Establishing Presence within the Service-Oriented Environment

Eric Konieczny, Ryan Ashcraft, David Cunningham, Sandeep Maripuri
Booz Allen Hamilton
8283 Greensboro Drive
McLean, VA 22102
703-902-5000
konieczny_eric@bah.com, ashcraft_ryan@bah.com, cunningham_david@bah.com, maripuri_sandeep@bah.com

Abstract—As Service Oriented Architectures continue to gain prominence as a mechanism to realize standards-based, distributed computing paradigms, the ability for traditional implementations to support bandwidth disadvantaged and runtime composition scenarios has been questioned.[1][2] Traditional approaches leverage centralized registry platforms to enable service discovery functionality, but this inherently introduces the possibility of stale metadata and registry information that does not reflect actual operating conditions. Service presence offers a fresh opportunity to redefine service discovery; while not typically viewed as a crucial element of SOA's runtime discovery solution space, service presence significantly enhances the real-time monitoring of services by introducing an omnipresent mechanism for capturing a service's state. This paper focuses on analyzing and evaluating the feasibility of utilizing Peer-to-Peer (P2P) approaches to better facilitate service presence and dynamic service discovery through discussion of experimentation conducted using an eXtensible Messaging and Presence Protocol (XMPP) driven prototype.

TABLE OF CONTENTS

1. INTRODUCTION

Service discovery is a central concept for fully realizing the benefits of Service-Oriented Architecture (SOA). A robust service discovery implementation enables the runtime fulfillment of the "Publish-Find-Bind" SOA paradigm and facilitates the execution of context-sensitive business policy. More importantly, by allowing service clients to *dynamically* locate and access services at runtime, SOA can better react to the malleable nature of highly distributed enterprises. Where there is frequent fluctuation in both services and service consumers, such as in mobile, bandwidth disadvantaged environments, paramount importance must be placed on an implementation's ability to seamlessly handle the change. The lack of an effective mechanism for dynamic service discovery can result in the significant degradation of overall system capability and quality.

As SOA continues to gain traction within the command and control (C2) community and becomes the underlying technology vehicle driving real-time production systems deployed to theater, the effectiveness of traditional SOA implementations within non-traditional environments must be evaluated. The capabilities of a SOA component deployed on a fighter jet, tank, satellite, missile, or other non-traditional asset should be synonymous to those of a component deployed to a traditional, stationary server. Similarly, given the recent industry trend centered on web service enabling dynamic enterprise resources, such as sensors [1], and their corresponding integration into SOA infrastructures, traditional approaches and commonly accepted patterns may not suffice. More specifically, the needs and capabilities of a dynamic service discovery implementation deployed to a volatile SOA environment should be re-addressed.

Traditional industry de-facto standards providing dynamic service discovery capabilities have generally employed standards-based, centralized registry platforms, such as Universal Description, Discovery and Integration (UDDI) and Electronic Business using eXtensible Markup Language (ebXML). These models have several strengths and provide a powerful and flexible mechanism to store and retrieve service metadata in an interoperable fashion. As a result, centralized registry based service discovery implementations have gained wide acceptance and popularity within the user community. However, due to the lack of a dynamic feedback mechanism, the overall effectiveness of these centralized models is limited by their storage of stale snapshots of the transient environments in which they operate.

Previous research and development focusing on better facilitating dynamic service discovery in distributed environments has produced several service discovery protocols [2], including Jini Network Technology and Universal Plug and Play (UPnP). However, these technologies are specifically designed as lower-level communication protocols and as a result, have not been widely applied at the enterprise scale. This paper will

1

[1] 978-1-4244-2622-5/09/$25.00 ©2009 IEEE.
[2] IEEEAC paper #1311, Version 2, Updated January 9, 2009

introduce the notion of establishing service presence as a solution to help alleviate some of the issues surrounding the usage of traditional, registry-based service discovery platforms in dynamic environments, on an enterprise SOA level. Service presence offers a fresh opportunity to refine dynamic service discovery by enabling the real-time monitoring of services through the omnipresent capturing of a service's state. This paper will focus on how service presence can be applied to a wide variety of C2 applications, both within the military and civilian domains, to better solve challenging use cases associated with a SOA's operation in non-traditional, unpredictable environments. Additionally, this paper will discuss the establishment of service presence through a standards-based, eXtensible Messaging and Presence Protocol (XMPP) driven solution, leveraging a rich set of presence and event-driven capabilities, to augment the service metadata publishing and retrieval infrastructure that is already in wide use across the industry.

2. SERVICE ORIENTED ARCHITECTURE

Service-Oriented Architecture (SOA) is a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains. It provides a uniform means to offer, discover, interact with, and use capabilities to produce desired effects consistent with measurable preconditions and expectations[12]. More concretely, SOA is similar to traditional component models, and relates different functional capabilities by providing uniform, platform-agnostic access to these capabilities via defined interfaces. These capabilities, or services, provide a platform-neutral interface contract that allows for service interaction across distributed and heterogeneous platforms in a uniform and universal manner. Services can be made available by publishing the service to a service registry where it may be discovered by a service consumer.

From an enterprise architecture perspective, the physical infrastructure to provide the underpinnings of this data sharing platform, in part, lies in the adoption of a SOA, implemented using Web Services and other XML-driven initiatives. Within a SOA, services are visible to the network at large by providing physical interfaces over enterprise assets. These services are platform neutral, and are described with application-agnostic service descriptions that can be published to metadata registries. Thus, network-enabled users are provided an open, standards-based means to discover relevant services and to invoke them either individually or within the framework of a larger composite process that leverages several services across the enterprise. SOA provides a foundational layer for an information-centric enterprise that satisfies new and changing business needs by enabling the dynamic sharing and aggregating of information across organizational boundaries via individual service-enabled enterprise assets [13].

However, SOA, by itself, is merely an abstract architecture specification; World Wide Web Consortium (W3C) [14] and the Organization for the Advancement of Structured Information Standards (OASIS) [15] endorsed Web Service standards (WS-*), built over XML and Web technologies, represent the latest attempt to realize the full capabilities of SOA. These standards provide many of the essential physical infrastructure components required of a SOA platform:

(1) ***Message Encoding:*** SOAP is a standardized specification for encoding message payloads between services.

(2) ***Service Interface Description:*** Web Service Description Language (WSDL) describes a Web service's capabilities as collections of communication endpoints capable of exchanging messages.

(3) ***Service Metadata Registry:*** Universal Description, Discovery, and Integration (UDDI) is a registry for services to expose their interface descriptions for discovery on the network.

Figure 1 – Traditional web services basic protocol stack

In particular, service discovery is a key enabling component of SOA's "Publish-Find-Bind" architectural pattern. As described in Figure 2, a SOA's service discovery capability enables a service consumer to find the information needed to access services at runtime by providing the relevant set of business context metadata describing the desired service's classification, functionalities, and other relevant exposed metadata. Once provided with needed invocation metadata, the service consumer can dynamically leverage the desired service programmatically at runtime. As a result, service consumers can execute business logic and perform policy decision-making in real-time based on relevant business events.

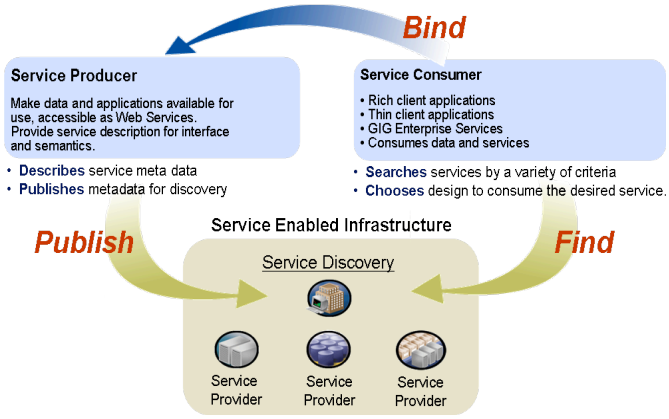

Figure 2 – Publish-find-bind SOA paradigm

From an enterprise computing standpoint, service-oriented architecture implementations have resulted in a fundamental shift from application-centric computing, where only a privileged set of users access data from isolated applications that perform a limited set of tasks, to a more situation-centric model, where a consumer interacts with a loosely-coupled system that provides dynamic, context-sensitive capabilities based on the operational needs of the problem at hand. Essential to this vision of improved interoperability and coordination is enabling services and their consumers – doers and users - to better coordinate the sharing of information. This fundamental support of loose coupling significantly increases the value of a SOA infrastructure by eliminating integration barriers of new business components and functionalities. Without an effective service discovery mechanism, a SOA implementation's components become increasingly interdependent and unable to react to one another as the system grows in size and function. Service consumers may be limited to the design time, hard-coded usage of available services. As a result, a SOA's ability to handle infrastructure change is significantly hindered as labor-intensive code-base modifications may be required for even trivial modifications such as service endpoint changes. A robust dynamic service discovery solution can enhance a SOA's capacity to react to changing conditions by reconfiguring itself dynamically through the seamless integration of new, previously unknown services.

3. MOTIVATION

With the recent advances and the rapid adoption of pervasive and grid computing, SOA has emerged as a key architectural paradigm for effectively facilitating communication between increasingly heterogeneous assets. In dynamic, distributed environments, a SOA's ability to seamlessly handle change is paramount. Accordingly, the effectiveness of a SOA's dynamic service discovery solution in providing service consumers with accurate, up-to-date discovery functionality becomes even more prevalent than in a more stable infrastructure. The negative impact of network fluctuations, service outages, re-classifications, and transient enterprise situations can all be reduced through the effectual use of dynamic service discovery. Due to the high frequency and amount of variability in service providers, the overall capability of a SOA may become significantly degraded or even rendered useless barring the presence of a robust dynamic service discovery solution. Several industry areas, including federal domains such as military, defense, and intelligence, as well as commercial domains focused on mobile computing platforms, have a strong business need for a service discovery implementation that is able to adapt within transient environments. In particular, due to the highly mobile and variable nature of the environments in which they typically operate, C2 applications can especially benefit from this type of enhanced service discovery mechanism that is capable of effectively functioning in volatile enterprise situations.

Typical Industry Approaches to Service Discovery

Traditional industry approaches to dynamic service discovery have generally leveraged centralized registry platforms. These models often utilize well known standards such as UDDI and ebXML. Centralized registry implementations typically provide a powerful and flexible system for storing and retrieving service metadata, and as a result, have achieved significant buy-in from and adoption within the enterprise community. In most cases, the process by which a service provider is registered to a service registry is a static process: a developer or system administrator manually publishes the relevant metadata describing the service to the registry. Similarly, if a previously published service is updated or removed, the corresponding metadata changes must be manually applied to the service registry.

While static, human-driven service registration processes may be acceptable within more stable environments, this approach is generally not viable in dynamic, distributed enterprises. The presence of highly variable service providers can create an abundance of stale metadata orphaned within a centralized registry. As services fluctuate due to loss of communications, re-classifications, and other dynamic enterprise interruptions, the metadata being exposed by the service registry to service consumers for dynamic service discovery purposes remains the same, unless physically updated by the appropriate party. Based on the frequency of service provider changes and service consumers accessing the service registry, manually synchronizing the registry with updated service metadata may not be feasible.

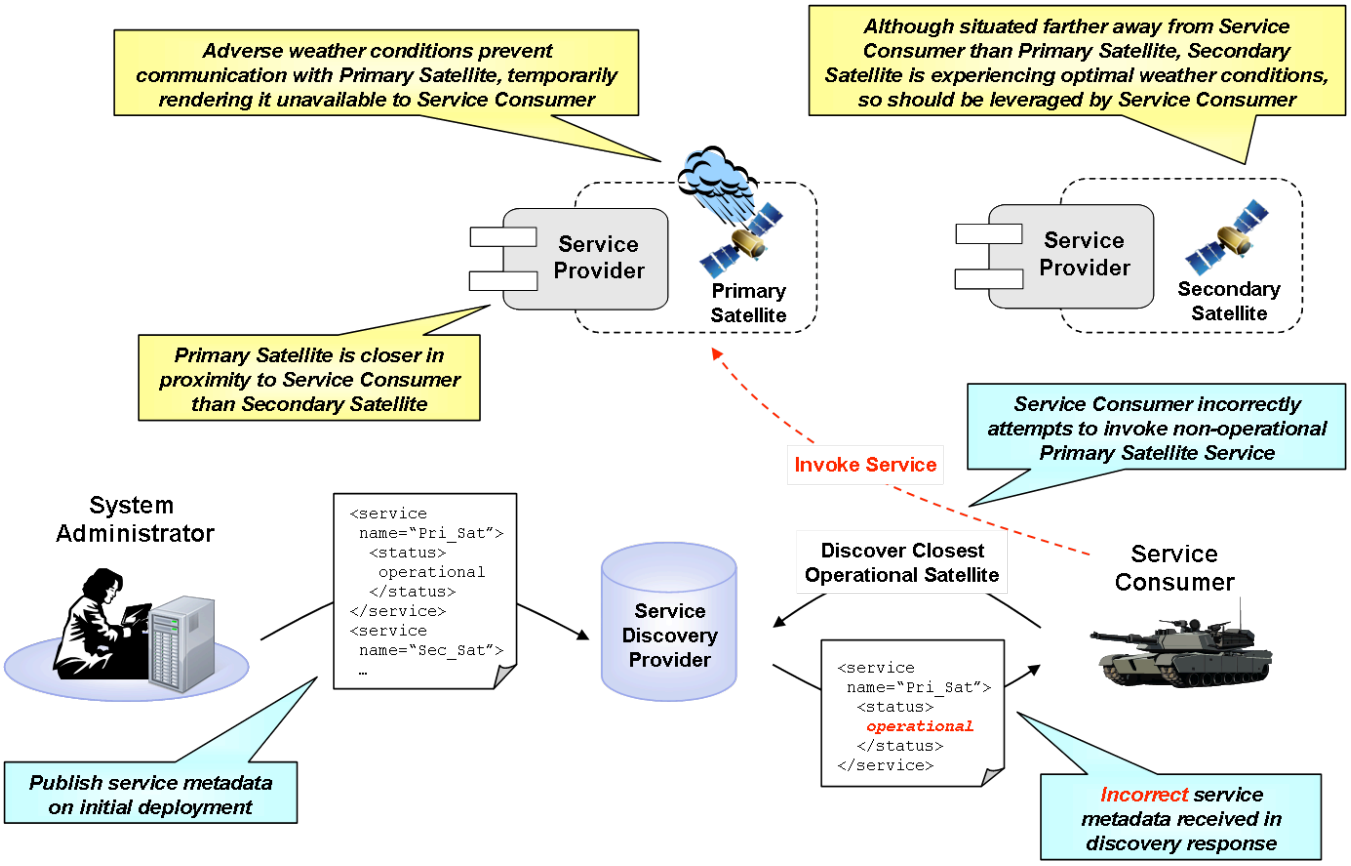

Figure 3 – System degradation as a result of invalid service metadata

As service consumers attempt to dynamically discover desired services, they are provided with outdated, invalid service information from the service registry. This has several negative consequences, as consumers may attempt to programmatically invoke services that are not available, or potentially initiate more serious processing errors by kicking off a sequence of incorrectly executed business logic based on service misinformation. As invalid service metadata is propagated and leveraged throughout the environment, the overall effectiveness of the system is drastically reduced. In highly dynamic SOA enterprises, the negative effects associated with the presence of invalid service metadata are significantly amplified in comparison to a more stable environment.

Challenges to Service Discovery in C2 Applications

In comparison to many other industries, SOA infrastructures deployed within the C2 domain are typically subject to greater amounts of variability and instability than traditional SOA deployments. This is largely due to the highly mobile and potentially harsh nature of the environments in which these services frequently operate. As a result, the impacts of stale service metadata discussed above, stemming from the use of traditional registry implementations for service discovery, can be comparatively pronounced in SOA-driven C2 applications. Within both the military and civilian sectors of C2, several difficult technical challenges exist as a result of the lack of a service discovery implementation that is capable of effectively performing in dynamic C2 environments. For example, consider the scenario presented in Figure 3, in which a mobile tank unit is provided with network connectivity via a military satellite system. Assuming a service oriented view of a network access capability, each individual tank might act as a service consumer and dynamically discover and invoke available network-providing satellite services, which are published to a service broker. Due to adverse weather conditions, satellite position, and other variable factors, a satellite may not be available for use or may provide unacceptably slow network access speeds. Using the traditional, static service registration approach discussed above, a temporarily disabled or disadvantaged satellite might be incorrectly described as fully operational within the relevant service registry. Nearby units may inappropriately attempt to consume these non-functional satellite services and as a result, degrade their tactical capabilities.

Another related challenge area involves communications coverage loss in aerial units. For example, consider a unmanned aerial vehicles (UAV) that has been tasked on a low-priority reconnaissance mission. As the UAV travels across the specified target area, it may periodically enter in and out of dead-zones during which it loses network connectivity and contact with its tasking entity. Assuming that the UAV can be considered as a service provider within

a SOA, the dynamic state of the UAV, in terms of its availability to be communicated with and re-tasked, will not likely be reflected in the relevant service registry if a traditional service discovery and publishing paradigm is employed. The corresponding presence of invalid metadata describing the operating state of the UAV may hinder or even prevent the dynamic re-tasking of the unit to perform time-sensitive, higher priority work.

Another scenario illustrating the need to re-address service discovery patterns occurred during Joint Task Force Operation Burnt Frost, in which a non-functioning U.S. National Reconnaissance Office satellite was intercepted and shot down [16]. One of the major challenges of Burnt Frost involved the assessment of the operation's effect on other nearby satellites, specifically in terms of whether or not the remnants of the destroyed satellite collided into other satellites, which could potentially cause significant damage and affect satellite operational capabilities. Assuming that nearby relevant satellites were deployed within a SOA environment, each satellite might be manually published as a service provider to a traditional registry platform. While Burnt Frost was an important success, the level of effort required by the operation's team may have been lowered if the pre-existing IT infrastructure, specifically the service registry, could have been leveraged to immediately determine which satellites had been affected by the operation and were still fully functional and available.

Previous Research on Discovery in Volatile Environments

Previously conducted research efforts attempting to address issues surrounding the enablement of an effectual dynamic service discovery implementation in a distributed SOA generally utilize P2P-driven, announcement-based mechanisms over lower level communication protocols, such as Jini and JXTA [3] – [5]. While these discovery models have several merits, they lack significant buy-in from the enterprise community. One reason for this lack of support may be because the underlying technologies used are typically tied to a specific technology or language, such as Java, as opposed to being based on open standards. Additionally, other ad-hoc techniques may be applied to resolve stale service metadata storage issues, such as enabling a service registry platform to periodically prune its invalid entries. While this begins to address the problem, a more robust architectural pattern is needed. Given the prevalence of centralized registry platforms in SOA implementations in existing deployments, a mechanism is needed that will easily complement pre-existing discovery infrastructures.

4. SERVICE PRESENCE

In order to alleviate the aforementioned issues associated with utilizing traditional mechanisms for dynamic service discovery, the notion of service presence can be applied. Service presence refers to the real-time monitoring of a service by introducing an ubiquitous mechanism for capturing service state. Depending on the exact nature of the relevant system's requirements, service presence can track different aspects and levels of service state. For example, in a pervasive environment composed of mobile, transient services that frequently fluctuate between an up and down state, service presence might be employed exclusively from the standpoint of maintaining an omnipresent reference of whether or not services are available for use. On the other hand, other systems may find it useful to apply service presence to track other facets of service state, such as advertised metadata intended to be used by service consumers for discovery purposes, or even the nature of the data content provided by a service. The architectural pattern of service presence can be leveraged in a number of different ways to provide consistently valid, up-to-date service information for utilization by a dynamic service discovery solution.

While not typically viewed as a crucial element of a SOA's runtime discovery solution space, the application of presence offers a fundamentally different, proactive approach to dynamic service discovery. The establishment of service presence as a means to capture and leverage service state in real-time for use within a dynamic service discovery implementation has significant benefits to a SOA. By seamlessly improving the quality of a system's underlying service discovery infrastructure, a SOA's adaptability and capacity for self-reconfiguration based on business events and policy is similarly enhanced. Presence can be used as a key component in enabling effective dynamic service discovery within volatile enterprises. While the benefits of utilizing service presence in a more stable enterprise, in which there is little change in service composition and availability, appear to be comparatively low, it can be used to reduce or eliminate the need for human-in-the-loop interactions on service registries. By effectively employing service presence, a SOA infrastructure can provide qualitatively better dynamic service discovery capabilities. Specifically, a dynamic service discovery implementation can increase its capacity to provide more accurate, dependable service discovery results by utilizing near perfect service information as provided by a service presence mechanism; the service metadata used by the discovery implementation is, to an extent, guaranteed to be representative of the real-time state of the service at the time of the discovery request.

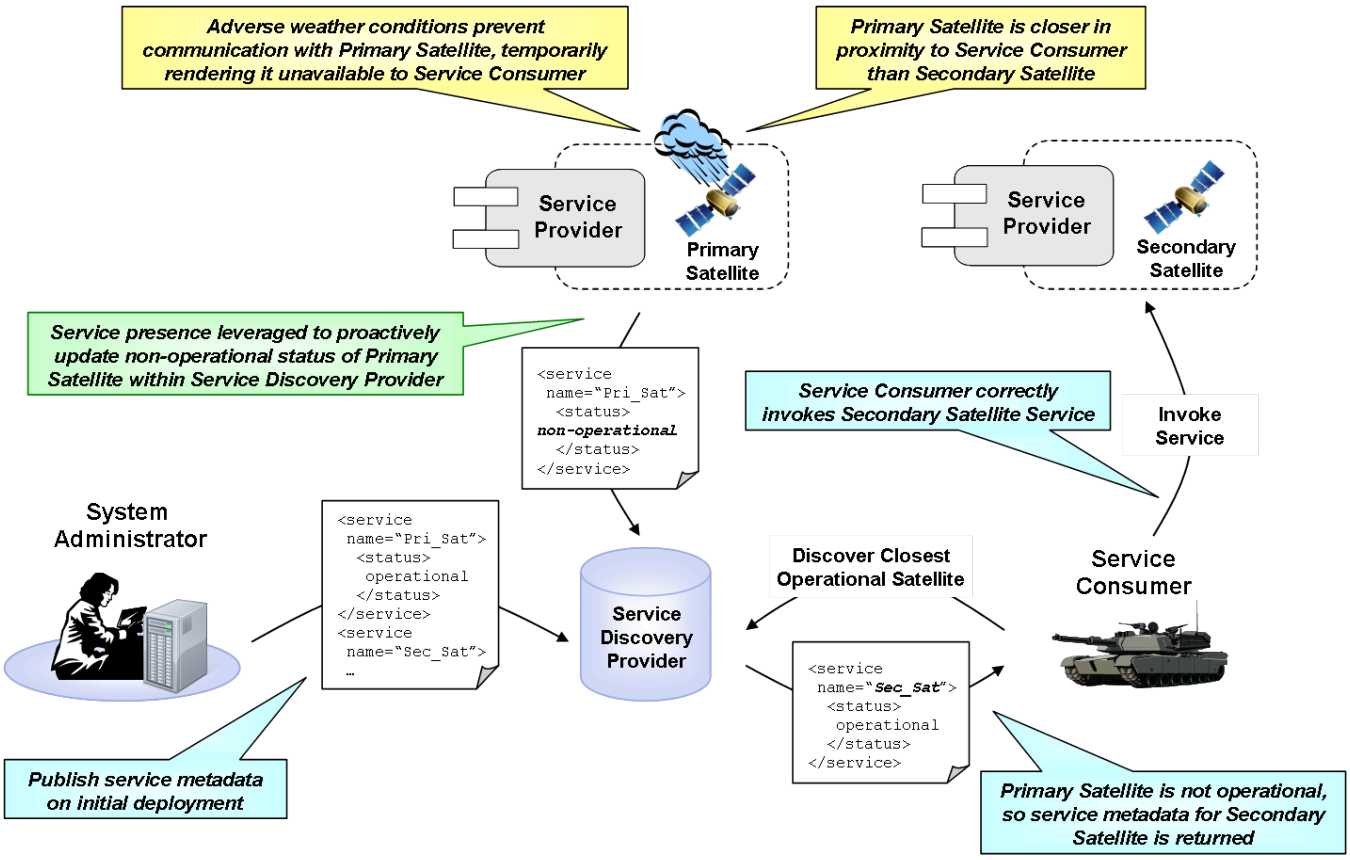

Figure 4 – Application of service presence within C2 problem domain

From a service consumer's standpoint, the quality of dynamic service discovery responses is transparently improved without any extra effort in terms of initiating more expressive discovery requests. Additionally, the added reliability of the received discovery results precludes any further work that must be done at runtime to ensure the validity of those results, such as determining whether or not a returned service is actually available and deployed to the given endpoint.

Benefits of Service Presence within C2 Applications

Service presence can be employed to help enhance the quality and capabilities of dynamic service discovery implementations, especially within the context of C2 specific applications. Revisiting the use cases described in Section 3, service presence can, from a service consumer perspective, help increase the situational awareness of service availability and state. The establishment of presence enables consistently accurate information regarding dynamic elements, such as satellite availability for network access or UAV communication connectivity for dynamic re-tasking, to be proactively published to a service discovery provider and seamlessly leveraged by consumers. As illustrated in Figure 4, by maintaining a near perfect knowledge of satellite, tank, and aerial unit state in terms of accessibility, geo-location, and other dynamic metadata aspects, several difficult challenges hindering a C2 SOA's effectiveness can be addressed.

Another relevant domain where the pattern of service presence can be utilized involves communications coverage loss in sensor networks. Wireless, ad hoc sensor networks play a key role in enabling intelligent, ubiquitous computing models, but are typically very unpredictable in terms of their connectivity to the outside world and quality of service [11]. Additionally, wireless sensor networks are leveraged across the C2 domain for a wide variety of uses, including aircraft engine health monitoring [17]. Especially given the recent industry trend to service enable sensor units for usage within SOA infrastructures, service presence can be established to manage the fluctuating on and off state of sensor resources within the context of dynamic service discovery.

Previous Research on Service Presence

The application of service presence within enterprise SOA environments is a comparatively undocumented and unexplored topic. Some research has been performed investigating the potential of utilizing lower level service discovery protocols, such as Jini and JXTA, to allow services to proactively announce their presence to a group of service peers [6]. However, these approaches are more focused on providing a high-performance mechanism to enable the self-registration of a service and the initial advertisement of its capabilities, as opposed to facilitating the continual real-time monitoring of service state. Additionally, the success of these models may be limited

within the enterprise domain by their inherent dependence on Java and the presence of a Java Virtual Machine (JVM). Other relevant research has presented the idea of extending Session Initiation Protocol for Instant Messaging and Presence Leveraging Extensions (SIMPLE) messages with Web Service Description Language (WSDL) service profile snippets to enable the exchange of WSDL defined service metadata within a presence framework [7]. This idea has several distinct benefits, in that to an extent, it provides a mechanism for tracking service state in real-time. However, the proposed system may be limited by its focus on providing an integrated presence framework, as opposed to a service discovery solution; only the method by which services transmit presence information is described, services must have some prior knowledge of each other's address and access protocol before being able to participate.

5. IMPLEMENTATION

Experimentation was conducted to develop a prototype capable of establishing an effective service presence mechanism that could be leveraged to provide improved dynamic service discovery capabilities within dynamic SOA environments. A key aspect of the developed prototype was to ensure that the implemented service presence solution could be easily integrated into a pre-existing SOA infrastructure to augment the service discovery capabilities already provided by a traditional, centralized service registry. With both of these overarching requirements in mind, XMPP was selected as the underlying technology vehicle to provide the needed service presence capabilities.

XMPP as a Means to Establish Service Presence

XMPP is an open standards, XML-inspired protocol designed for near real-time, extensible instant messaging and presence information [8]. The technology is best characterized by its simple, flexible, and lightweight architecture. Most XMPP server implementations leverage long-lived XML stream parsing models, which can enable XMPP-driven applications to achieve strong performance and high message traffic throughput. Additionally, a powerful set of event publishing and notification functionalities are provided through a publisher-subscriber XMPP protocol extension [9]. Given its previously described abilities, XMPP was chosen as a more viable candidate than lower level service discovery protocols, such as Jini and JXTA, to act as the core technology used for establishing service presence. While a lower level protocol may have provided better performance, XMPP was determined to be a better fit for use at the enterprise SOA level due to its standards-based nature and rich set of built-in presence, publisher-subscriber, and messaging functionalities.

Notional Architecture

The notional architecture for the developed dynamic service discovery prototype can be seen in Figure 5. In the prototype, an XMPP client proxy wrapper is instantiated for each service and service consumer deployed within the SOA, which enables the real-time sharing of presence information and the usage of XMPP's publisher-subscriber capabilities. Similarly, the pre-existing UDDI registry is wrapped by an XMPP server instance, which acts as a proxy to service registration and metadata update requests.

When a service is initially deployed to the enterprise, the service self-registers its presence and state information through its corresponding XMPP client proxy. In turn, the XMPP client proxy sends the appropriate message to the XMPP server announcing the presence and advertised state metadata of the new service. This message, as shown in Figure 4, encapsulates two different types of relevant service metadata. The first type is metadata that will remain fairly static for comparatively long periods of time and should be published and maintained within the UDDI registry. This may include taxonomy classification, WSDL endpoint, and other stateful information that will most likely not change at a frequency that could be prohibitive to publishing the new relevant updated metadata value in the UDDI registry on each change. For extensibility and expressivity purposes, this static metadata is represented in the service announcement message as a tModel, which is provided for the service at design time. The second type of metadata contained within the service announcement message is any information that is highly volatile and as a result, cannot be easily published to the underlying UDDI registry due to performance and scalability constraints. This may include geo-spatial information describing the service's deployment location, whether or not the service is available for use at a given point in time, or even certain aspects of the data content provided by the service. As this dynamic information should not be published to the UDDI registry, a mechanism is needed that will proactively notify service consumers who are interested in actively knowing when the relevant metadata has changed, or more specifically, if the metadata has entered a particular threshold. This event driven capability is provided through XMPP's publisher-subscriber infrastructure. The volatile service metadata specified in the service announcement message is published to a pubsub node on the XMPP server.

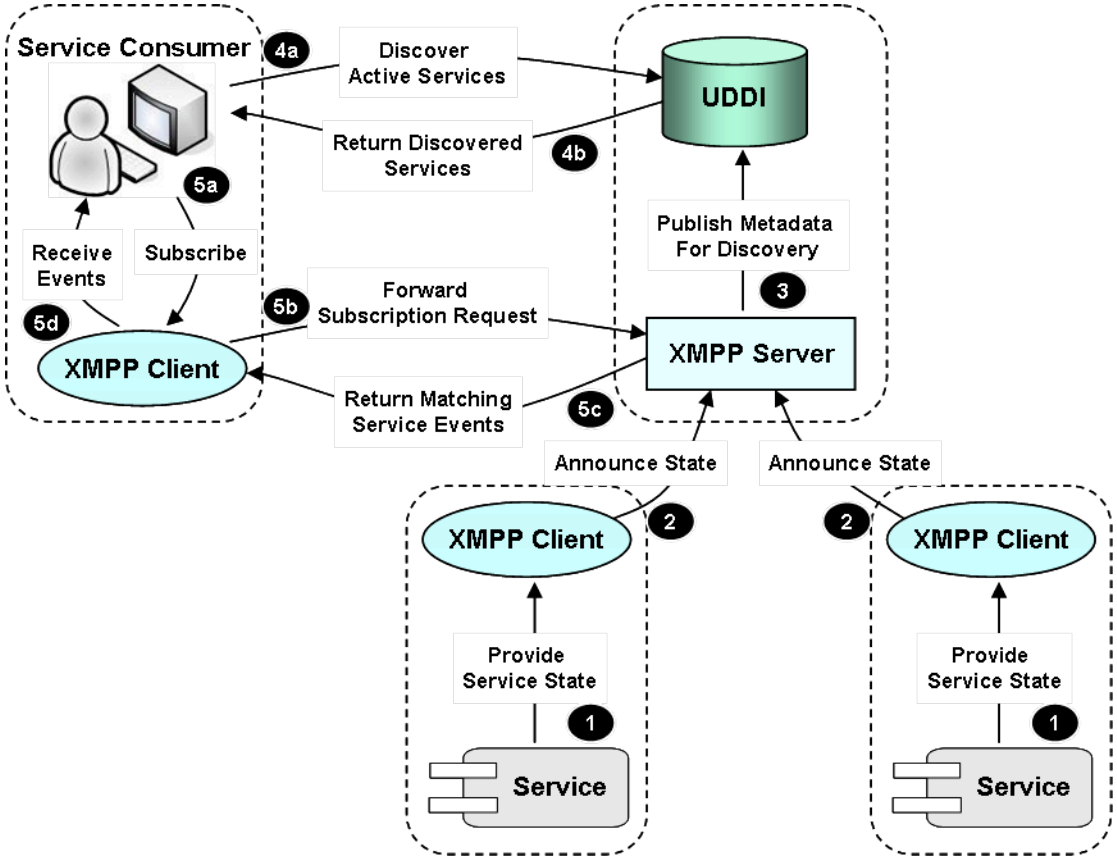

Figure 5 – Notational architecture of developed dynamic service discovery prototype

For consistency and simplicity purposes, the name of the pubsub node that is published to is the same as the service's specified taxonomy classification node name.

```
<iq type='set' from='service-provider@service-endpoint.com'to='pubsub.xmpp-service-discovery-provider.com' id='publish1'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <publish node='urn:xmpp-service-discovery:taxonomy:generic-services'>
      <item>
        <service-provider-metadata
            xmlns='urn:xmpp-service-discovery:service-provider-metadata'>
          <tModel
            tModelKey="uuid:5DD52389-B1A4-4fe7-B131-0F8EF73DD175" xmlns:uddi='urn:uddi-org:api_v3'>
            <name>Sample Service Provider</name>
            <description xml:lang="en">Sample service provider within the XMPP Service Discovery Provider
            </description>
            <categoryBag>
              <keyedReference
tModelKey="uddi:uddi.org:wsdl:porttypereference"
keyName="wsdl:portType Reference"
keyValue="uuid:340e72e4-12d0-44a1-9a08-3ef8733c27dc"
/>
              <keyedReference tModelKey="uuid:cf4c1ad8-37cb-412c-ac40-0ad8b419a0e3" keyName="Sample Service Provider Taxonomy" keyValue="urn:xmpp-service-discovery:taxonomy:generic-services" />
              ...
            </categoryBag>
          </tModel>
          <subscribable-metadata>
            <available-for-use>true</available-for-use>
            <current-location>
              <kml xmlns="http://www.opengis.net/kml/2.2">
                <Placemark>
                  <name>New York City</name>
                  <description>New York City</description
                  <Point>
                    <coordinates>
                      -74.006393,40.714172,0
                    </coordinates>
                  </Point>
                </Placemark>
              </kml>
            </current-location>
            ...
          </subscribable-metadata>
        </service-provider-metadata>
      </item>
    </publish>
  </pubsub>
```

</iq>

Figure 6 - Sample service presence and metadata announcement sent by a XMPP client wrapper on behalf of a service

In addition, using XMPP's core built-in capabilities, a connection used for message traffic flow and presence information exchange is established and maintained between the XMPP client proxy and the XMPP server. By deploying the XMPP client proxy and the relevant service in conjunction on the same network entity, unexpected network fluctuations and outages preventing consumer interaction with the service can be automatically detected by the XMPP server through the disruption in its connection with the corresponding XMPP client proxy. The XMPP server can then leverage the appropriate service metadata storage facilities, as described below, to appropriately update the relevant service information to reflect the network connectivity loss and unavailable state of the service.

Once the XMPP server receives and validates the service announcement message, it generates and sends the appropriate SOAP web service invocation to the UDDI registry to publish the service's static metadata. Concurrently, the XMPP server publishes the service's transient metadata to the appropriate pubsub node on the server, which proactively alerts any service consumers who are subscribed to the presence of services with metadata matching that of the newly published service. Similarly, when the state of a previously published service is modified, the XMPP client proxy wrapper generates and forwards the appropriate service announcement message to the XMPP server. While the presence of the published service in terms of network connectivity can be seamlessly monitored by the XMPP client proxy and XMPP server using XMPP's built-in presence infrastructure, the service has the responsibility of providing the XMPP client proxy with any other updated service metadata. Generally, this can be accomplished through the effectual use of the Observer pattern within the service.

As previously mentioned, the developed dynamic service discovery prototype enables service consumers to receive real-time notifications regarding services of interest through an event-based subscription mechanism. Through their corresponding XMPP client proxy, a service consumer can subscribe to published services that meet a desired taxonomic classification and volatile metadata criteria. As new, previously unknown services that meet the desired subscription criteria are published to the service discovery capability, or as previously subscribed to services are modified, updates regarding the state of published services are pushed to the service consumer in real-time. A Listener pattern interface is provided such that the consumer can automatically invoke custom business logic in response to certain subscribed service events. While this event driven capability is provided to enable service consumers to monitor a service's dynamic state metadata, similar functionality can be realized with a service's static state metadata published to the UDDI registry by leveraging UDDI's built-in subscription capabilities [10]. Additionally, service consumers may still dynamically discover services from the UDDI registry by invoking typical request-response style discovery commands that now have the added benefit of utilizing more accurate and dependable service information.

Analysis of the Developed Prototype

The implemented prototype offers several advantages to traditional, standalone service registry implementations in dynamic SOA environments through the establishment of service presence. By enabling different mechanisms for persisting and broadcasting service state based on frequency of modification, the proposed system attempts to bring together the strengths of both XMPP's presence and messaging capabilities and UDDI's metadata storage and retrieval infrastructure. Additionally, service consumers are transparently presented with more dependable dynamic service discovery results through the underlying establishment of service presence, and are also given the option of leveraging a more proactive, event-driven discovery mechanism.

While the implemented prototype was designed to demonstrate the viability of establishing service presence at the enterprise level, there are alternate strategies that may improve the system's dynamic service discovery capabilities. For example, there may be a more robust and extensible fashion to determine which advertised service metadata is persisted through the UDDI registry or through the XMPP-provided publisher-subscriber infrastructure. Instead of using an ad-hoc, predefined approach, the service metadata's modification frequency could be monitored in real-time and depending on the requirements of the system, a policy could be developed that could dynamically promote or demote metadata elements from storage within either infrastructure. An additional aspect of the developed prototype where there is opportunity for growth lies in the schema used to define and create subscriptions to a service's volatile metadata set. Currently, an ad hoc system is used to create a basic metadata filter that is applied to select the appropriate sets of services to which to subscribe. A more extensible and flexible framework for defining the criteria on which subscription filtering is performed, potentially similar to the Hibernate Criteria API, would significantly increase the usability of the developed prototype from a service consumer's perspective. Similarly, this subscription filtering framework could be extended to allow for the definition of custom types of subscription events. Currently, subscription events are statically defined and are only generated for instances where a service matching the provided subscription criteria is published, unpublished, or has its advertised metadata modified to equal some specified value.

Applicability of Prototype within C2 Industry

There is significant potential to apply the architectural patterns used by the implemented prototype to realize dynamic service discovery related business use cases within the C2 domain. For example, consider the scenario of a ground station tasking a series of satellites in orbit to collect imagery data needed for analysis and planning. For any given collection scenario, there are a series of primary satellites that are most desired for use based on their ability to optimally capture image data for the given situation, potentially due to several reasons, including the satellites' available imaging capabilities, relative location, and line of sight. Conversely, there are a series of secondary satellites that can also be utilized to capture the needed data but will not produce the most optimal results. In certain situations, a primary satellite that is tasked by the ground station to provide coverage for a particular area will, during its normal orbit path, come into close proximity with the sun such that it is temporarily rendered inactive. During this period of inactivity, a secondary, backup satellite must be tasked to temporarily provide the needed coverage until the primary satellite has traveled back into a position appropriate for utilization. In order to realize this use case, the notification-based, event-driven architecture developed for the prototype can be applied to enable the real-time handling of important changes in service state. Specifically, the appropriate ground station service consumer could create a subscription requesting to be notified when the set of metadata describing a satellite service's geo-location reaches a certain threshold indicating that the satellite is located within a certain distance from the sun. The ground station consumer could then process notification events that are proactively pushed to it to determine in a dynamic, real-time fashion which satellites should be utilized to best maximize overall image coverage.

6. CONCLUSION

This paper presented an approach for establishing service presence as a vehicle to enhance the dynamic service discovery capabilities of a traditional centralized service registry platform. Through the introduction of a ubiquitous mechanism for capturing service state, an enterprise SOA deployed to a transient environment can qualitatively improve its dynamic service discovery implementation and alleviate critical issues related to stale service information stemming from the traditional usage of a centralized service registry. Additionally, this paper presented an architecture for a dynamic service discovery prototype integrating service presence and event-driven capabilities provided by XMPP, with the service metadata storage and retrieval infrastructure provided by a UDDI registry. While the proposed application of service presence has significant potential for enhancing service discovery within dynamic environments, such as those typically encountered in C2, more exploration and research is still needed, specifically in the realm of scalability, performance, security, and governance.

REFERENCES

[1] M. Botts, G. Percivall, C. Reed, and J. Davidson, "OGCSensor Web Enablement: Overview And High Level Architecture," http://www.opengeospatial.org.

[2] F. Zhu, M. W. Mutka, and L. M. Ni, "Service Discovery in Pervasive Computing Environments," *IEEE Pervasive Computing*, vol. 4, no. 4, pp. 81-90, Oct-Dec. 2005.

[3] Ken Arnold, "The Jini Architecture: Dynamic Services in a Flexible Network," *Design Automation Conference*, vol. 0, no. 0, pp. 157-162, 36th Annual Conference on Design Automation (DAC'99), 1999.

[4] F. Xhafa, L. Barolli, R. Fernández, T. Daradoumis, and S. Caballé, "Extension and evaluation of JXTA protocols for supporting reliable P2P distributed computing," *International Journal of Web Information Systems*, vol. 4, no. 1, pp. 121-135, 2008.

[5] M. Pirker and M. Berger, "An approach for fipa agent service discovery in mobile ad hoc environments," in *Workshop on Agents for Ubiquitous Computing, in conjunction with AAMS 2004*, July 2004.

[6] N. Furmento, J. Hau, W. Lee, S. Newhouse and J. Darlington, "Implementations of a Service-Oriented Architecture on top of Jini, JXTA and OGSI," in *Grid Computing: Second European AcrossGrids Conference, AxGrids 2004,* Nicosia, Cyprus , Jan. 2004, pp. 90-99.

[7] R. Liscano, A. Ghavam, and M. Barbeau, "Integrating Service Discovery Protocols with Presence-based Communications for Ad hoc Collaborative Scenarios," in *Communication Networks and Services Research Conference (CNSR 2004)*, Fredericton, Canada, May 19-21, 2004. pp. 357-61.

[8] P. Saint-Andre, "Extensible Messaging and Presence Protocol (XMPP): Core," IETF RFC 3920, 2004.

[9] R. Meijer, P. Millard, and P. Saint-Andre, *XEP-0060: Publish-Subscribe Version 1.12 Specification,* XMPP extension specification, Oct. 2008, http://xmpp.org/extensions/xep-0060.html.

[10] T. Bellwood, et al., *UDDI Version 3.0.2 Specification,* OASIS standard, Oct. 2006, http://www.uddi.org/pubs/uddi_v3.htm.

[11] S. Meguerdichian , F. Koushanfar, M. Potkonjak and M. Srivastava, "Coverage Problems in Wireless Ad-hoc Sensor Networks," in *INFOCOM 2001, Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies*, Anchorage, Alaska, 2001, vol. 3, pp. 1380-1387.

[12] OASIS Reference Model for Service Oriented Architecture 1.0, http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.pdf.

[13] Margaret Arney, Brad Cohen, Brad Medairy, "Impact of Advanced Collaborative Architectures on Intelligence Analysis," *2004 IEEE Aerospace Conference Proceedings*, March 2004.

[14] World Wide Web Consortium, http://www.w3.org/.

[15] Organization for the Advancement of Structured Information Standards, http://www.oasis-open.org/.

[16] J. Raymond, "Operations Group blazes new trail during Operation Burnt Frost," Peterson Air Force Base, Colorado Springs, Colorado, March 11, 2008.

[17] H. Bai, M. Atiquzzaman, and D. Lilja, "Wireless Sensor Network for Aircraft Health Monitoring," *Broadband Networks (BROADNETS'04)*, 2004, pp. 748 – 750.

BIOGRAPHIES

Eric Konieczny *is a Consultant with Booz Allen Hamilton, Inc., providing software development and technology consulting services to the federal government. His skill set and focus areas include Java/JEE software development, Service Oriented Architecture (SOA), and Semantic Web technologies. He is currently participating in several research and piloting activities, specifically involving innovative approaches to dynamic service discovery and dynamic service orchestration. He holds a BS in Computer Engineering from Virginia Polytechnic Institute.*

Ryan Ashcraft *is an Associate with Booz Allen Hamilton. He has nine years of professional experience in delivering technology consulting services to the government. He has spent much of his career specializing in Model Driven Architecture (MDA) systems while also evangelizing iterative software development lifecycle practices. Mr. Ashcraft is currently focusing on SOA, providing architectural direction and mentoring on several research efforts for Defense and Intelligence Community clients. He has a BS in Computer Science and Economics from Vanderbilt University.*

David Cunningham *is an Associate with Booz Allen Hamilton Inc. He has over 8 years of proven successes in the development of enterprise systems and architecture across the Department of Defense (DoD) and Intelligence Community (IC). Mr. Cunningham has hands-on experience leading all stages of application development efforts including requirements definition, architecture and design, development, testing, and production support for enterprise systems. He has a deep understanding of SOA methodologies and standards and leads initiatives across the DoD and IC to ensure interoperability across the two communities.*

Sandeep Maripuri *is a Senior Associate with Booz Allen Hamilton, Inc. He has over 10 years of professional experience in delivering technology consulting services to the government and industry, as well as significant experience in the COTS marketplace. His skill set and focus areas include applying advanced technologies, such as semantics-based technologies and grid computing to Net-Centric architectures. He is currently managing research efforts aimed at prototyping transformational, next-generation SOA solutions for Defense and Intelligence Community clients. He holds a BS in Mechanical Engineering, minor Computer Science, from the University of Illinois at Urbana – Champaign.*