missile defense, space situation awareness, ISR, border protection, and cyber defense. In this paper we present a concept and architecture for resource brokering and describe current applications. Our architecture is aligned with the U.S. DoD's NCES (Net-Centric Enterprise Services).

Keywords: Net-centric, architecture, NCES, tasking, agents, planning, brokering, semantic

1. INTRODUCTION

Net-centric[1] approaches to military and other government sense-decide-act systems hold great promise for improving effectiveness, efficiency, and a number of other factors. Net-centric operations seeks to translate an information advantage enabled in part by information technology into a competitive warfighting advantage through robust networking of well-informed geographically dispersed forces[2]. An important aspect of net-centric systems is sharing of sensors, processors, and other information resources. Net-centric approaches can enable unanticipated users and uses of limited information resources through run-time service and information discovery and binding.

An important consideration in net-centric systems is timely and efficient resource allocation. Figure 1 is a conceptual graph that addresses complexity of time-constrained dynamic resource allocation. This Figure compares the number of resources and task processing steps required under time constraints by several U.S. DoD and other Government applications. In this context, resources include sensors, data feeds, processors, platforms, communications, weapons, and even human analysts. The lower left corner of this Figure depicts notional bounds on the ability of most humans and of expert operators to successfully administer complex resource allocation tasks under time constraints. Under time constraints, even the best human operators generally cannot manage complex dynamic resource allocation for many important sense-decide-act applications.

The complexity of dynamic resource allocation under time-constraints is a large contributor to the stove-piped nature of many deployed systems. As depicted in the left side of Figure 2, stove-piped architectures result in tightly coupled sensing, communications, processing, decision-making, and weapons functions that are locked within individual domains. In stove-piped systems, resources are allocated at design time or upon deployment. Stove-piped systems cannot

[*] This work is sponsored by the United States Air Force under Air Force Contract # FA8721-05-C-0002. Opinions, interpretations, recommendations and conclusions are those of the authors and are not necessarily endorsed by the United States Government.
[†] Contact author: Daniel J. Van Hook, Email: DVanHook@LL.MIT.EDU Phone: 781-981-4139

Defense Transformation and Net-Centric Systems 2010, edited by Raja Suresh, Proc. of SPIE
Vol. 7707, 77070K · © 2010 SPIE · CCC code: 0277-786X/10/$18 · doi: 10.1117/12.850588

be easily employed for missions other than those originally envisioned. Under-utilized sensors cannot be dynamically reallocated to support other pressing missions. New capabilities cannot be dynamically composed from functions available inside different systems to meet new threats and adversaries. Data is locked up within stove-pipes. Supporting unexpected users and uses in a timely and efficient fashion is impossible or at best difficult. Stove-piped systems are globally inefficient with respect to resource utilization and are inflexible. These are clearly undesirable attributes given agile adversaries, rapidly evolving threats, disruptive technologies, and an increasingly constrained budgetary environment. In contrast, net-centric systems enable dynamic allocation and flexible interconnection of information resources to meet changing mission needs, as implied by the information bus depicted on the right side of Figure 2.

While stove-piped systems are clearly undesirable they offer important guarantees on performance, security, resilience, and resource availability. The challenge for net-centric systems is maintaining these kinds of guarantees while also achieving timely and efficient resource allocation. This paper presents our approach to this problem.

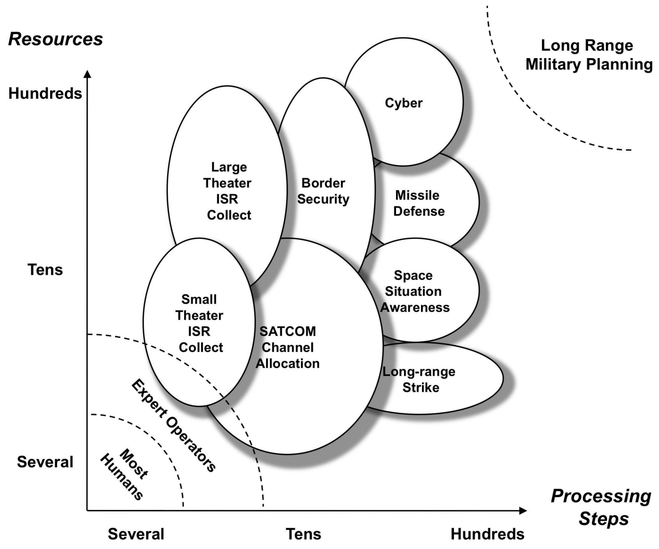

Figure 1. Time-constrained dynamic resource allocation for important U.S. DoD and Government applications is mostly beyond the abilities of even expert human operators. This is a large contributor to the stove-piped nature of many systems.

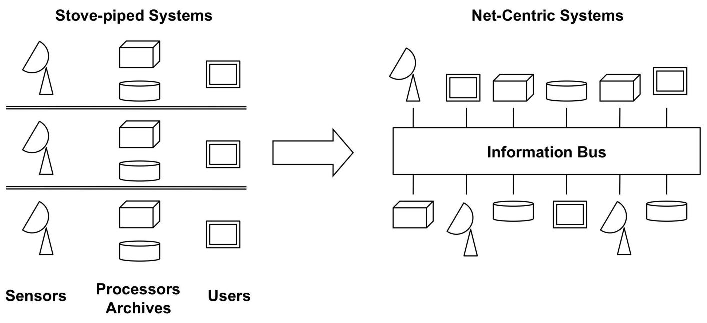

Figure 2. Stove-piped systems limit information resource sharing. In contrast, net-centric systems facilitate sharing with benefits that include improved efficiency, utilization, and effectiveness. Timely and efficient information resource allocation is an enabler for net-centric systems.

This paper is organized as follows. In Section 2 we give an overview of several important concepts related to the resource brokering approach. In Section 3 we provide a detailed description of our resource brokering software architecture. Then, in Section 4, we describe two applications in which we are applying our resource brokering architecture. In Section 5 we list several target areas that we think are important for further exploration that are pertinent to future development of the resource brokering concept and architecture. Finally, we finish up with a summary and references.

2. RESOURCE BROKERING CONCEPTS

The top-level goal of our resource brokering approach for net-centric systems is timely and efficient information resource allocation and tasking. In this section we present several key concepts including:

- Decoupling of information consumers from information providers
- Dynamic composition of information resources into processing chains based on users' information needs
- Direct data flows from resources to information consumers
- Semantic resource descriptions that support automated reasoning
- Federated architecture

Important metrics for evaluating our resource brokering concept include planning latency, planning success rate, tasking latency, overhead (e.g. messaging), processing and memory usage, time-to-failure, time-to-recovery, resilience in the face of cyber attacks, and scaling behavior with number of resource brokers, consumers, resources, etc.

2.1 Decoupling

The concept of decoupling information consumers from the resources that provide information is illustrated in Figure 3. The left side of Figure 3 shows information consumers coupled directly to dedicated information resources. To get information, consumers must have a priori knowledge of relevant information resources and are responsible for tasking resources directly. For example, within the ISR domain, an intelligence analyst searching for imagery of a specific location must manually find the best available airborne platform capable of collecting the needed data and task this resource by tail number. In contrast, the right side of Figure 3 shows information consumers decoupled from information resources. Rather than directly tasking resources, information consumers instead request information from a resource broker. The resource broker in turn selects an appropriate resource from the available resources and tasks it on behalf of the consumer. The resource provides data that satisfies the information consumer's request.

The idea of a resource broker is analogous to the idea of brokering agents from everyday life such as real-estate brokers, stock brokers, mortgage brokers, etc. All of these agents mediate between parties, usually a buyer and a seller. Within the context of distributed DoD and other government systems, a resource broker is an agent that mediates between a consumer of information and resources that provide information.

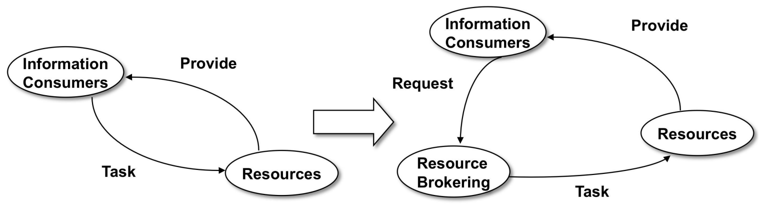

Figure 3. Decoupling information consumers from the resources that provide information enables resource sharing, increased utilization, opportunities for optimization, and independent evolution.

Decoupling information consumers from the resources that provide information via a resource brokering service offers a number of benefits including:

- Sharing. Resources can be shared across missions and used by more than one information consumer. This can improve timeliness, quality, and diversity of data and services available.
- Improved utilization. Idle resources can be tasked to satisfy users requests for information.
- Optimization. The best resource to satisfy a request can be selected.
- Independent evolution. Information consumers and resources can be upgraded and deployed independently.
- Pre-emption. Resources can be pre-empted to service higher priority requests (subject to policy in force) in a more orderly fashion.

2.2 Dynamic composition

In our approach a resource broker dynamically composes processing chains (i.e. workflows) from resources to satisfy consumers' information requests. This concept is depicted in Figure 4, which builds upon the illustration in Figure 3. In Figure 4 the resource broker discovers an appropriate set of resources, constructs a resource processing chain, tasks the resources, and then monitors the execution of the processing chain to detect and recover from faults. The information requested is provided to the consumer. Processing chains consist of one or more resources. When the request has been satisfied the processing chain is torn down and the resources are released. Artificial Intelligence planning algorithms, automated reasoning, and semantic metadata descriptions of resource capabilities are fundamental to dynamic composition.

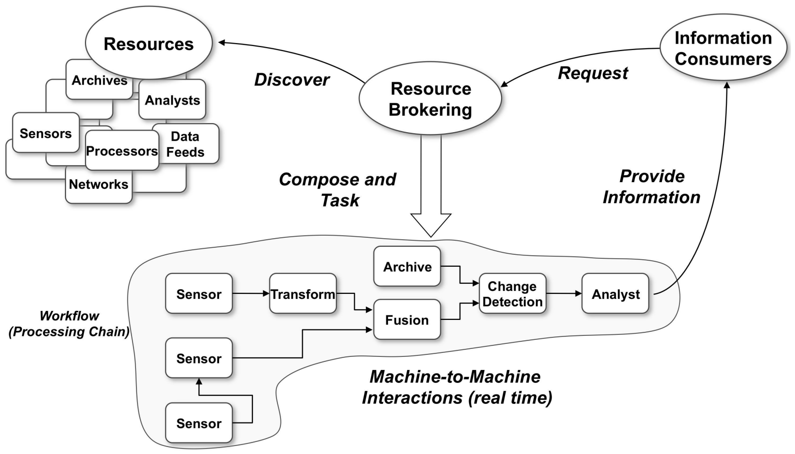

Figure 4. A resource broker dynamically discovers information resources, composes them into processing chains, and tasks them to satisfy consumers' information requests. Information flows in real-time from resources to consumers without intervention by the resource broker. The resource broker monitors execution to detect and recover from faults.

2.3 Direct data flows

In our approach, data flows in real-time directly between resources and from resources to information consumers as we depict in Figure 4. A resource broker does not relay data. A resource broker operates as a control plane alongside the resources rather than in line with the resources. Performance is determined by the underlying networks and protocols and is not limited by the resource brokering service.

2.4 Semantic resource descriptions

Semantic metadata descriptions of resources are central to the resource brokering concept. Metadata descriptions capture resource capabilities in an unambiguous, machine-processable form. A resource broker leverages resource metadata descriptions in conjunction with semantic reasoning techniques to accurately select and compose resources into processing chains to satisfy consumers' information requests. Examples of resource metadata include data types, data formats, protocols, policy specifications, process models, tasking mechanisms, processing and memory requirements, physical characteristics, and limitations.

In the resource brokering architecture, resource capabilities are decoupled from the resource broker. Knowledge is represented in the resource metadata only and is not encoded in resource brokering software. This architecture characteristic allows new resources with accompanying metadata descriptions to be introduced into the system without modifying the resource brokering software.

By exploiting semantic resource metadata we gain significant improvements and new functionality that would be considerably harder to achieve by other methods. These benefits include:

- Improved resource discovery through automated reasoning about resource metadata, e.g. exploiting sub-class, equivalence, transitivity, and other relationships
- Enhanced integration across domains through mapping ontologies that capture inter-domain relationships represented by different domain-specific ontologies

- Independent evolution by allowing new knowledge to be integrated without impacting existing knowledge
- Automated verification by providing a means to capture relevant performance, process model, and interface knowledge in a machine-processable format

2.5 Federated architecture

Resource brokers can be federated to realize cross-domain information requests. Figure 5 shows a federation of resource brokers with each broker responsible for a set of resources. Resource brokers collaborate using the federation mechanism to share resources, formulate plans, and execute plans. We have evaluated a range of federation mechanisms including blackboard, peer-to-peer, client-server, and various hierarchical schemes using simulation. Our current preferred approach is a blackboard[3]. We are preparing a paper for future publication describing our results in this area.

Resources are partitioned into domains with a resource broker per domain. The resource brokering concept does not specify particular criteria for partitioning resources into domains. Criteria include resource type, policy considerations, security requirements, geography, command hierarchy, or other administrative boundary. For example, resources ma be partitioned by service alignment (e.g. Navy, Army, Marines, Air Force), geographic location (e.g. Europe, Mideast, Southeast Asia) or by resource type (e.g. radars, cameras, ELINT, processors, networks, etc.). Criteria can of course be combined (e.g. radars operated by the Army in the Mideast).

A federated approach has a number of benefits including:

- Improved scalability
- Support for local administrative control and policy specification/enforcement
- Ability to flexibly tailor resource brokering algorithms to a specific resources and domain requirements
- Independent evolution of different domains
- Increased resilience against faults, both stochastic and malicious, through redundancy

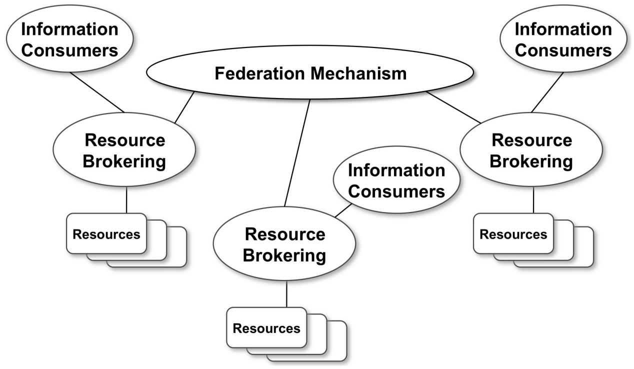

Figure 5. Resource brokers are federated and collaborate to allocate and task resources and compose workflows across mission domains. Federation enhances scalability and supports tailoring to the requirements of each domain including per-domain administration and policy enforcement. Domains are flexibly defined to address administrative, policy, security, and performance requirements.

3. RESOURCE BROKERING SOFTWARE ARCHITECTURE

In this section we describe our resource brokering software architecture, which is depicted in Figure 6. This architecture and its implementation embody the resource brokering concepts that we outlined in the previous section.

The resource brokering software architecture is a framework of replaceable components with fixed interfaces. Resource broker components communicate with each other through SOAP/WSDL web services (Simple Object Access Protocol[4]/Web Services Description Language[5]). SOAP/WSDL was selected as the primary mechanism for inter-component communication due to its widespread adoption, support form standards bodies, and compliance with NCES-style service security (Net-Centric Enterprise Services[6]). The architecture can easily support additional interface protocols such as REST (Representational State Transfer[7]) though we have not yet extended the implementation in this way.

The resource brokering software architecture supports pluggable domain-specific component implementations. We anticipate that domain-specific implementations may be needed. For example, a planning service (see Figure 6) for network resources and a planning service for sensor resources will almost certainly employ different algorithms. They will, however, support the same external interfaces. This approach enables federated interoperation across domains while facilitating independent evolution and deployment. This is critical to support the emerging loosely-coupled net-centric paradigm.

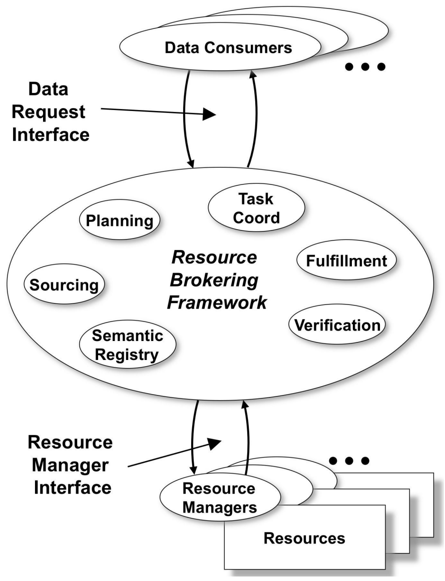

Figure 6. The resource brokering software architecture diagram showing key interfaces and components. The architecture is a framework of replaceable components with fixed interfaces that may be federated. Industry standard interfaces and protocols are used throughout.

In the following sections we describe the primary interfaces and components in the resource brokering software architecture that is depicted in Figure 6. In particular, we discuss:

- Resource manager interface
- Data request interface
- Task coordination service
- Planning service
- Resource registry
- Sourcing service
- Verification and assessment service
- Fulfillment service

3.1 Resource manager interface

Resources utilized within the resource brokering framework are in general heterogeneous: they can have different management and control interfaces. In order to accommodate this constraint, we have incorporated a common resource manager interface in the architecture as depicted in Figure 6. This interface supports resource tasking, registration, status monitoring, and other management and control functions.

Our implementation of the resource manager interface is a based on the OGC SPS (Open Geospatial Consortium Sensor Planning Service[8]). The resource management interface supports the schemas and semantics of SPS. However, unlike the REST interface of the OBD standard version, the developed implementation is exposed as a SOAP/WSDL web service that it can be secured using NCES-style service security.

The resource manager interface may be implemented in a number of ways and is designed to support multiple resources. For legacy resources, an existing sidecar[*] or other net-centric adaptor can implement the resource manager interface. We have also constructed a number of standalone agents that implement the resource manager interface. These are often referred to as resource managers though our focus is on a common resource manager interface and not an implementation. Our long-term vision is that sensors and other resources will be acquired and deployed "net-centric ready" with an integral resource management interface.

Resources are tasked via the resource manager interface. The tasking language is specific to each resource as expressed in the metadata capabilities stored in the resource registry. The example in Figure 7 illustrates a task request for collecting ten minutes of video of a point on the ground:

```
<sensorParam>
    <collectionTime>
        <duration>PT10M</duration>
    </collectionTime>
    <product>VIDEO</product>
    <designation>
        <surfacePoint>
            <latitude>42.36076</latitude>
            <longitude>-71.09005</longitude>
        </surfacePoint>
    </designation>
</sensorParam>
```

Figure 7. Example of a tasking request for a video camera. This example requests ten minutes of video of a particular surface point. The tasking request is conveyed to a resource such as a sensor via the resource manager interface, which is a SOAP/WSDL and NCES-securable version of the standard OGC Sensor Planning Service.

3.2 Data request interface

The data request interface allows consumers to request information and manage plan execution. Data requests are expressed in the W3C standard (World Wide Web Consortium) SPARQL (SPARQL Protocol and RDF Query Language[9]) query language. SPARQL-formatted queries are evaluated against an RDF graph modeling domain-specific resource capabilities in addition to cross-cutting domain-independent resource features.

The expressiveness and powerful filtering capabilities of SPARQL are illustrated in the example data request shown in Figure 8. In this example, the consumer requests video data of a particular surface location. The request also includes a priority that is used to prioritize and schedule resource usage and control access.

```
<dataRequest>
    <requestID>45692351-4674-2823-1945-842745629264</requestID>
    <query>
     PREFIX : <http://localhost:8080/ontology/LLResource.owl#>
     SELECT DISTINCT ?resource
     WHERE {
               ?resource  a :sensor.
               ?resource  :canProduce :video.
               ?resource  :fieldOfRegard ?field.
     FILTER (included(?field, '42.36076 -71.09005').
     }
    </query>
    <priority>5</priority>
</dataRequest>
```

Figure 8. An example of a consumer's information request using the W3C standard SPARQL query language. The request is for video of a particular surface location. Additional constraints can also be included in the request (not shown). This example also shows that data requests include a priority that is supplied by the requester and used by the resource broker to prioritize and schedule resource usage and to control access.

[*]A sidecar is a component framework that allows processing algorithms, net-centric adaption, and other functions to be added to existing sensors without impacting their primary mission. Sidecars are well known and widely deployed in some DoD communities and provide a loosely coupled connection to a host system and a platform for deploying and testing new technology.

3.3 Task coordination service

The task coordination service shown in Figure 6 above coordinates planning, plan selection, and plan execution across resource brokering components. While the service interfaces exposed by all of the resource brokering components are available to clients, the task coordination service provides a high-level data request interface that is more convenient for most applications. Its primary function is to obtain plans from a planning service in response to consumers' data requests, select resulting plans for execution, request that a fulfillment service execute the plan or plans, and provide feedback to consumers. The task coordination service uses multiple factors including feasibility, cost, and user preferences to select plans for execution.

3.4 Planning service

The planning service constructs plans for collections of resources that together satisfy consumers' requests for information. Planning takes into account user specified priorities. In the context of the resource broker, plans consist of resources, tasking, specifications for data flows between resources and from resources to consumers, and timing and ordering constraints. Plans are encoded using XML allowing straightforward XSLT-based translations to other formats more easily consumable by plan execution environments.

The planning service supports pluggable integration of multiple planning engines. The most appropriate planning engine is dynamically selected based on the nature of the data request provided. Our initial planning service implementation uses JSHOP2[10], which is a hierarchical task network (HTN) planner.

In our approach we leverage composable plan templates to provide heuristics to drive the planning process and limit the search space. Plan templates also provide control structure and constraints on resources used in plans. Additional constraints derived from consumers' data requests and from resource capabilities are propagated into the planning process and the resulting plans. The planning service uses the sourcing service to find suitable resources and the verification and assessment service to determine the feasibility and costs of generated candidate plans.

3.5 Resource registry

The resource registry stores semantic metadata that describes resource capabilities. Resource managers are able to dynamically register semantic metadata capabilities documents in the registry on behalf of their managed resources. Resource metadata may also be updated as needed.

RDF (Resource Description Framework[11]) and OWL (Web Ontology Language[12]) standards promulgated by the W3C (World Wide Web Consortium) are employed for defining resource metadata. In addition, the run-time conversion of SensorML[13] structured metadata into RDF is supported. SensorML is an Open Geospatial Consortium standard for XML encoding geometric, dynamic, and observational characteristics of sensors and sensor systems. A plug-in framework has been developed to facilitate conversion of other XML-based resource metadata formats into RDF.

The resource registry is intended to store only static or relatively slowly changing resource metadata. Rapidly changing data about resources is obtained directly from the resource. For example, persisting slowly changing information such as a radar sensor's maintenance status or current theater of operation in the resource registry is acceptable. In contrast, the current pointing vector for a rapidly steerable sensor or the current position or orientation for a highly mobile platform such as an airplane should be not be stored in the resource registry.

The resource registry is implemented using a high performance persistent triple store. We have selected the OWLIM[14] semantic repository as the underlying storage mechanism for the resource registry. OWLIM is part of the open source Sesame RDF framework developed by openRDF.org[15].

3.6 Sourcing service

The sourcing service provides querying and inferencing services for knowledge stored in the resource registry. The planning service, described above, operates on the level of data requests. Satisfying data requests is largely dependent on the capabilities of the underlying resources. The sourcing service accepts requests for resources with resource capabilities as the primary input parameters. It uses the semantic metadata stored in the resource registry to find resources that match the required capabilities.

The sourcing service uses semantic reasoning agents to perform matching. We are currently using the SwiftOWLIM[16] reasoning agent for this function. An example of semantic reasoning in the sourcing service is subclass relationship understanding. For example, if a resource with imaging capabilities is needed, the semantic reasoning agent can find resources capable of providing EO, IR, and SAR images, since these are all subclasses of the more general class of imagers. Another example of build-in semantic reasoning capabilities is cross-domain concept equivalences. As described in the previous section, OWL is employed to capture domain ontologies. The semantic reasoning agent can understand similar concepts represented differently in different domains by exploiting cross-domain relationships captured in mapping ontologies.

We have partitioned querying and inferencing services from the storage services provided by the resource registry to more easily allow different implementations and composition.

3.7 Verification and assessment service

The verification and assessment service determines plan feasibility and cost. This service determines feasibility and cost estimates for each step of a plan and rolls these estimates up into overall estimates. Estimates are obtained by querying each resource used in the plan through its resource manager interface. The overall results are provided to the requester and are used to select one or more plans for execution.

The mechanisms employed by resources to calculate feasibility and cost are resource specific. The algorithms can incorporate any number of relevant factors into account including physical limitations, current and projected future state, learned models, schedule constraints, consumables, tactical situation, priorities, and policy. However, feasibility and cost are represented on common numeric scales so that measures obtained from different resources may be combined and compared.

3.8 Fulfillment service

The fulfillment service executes plans (fulfills information requests). It supports multiple concurrent plan executions. Executing a plan consists of allocating and tasking resources and then monitoring and controlling executions for faults, user intervention, or completion.

We are using the BPEL[17] standard (Business Process Execution Language) as the orchestration language. We selected BPEL because it is a well-developed and supported standard with open source support that offers tremendous flexibility and off-the-shelf functionality. A BPEL process description is generated from each plan supplied to the fulfillment service via an XSLT-based translation and then executed. We are using Apache ServiceMix[18] for execution because it is a mature, open source ESB (Enterprise Service Bus) that has multiple robust options for BPEL service engines and binding components. This approach allows us to support arbitrary communication protocols between resources and from resources to clients.

4. RESOURCE BROKERING APPLICATIONS

We are currently applying and evaluating the resource brokering architecture in several application areas including missile defense, space situation awareness, ISR systems, border security, and cyber defense. This section gives an overview of two representative applications. These applications illustrate many of the key ideas that we presented earlier in this paper.

4.1 Cross-mission time-critical radar sensor allocation and tasking

In this example we focus on resource brokering applied to cross-mission time-critical radar sensor allocation and tasking. We recently developed and demonstrated this system in live field tests. Figure 9 depicts a schematic view of this application. This Figure shows high-resolution and lower-resolution radars that are capable of providing feature data and tracks, respectively. These radars have overlapping fields of regard and are managed by a resource broker. The resource broker also manages other radars not depicted in Figure 9 that can provide feature data and tracks but are not co-located and hence have different fields of regard. Sidecars supporting the resource manager interface as well as a number of other details are not shown in Figure 9 and are omitted from this description. Message flows are conveyed using secure NCES messaging and information is encoded using standard UCore2[19] schemas.

The radars support two missions: an ongoing routine mission that involves collecting feature data on routine targets; and a priority mission that is unanticipated, infrequent, and high priority. Two command centers, one associated with each mission, are also shown in Figure 9.

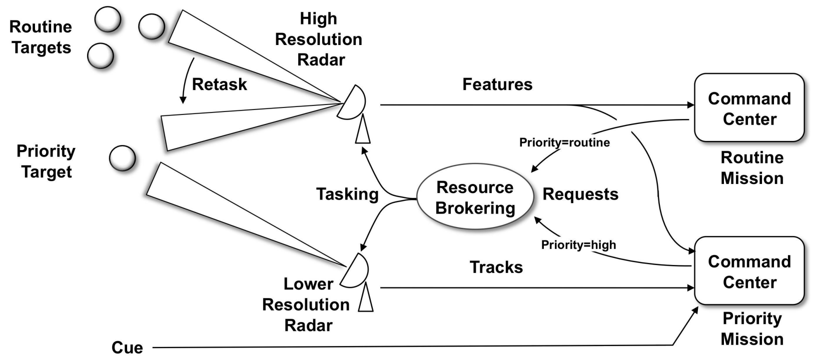

Figure 9. Application of resource brokering to cross-mission time critical radar sensor allocation and tasking. A high-resolution radar is re-tasked from a routine mission to satisfy a priority mission's time critical information need subject to user priorities and policy in force.

Referring to Figure 9, the sequence of events for this application is as follows:

- Feature data for the routine targets is needed by the routine mission command center on an ongoing basis. The routine mission command center requests feature data for the routine targets from the resource broker. The resource broker allocates and tasks the high-resolution radar because it is the best radar available to satisfy the request: it is capable of providing feature data, available to be tasked, has a field of view that includes the routine targets, as well as other considerations. Feature data for the routine targets are provided to the routine mission command center.
- An unanticipated priority target pops up. A cue indicating the presence of a priority target is sent to the priority mission command center.
- The priority mission command center needs to track the priority target. The priority mission command center requests tracks from the resource broker. The request has a high priority. The resource broker allocates and tasks the lower-resolution radar because it is capable of providing tracks, is available to be tasked, and has a field of view that includes the priority target. Tracks for the priority target are provided to the priority mission command center by the lower-resolution radar.
- After analyzing the tracks on the priority target, the priority mission command center decides it needs feature data on the priority target to support further analysis and decisions. The priority mission command center requests feature data for the priority target from the resource broker with a high priority. The resource broker finds that the high-resolution radar can provide feature data, the priority target lies within the field of view of the high-resolution radar, but the high-resolution radar is busy supporting the routine mission. The priority mission command center's request for feature data is at a higher priority and the policy in force allows pre-emption under these circumstances. As a result the routine mission is pre-empted and the routine mission command center is notified. The high-resolution radar is re-tasked to support the priority mission. Feature data for the priority target are provided to the priority mission command center.
- Upon termination of the priority mission, the resource broker re-tasks the high-resolution radar to support the routine mission, which is resumed. The lower-resolution radar is returned to its idle state. Feature data for the routine targets are provided to the routine mission command center.

4.2 Multi-camera, multi-user airborne video camera resource brokering

In this example we describe an application of resource brokering to allocation and tasking of airborne video cameras. Figure 10 is a high-level depiction of the system we have developed. Our system is a laboratory-based emulation at this point in time but supports interfaces and standards that will allow transition.

Our system uses the MIT Lincoln Laboratory Reconfigurable Sensor Simulator, which is a sensor simulator developed at Lincoln Laboratory to support various sensing and decision support activities. We are using this sensor simulator to emulate pointable airborne EO and IR video cameras. The sensor simulator includes a commercial image generation system, control interfaces, network interfaces, and sensor models for various sensors including EO and IR video cameras. Video is published in standard STANAG 4609 format[20] using IP multicast.

In this system users can request video by clicking on a location of interest on a FalconView[21] display. Mouse clicks result in data requests to a resource broker for video from the specified location. The resource broker allocates and tasks the best video camera, which provides the desired video to the user for display. The current optimization algorithm is simple - closest camera that is not already tasked – but can be made more sophisticated by taking into account other resource capabilities such as resolution, modality, platform state, etc. Cameras are attached to simulated airborne platforms. We use DIS[22] (Distributed Interactive Simulation) simulations including JSAF[23] (Joint Semi-Automated Forces) to model airborne and other entities. We are currently using VLC[24], an open source multi-media framework, for video.

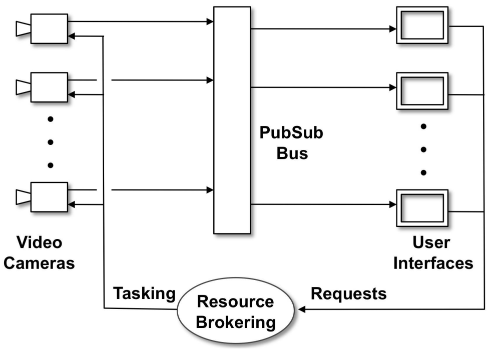

Figure 10. Application of resource brokering to allocation and tasking of video cameras. Decoupling the users from the sensors by the resource brokering and net-centric publish-subscribe approaches enable sensor sharing and optimization.

5. FUTURE DIRECTIONS

Resource brokering is a promising approach with significant benefits for a wide range of net-centric DoD and other government systems. However, we believe that there are a number of areas in which future research and development is needed. This section briefly lists a number of these topics areas.

- Federated brokering, both for distributed planning and distributed plan fulfillment.
- Trust representation and algorithms.
- Semantic metadata resource descriptions – ontologies - for a wide range of relevant mission areas along with reasoning algorithms. There is scattered activity on this topic in the community but significant work remains.
- Resource scheduling, both architectures and optimization algorithms, as components of and services employed by the resource brokering architecture.
- User preferences incorporated into plan generation as heuristics that improve planning efficiency.
- Brokering network resources, i.e., allocating and managing networks and their capabilities including bandwidth, links, channels, spectrum, routing, and quality-of-service related parameters as a resource.
- Model-based planning algorithms that treat planning and fulfillment as a real-time, goal-driven control problem.
- Learning, i.e., learning models for resource capabilities, user preferences, plan performance, and other run-time information and then leveraging the learned models for prediction and optimization.
- Verification and assessment algorithms, e.g., based on predictive simulations and models for the physical environment, tactical situation, and resource capabilities.

6. SUMMARY

In this paper we have presented concepts, software architecture, and example applications for resource brokering. Our approach addresses timely and efficient information resource allocation and tasking. Resources that we consider include sensors, processors, data feeds, archived data, networks, and even human analysts. Resource brokering has great potential for breaking down stove-pipes, cross-mission sensor and data sharing, and timely support for unanticipated users and uses.

The most important top-level concepts of our resource brokering approach are as follows:

- Decoupling of information consumers and information providers
- Dynamic composition of information resources into processing chains based on users' information needs
- Direct data flows from resources to information consumers
- Semantic resource descriptions that support automated reasoning
- Federated architecture

REFERENCES

[1] William A. Owens, "The Emerging U.S. System-of-Systems," February 1996.
[2] Department of Defense, "The Implementation of Net-Centric Warfare," Washington DC, 2005.
[3] H. P. Nii. Blackboard Systems: The Blackboard Model of Problem Solving and the Evolution of Blackboard Architectures. AI Magazine, 7(2):38-53, 1986.
[4] http://www.w3.org/TR/soap
[5] http://www.w3.org/TR/wsdl
[6] http://www.disa.mil/nces
[7] http://en.wikipedia.org/wiki/Representational_State_Transfer
[8] http://www.opengeospatial.org/standards/sps
[9] http://www.w3.org/TR/rdf-sparql-query/
[10] O. Ilghami and D. S. Nau, "A general approach to synthesize problem-specific planners," Technical Report CS-TR-4597, UMIACS-TR-2004-40, University of Maryland, October 2003.
[11] http://www.w3.org/RDF/
[12] http://www.w3.org/TR/owl-features/
[13] http://www.opengeospatial.org/standards/sensorml
[14] http://www.ontotext.com/owlim/
[15] http://www.openrdf.org/
[16] http://www.ontotext.com/owlim/
[17] http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html
[18] http://servicemix.apache.org/
[19] https://ucore.gov/ucore/
[20] http://www.nato.int/structur/AC/224/standard/4609/4609.htm
[21] http://www.falconview.org/trac/FalconView
[22] IEEE, "IEEE Standard for Distributed Interactive Simulation - Application Protocols," IEEE Standard 1278.1, 1995.
[23] http://www.jfcom.mil/about/fact_jsaf.html
[24] http://www.videolan.org/vlc/