Available online at www.sciencedirect.com
ScienceDirect
Computer Communications 30 (2007) 546–563
computer communications
www.elsevier.com/locate/comcom

OSDA: Open service discovery architecture for efficient cross-domain service provisioning

Noura Limam [a,*], Joanna Ziembicki [a], Reaz Ahmed [a], Youssef Iraqi [a], Dennis Tianshu Li [a], Raouf Boutaba [a], Fernando Cuervo [b]

[a] University of Waterloo, 200, University Avenue West, Waterloo, Ont., Canada N2L 3G1
[b] Alcatel Internetworking, Inc. 600 March road, Ottawa, Ont., Canada

Available online 11 January 2006

Abstract

Emerging service-oriented architectures are pushing towards on-demand and "on the fly" composition of applications and business processes. In order to support service composition, the underlying infrastructure must provide a facility for on-demand discovery of services and service components. Discovery becomes challenging when services span heterogenous and independently administrated domains. For inter-domain discovery to be achieved independently of domain-specific service discovery technologies, a middleware is needed to interface between the different discovery systems. In this paper, we present a novel open service discovery architecture (OSDA) designed to serve as an open, scalable and fault-tolerant middleware for cross-domain discovery. We demonstrate the implementation of OSDA using a set of mature technologies.

Keywords: Service discovery; Cross-domain service infrastructure; Peer-to-peer systems; XML-based communication; Web services

1. Introduction

The emergence of service-oriented architectures for enabling the composition of applications, business processes and services, creates a rich breeding ground for widely distributed applications that span heterogeneous networks and administrative domains.

Composing services on the fly requires an underlying service infrastructure that provides the facility of discovering resources and service components. Service discovery technologies (e.g., SLP, Jini, UPnP, etc.) are good potential candidates to serve as a basis for such an infrastructure.

In our previous work [1] we have studied in detail various current approaches to service discovery and have defined a set of design requirements for a large-scale cross-domain service discovery system. From these requirements we have derived evaluation criteria, against which we have compared existing service discovery systems. The comparison criteria include fault-tolerance, performance, scalability, interoperability with other discovery systems, platform independence, standardization and availability of a mature implementation. Our study has revealed that no existing discovery system satisfies all of the requirements, but many of them contain desirable characteristics.

The continuing proliferation of heterogeneous, inherently non-interoperable service discovery technologies creates an incentive for providing a middleware that would enable their interworking. Such a middleware would abstract domain-specific service discovery systems at the inter-domain level, while allowing cross-domain publication and discovery of services and resources.

Providing a common resource and service access interface over heterogeneous environments has been demonstrated by platform-independent middleware (e.g., CORBA, DCOM, Grid services, etc.). In this work, we wish to apply a similar concept to service discovery. In particular, we aim to design a common middleware architecture for seamless cross-domain service and resource discovery.

* Corresponding author.
E-mail address: nlimam@bbcr.uwaterloo.ca (N. Limam).

0140-3664/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2005.11.017

Our proposal, the open service discovery architecture (OSDA) provides a scalable and efficient model for cross-domain service discovery by allowing service providers and consumers to transparently discover services advertised outside their own domain boundaries using their domain-specific service discovery mechanisms. OSDA achieves this goal by establishing an inter-domain distributed information storage and querying model and a unified information representation.

OSDA incorporates some of the most useful elements of existing discovery systems and service infrastructures in the design of our proposed architecture. The flexibility and scalability of our system is ensured by its stateless, modular, loosely coupled components and the use of well-accepted, web-based technologies.

The remainder of the paper is organized as follows. In Section 2 we present the motivation behind the design of OSDA and the contribution of our work. Section 3 summarizes related works. In Section 4 we describe the high-level architecture design and our unified message formats, while Section 5 presents a detailed specification of the OSDA components. In the following Section 6 we present our choice of implementation technologies, demonstrate the system flow using a case scenario, and report on the status of our implementation. Section 7 concludes the paper, discussing unsolved issues and future works.

2. Motivation and contribution

In the illustrated case scenario above (Fig. 1), an administrator desires to establish an optical link (lightpath) between two campuses, University of Waterloo and University of Quebec, in order to link some workstations in the former to a server in the latter. Given that there is no direct path that links both campuses, an end-to-end lightpath must be composed across several domains, here ORION, CA*NET and RISQ, by cross-connecting several sub-paths. A highly desirable method of composing an end-to-end path on the fly is to have a service discovery mechanism that is capable of searching and retrieving every single existing lightpath that could compose the end-to-end path. This mechanism should operate over and across different networks and independently administered domains. Moreover, given that domains may already locally use heterogeneous service discovery mechanisms and different schemes to describe their deployed or owned lightpaths, the inter-domain discovery system should be capable to interoperate with domain-specific systems. Scalability and interoperation with other discovery systems are hence hard requirements for the envisioned system.

Over the past few years, many service discovery approaches have been proposed by academia (INS [2], INS/Twine [3], SSDS [4], Splendor [5] etc.) and industrial standardization bodies (Bluetooth SDP [6], SLP [7], UPnP [8], Jini [9], Salutation [10] and UDDI [11]). In our previous work [1], we have evaluated the feasibility of applying current discovery systems to our multidomain and large-scale context based on a set of evaluation criteria we deem essential for such a context. Although, these systems provide the same basic functionality of service discovery, they differ significantly in architecture, message exchange pattern, expected operating environment (e.g., mobile vs. stationary services) and service representation/description. These differences make their interoperation difficult. Besides, paying closer attention to the scalability, performance, fault-tolerance and platform independence aspects of these mechanisms, reveals most of the discovery systems are not suitable for large-scale deployment.

Service Location Protocol (SLP) and Jini are aimed at a single administrative domain, as found in enterprise networks. Both rely on dedicated directory entities for caching service advertisements. Several directory entities can co-exist in the same network, but they do not communicate with each other for routing queries, or for exchanging advertised information. These approaches do not scale with increasing number of discovery operations and suffer from central points of failure. To avoid reliance on static configurations, SLP and Jini use network-level multicasting for locating directory components. Such reliance not only affects performance, but also makes the systems network dependent. Moreover, both Jini and SLP have their own way of defining service handles (service URLs [12]) for SLP and RMI stub for Jini. These representations are too technology-specific, which makes interoperation difficult.

Universal Plug and Play (UPnP) and Salutation architectures aim to allow automatic joining and seamless

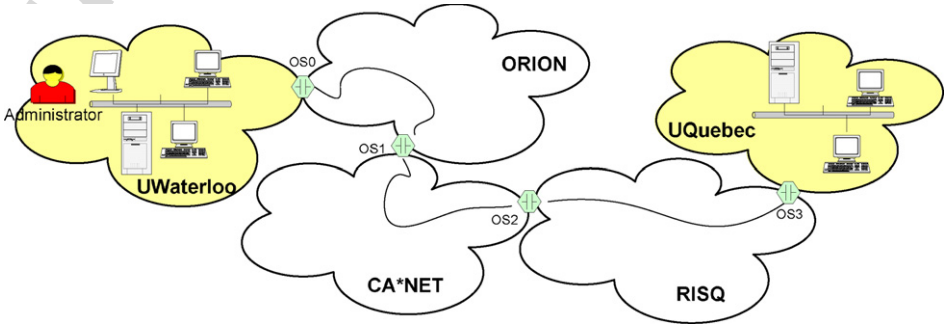

Fig. 1. Cross-domain lightpath composition.

discovery of network-enabled devices. Neither of them uses dedicated directory entities for advertisement caching, instead, they rely on multicast to discover other services, or to announce the presence of a service. Although, suitable for LAN operations, they do not scale well in large-scale networks and have significant performance issues. The benefit of UPnP is its use of standard web technologies (like XML, SOAP, HTML and HTTP) for service representation and communication, which could potentially enable interoperation with other technologies. Salutation uses an additional layer (i.e., the Salutation Manager) to achieve transport independence.

INS [2] expresses service descriptions as a tree-like hierarchy of descriptive attributes and values. Discovery is performed using an application-level overlay of directory entities. Advertisements are disseminated using a logical spanning tree topology, and replicated in each directory. Although the replication strategy handles fault-tolerance well, INS does not scale to large scale networks. Twine [3] was introduced to solve this scalability problem. Twine arranges directory entities in a logical ring and uses the Chord [13] Distributed Hash Table (DHT) to distribute advertisements over the directory entities. Because of the unique tree-like data representation, interworking INS and Twine with other service discovery protocols is difficult. However, we have found Twine's technique for creating index keys from hierarchical service descriptions to be an elegant and useful idea, and have taken advantage of it in the design of OSDA.

Secure Service Discovery Service (SDS) is intended for wide-area deployment and emphasizes security by allowing user/service level authentication and authorization. Public key and symmetric key encryption is used for communication privacy. SDS arranges directory entities into logical hierarchies, using Bloom Filters [14] to aggregate advertisement information. SDS can be used to perform service discovery across administrative domains, but the system does not address the tolerance to directory failures; SDS uses a tree-like hierarchy for index distribution which makes the system very sensitive to the failure of higher level nodes. In addition, SDS uses proprietary protocols for messaging and hence can hardly interoperate with other discovery system.

Universal Description, Discovery and Integration (UDDI) is a registry-based approach for Web Service discovery. The UDDI specifications defines SOAP APIs for querying and publishing service descriptions, an XML-based representation of the registry data model and interfaces in the Web Service Description Language (WSDL). The registry architecture is however left open. Although UDDI is mainly designed for large-scale deployment, its interoperation with current service discovery systems is not obvious. In fact, the interface-oriented service representation scheme (WSDL) does not support the description of generic service capabilities as opposed to all other discovery systems.

From these observations we conclude that none of the existing discovery systems match the requirements for inter-domain discovery. This leaded us to further study existing works on interworking discovery systems (see Section 3), and finally, to design our architecture for inter-domain service discovery.

3. Related works

As shown in Fig. 2, a number of works ([15–20] and [21]) have been conducted to enable interoperation between two different service discovery technologies. Each of these work employs a kind of "bridge" for protocol and data conversion.

Koponen and Virtanen [19] have presented an architecture for Jini and SLP interoperability. At the core of this architecture are a service broker and an adapter. The adapter has twofold functionality: it acts as directory service (i.e., *directory agent* for SLP and *lookup service* for Jini) and it registers services in other domains with the local directory service. An adapter captures local advertisements and forwards them to the broker. The broker in turn registers these advertisements with the directory service of each domain using the adapter in the respective domain. A client can discover services in remote domains, simply by querying its local directory service. This approach is not suitable for networks with a large number of domains, due to two reasons. First, all advertisements are mirrored in the directory service of each domain, which raises a scalability issue. Second, the broker is a single point of failure and a performance bottleneck.

Another work [18] presents an architecture for interworking Jini and UPnP, where virtual clients and services are placed in each domain. For a service that is discovered by a virtual client in one domain, a corresponding virtual service is created in the other domain. The virtual service registers itself to Jini Lookup Service (in Jini domain) or multicasts its existence (in UPnP domain). A client can discover and access a service in a remote domain using

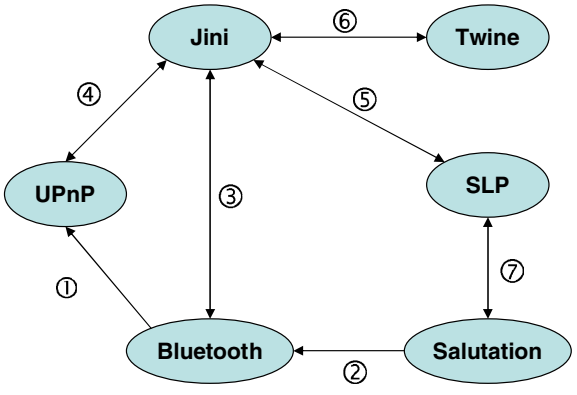

Fig. 2. Existing approaches for interworking service discovery technologies: (1) Bluetooth–UPnP [15], (2) Bluetooth–Salutation [16], (3) Bluetooth–Jini [17], (4) Jini–UPnP [18], (5) Jini–SLP [19], (6) Jini–Twine [20] and (7) Salutation–SLP [21].

the virtual service present in its own domain. This approach is not efficient for connecting a large number of domains since all the services of all domains are mirrored in each domain.

A different approach [20] for interworking Jini and Twine adds a proxy component between both domains. The proxy acts as a lookup service in the Jini domain and both as a client and service in the Twine domain. It forwards both advertisements and queries coming form the Jini domain to the Twine domain. Hence, Jini services are registered in the Twine domain, and while queries coming from Jini clients are solved both in the Jini and Twine domains, queries coming from Twine clients are resolved only in the Twine domain. Applying such an approach to different discovery systems, for instance UPnP instead of Twine, would assume that both domains are part of the same network, for instance local area network. Such an assumption is not suitable for service discovery in wide area networks.

In contrast to all the existing approaches to bridging service discovery systems, OSDA can operate on multiple domains with diverse discovery mechanisms, and can support a large number of services/users participating over a wide area network. For cross-technology service discovery, OSDA uses an open and interoperable service description scheme. OSDA is designed to be platform-independent, extensible and fault-tolerant and provides straightforward ways to introduce access-control policies.

4. Open service discovery architecture

In the remainder of the paper, a "domain" is defined as a federation of network components (users and services) controlled by a single service discovery technology. The main motivation of our work is the need to federate users and services spanning different domains whether they belong to the same or different networks. To this end, we build on the existing domain-specific discovery systems by providing the following facilities:

- As an alternative to mirroring shared services or broadcasting queries in all the involved domains and networks (which do not scale with an increasing number of domains and services), a **structured peer-to-peer overlay** is created as an inter-domain and inter-network space where shared services are advertised and queries are solved.
- Programmable **service brokers** are deployed in domains to act as an interface between the intra-domain and inter-domain discovery systems.
- As an alternative to converting service advertisements to all involved service description schemes, a **Unified Service Description scheme** is used for the advertisement of services in the inter-domain space.

In the following section, we will describe the high-level architecture of OSDA. Later, we will introduce USD, the Unified Service Description schema used to advertise and query services in our framework.

4.1. High-level architecture

In Fig. 3 we present a high-level overview of OSDA. The architecture assumes that the following local service discovery components are in place:

User Agent: acts as an interface between the end user and the discovery system. It is dependent of the domain-specific discovery technology.

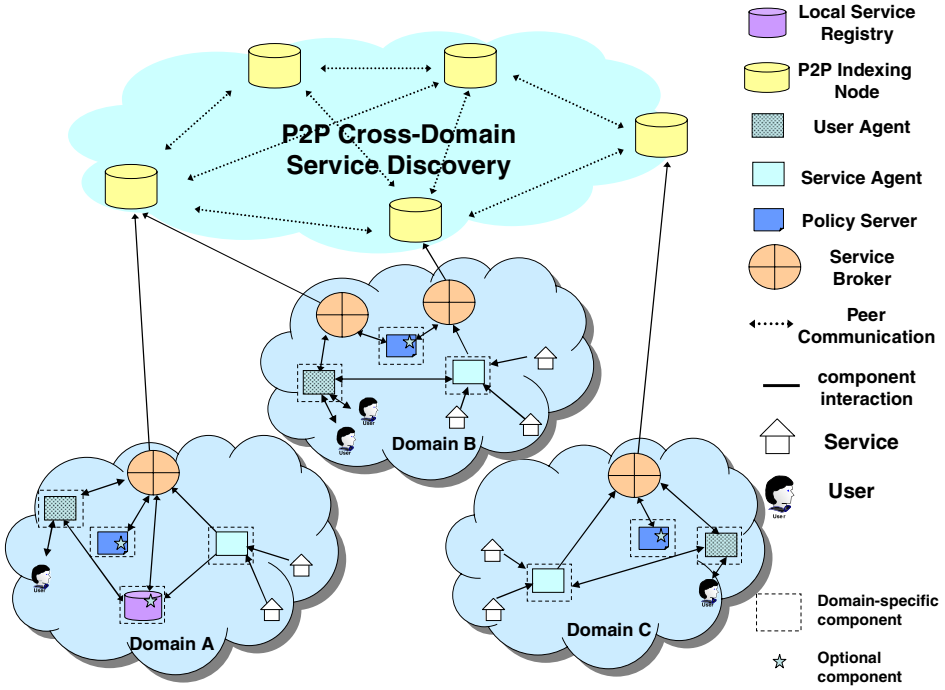

Fig. 3. OSDA: high-level architecture.

Service Agent: interfaces between the service provider and the discovery system. It handles service advertisement, advertisement renewals and/or de-registrations.

Local Service Registry: an optional component whose presence depends on the intra-domain service discovery mechanism. It is responsible for storing service records and responding to service discovery requests.

It is worth noting that there can be zero or more service registries in a domain depending on the service discovery mechanism used in the domain.

- If the domain uses a fully decentralized local service discovery mechanism (such as UPnP), then there may be no Service Registry in the system.
- If a directory-based local service discovery mechanism is used in the do main, then one or more than one service registry can be used in the system.

Policy Server: is an optional component responsible for providing discovery policies that control the publication of service advertisements and service queries outside the domain boundaries. These policies are domain-specific and are implemented by domain administrators.

As stated before, we introduce in each domain one or more than one service brokers and build a peer-to-peer indexing network in the inter-domain space. The canonical setup would involve one broker per domain and a one-to-one association between brokers and indexing peer nodes, however, the architecture supports multiple brokers per domain and many-to-many association between brokers and peers.

The following components are integral parts of our system:

Service Broker: is responsible for handling and processing cross-domain advertisements and queries. It acts as an interface between the local discovery technology and the inter-domain discovery system. The broker can be divided into two layers; a technology-dependent lower layer and a technology-independent upper layer.

The technology-dependent layer (referred to as an *Adapter*) abstracts the local service discovery system. It intercepts and processes requests coming from User Agents and Service Agents and converts advertisements and queries to a well-defined service description and query language. It also processes requests coming from the inter-domain system and executes them in the local discovery system.

The technology-independent layer handles broker-to-peer and broker-to-broker communication. It provides the necessary interfaces for the broker to be accessed by the entities involved in the inter-domain discovery process.

Note: Since there is a fairly clear separation between the technology-dependent and the technology-independent layers of the broker, we will occasionally refer to them as two separate entities: the *Adapter* and the *Broker*, respectively.

Peer-to-peer Indexing Node: is responsible for distributing the service information in the peer-to-peer discovery network and allowing the discovery of services in multidomain networks. The peer-to-peer network uses a Distributed Hash Table (DHT)-based architecture to store service information and solve queries.

Our system supports the two main functions of a service discovery system: *advertisement* and *querying*. Below, we give a brief outline of these functions. We use the case scenario described in Section 2 to better illustrate these functions. A more detailed description can be found in Section 5.

Advertisement: In the example, lightpaths are provided in the ORION, RISQ and CA*NET domains and advertised to users in each domain's service discovery technology. Upon the interception of a lightpath advertisement, the adapter will look into the domain's policies to see whether or not the service can be advertised outside the domain boundaries, and eventually to filter on the information that will not be published (e.g., security-sensitive information like the access point to the service).

Assuming that lightpaths can be leased to non-local users, the adapter will first map the description of the lightpath service to a "public" lightpath service template. This latter is either provided by the ORION domain, RISQ domain, CA*NET domain or any other domain to service as a template for describing lightpath services. The adapter composes an inter-domain advertisement based on the resulting lightpath description and additional meta-information, like the expiry time of the service, and sends it to the broker.

The broker will complement the advertisement with additional meta-information, typically its URL and the URLs of those brokers responsible for processing queries that relate to the advertised service. The advertisement is then sent to the peer-to-peer overlay, which distributes the service information among selected nodes.

Querying: Querying in OSDA is a two-step process. First, upon interception of the administrator's query for a lightpath service to connect both campuses, the adapter in the UWaterloo network looks into the domain's policy in order to decide to publish or not the administrator's query in the inter-domain space.

The administrator query is expressed in the UWaterloo domain's discovery technology and is meant to be mapped against any advertisement that describes a lightpath service with a specific (as apposed to any) template. Similarly to advertisements, the query will be mapped to a "public" lightpath service template, converted to a well-defined format and then sent to the broker.

The broker forwards the query to a peer indexing node so that it is resolved in the peer-to-peer overlay. It will receive back the set of broker URLs that have been advertised along with the lightpath services matching the query. Later, this same broker contacts one or more brokers from the received list in order to retrieve the whole information about the offered lightpath service, especially its access point. This second step involves the contacted broker to execute queries in the local discovery system.

Splitting up the querying process in this way allows the domains to control which parts of service information are advertised to the world at large, and which parts are made available only to selected domains. It also allows the domains to control the amount of data forwarded to the peer-to-peer overlay by sending a single inter-domain advertisement for a set of similar services (aggregation).

The resulting architecture acts as a unifying "glue" that connects diverse local service discovery systems such as SLP, Jini or Salutation. The multidomain service discovery system *does not depend on the local service discovery mechanism* (i.e., both centralized and decentralized service discovery approaches are supported). If a directory-based local approach is used, then the service registry will be contacted for local service discovery. Otherwise, multicast or broadcast messages will be sent for local discovery.

4.2. Service naming, description and querying

Each service discovery system has its own way of describing a service. Supporting interoperability among a variety of service discovery systems needs some means of vocabulary translation. Vocabulary translation can be carried out directly from one technology to the other: a Jini advertisement (or service description) can be converted to an equivalent SLP advertisement and advertised in the SLP system. However such a scheme would require $O(N^2)$ mappings, where N is the number of supported service discovery technologies. Clearly, the preferred method would be to design an intermediate, unified scheme for inter-domain advertisements. In this case, an advertisement from a particular domain can be translated to the agreed inter-domain format and can then be advertised in an other technology after another step of conversion. Because of its expressive power and acceptance in the Internet community, XML seems to be most appropriate as a basis for such a format, while the selection of appropriate elements for the XML description needs more detailed analysis of the existing service discovery technologies.

To achieve this kind of interoperability, we propose the Unified Service description (USD), a meta-service description schema that can interoperate with most of the major service discovery systems.

As shown in Fig. 4, the USD scheme consists of two major nested parts. The main envelope contains the meta-information for the advertisement, such as the type, location and expiry time of the service. The *description* component specifies the properties and capabilities of the service itself. Table 1 summarizes the fields contained in the USD scheme.

For better understanding of the discovery process, let us focus on two key fields of the Unified Service Description: *serviceID* and *description.*

serviceID: In OSDA, a service identifier consists typically of two parts: a globally unique domain identifier, and a domain-unique service identifier. Together, these two parts form a globally unique identifier for each service, obviating

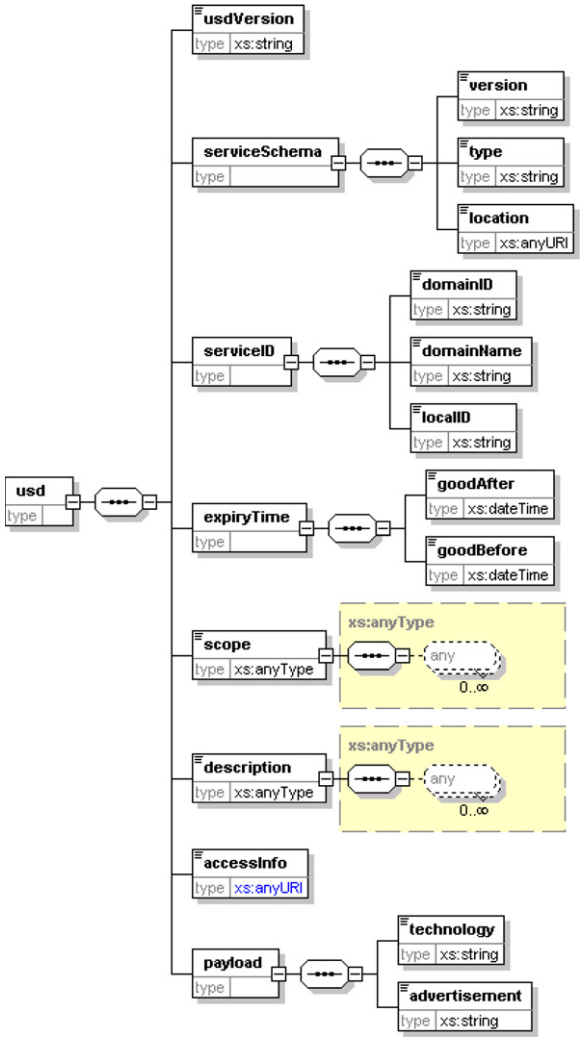

Fig. 4. USD: Unified Service Description.

the need for a central naming authority that assigns names to each participating service. The format of the service identifier is left up to the domain administrator, giving each domain discretion in naming its services. The domain identifier is more difficult to define, since it must be globally unique. In the case where "domain" means simply "the administrative domain", we recommend the use of DNS names [22]. In other cases, uniqueness can be accomplished by using a Universal Unique Identifier (UUID) [23]. If a non-human-readable domain identifier is used, such as the UUID, we recommend the use of the *domainName* field to include a human-readable domain name with the USD.

description: OSDA supports any description schema (or service template) provided it is written in XML. Typically, service templates are generated by service providers and may be commonly used by a number of service providers as the "standard" template for inter-domain advertisements.

In addition to the meta-service description scheme, we also design two XML message formats: *Unified Request* and *Unified Response* used, respectively, to wrap inter-domain advertisement or discovery requests, and discovery responses.

Table 1
Unified Service Description

Field	Description
usdVersion	Specifies the USD version from which the service template was derived
serviceSchema	Refers to the XML Schema Definition (XSD) which serves as a template for the service description **Version**: specifies the version of the service template. It allows incremental upgrading of service descriptions with backward compatibility **Type**: a gross category of the service referring to the service template **Location**: a URI specifying the location of the service template XSD document
serviceID	The service identifier used to globally and uniquely identify a service. It contains the following information: **domainID**: unique identifier for the domain (can be non-human-readable) **domainName**: human-friendly domain name supplied by the system administrator **localID**: the name used to uniquely identify a service within a domain
expiryTime	The time when an advertisement expires. It consists of two fields: **goodAfter**: time by which the advertisement starts to be valid **goodBefore**: time by which the advertisement is no longer valid
scope	May contain (a reference) the list of domains that are allowed to discover the service. It allows domain level access-control for broker-to-broker communication
description	The capability description of a service. It is a set of attribute–value pairs in a hierarchical relationship. We propose the use of XML Schema Definition (XSD) for describing the capability template (analogous to the service template in SLP or UPnP)
accessInfo	A URL that can be used to invoke a service. The URL may point to a static document (e.g., WSDL document) that describes the interfaces to access the service, or embed the request needed for obtaining the service access point. The URL may also point to an intermediate entity (e.g., proxy) that mediates the invocation syntax and semantics
payload	Used to provide the client with the original description of a discovered service as advertised in its domain. The payload information consists of two components: **technology** refers to the local discovery technology (e.g., Jini, SLP, etc.) **advertisement** the description of the service as advertised in its domain The payload information may allow a client aware of a specific technology to perform technology-specific operations on the discovered service

5. Module specification

As described in Section 4, OSDA is comprised of several components. Each of these components consists, in turn, of a number of smaller modules. The high-level objective of the design of OSDA modules is maximizing the system modularity through a clean separation of responsibilities between components and a clean identification of functionalities for each component in order to reducing maintenance effort and code duplication.

In the following section, we will examine the module-level design of OSDA (see Fig. 5), by explaining the functionalities and interfaces of the individual modules.

5.1. Adapter modules

The Adapter is composed of the following four modules: the *Registration Advertisement Handler*, the *Discovery Request Handler*, the *Directory Handler* and the *Converter*.

(1) *Registration Advertisement Handler.* The Registration Advertisement Handler is responsible for processing advertisement requests coming from local Service Agents. Upon interception of an advertisement, the Registration Advertisement Handler contacts the policy server (if present) so that related domain policies are applied. If the advertisement is allowed to be propagated in the inter-domain discovery system then the Advertisement Registration Handler first converts it to the USD format and then submits it to the Broker's Advertisement Propagator.This process is illustrated in Fig. 6. Note that the policing step is omitted.

(2) *Discovery Request Handler.* This module is responsible for intercepting and handling service discovery requests. It steers queries either to the local or global discovery systems according to the domain policies. If a service discovery request is allowed to be propagated to the inter-domain discovery system, it is first converted to the USD-based query format, and then submitted to the Broker's Global Discovery Handler (see Fig. 7). Similarly, upon reception of a response to a query, the response is first converted to the local description format and then forwarded to the User Agent that generated the query (see Fig. 8).

(3) *Directory Handler.* The Directory Handler is responsible for handling Broker's service discovery requests in the second step of the querying process (see Section 4). It takes care of converting queries into the local query format and generating discovery requests in the local discovery system (see Fig. 8). The Directory Handler implements a user agent interface; generated queries are either sent to the service repository or multicasted/brodcasted over the network depending on the discovery mechanism deployed in the domain.

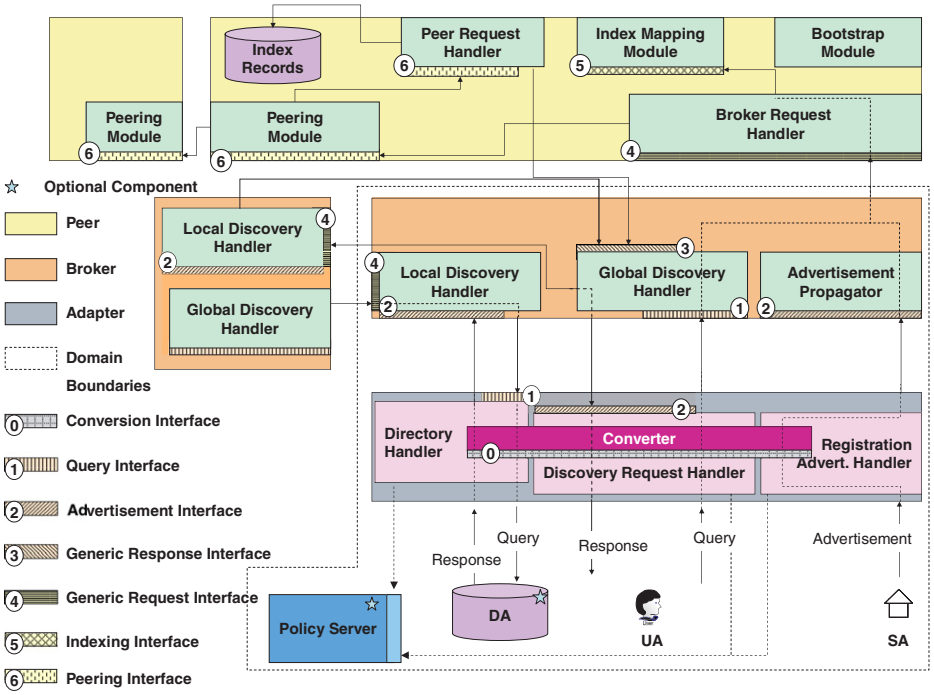

Fig. 5. OSDA: architecture specification.

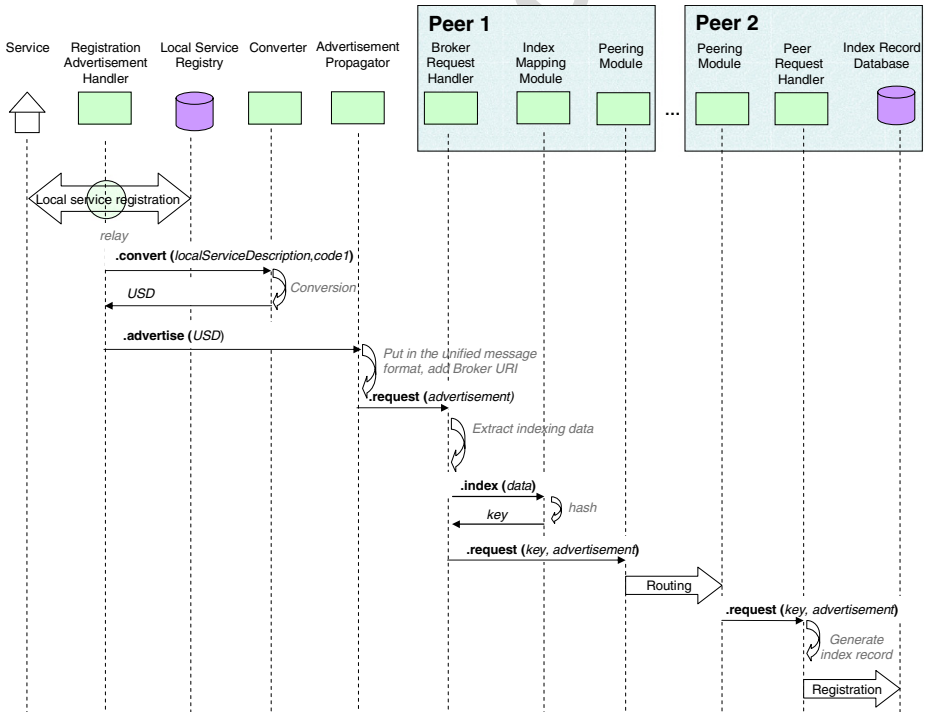

Fig. 6. Sequence diagram: service advertisement.

(4) *Converter.* The Converter module is used to convert queries and service descriptions into a USD-based format, or from the USD-format to the local format. A mapping between local service templates and USD templates as well as a mapping between the local query format and the Unified Query format are both required in the conversion process.

5.2. Broker modules

The Broker is responsible for handling cross-domain advertisement and service discovery requests. It provides well-defined interfaces that are invoked to post requests and responses. The broker is composed of the following three modules: *Advertisement Propa-*

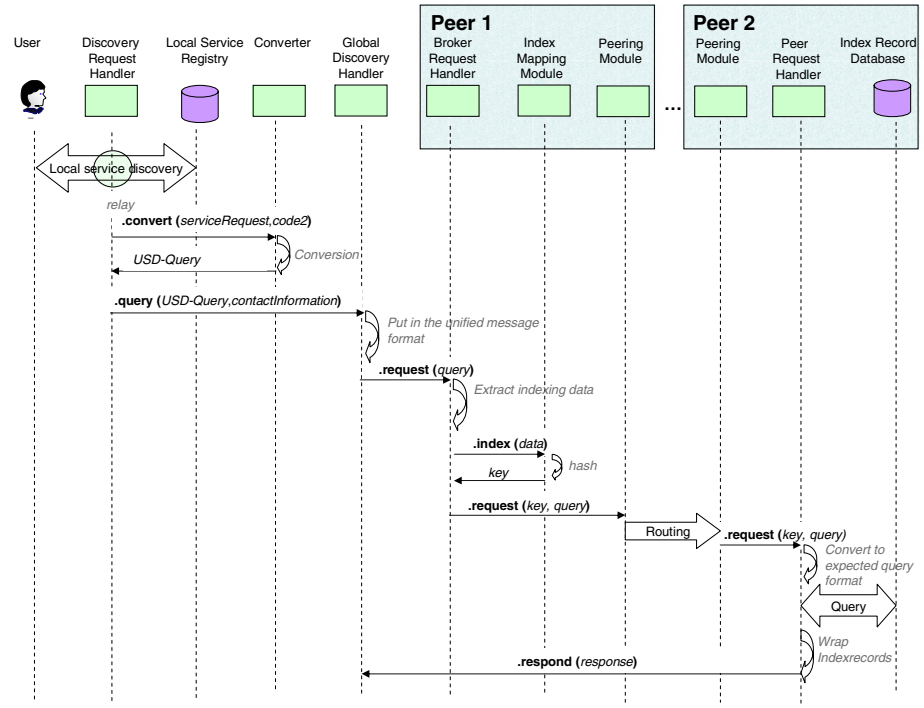

Fig. 7. Sequence diagram: service discovery, step 1.

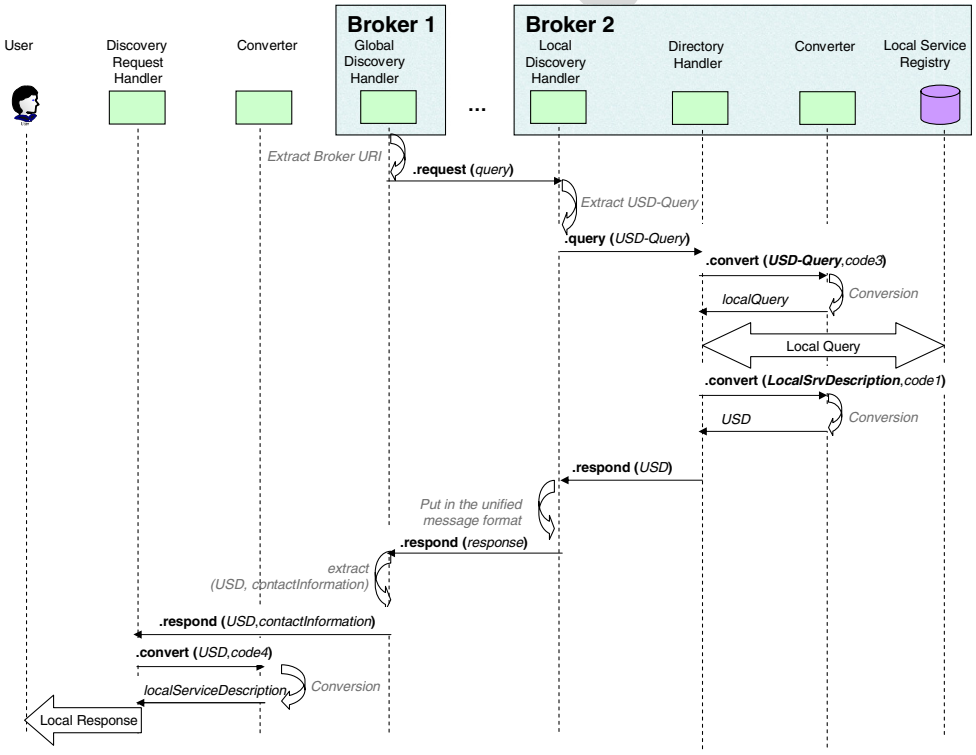

Fig. 8. Sequence diagram: service discovery, step 2.

gator, *Global Discovery Handler* and *Local Discovery Handler*.

(1) *Advertisement Propagator.* The Advertisement Propagator is responsible for propagating advertisements to a peer-to-peer indexing node (see Fig. 6).

(2) *Global Discovery Handler.* This module is responsible for processing inter-domain service discovery requests. As stated in Section 4, inter-domain service discovery is achieved in two steps. In the first step (see Fig. 7), the Global Discovery Handler is responsible of the propagation of service discovery requests to the peer-to-peer

network. In the next step (see Fig. 8), the Global Discovery Handler sends the same request to one or more than one broker listed in the response of the peer-to-peer network. A selection mechanism over the received list may be implemented. Responses from the contacted brokers will contain the USD of services that match the discovery request. As discussed in Section 6, the Global Discovery Handler can also be contacted directly, by services and clients, without having to go through a pre-existing local discovery system.It is worth noting that the number of contacted brokers influences both the inter-domain query overhead and the completeness of the query result.

(3) *Local Discovery Handler.* The Local Discovery Handler is responsible for processing the requests sent by remote Global Discovery Handlers. Queries embedded in the received requests are sent to the Directory Handler in order to be solved into the USD of matching services. USDs are sent back to the Global Discovery Handlers that generated the requests.

5.3. Peer node modules

The network of OSDA peer nodes forms a distributed hash table. The nodes are responsible for storing index records corresponding to previous inter-domain advertisements, and for solving cross-domain queries. A peer node is composed of a *Broker Request Handler*, an *Index Mapping Module*, a *Peering Module*, a *Peer Request Handler* and a *Bootstrap Module.*

(1) *Broker Request Handler.* The Broker Request Handler is the access point to the peer-to-peer network. It intercepts and processes inter-domain advertisement and query requests sent by Brokers and then directs them in the peer-to-peer network. Given a service advertisement or query, the Broker Request Handler first internally extract from the capability description section the data that will serve to index the request. The data is then passed on to the Index Mapping module, to convert it into one or more hash keys. Each hash key, along with the original request, is passed to the Peering Module in order to be routed to the appropriate peer.

(2) *Index Mapping Module.* The Index Mapping Module uses a hash function to convert the data received from the Broker Request Handler to a key. The hash is returned back to the Broker Request Handler.

(3) *Peering Module.* Each peer in the peer-to-peer overlay is responsible for a set of keys. Upon receiving a key–request pair, the Peering Module routes the request to the appropriate peer. If the given key is one of this peer's own keys, the key–request pair is passed to the local Peer Request Handler.

(4) *Peer Request Handler.* A peer-to-peer node maintains a database which associates each hash key with one or more than one index record.Upon receiving a request–key pair, the Peer Request Handler either executes the contained query or stores the contained advertisement in the index record database within the set of index records associated with the given key. Before being executed, a query is first converted to the format supported by the database.The advertisements stored in the peer-to-peer network are soft-state: each stored index record contains the information about its lifetime. While the advertisement renewal process is handled by the Discovery Request Handler in the domain level, cleaning the peer database from expired index records must be handled by the Peer Request Handler if not supported by the used database management system.

(5) *Bootstrap Module.* The Bootstrap Module is responsible for joining the peer-to-peer network. The join process is specific to the DHT architecture used in the peer-to-peer network.

It is worth noting that, the inter-component communication is asynchronous and the modules are stateless, i.e., they do not keep track of current requests. This enhances the system's tolerance to "short" failures or disconnections. For example, if multiple brokers are known by an adapter, they can be used interchangeably if one of them fails. Moreover, since information about the source of the query is embedded in each request, a broker can be restarted or replaced between sending the query and receiving the responses. If a broker goes down while waiting for query responses to arrive, those responses can be sent to an alternate broker (if known), or to the originating broker, if it becomes accessible in the meanwhile.

All OSDA components, and most of the modules are designed to be loosely bound. Contingent on access-control policies, an adapter can communicate with any broker in its domain (allowing several adapters per broker, or vice versa), while a broker can communicate with any peer indexing node. Such flexibility and degree of fault-tolerance is mandatory, since OSDA is designed to be a core supporting function for an Internet-scale network of resources and services.

6. Implementation and validation

The design of OSDA meets the main requirements for an inter-domain service discovery middleware.

- Scalability: the filtering and aggregation features of the service broker allows OSDA to scale with an increasing number of advertised services. In addition, the use of a structured peer-to-peer overlay, as opposed to unstructured systems, reasonably bounds advertisement and query routing at the inter-domain layer, and hence minimize the communication overhead.
- Openess: service brokers make OSDA a pluggable solution that does not require any change in the local discovery systems and that is extensible to new participating domains.

- Fault-tolerance: service registration in the peer-to-peer overlay is soft-state; advertisements are evicted if the corresponding services are not re-advertised. This way OSDA guarantees the consistence of query responses and is tolerant to service failures. In addition, given that brokers an peers are stateless and loosely coupled, OSDA is able to recover their failures.

The implemented OSDA prototype maximizes these features. In fact, in addition to choosing open and well-accepted technologies, we have designed standard interfaces and asynchronous messaging protocols. The resulting system has been tested an validated through some case scenarios.

6.1. Implementation technologies

As the name of our architecture would imply, its main goal is to be as open and universal as possible. We used this motivation as a guide in our choice of tools and technologies that would implement our system. We focussed on well-known, tested, freely available and open-source components such as JXTA [24], Chord [25], JBoss [26], INS/Twine [27], Jetty [28] and web-based technologies such as SOAP. Because of the need for platform independence, we used Java as the programming language, HTTP as the transport protocol for broker-to-broker and broker-to-peer communications and XML as the format for all communications. In addition to greatly simplifying the implementation process (in comparison to re-inventing the communication/data format wheel), using primarily web-based technologies allowed us to create a flexible and modular system. If needed, many of the components, such as the index record database, or the peer-to-peer routing mechanism can be relatively easily replaced with other technologies. Moreover, because of the loose coupling between components, the individual parts of the system can be flexibly deployed on separate systems or even all on the same machine, allowing OSDA to scale gracefully in face of increasing load.

- *Broker*: The broker functions as a standalone Enterprise Java server. The broker modules are implemented as stateless-session Enterprise Java Beans (EJBs) running in the JBoss application server, and deployed as Web Services. This way, the broker components may be invoked either using RMI/IIOP or SOAP/HTTP. Currently, Local and Global Discovery Handlers are accessed by peers and other brokers, cross-domain boundaries, through SOAP/HTTP which offers many attractive advantages like the support of security mechanisms and the ability to work through firewalls.One of the most interesting advantages of this implementation is that it does not require a local service discovery system to be deployed in a domain. Because the Global Discovery Handler is implemented as a Web Service, a web client can be easily created and then used to discover services in other domains.
- **Peer Nodes**:
  (1) *Broker Request Handler:* The Broker Request Handler is implemented as a SOAP service and runs on top of the lightweight Jetty HTTP server. We use INS/Twine libraries to extract from advertisements and queries the data that will be used to generate indexing keys. The service capabilities part of advertisements is split into strands, using the INS/Twine model, and all strands are hashed into keys by the Index Mapping Module (see Fig. 13). Advertisements are routed to and then stored/replicated in each peer corresponding to one of the resulting hash key. On the other hand, queries are forwarded to the peer that is associated to the key resulting from the longest strand.
  (2) *Index Mapping Module:* The Index Mapping Module uses an MD5 [29] hash to turn strands into hash keys later used for routing.
  (3) *Peering Module:* The Peering Module is implemented as a JXTA peer to take advantage of the bootstrapping, authentication, and secure communication facilities of the JXTA environment. Peering Modules are organized in a Chord ring, where Chord is used to route requests between peers.We chose Chord as a base for peer-to-peer network because of its fault-tolerance and self-stabilizing properties as well as its effective method of evenly distributing information and query process load. This design is necessary in dealing with the large volume of advertisement messages and query operations expected in an Internet-scale network. Below, we provide a short analysis of the query costs in the OSDA implementation.Unlike broadcast-based approaches, the Chord DHT guarantees an upper bound of $O(\log N)$ hops between peers for each routed request, where N is the number of peers in the overlay. After being advertised locally, each advertisement generates k strands, and hence k keys to be distributed among the peer-to-peer nodes, resulting in $O(k \log N)$ messages. The advertisement overhead therefore combines the cost local advertisement with the peer-to-peer advertisement overhead. Additionally, since the peer-to-peer layer uses a soft-state approach, each advertisement must be periodically re-advertised to remain valid, which makes the total overhead dependent on the frequency of advertisement renewal. Because a query generates a single strand, the peer-to-peer query overhead is lower at $O(\log N)$ messages.
  (4) *Peer Request Handler:* The Peer Request Handler stores and retrieves XML-based index records to and from a database. For this purpose, we used the eXist [30] open-source native-XML database. Like JXTA, and our SOAP interfaces, eXist relies on the lightweight Jetty HTTP server [28], and includes many useful features such as a web inter-

face and XPath/XQuery [31] processing. While the canonical INS/Twine search would return all service descriptions corresponding to the given key, the Peer Request Handler goes a step further by converting the original query (possibly containing complex predicates – currently, XSet [32] range predicates are supported) into an XQuery. The XQuery is then performed against the index records corresponding to the given key, returning only the relevant set of matching documents.

6.2. Testbed and implementation

Currently, our testbed consists of 2 Linux PCs, each running a broker and an SLP daemon, and 3 Sun Solaris 8 machines each running a peer indexing node. We have implemented an SLP adapter which acts as a virtual SLP Directory Agent for intercepting advertisements and queries coming respectively from users and services. The intercepted messages are then forwarded to both the real Directory Agent and the broker. The SLP adapter also runs a process which intercepts queries coming from the broker at the second step of inter-domain queries. Responses to queries coming both from the local Directory Agent and from the broker are grouped and then forwarded to users. We have also implemented a web portal for issuing advertisements and queries directly to the broker, and some java beans that act as an adapter between the portal JSP pages and the broker. We use the portal for simulating a non-SLP domain. Using this setup, we have successfully validated the advertisement and inter-domain query processes.

Because cross-domain service discovery is such a complex topic, we were clearly not able to address all the related issues in the current implementation. For example, in the current implementation of OSDA a broker is statically configured to contact a specific peer in the peer-to-peer network. We intend to work on implementing a mechanism that allows brokers to dynamically find a peer node in the peer-to-peer network. Moreover, we are searching the problem of including more sophisticated hash function in the peer-to-peer network that would enable complex and simple advertisements/queries to be handled equally. Currently, complex advertisements and queries are simplified, for existing hash functions do not provide any mapping between the hash of a range-value and the hash of values contained in the range. For example, our DHT does not support matching a query containing $attribute > x$ to a previous advertisement containing $attribute = x + 1$. This functionality is instead provided at the database level, by converting the previously ignored range query predicates into an XQuery, and using it to narrow down search results.

6.3. Case scenario

As a proof of concept of our system, we have simulated on our testbed the following case scenario (see Fig. 9) inspired by the cross-domain lightpath establishment use case (Section 2).

In this scenario, the CA*NET domain consists in one of the Linux PCs running an SLP daemon and a broker. The UW domain consists in the second Linux PC that runs a broker, on which we have installed the web portal to simulate a non-SLP domain.

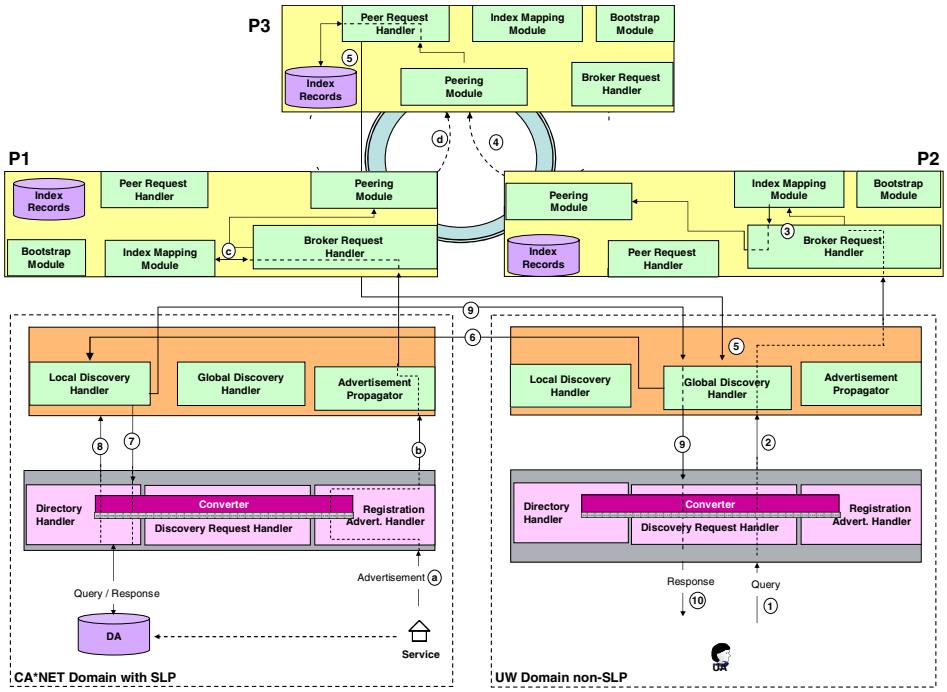

Fig. 9. Implementation use case.

We have designed an SLP template for the lightpath service that will be used by the CA*NET domain to advertise a number of lightpaths with different descriptions (see Fig. 10), i.e., offering different bandwidth, linking different optical switches, etc. For simplicity, we will closely follow the SLP template in creating the USD template of the lightpath service (see Fig. 11).

In the remainder of this section, we will describe the message flow involved in both the advertisement and discovery processes.

6.3.1. Advertisement

Upon intercepting (step (a)) the SLP advertisement, the adapter converts it to the USD format as shown in Fig. 12.

As we can see, there is a clear mapping between a number of USD tags and SLP advertisement fields. For example, the *LocalID* field of USD template is mapped to the *URL* field of the SLP advertisement message, and the value associated to *goodBefore* is calculated based on the current time, which is given to the *goodAfter* field, and the value associated to the *Lifetime* field of the SLP advertisement. Note that here we have considered a simple advertisement, i.e., an advertisement that does not contain any complex value (no ranges or enumerations).

In step (b) the *AdvertisementPropagator* EJB is invoked and the USD-based description of the lightpath service is sent. The *description* part of the USD is extracted along with the service *type*, *domainID* and *expiryTime*, all are wrapped in a *Unified Request*.

In step (c), the Unified Request is sent to the peer. The Broker Request Handler is invoked to process the request. Since the advertisement is not complex, it is directly turned into a tree and split into INS/Twine-like strands (see Fig. 13). Note that the USD service type is always set as a root in the strands. This is done on purpose so that every generated key is more likely to belong exclusively to a specific service type.

In step (d) keys along with the corresponding Unified Request are routed in the Chord ring. Here, the peer node P3 is responsible of K1 (Fig. 13), it will extract the index record from the Unified Request and store it in the database.

6.3.2. Query

On request from the administrator, the web portal will load the USD template of the lightpath service to assist the administrator in formulating the query (see Fig. 14).

Version = 2.0
Function = srvReg
XID = 78643
Language = en-US
URL = service:lightPath:http://marakech.uwaterloo.ca:8888/axis/lightPathManager.jws
Lifetime = 10800
Scope = DEFAULT
Attributes =
(source = OS1),
(destination = OS2),
(bandwidth = 524288),
(owner = CA*NET),
(lease-expiry-time = 2005-11-01T00:00:00.000-0500),
(bidirection = true)

Fig. 10. SLP advertisement for the lightpath service.

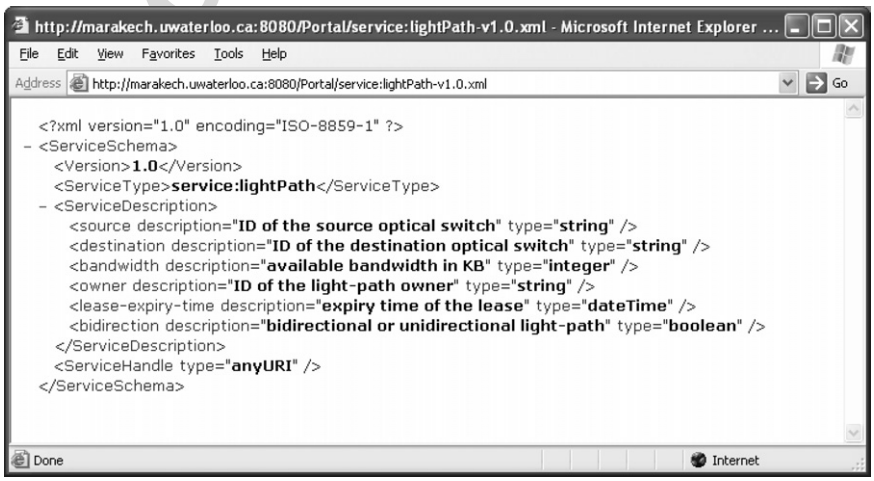

Fig. 11. USD template for the lightpath service.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <Usd>
  <UsdVersion>1.0</UsdVersion>
- <ServiceSchema>
  <version>1.0</version>
  <type>service:lightPath</type>
  <location>http://marakech.uwaterloo.ca:8080/Portal/service:lightPath-v1.0.xml</location>
  </ServiceSchema>
- <ServiceID>
  <domainID>UUID:58f202ac-22cf-11d1-b12d-002035b29092</domainID>
  <domainName>CA*NET</domainName>
  <localID>OS1-OS2-lp0</localID>
  </ServiceID>
- <ExpiryTime>
  <goodAfter>2005-05-01T00:00:00.000-0500</goodAfter>
  <goodBefore>2005-05-02T00:00:00.000-0500</goodBefore>
  </ExpiryTime>
  <Scope>default</Scope>
- <ServiceDescription>
  <source>OS1</source>
  <destination>OS2</destination>
  <bandwidth>524288</bandwidth>
  <owner>CA*NET</owner>
  <lease-expiry-time>2005-11-01T00:00:00.000-0500</lease-expiry-time>
  <bidirection>true</bidirection>
  </ServiceDescription>
  <accessInfo>http://marakech.uwaterloo.ca:8888/axis/lightPathManager.jws?wsdl</accessInfo>
  </Usd>
```

Fig. 12. USD of the CA*NET lightpath service.

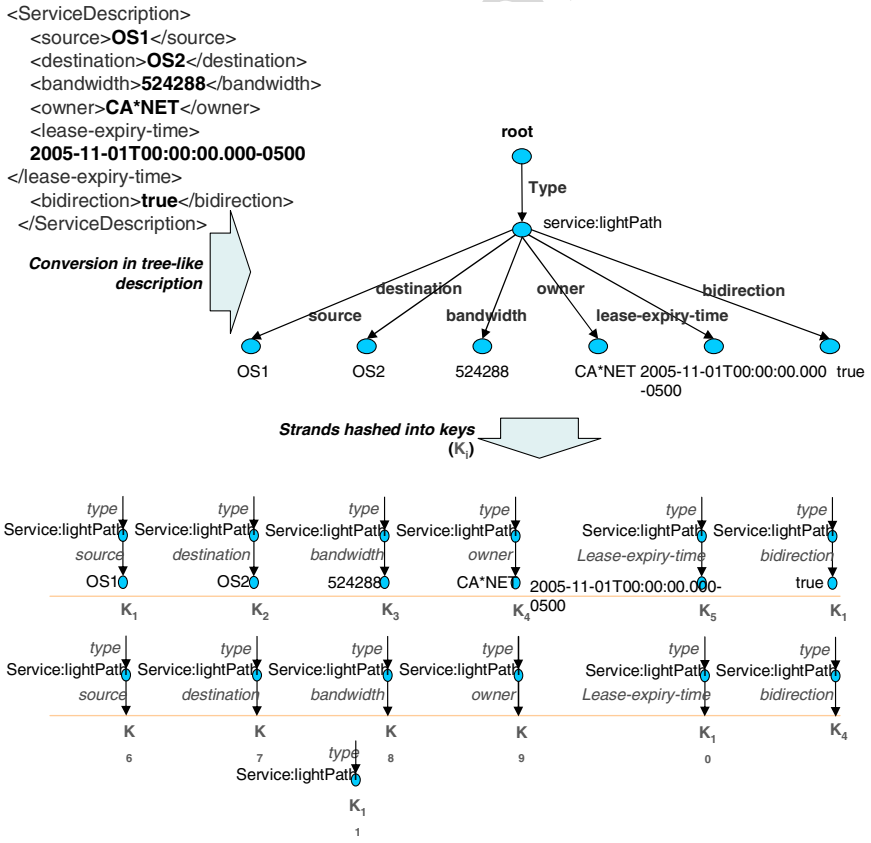

Fig. 13. Advertisement stranding.

In step (2) the query is wrapped into a Unified Request. Note that the request will contain the set of access points needed to route back the query response; namely the endpoint of the Global Discovery Handler (being deployed as a Web Service), followed by the access point to the Discovery Request Handler and a reference to the bean that is in charge of receiving the response to the query.

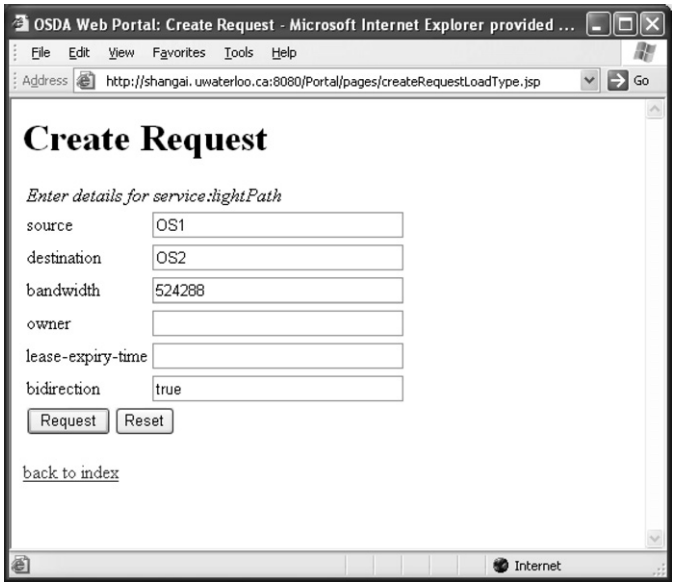

Fig. 14. Query for a lightpath service.

At the peer level (step (3)), the Broker Request Handler splits the query into strands; since the query does not contain any complex predicates such as ranges or enumerations, no intermediary query simplification is needed. The longest strand is chosen to be turned into a key. Assuming that the generated key K is equal to K1 (see Fig. 13), the Unified Request is then routed to P3 (step (4)) where it is resolved. As a result to the query, the database returns the Index Record of the lightpath service advertised by the CA*NET SLP domain. The Index Record along with the original Unified Request are wrapped in a Unified Response (see Fig. 15) and sent back to the Global Discovery Handler (step (5)). The same request is sent to the Local Discovery Handler whose endpoint is specified by the *BrokerURL* field of the received response, i.e., to the CA*NET SLP domain's broker (step (6)).

In step (7) the query is converted to the SLP query format (LDAP-filter) based on the mapping between the USD template of the lightpath service and the SLP *service*:*lightPath* template. The converted query is executed by the Directory Agent and, as a result, the exact service description that is shown in Fig. 10 is returned back. The conversion of this advertisement leads to the same USD as described in Fig. 12 (step (8)).

In step (9) the USD of the printing service is wrapped in a Unified Response along with the original Unified Request. The response is sent back to the Global Discovery

```
<?xml version="1.0" encoding="UTF-8" ?>
- <UnifiedResponse>
- <UnifiedRequest>
<Command>query</Command>
<MessageID>753792144</MessageID>
- <Source>
<ContactInfo>http://shanghai.uwaterloo.ca:8088:jboss-net/services</ContactInfo>
<ContactInfo>shanghai.uwaterloo.ca:33636</ContactInfo>
</Source>
- <Content>
- <ServiceDescription>
<Type>service:lightPath</Type>
<source>OS1</source>
<destination>OS2</destination>
<bandwidth>524288</bandwidth>
<bidirection>true</bidirection>
</ServiceDescription>
</Content>
</UnifiedRequest>
- <Results>
- <Content>
<DomainID />
<ExpiryTime>2005-05-02T00:00:00.000-0500</ExpiryTime>
<BrokerURL />
- <ServiceDescription>
<Type>service:lightPath</Type>
<source>OS1</source>
<destination>OS2</destination>
<bandwidth>524288</bandwidth>
<owner>CA*NET</owner>
<lease-expiry-time>2005-11-01T00:00-00.000-0500</lease-expiry-time>
<bidirection>true</bidirection>
</ServiceDescription>
</Content>
</Results>
</UnifiedResponse>
```

Fig. 15. Unified Response: response from peer nodes.

Handler. Upon receiving this response, the Global Discovery Handler forwards the embedded USD to the user (here a binding bean of the web portal) whose access point appears in the *source* field of the request.

7. Conclusions

In this paper, we have presented OSDA, a novel cross-domain service discovery architecture. It allows the service providers and consumers seamless access to service and resource discovery across domains using local discovery mechanisms. We have used our previous in-depth study [1] of the existing service discovery technologies and the requirements for a large-scale, inter-domain service discovery system, as guideline to the design of OSDA.

The proposed architecture is designed to be scalable, extensible, efficient and robust. Its interoperation with local discovery mechanisms is enabled through programmable components (message interceptors and translators) and standalone brokers. Because of the loosely coupled nature of OSDA components, the system is able to evolve over time and gracefully recover from faults. The use of DHT-based peer-to-peer overlay for cross-domain discovery guarantees a bounded query response time and can manage large volumes of service advertisement and discovery operations.

In our initial system prototyping, we have succeeded in incorporating a set of mature technologies in an end-to-end OSDA implementation. Although the full implementation is not yet complete, the results are promising. Using well-defined standard interfaces (i.e., EJBs, Web Service and JXTA), communication protocols (i.e., SOAP and JXTA end-point communication pipes), and unified data representation (i.e., XML for messages and service descriptions), we can successfully bridge the differences among heterogeneous local discovery systems to provide cross-domain service discovery.

In addition to working on completing the OSDA system implementation, we are currently exploring the use of ontologies in defining and integrating service description vocabularies. Using Semantic Web tools such as Resource Description Framework (RDF) [33] and Web Ontology Language for Services (OWL-S) [34] at the cross-domain discovery-layer we aim to improve search correctness (by ensuring that the search terms are relevant to the domain of discourse through the use of URIs instead of keywords) and search completeness (by exploiting the use of synonyms and ontological relationships between search terms). Finally, we plan to exploit these ontological relationships to create a semantic peer-to-peer cross-domain overlay which would improve search performance.

Acknowledgements

This work has been sponsored in part by Alcatel Inc. and the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

[1] R. Ahmed, R. Boutaba, F. Cuervo, Y. Iraqi, D. Li, N. Limam, J. Xiao, J. Ziembicki, Service Discovery Protocols: A Comparative Study, in: Proceedings of IM 2005 (Application Session), May 15–18, 2005, Nice, France. URL: <http://bcr2.uwaterloo.ca/ălcatel/publications/IM05/IM05-discovery.pdf/>.

[2] W. Adjie-Winoto, E. Schwartz, H. Balakrishnan, J. Lilley, The Design and Implementation of an Intentional Naming System, in: Symposium on Operating Systems Principles, 1999, pp. 186–201. URL: <citeseer.nj.nec.com/adjie-winoto99design.html/>.

[3] Balazinska M., Balakrishnan H., D. Karger, INS/Twine: A scalable peer-to-peer architecture for intentional resource discovery, in: Proceedings of the First International Conference on Pervasive Computing, Springer-Verlag, 2002, pp. 195–210.

[4] S.E. Czerwinski, B.Y. Zhao, T.D. Hodes, A.D. Joseph, R.H. Katz, An architecture for a secure service discovery service, Mobile Computing and Networking (1999) 24–35, URL: <citeseer.ist.psu.edu/czerwinski99architecture.html/>.

[5] F. Zhu, M. Mutka,, L. Ni, Splendor: A secure, private, and location-aware service discovery protocol supporting mobile services, in: Proceedings of the First IEEE International Conference on Pervasive Computing and Communications (PerCom03), 2003, pp. 235–242. URL: <http://www.cse.msu.edu/z~hufeng/splendor.pdf/>.

[6] Bluetooth SIG, Specification of the Bluetooth System, Vol. 1, Core, Rev. 1.1, Tech. rep., Bluetooth SIG, 2001.

[7] E. Guttman, C. Perkins, J. Veizades, M. Day, RFC 2608: Service location protocol, version 2, status: Proposed Standard, 1999. URL: <http://www.ietf.org/rfc//>.

[8] UPnP Forum, UPnP device architecture 1.0 (May 2003). URL: <http://www.upnp.org/resources/documents/CleanUPnPDA101-20031202s.pdf/>.

[9] Sun Microsystems, Jini Technology Architectural Overview, Tech. rep., Sun Microsystem, Inc., 1999. URL: <http://wwws.sun.com/software/jini/whitepapers/architecture.pdf/>.

[10] Salutation Consortium, Salutation architecture specification (part-1), June 1999. URL: <ftp://ftp.salutation.org/salute//>.

[11] UDDI Consortium, UDDI Technical White Paper, 2002. URL: <http://www.uddi.org/pubs//>.

[12] E. Guttman, C. Perkins, J. Kempf, RFC 2609: Service Templates and Service: Schemes, June 1999.

[13] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, H. Balakrishnan, Chord: A scalable peer-to-peer lookup service for internet applications, in: Proceedings of the ACM SIGCOMM 2001 Conference, San Diego, CA, 2001, pp. 149–160. URL: <http://www.pdos.lcs.mit.edu/chord/papers/paperton.pdf/>.

[14] B. Bloom, Space/Time Trade-offs in Hash Coding with Allowable Errors, Communications of ACM 13 (7) (1970) 422–426.

[15] B. r. c. Palowireless, Extended service discovery profile for universal plug and play. URL: <http://www.palowireless.com/infotooth/tutorial//>.

[16] B. Miller, Bluetooth whitepaper, mapping salutation architecture apis to bluetooth service discovery layer, document number 1.C.118/1.0, version 1.0, 1999. URL: <http://www.salutation.org/whitepaper/BtoothMapping.PDF/>.

[17] S. Kasper, L. Bhrer, Jini discovers Bluetooth, semester Thesis SA-2002.30, Institut fr Technische Informatik und Kommunikationsnetze, 2002. URL: <http://www.tik.ee.ethz.ch/beutel/projects/sada//>.

[18] J. Allard, V. Chinta, S. Gundala, G.R. III, Jini meets upnp: An architecture for jini/upnp interoperability, Department of Computer Science, University of New Orleans.

[19] T. Koponen, T. Virtanen, A service discovery: A service broker approach, in: Proceedings of the 37th Hawaii International Conference on System Sciences, 2004. URL: <http://csdl.computer.org/comp/proceedings/hicss/2004/2056/09/205690284b.pdf/>.

[20] S.R. Livingstone, Service Discovery in Pervasive Systems, The School of Information Technology and Electrical Engineering, University of

Queensland, Autralia, 2003, URL: <http://innovexpo.itee.uq.edu.au/2003/exhibits/s370816//>.
[21] P.S. Pierre, T. Mori, Salutation and SLP, the Salutation Consortium. URL: <http://www.salutation.org/techtalk/slp.htm/>.
[22] P.V. Mockapetris, RFC 1035: Domain names – implementation and specification (November 1987). URL <http://www.ietf.org/rfc/rfc1035.txt/>.
[23] P.J. Leach, R. Salz, UUIDs and GUIDs, status: Internet-Draft, February 1998. URL: <http://hegel.ittc.ukans.edu/topics/internet/internet-drafts/draft-l/draft-leach-uuids-guids-01.txt/>.
[24] Project JXTA, JXTA J2SE 2.2.1, 2004. URL: <http://platform.jxta.org/java/release/2004Q1Churrasco/releasenote.html/>.
[25] I. Stoica, R. Morris, D. Liben-Nowell, D.R. Karger, M.F. Kaashoek, F. Dabek, H. Balakrishnan, Chord: a scalable peer-to-peer lookup protocol for internet applications, IEEE/ACM Transactions on Networking (TON) 11 (1) (2003) 17–32.
[26] JBoss Inc., Professional Open Source from JBoss Inc., 2004. URL: <http://www.jboss.com//>.
[27] MIT, INS/Twine v2, 2002. URL: <http://nms.lcs.mit.edu/software/instwine/ins-2-0.tgz/>.
[28] Jetty java http servlet server. URL: <http://jetty.mortbay.org/jetty//>.
[29] R. Rivest, RFC 1321: The MD5 Message-Digest Algorithm, 1992.
[30] eXist: Open source native xml database. URL: <http://exist.sourceforge.net//>.
[31] W.W.W.C. W3C, Xquery, 2004. URL: <http://www.w3.org/XML/Query/>.
[32] B. Zhao, The Xset XML search engine and XBench XML query benchmark, Tech. Rep. UCB/CSD-00-1112, University of California, Berkeley, 2000. URL: <citeseer.ist.psu.edu/zhao00xset.html/>.
[33] G. Klyne, J.J. Carrol, Resource description framework (RDF): Concepts and abstract syntax, February 2004. URL: <http://www.w3.org/TR/rdf-concepts//>.
[34] D. Martin et al., OWL-S: Semantic markup for web services. URL: <http://www.daml.org/services/owl-s/1.1/overview//>.

Fernando Cuervo is currently a Product Manager in Alcatel's IP Network Management division, he has transitioned from Research & Innovation where he was for the past two years a Senior Researcher for the Next Generation Network and Service Management strategic project. Within this project Fernando was the Chief Architect for the Alcatel Policy Architecture and the Cross Domain Integration Project. These projects address the needs of operators to be more efficient providing flexible network services and service management. Fernando's experience covers a wide range of management topics, including, information modelling, performance of management systems, management process re-engineering, workflow integration and integration of management and control. Fernando has also contributed to standards and industry fora such as IETF and MSForum. Fernando has a bachelor degree in electrical engineering from Universidad de Los Andes, Colombia, and a master of science in computer science from the University of Western Ontario.

Joanna Ziembicki received a B. Math degree in Pure Mathematics and Computer Science (Co-op) from University of Waterloo, Canada, in 2003. She is now working towards her M. Math degree at the School of Computer Science at the University of Waterloo. Her research interests include web semantics, service discovery, peer-to-peer networking and large-scale network management.

Noura Limam received the B.S. degree from the National School of Computer Science, Tunisia, in 2001, and M.S. degree in networking from the University of Paris VI, France, in 2002. In 2003, she was with Ucopia Communications Inc., France, at the R&D Department where she was involved in the development of a management tool for enterprise wireless networks. She is working towards the Ph.D. degree and is currently a Research Assistant at the University of Waterloo, Canada. Her research interests include network and service management, service discovery and service-oriented architectures.

Reaz Ahmed is a Ph.D. student at the School of Computer Science at University of Waterloo, Canada. His interests include service discovery architectures, distributed indexing schemes and peer-to-peer networks.

Tianshu Li received his Master's degree in Computer Science from the University of Waterloo, Canada. Before that, he received a B.Sc degree in Computer Science from the University of Victoria, Canada. He was a Research Associate in the group of Networking and Distributed Computing in the school of Computer Science, University of Waterloo, where his research interests span across Distributed Computing, Resource and QoS management in the Internet, Network and Service management. During this period, he was involved in several projects including the grid-based User Controlled LightPath (UCLP) system, which had been deployed on the fourth generation of Canadian Research and Education optical Network (CA*Net4). He joined Kitware Inc. in September 2004. His current research interests include: grid computing, web-based computing, parallel computing, and applying distributed computing techniques in large scale scientific computing and visualization.

Dr. Raouf Boutaba is an Associate Professor in the School of Computer Science of the University of Waterloo. Before that he was with the Department of Electrical and Computer Engineering of the University of Toronto. Before joining academia, he founded and was the director of the telecommunications and distributed systems division of the Computer Science Research Institute of Montreal (CRIM). Dr. Boutaba conducts research in the areas of network and distributed systems management and resource management in multimedia wired and wireless networks. He has published more than 150 papers in refereed journals and conference proceedings. He is the recipient of the Premier's Research Excellence Award, the NORTEL Networks research excellence Award and several Best Paper awards. He is a fellow of the faculty of mathematics of the University of Waterloo, has served as a distinguished lecturer of the IEEE Computer Society and is currently a distinguished lecturer of the IEEE Communications Society. Dr. Boutaba is the Chairman of the Working Group on Networks and Distributed Systems of the International Federation for Information Processing (IFIP), the Vice Chair of the IEEE

Communications Society Technical Committee on Information Infrastructure, and the Director of standards board of the IEEE Communications Society. He is the founder and acting editor in Chief of the IEEE Transactions on Network and Service Management published electronically, on the advisory editorial board of the Journal of Network and Systems Management, on the editorial board of the KIKS/IEEE Journal of Communications and Networks, the editorial board of the Journal of Computer Networks and the Journal of Computer Communications. He has also served as a guest editor of several special issue of IEEE Journal of Selected Areas in Communications (JSAC), the Journal of Computer Networks, the Journal of Computer Communications and the Journal of Network and System Management. He acted as the program chair for the IFIP Networking conference and the IEEE Consumer Communications and Networking Conference (CCNC), and a program co-chair for the IEEE/IFIP Network Operation and Management Symposium (NOMS), the IFIP/IEEE Conference on Management of Multimedia Networks and Services (MMNS), the IEEE Feature Interaction Workshop, the IEEE Autonomic Computing and Communications (ACC) and two IEEE International Conference on Communications (ICC) symposia.

Youssef Iraqi received the B.Sc. in Computer Engineering, with high honors, from Mohamed V University, Morocco, in 1995. He received his M.S. and Ph.D. degrees in computer science from the University of Montreal in 2000 and 2003. He is currently a research assistant professor at the School of Computer Science at the University of Waterloo. From 1996 to 1998, he was a research assistant at the Computer Science Research Institute of Montreal, Canada. His research interests include network and distributed systems management, resource management in multimedia wired and wireless networks, and peer-to-peer networking.