Service Discovery Protocols
A Comparative Study

Reaz Ahmed, Raouf Boutaba, Fernando Cuervo, Youssef Iraqi, Dennis Tianshu Li, Noura Limam, Jin Xiao & Joanna Ziembicki

School of Computer Science
University of Waterloo

Abstract

With the increasing need for networked applications and distributed resource sharing, there is a strong incentive for an open large-scale service infrastructure operating over multi-domain and multi-technology networks. Service discovery, as an essential support function of such an infrastructure, is a crucial research challenge today.

Cross-domain service discovery is essential for seamless service composition in large-scale, multi-domain, and multi-technology networks. In this paper, we compare the current state-of-the-art service discovery systems to their competence with such requirements. The comparison is conducted based on a set of well-defined criteria, leading to a selection of few approaches that can serve as the guideline in designing a global service discovery system for large-scale and multi-technology networks.

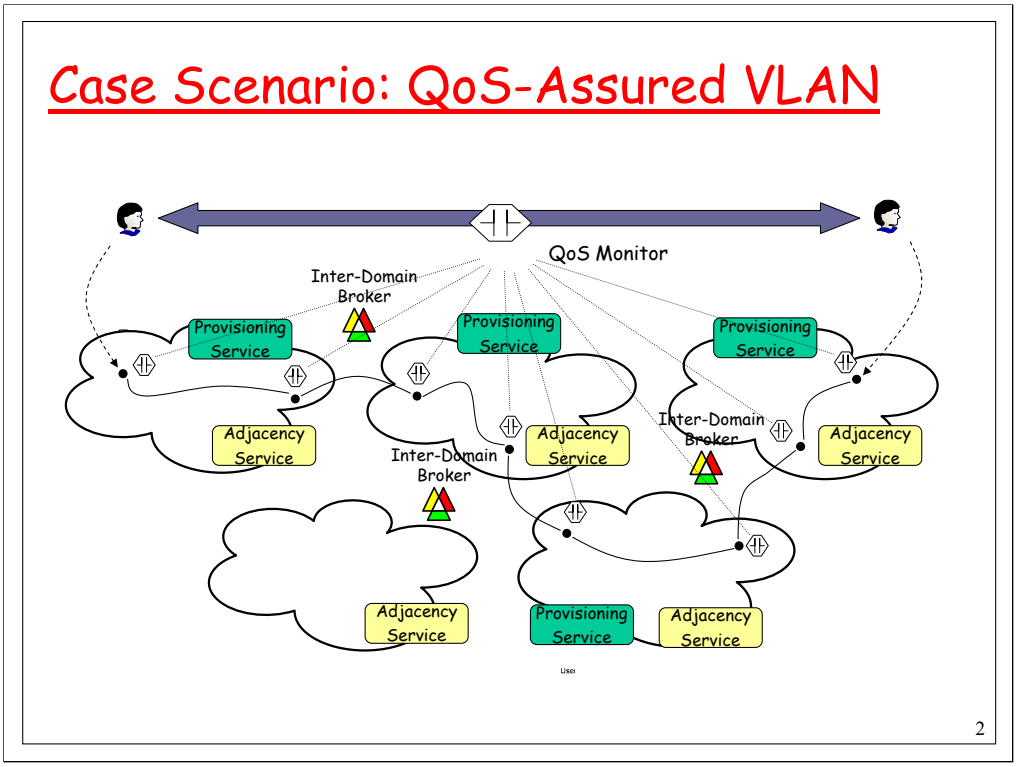
Keywords

Service discovery, multi-domain, multi-technology, large-scale service infrastructure, Web Services, Grid Services, Peer-to-Peer systems.

Authors

Reaz Ahmed	r5ahmed@uwaterloo.ca
Raouf Boutaba	rboutaba@bbcr.uwaterloo.ca
Fernando Cuervo	fcuervo@alcatel.ca
Youssef Iraqi	iraqi@bbcr.uwaterloo.ca
Dennis Tianshu Li	dtianshu@bbcr.uwaterloo.ca
Noura Limam	nlimam@bbcr.uwaterloo.ca
Jin Xiao	j2xiao@uwaterloo.ca
Joanna Ziembicki	jiziembi@uwaterloo.ca

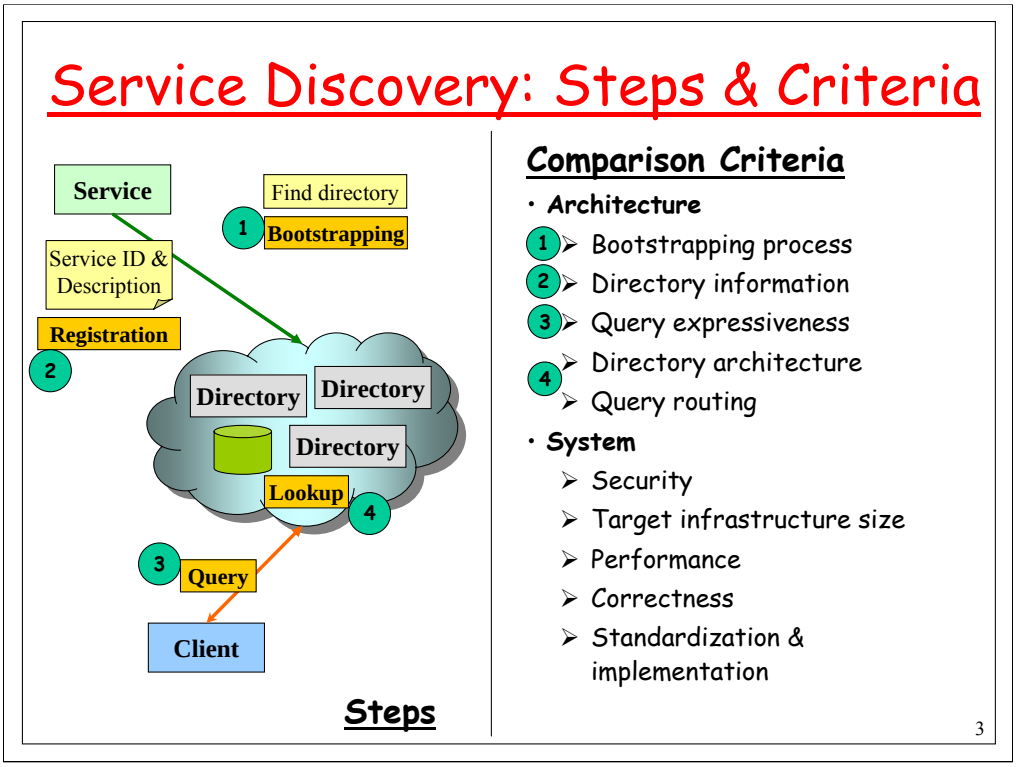
Acknowledgement: This work has been sponsored in part by Alcatel Inc. and the natural Sciences and Engineering Research Council (NSERC) of Canada



For next generation services across multiple provider networks, distributed service management applications are envisioned to perform the provisioning, configuration and management tasks dynamically. To support the creation and deployment of such management applications, service/resource discovery establish itself as a fundamental function. In a multi-domain context, a service/resource must be globally identifiable. For a service/resource to be discovered globally across provider networks and in Internet scale, an efficient, robust and scalable discovery mechanism is essential.

In the illustrated case scenario above, Alice (left) desires to establish a VLAN service to Bob (right), with some QoS assurance. To support such on-demand service creation, a large number of distributed management functions must be composed together. For example, adjacency services must be present to provide inter-domain routing information, intra-domain provisioning services are needed to establish end-to-end intra-domain paths, and inter-domain brokers are required to conduct inter-domain traffic exchange and peering contracts. Moreover, the QoS monitoring of the VLAN service relies on large number of QoS sensors distributed across the end-to-end path. All of these management functions are self-contained and reusable applications that need to be sought after at runtime. A highly desirable method of achieving this goal is to have a cross-domain discovery mechanism that is capable of searching and retrieving these functions at runtime.

In this paper, we evaluate the feasibility of applying current discovery systems, namely SLP [1], Jini [2], JXTA [3], UPnP [4], Salutation [5], INS [6], Twine [7], Splendor [8], Web Services [9] and file-sharing peer-to-peer systems [10, 11, 12], to our multi-domain and large-scale context based on a set of evaluation criteria we deem essential for such discovery systems.



Components found in most of the service discovery systems are:

- **Service**: abstracts a set of functionalities offered by a networked entity.
- **Client**: clients are entities who expect to discover available services in an unknown network with no or little configuration.
- **Directory**: directories are components responsible for caching advertisement from available service and perform lookup to satisfy discovery requests from clients. Some approaches has explicit directory agents. Others implement the directory as a part of the service/client implementation.

From the clients' perspective, service discovery involves bootstrapping, querying, and obtaining service handle(s) as a result of the query. A service first bootstraps and then registers itself with a directory agent if present.

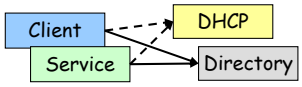
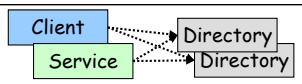
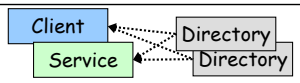
In this work we compare a number of service discovery approaches based on a set of evaluation criteria. These criteria are derived from the steps in the discovery process (i.e. criteria related to the **architecture**) and the factors determining the quality of the discovery process (i.e. criteria related to the **system**'s operation).

Bootstrapping specifies how users and services establish contact with the discovery system. **Directory architecture** defines the way directory information are stored and accessed in the system and how directory components coordinate with each other. **Directory information** is usually generated from service advertisements and is stored in the directory with some other maintenance information. **Query expressiveness** examines the power of a query, while **query architecture** specifies the way a query is routed across directory agents.

System level criteria are concerned with systems implementation. Any quantitative comparison based on these criteria requires an appropriate benchmarking tool. Due to the lack of such tools, we will restrict ourselves to qualitative comparisons.

Bootstrapping

- Requires some pre-configured knowledge
- Three major approaches:

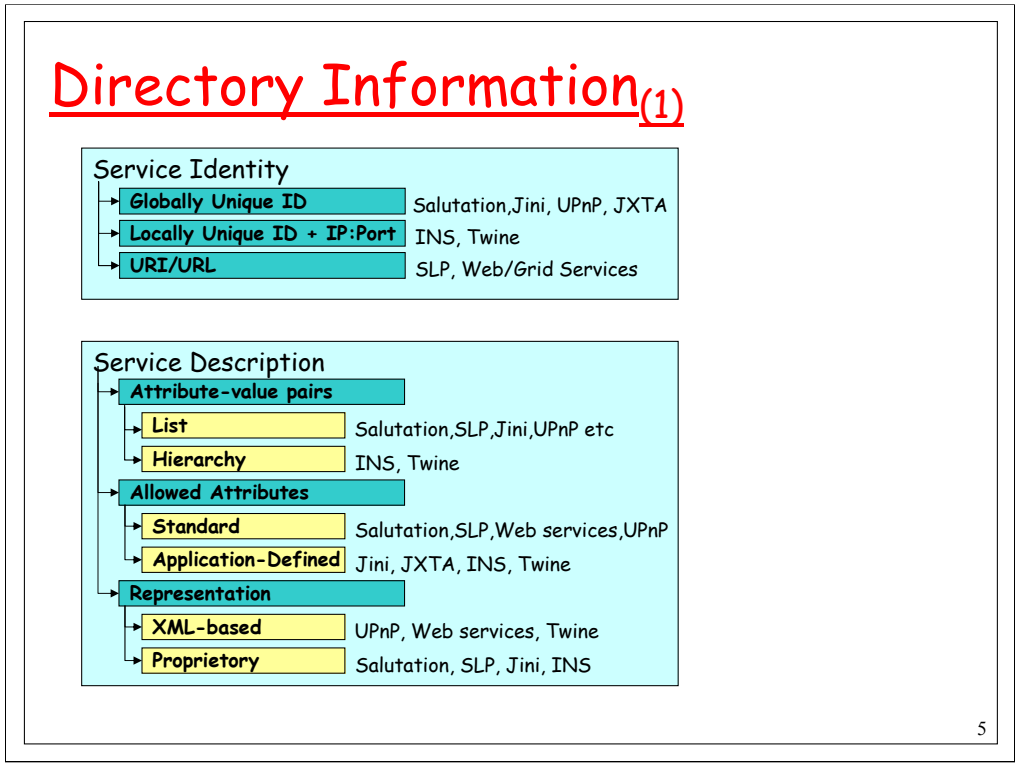
Approach	Figure	Bandwidth requirement	Fault-tolerance	Network Dependent
Unicast to directory. (IP obtained from DHCP server/ out of band)	 INS,Twine,SLP,P2P,Grid	Low	Low	No
Multicast by client/service	 SLP, Jini,Splendor	High	High	Yes
Periodic **multicast advertisement** by directory	 SLP, Jini, Splendor	Moderate	High	Yes

Bootstrapping is the first step in the discovery process and some a priori knowledge or pre-configuration is required by most discovery approaches.

Three basic techniques used for the bootstrapping process are:

- Unicast communication on a network address, on which a directory agent or other bootstrap information provider is listening. This technique is adopted by many approaches including INS, Twine, Web and Grid Services and content sharing P2P systems. SLP uses DHCP for disseminating IP addresses of directory agents while clients/services join the network.
- Multicasting from the client/service to a directory agent is also used in some other approaches like SLP, Splendor and Jini. With this method, the bootstrapping entity announces its existence using a well-known multicast address on which directory agents are listening. Upon receiving the message, the directory agent usually responds to the client/service with necessary information or authentication request on another unicast channel opened by the client/service.
- Another approach is to listen for the periodic multicast of directory advertisements on a well-known multicast address. This is the opposite process of the previous technique and is adopted by a few approaches such as Jini and SLP. In Splendor, this technique is used for providing location information to the mobile clients/services.

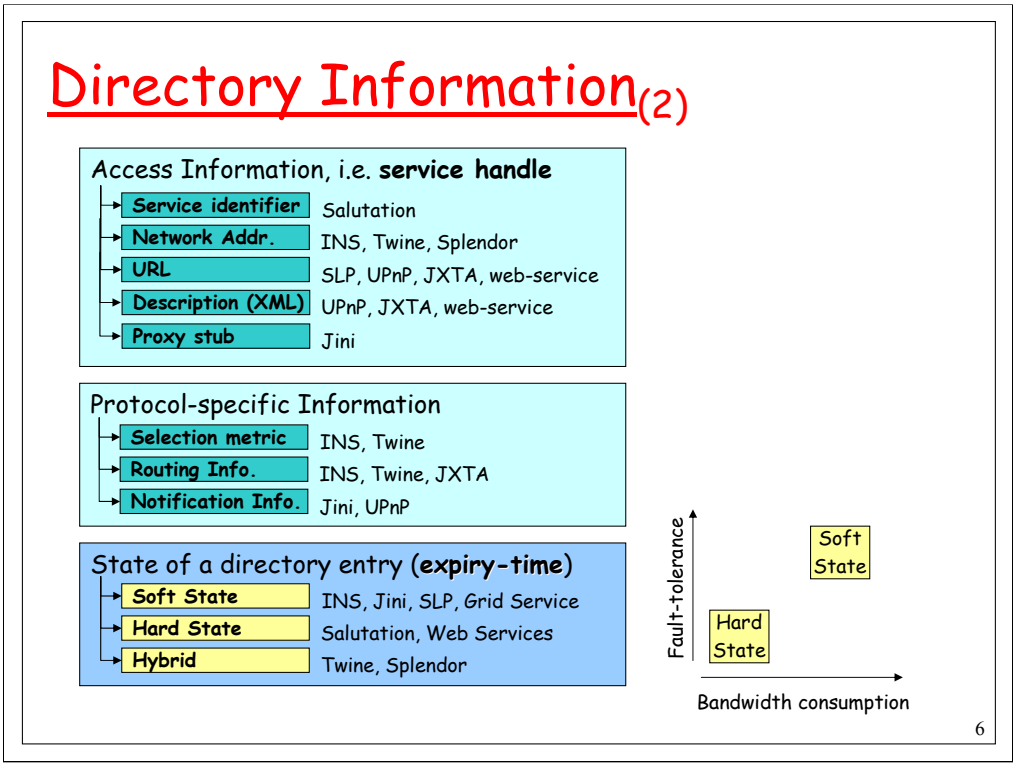
Using a well-known unicast address yields better bandwidth utilization but forces the directory to reside in a known location. On the other hand, multicasting from client/service to directory agents consumes more bandwidth in exchange for higher flexibility.



From the client side, a service request is transformed into a query. Similarly, from the service provider side, a service needs to be abstracted and formally represented. This is the role of directory information, called *service record* in INS, *UDDI* in Web services, *index entry* in P2P systems, etc. Two important aspects related to directory information are: expressiveness and formalism. Expressiveness pertains to how versatile a directory information can be and what content it can contain, while formalism of the directory information examines how it is structured. As a query is matched against the directory information, the aspects of directory information have an immediate impact on the effectiveness, correctness, and scalability of the discovery service.

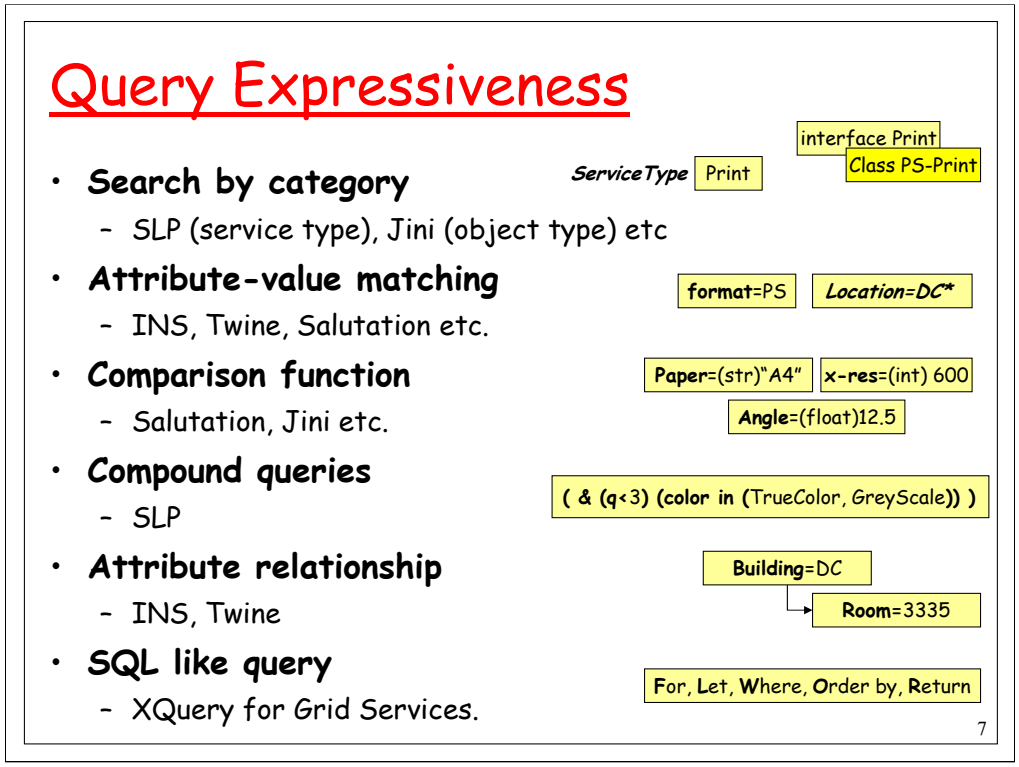
Information stored in directory by most of the service discovery protocols are as follows:

- **Service Identity**: All the service discovery approaches store an identifier for uniquely identifying a service among all the advertised services. In some cases a globally unique flat identifier (as in Salutation) is used and the identifier is generated locally by the service. Uniqueness is ensured with high probability, using a suitable generator function. In other cases the identity is composed of different pieces of information like locally unique identifiers combined with the network address (IP:port pair) of the service.
- **Service Description**: Most discovery approaches offer *yellow pages* service, i.e. allow service lookup based on given description. Service description is usually stored as a set of attribute-value pairs. Some discovery approaches (like SLP, Jini, etc.) store attribute-value pairs as lists and do not relate attributes with each other. On the other hand, INS and Twine relate attributes based on dependency relationships and store them in a tree-like hierarchy. Some systems (like SLP, Salutation etc.) use a predefined set of attributes for each service type, while others (like Jini, INS etc.) impose no such restriction on the allowed set of attributes. Finally, XML-based representation is used by many discovery approaches for extensibility and openness.



- **Service Handle**: Access to a service is facilitated through its service handle. The retrieval of service handle, should be an outcome of service discovery. The service handle can be any form of reference to the service. Some of the possible returned results after a service lookup are: service identifier that can be used to identify and invoke the service, network address of the service (e.g IP:port), a URL for the service, service description as a set of attribute-value pair or in some standard language like XML, and proxy stub for the service that can be used for service invocation or further communication with the service.

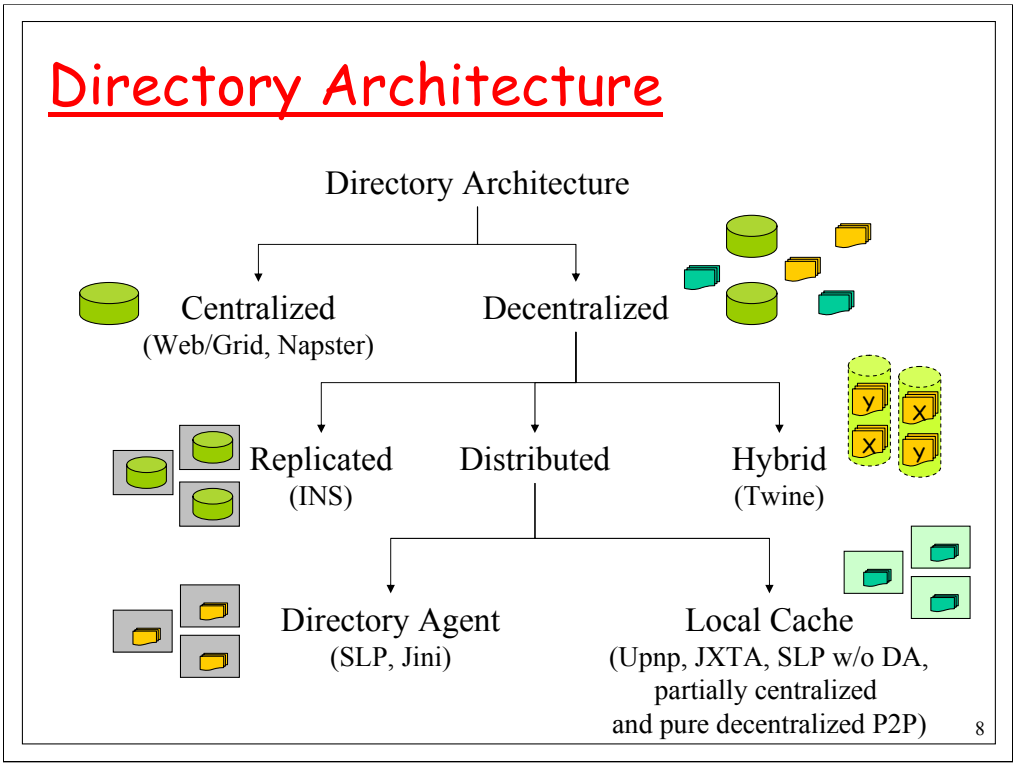
  Network address of a service is the minimum information provided by a discovery approach (INS, Twine, Splendor, and most of the peer-to- peer file sharing systems). SLP returns the URLs of the matching services. UPnP, Grid Service, Web Services and JXTA return XML-based description of the service in addition to the service URL. Finally, JXTA and Jini return proxy objects for the matched services.
- **Protocol Specific Information**: Some discovery approaches store protocol-specific information in addition to service identity, service description, and service handle. For example, INS and Twine store selection metrics and routing information for dynamic service selection by directory agent and application-level multicasting respectively.
- **Expiry Time**: Two basic techniques are adopted for directory information update and consistency maintenance: hard-state and soft-state registration. Hard-state registration has a lower bandwidth requirement and provides lower fault-tolerance. In contrast, soft-state registration provides higher tolerance to service failures, but at the expense of higher bandwidth consumption. Hence there is a tradeoff between fault tolerance and bandwidth consumption. SLP, Jini, INS and Grid Services adopt soft-state registration, while Salutation and Web Services use hard-state registration. On the other hand, Twine and Splendor use a mix of these two approaches. They use a proxy for each service. The service-proxy registration is soft-state and the proxy-directory registration process is hard-state.



The process of querying involves matching the client's request (expressed as a query) against all services to obtain a satisfying subset. The querying language adopted by a discovery system, allows the client to impose different types of restriction on this satisfying subset and facilitates discovery of more relevant services. Different levels of expressiveness are observed by various service discovery approaches:

- **Search by category**: Many service discovery approaches, including SLP, Jini and UPnP, classify services into categories and allow clients to discover services by category.
- **Attribute-value matching**: Systems like INS, Twine, Salutation etc. form queries as conjunction of a set of attribute value pairs that should be matched against all the advertised service descriptions. In addition INS and Twine support wildcard matching in the value side of an attribute-value pair.
- **Comparison function**: Salutation and Jini support more than string matching. In these systems, data-type for each attribute can be specified implicitly (in Jini) or explicitly (in Salutation). The specified data-type is used in selecting appropriate comparison function during query processing.
- **Compound queries**: SLP allows logical, relational and set membership operators in queries.
- **Attribute relationship**: INS and Twine allow dependency relationship between attributes, resulting into tree-like service descriptions and queries.
- **SQL like Query**: XQuery [14] as adopted in Grid services provides more flexible SQL-like query semantics.

Among these different levels of query semantics, XQuery and compound queries are the most powerful and flexible. Tree-like query and description as adopted in INS and Twine are useful in narrowing down search scope for large systems.



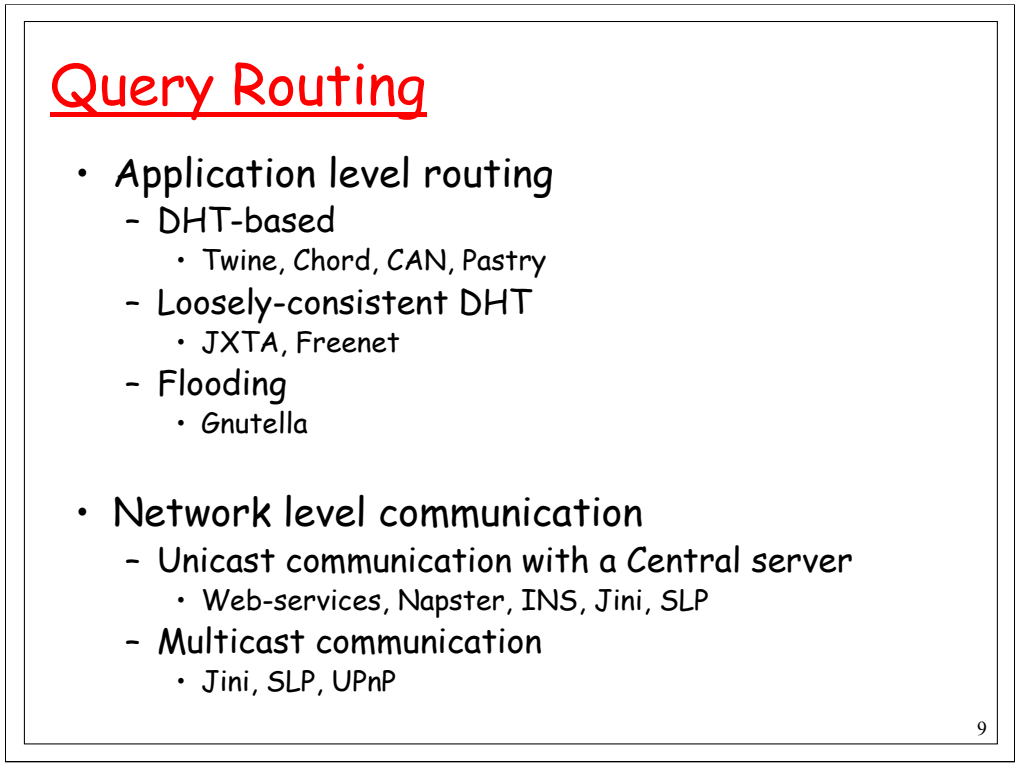
The directory architecture adopted by different service discovery approaches can be broadly classified as centralized or decentralized. In centralized architectures (e.g. Grid Service, Web Services and Napster), the directory information is stored in a central location in the network.

In decentralized architectures, directory information is stored at multiple network locations. Decentralized directory systems can be categorized as replicated, distributed and hybrid. In the replicated case, the entire directory information is stored at different network locations (as in INS). In distributed case, the directory information is partitioned, and the parts are stored at different network locations. Finally in the hybrid case, both replication and distribution are used (as in Twine).

For distributed directory architectures the directory information can be stored by dedicated servers, i.e. directory agents (DA) (as in SLP, Jini and Salutation) or can be cached locally by the service providers in the system (e.g. UPnP, JXTA, SLP without DA, pure decentralized, and partially-centralized P2P systems [13] *).

A centralized directory architecture is not suitable for large systems. In this case, the directory becomes a performance bottleneck and a single point of failure. On the other hand, consistency of the replicas is a major issue in the replicated architecture. Distributed and hybrid directory architectures scale well with the growth of network and provide better level of fault-tolerance.

* P2P file-sharing systems are broadly classified as hybrid-decentralized, partially-centralized, and pure-decentralized by structure and as unstructured, loosely-structured, and structured by index organization [13]



The query mechanisms may use a lookup in local cache, single directory or multiple directories. In UPnP and JXTA, all announcements from local services, plus some remote services already discovered during previous querying processes, are stored in local cache. Lookup is done at local cache first. In Jini, SLP, INS and Splendor, the directories are designed to be independent and can resolve a query based on local information.

On the other hand, in Twine, and ***DHT-based*** structured peer-to-peer systems (e.g. Chord, CAN, Pastry, Tapestry) directories (or peers) share their local information to resolve a query and the request may be forwarded to multiple directories/peers for the results. DHT-based structured peer-to-peer systems distribute the routing information across the peers. A query is routed to the peer containing the result in a finite number of hops. Partially structured peer-to-peer systems (like Freenet and JXTA) use ***loosely-consistent DHT*** techniques and route a query based on local routing decisions. For such systems the query is likely to be routed to correct destination, but the required number of hops is not deterministic. Initial peer-to-peer systems, like Gnutella, adopt ***flooding*** for query routing. To restrict the overhead generated from query traffic TTL (time-to-live) limit is used.

During the querying process, most approaches use unicast to communicate with a directory. In the absence of a directory, Jini and SLP can also use multicast to communicate with the service providers directly. UPnP relies on multicast for discovering available service in the network.

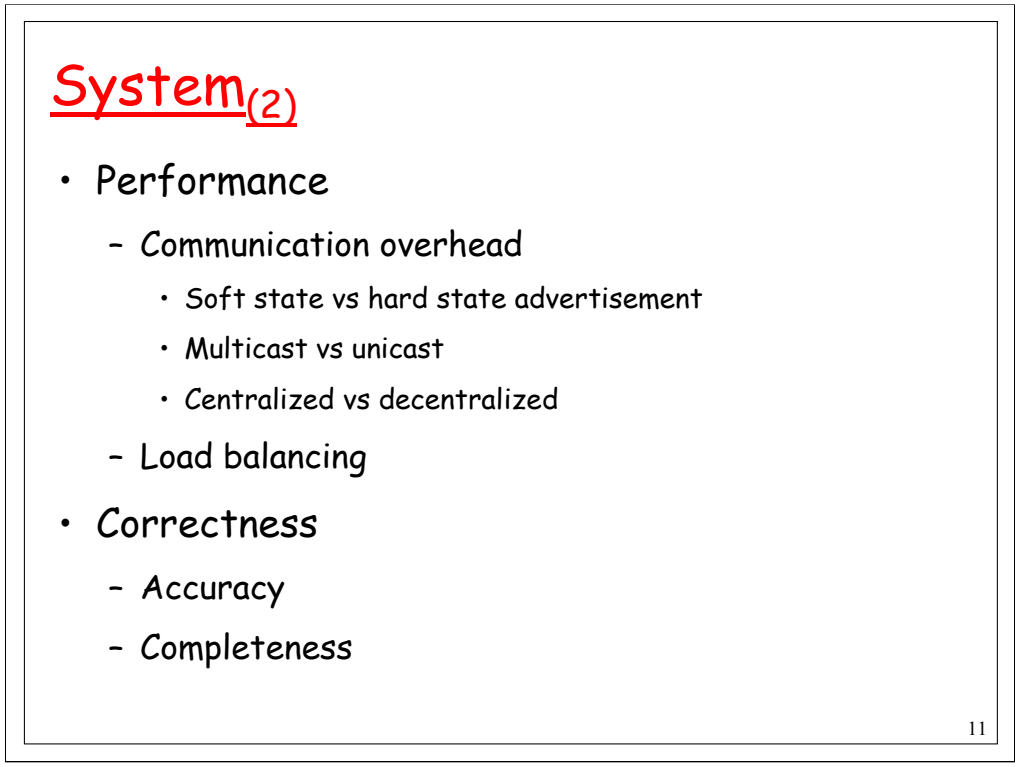
System(1)

- Security
  - None : INS, Twine, UPnP
  - Authentication : Salutation, Jini, Grid, Splendor
  - Integrity : SLP, Jini, Grid, Splendor, JXTA
  - Confidentiality : Jini, Grid, Splendor, JXTA
- Target infrastructure size
  - LAN : INS
  - Enterprise Network: Jini, SLP, UPnP, Salutation
  - WAN: Web/Grid services, File-sharing P2P systems
  - Large-scale:
    - Twine : aims to handle $O(10^8)$ services/resources
    - JXTA: aims for Internet-scale

10

- **Security**: Three aspects of security are examined: integrity, confidentiality and authentication. The integrity aspect considers whether the integrity of exchanged data is maintained throughout the process of discovery; the confidentiality aspect is concerned with the secrecy of user's request and returned service content; the authentication aspect notes the presence (or lack) of these procedures before any directory information can be accessed. Security is not considered in INS, Twine and UPnP. SLP provides only message integrity, while Web Services provides message level security. Salutation and most P2P systems use simple authentication of login and password. In contrast, Jini has its own standardized security package, which incorporates authentication, confidentiality, and integrity. Grid has a full security infrastructure. JXTA mandates its implementation to have confidentiality, and data integrity. JXTA protocols are compatible with accepted security protocols, such as TLS and IPSec. Splendor also has its own scheme for data integrity, authentication.

- **Target infrastructure size**: Most of the service discovery approaches studied are targeted for Internet-scale networks except SLP, Jini, and INS. SLP and Jini are mostly designed for service and resource sharing in a local area network. By using a relatively low value of Time To Live (TTL) to limit the overhead of multicast messaging, UPnP can be used in a relatively larger scale. Unlike Web Services, OGSA was initially designed for forming a virtual organization. However, recent trend on merging Web Services and Grid Services sheds some light on the future of building scalable Grid Services. INS does not scale well in a wide area network due to its expensive name-tree processing. Twine on the other hand is targeted for a large scale network.



It is very difficult to evaluate the performance of different approaches without a clearly defined benchmark. Here we consider two broad criteria: communication overhead and load balancing, and try to qualitatively compare the performance of different approaches.

- **Communication overhead***:* The communication overhead often depends on design choices and considerations. Following are the major conceptual choices that influence network load.
  - *Soft-state vs. hard-state advertisements***:** Approaches adopting soft-state model, like SLP, Jini, UPnP etc., require periodic announcement or registration and certainly generates more communication overhead than the hard-state model.
  - *Multicast vs. unicast***:** Multicast and flooding generate communication overhead. UPnP uses multicast for both queries and announcements, and Gnutella uses flooding; both are bandwidth-consuming.
  - *Centralized vs. decentralized architecture***:** In general, a service discovery approach using a centralized architecture generates less messaging overhead than a service discovery approach using a decentralized architecture. In distributed case, caching (as in UPnP, Salutation) and replication (as in INS) can reduce traffic generated from queries but at the expense of temporary inconsistency.
- **Load balancing***:* Centralized approaches often suffer from overload problems as the number of services registered with the registry and the number of client queries increase. Load balancing in that case can be handled through replication (e.g. UDDI, SLP and Jini). INS implements dynamic load balancing, i.e. replicates an overloaded directory agent in run-time.

Correctness: The correctness of the discovery service is examined in terms of two aspects: accuracy and completeness. The accuracy criterion evaluates the validity of the returned result, i.e. how closely the discovered service matches the user's request. The completeness criteria evaluates if the discovery process can find all services satisfying the query. Approaches with dedicated and conceptually centralized repositories (such as Web Services and Grid), or Structured routing (such as Chord, CAN and pasty) seem to be the best choices.

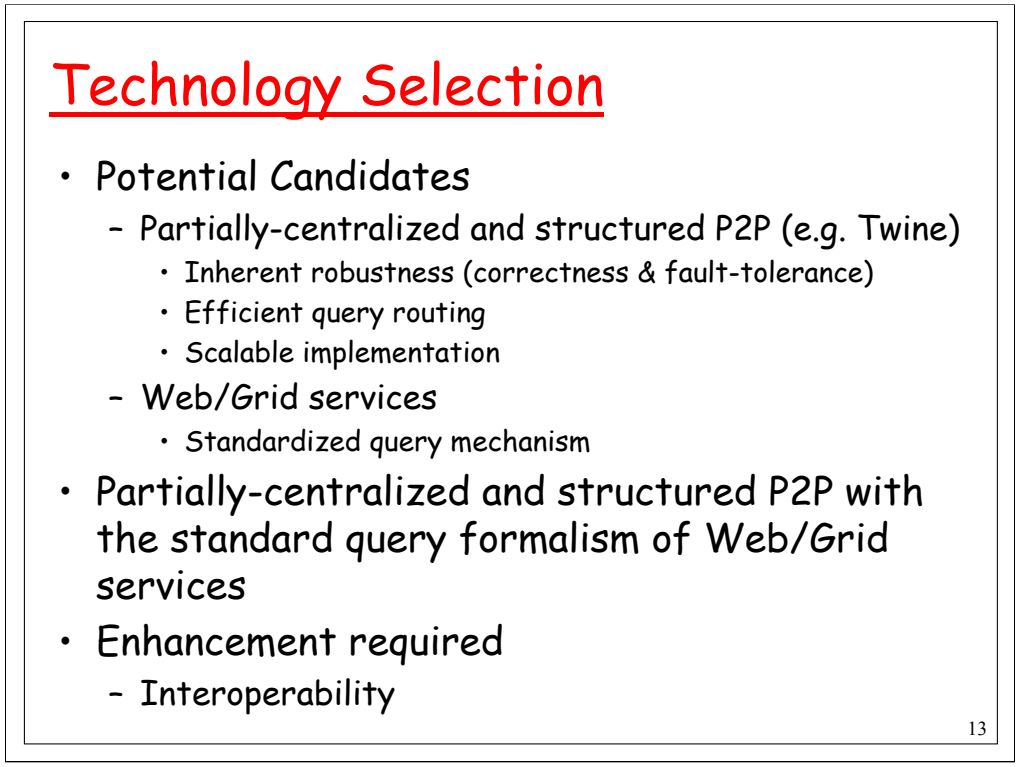
System(3)

- Standardization & Implementation
  - Industry Standards
    - SLP : IETF SrvLoc group
    - Jini : Sun Microsystem
    - UPnP : Microsoft Corporation
    - Salutation : Salutation Consortium
    - Grid and Web services: Globus Alliance, W3C, UDDI Consortium, OASIS etc.
  - Major Research projects
    - INS and Twine : MIT
    - SSDS : UC, Berkley
    - Splendor : MSU

12

Several international organizations have been involved in standardizing or promoting service discovery approaches. Specifically, they are the IETF for SLP, Globus Alliance for OGSA Grid Services, the W3C and UDDI Consortium - a section of the Organization for the Advancement of Structured Information Standards (OASIS) - for Web Services/XML/SOAP/WSDL/UDDI, Salutation Consortium for Salutation, and UPnP Forum for UPnP. There are a number of existing implementations and development platforms for Salutation, SLP, Jini, UPnP and JXTA.

- The OpenSLP project delivers an open-source implementation of SLP.
- Sun Microsystems has released the Jini Technology Starter Kit to build Jini-enabled clients, services, and directories.
- A number of UPnP SDKs are available, including the ones developed by Intel and the open source SDK for the Linux platform.
- IBM provides a Salutation Software Development Kit (SDK) and Salutation-Lite is provided in Open Source at the Salutation Consortium web site.
- An implementation of the JXTA Protocols using Java 2 Standard Edition (J2SE), and another one in C for Windows and Linux platforms are available.
- WebSphere, .NET, and SUN ONE web services are the main existing implementations of Web Services.
- Globus provides a development toolkit (Globus Toolkit) to implement Grid Services.
- The current implementation of INS and Twine uses Java 2.0 platform and is available at MIT website.



Based on our investigation, we find partially-centralized and structured peer-to-peer systems (such as Twine) and Web/Grid service discovery mechanism have some salient benefits for service discovery in multi-domain and large scale networks. A P2P system like Twine are designed to operate over large volume of information dispersed widely across networks. The directory architecture is inherently robust as there does not exist failure bottlenecks and the querying process has guaranteed correctness. Query routing in such system is theoretically bounded to logarithmic scale in terms of the number of directory nodes in the network. Such system can be implemented to be scalable both in terms of information storage and request processing. Web/Grid service discovery can also serve as a good candidate as they are designed to target the Internet. Unified query representation in XML and standardized query mechanism using new technologies such as XQuery gives added attractiveness.

It is also possible to integrate the partially-centralized and structured P2P architecture with the query formalism found in Web/Grid services. Such hybrid approach may take some initial effort to design and implement, but we believe will yield a better fit in the end, compared to readily adopting one of the existing discovery architectures.

In the multi-domain context, it is imperative the discovery system in question must interoperate with existing discovery mechanisms. As more and more service providers are employing their own service discovery mechanism within their administrative scope, it will be impractical to require all service providers to conform to a single discovery mechanism across all administrative domains. Unfortunately, as demonstrate in this paper, such interoperability is sourly lacking in current service discovery systems.

References

[1] E. Guttman, C. Perkins, J. Veizades, and M. Day. RFC 2608: Service location protocol, version 2, June 1999.

[2] Sun Microsystem Inc. Jini technology architectural overview. Technical report, Sun Microsystem, Inc, 1999. http://wwws.sun.com/software/jini/whitepapers/architecture.pdf.

[3] Sun Microsystems. JXTA technology: Creating connected commu-nities, January 2004. http://www.jxta.org/project/www/docs/JXTA-Exec-Brief.pdf.

[4] UPnP Forum. Upnp device architecture 1.0, May 2003. http://www.upnp.org/download/UPnPDA10 20000613.htm.

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

[6] William Adjie-Winoto, Elliot Schwartz, Hari Balakrishnan, and Jeremy Lilley. The design and implementation of an intentional naming system. In Symposium on Operating Systems Principles, pages 186–201, 1999.

[7] Paul J. Leach and Rich Salz. INS/Twine: A scalable peer-to-peer ar-chitecture for intentional resource discovery, 1998.

[8] Feng Zhu, Matt Mutka, , and Lionel 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), pages 235–242, March 2003.[Chord] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In Proceedings of the ACM SIGCOMM '01 Conference, San Diego, California, August 2001.

[9] UDDI.org. Uddi technical white paper, 2002. http://www.uddi.org/pubs/Iru UDDI Technical White Paper.pdf.

[10] Gnutella Protocol Specification. Available at: http://dss.clip2.com/GnutellaProtocol04.pdf

[11] I.Clarke, O.Sandberg, and B.Wiley. Freenet: A distributed anonymous information storage and retrieval system. In In Proceedings of the Work-shop on Design Issues in Anonymity and Unobservability, Berkeley, Cal-ifornia, June 2000.

[12] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In Proceedings of the ACM SIGCOMM '01 Conference, San Diego, California, August 2001.

[13] Stephanos Androutsellis-Theotokis, A Survey of Peer-to-Peer File Sharing Technologies, Athens University of Economics and Business, Greece, Greece, 2002.

[14] W3C, W. W. W. C. 2004. Xquery. http://www.w3.org/XML/Query.