Pattern Language for Service Discovery

Juha Pärssinen
VTT
juha.parssinen@vtt.fi

Teemu Koponen
Helsinki University of Technology
teemu.koponen@hut.fi

Pasi Eronen
Nokia Research Center
pasi.eronen@nokia.com

December 20, 2004

Abstract

In this paper a pattern language for service discovery is introduced. These patterns were mined from several existing service discovery protocols, and from protocols which are generally used for service discovery, but have not been discussed under the theme of service discovery in the research community. In this paper everything preceding the actual service usage is considered as service discovery, regardless of the actual used mechanism.

This language gives to the reader an overview to different aspects of service discovery, and enables easier comparison of different existing approaches to this problem domain.

Language Context

Service discovery is about discovering services in a dynamic network environment. We consider basically everything that can be accessed over the network as services. A service may be low-level "infrastructure service" offering critical functionality such as IP connectivity or DNS name resolution. Other services are associated with physical devices such as printers, while web sites and web services are more independent of a particular computer implementing them. Certain services operate in more or less ad hoc networks—characterized by wireless connections, mobility, lack of fixed infrastructure, decentralized control, and little planning, configuration or administration—while others are implemented in professionally operated fixed networks with centralized control.

In this paper, the term "client" means the party that initiates service discovery, and "service" means the party being discovered. As the result of the service discovery the client obtains information necessary to choose the right service and communicate with it, such as its capabilities and addresses. However, this does not directly imply in service discovery one always has distinct roles for a "client" and a "server". For instance, in voice-over-IP the caller may need to discover the current location (address) of the called party, but both peer-to-peer and client-server architectures are possible.

While some of the presented patterns assume the client is a device operated by a person, such as a Personal Digital Assistant (PDA) or a laptop, the kinds of clients are not limited to

such. For example, an enterprise application component could implement service discovery to find the current location of a component offering a certain CORBA interface.

A road map of the pattern language is shown in Figure 1 below. In this road map the starting point is the node *How to discover?*, and a number in parentheses after each pattern name is the sequence number of pattern in this language.

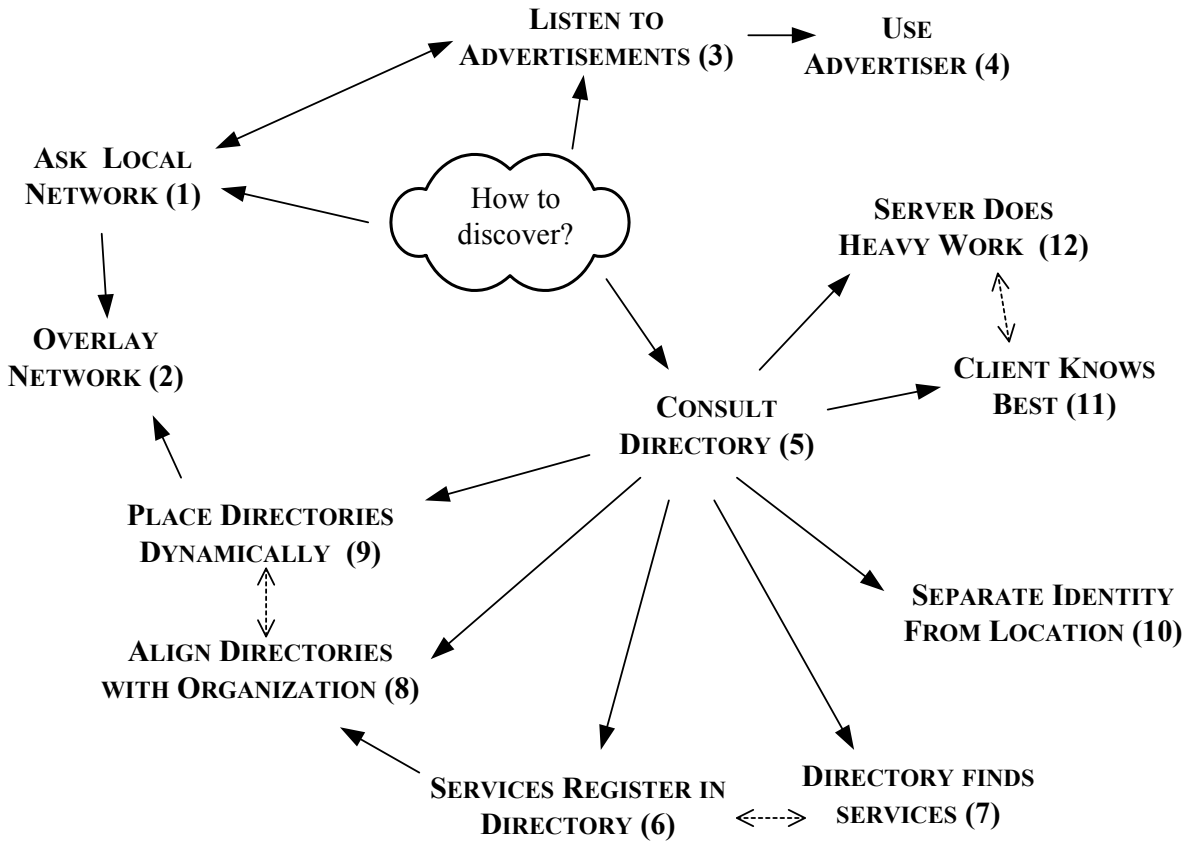

Figure 1: A pattern language for service discovery. (Solid arrows lead to patterns solving a problem in the resulting context; a dashed line indicates an alternative solution or trade-off.)

1. ASK LOCAL NETWORK

Context There is a dynamic network of parties; that is, any node in the network can join or leave, usually without involving any centralized administration. Some nodes are offering services to others, mostly to others within broadcast/multicast scope. Service in this context can be almost anything, and the roles of the nodes are not necessarily fixed: the client is the party who wishes to find a service, and service is the party being discovered.

Problem The client has an idea what kind of service it needs, but it does not have enough information to contact the service yet.

Forces

- Deploying new services should be easy, and involve as little manual administration as possible;
- Services themselves often know best where they are, and what kind of services they provide; and
- Introducing new network elements, such as dedicated directory servers, may not be feasible.

Solution The client sends a query to nodes that are near the client (in terms of network topology), indicating what kind of services it is looking for. The query is typically sent using some kind of multicast or broadcast message. All services listen to these messages, but only the nodes that have relevant information to the query will actually answer.

Resulting Context Clients can discover services that are in the broadcast/multicast scope without requiring fixed infrastructure (such as a directory server) or manual administration. Often the network topology approximates physical location and other relevant context, so the services the client is interested in are likely to be near the client in network topology.

However, the client may also be interested in services that are not near it in network topology. One option is to extend the multicast scope, either using normal IP multicast routing or an OVERLAY NETWORK (2). However, if the number of nodes in the network is large, using multicast messages may become inefficient. This is typically solved by introducing dedicated directory servers, discussed in CONSULT DIRECTORY (5) pattern.

If the client is interested in detecting when new services appear on the network, periodic polling is required. To avoid polling, this pattern is often combined with LISTEN TO ADVERTISEMENTS (3) pattern.

Choosing right service from the large amount of possible ones is considered in patterns CLIENT KNOWS BEST (11) and SERVER DOES HEAVY WORK (12).

Known Uses Service discovery protocols supporting operation without a centralized directory, such as Rendezvous [12], SLP [15], and UPnP [29], allow clients simply to broadcast their queries to everybody. Services considering the query to match their properties answer and thus inform client about their presence.

2. OVERLAY NETWORK

Context Clients ASK LOCAL NETWORK (1) to find services.

Problem The client's local broadcast/multicast scope does not contain the services the client is looking for. Moreover, only a limited set of nodes in the network containing the services is relevant for answering the query; thus, flooding the queries to everyone is inefficient. How to communicate with a possibly large amount of other nodes while still keeping the benefits of ASK LOCAL NETWORK (1)?

Forces

- The network may not provide efficient way to implement multicast e.g., normal IP multicast routing may not be available; and
- There may not be any pre-arranged centralized directories to ask.

Solution Clients and services build an "overlay network" that allows "application layer multicast" queries to be sent regardless of the underlying network topology. The deployed overlay network enables more intelligent techniques to more efficiently distribute queries than simple multicasting. For example, the overlay network nodes could implement caches to store query results. Thus, centralized directories can be avoided.

Resulting Context Application layer multicast facilitates multicast in environments which it not otherwise possible. However, the price is increased client/service complexity and increased usage of resources.

While an overlay network may not require any centralized directory servers that store information about other services, some nodes become more important than others. They are responsible for duplicating multicast messages for leaf nodes in the overlay network. Thus, PLACE DIRECTORIES DYNAMICALLY (9) and ALIGN DIRECTORIES WITH ORGANIZATION (8) patterns may be useful in selecting these more vital nodes. In other words, this pattern blurs the line between multicasting and (dynamically placed) directory servers.

Known Uses Peer-to-peer file sharing networks, and peer-to-peer applications in general, commonly use variety of application layer multicast approaches; see, for instance, JXTA Search [31].

3. LISTEN TO ADVERTISEMENTS

Context There is a dynamic network of parties; that is, any node in the network can join or leave, usually without involving any centralized administration. Some nodes are offering services to others, and usually many of the nodes within broadcast/multicast scope are interested in the services.

Problem Clients want to find out what services are available, when new services appear on the network, and when old services leave.

Forces

- Deploying new services should be easy, and involve as little manual administration as possible;
- Introducing new network elements, such as dedicated directory servers, may not be feasible; and
- The network has a large number of clients compared to the number of services.

Solution Services periodically send advertisement messages, usually using some kind of multicast or broadcast messages. Using advertisements is especially suitable if most clients are likely to be interested in the same services.

Resulting Context There is a trade-off between reactiveness and bandwidth usage: if the services send advertisements more often, clients can discover the current situation faster, but more bandwidth and other resources are required. For this reason, periodic advertisements are sometimes used together with ASK LOCAL NETWORK (1) pattern. If the number of services grows larger, other alternatives such as CONSULT DIRECTORY (5) pattern can be considered.

If a service is not able to advertise itself, it can USE ADVERTISER (4).

Known Uses Many service discovery protocols, such as SLP and UPnP [15, 29], use periodical multicast advertisements. Other examples include IEEE 802.11 wireless LAN Beacon messages [17] and ICMPv6 Router Advertisements [28].

4. USE ADVERTISER

Context Clients LISTEN TO ADVERTISEMENTS (3) to find services.

Problem A service may not be able or willing to advertise itself.

Forces

- The service may not support the service discovery protocol that is used in the current environment; and
- Advertising consumes limited resources of the service.

Solution Service delegates advertising to an advertiser which can, and is able to, advertise on behalf of the service. In order to advertise, the advertiser requires the service information. This transfer of information may be either manual or automatic. While the first requires support from service administrators, the latter requires advertiser to have a transfer interface available for services.

Resulting Context Services which are not able to advertise themselves, or are too busy serving clients, can now use LISTEN TO ADVERTISEMENTS (3) pattern. If in case of automatic delegation the service does not know an advertiser, it can LISTEN TO ADVERTISEMENTS (3), ASK LOCAL NETWORK (1) or CONSULT DIRECTORY (5) to discover one.

Known Uses Service discovery protocol implementations often support manual registration of legacy services (without support for modern service discovery protocols). For example, OpenSLP's SLP daemon supports manual service registrations [24].

5. CONSULT DIRECTORY

Context There is a dynamic network of parties; that is, any node in the network can join or leave any time. There is possibly a large number of nodes offering services to others. The service in this context can be almost anything from low-level infrastructure services such as DNS name resolution to web sites such as Amazon.com.

Problem Client has an idea what kind of service it needs, but it does not have enough information to contact the service yet.

Forces

- The services are not necessarily near the clients in the network topology;
- Using broadcast/multicast queries and advertisements may not scale to a large number of services; and
- Often centralized administration is involved in setting up a new service.

Solution Let the client consult a directory server that manages a database containing contact information of available services.

Resulting Context The network requires the presence of a directory or directories. Administrator or service discovery protocol developer must decide whether being a directory is a permanent role assumed by one or a group of nodes in the network. One can ALIGN DIRECTORIES WITH ORGANIZATION (8), or the network can PLACE DIRECTORIES DYNAMICALLY (9) by itself.

In order to give meaningful answers, a directory must be provisioned with the information to be shared for clients. Usually either SERVICES REGISTER IN DIRECTORY (6) or DIRECTORY FINDS SERVICES (7). Moreover, to contact the directory, the client needs to know its contact information. This "bootstrap provisioning" could be done manually (by configuring the network address), or the client can LISTEN TO ADVERTISEMENTS (3) or ASK LOCAL NETWORK (1) to find the directory.

Clients are also occasionally interested in the appearance of new services on the network. To avoid needless polling, the PUBLISHER-SUBSCRIBER [4] pattern, a variant of the OBSERVER [6] pattern, could be used to allow the directory to notify interested clients of new services.

Known Uses Several service discovery protocols have a directory entity. For example, SLP has Directory Agent (DA) [15], Jini has a lookup service [26]. Service discovery protocols also often support subscription of notifications about service appearances (e.g., [18] and [26]). Moreover, the network may have a LDAP directory [16] or authoritative DNS server [21].

General web search engines (e.g., [14]) and directories, such as Google Directory, Yahoo!, and telephone directory services, are also examples of the pattern.

Related Patterns In small networks directories can be avoided using ASK LOCAL NETWORK (1) and LISTEN TO ADVERTISEMENTS (3) patterns.

The LOOKUP pattern [8] describes a directory service, but combines querying, registering, bootstrapping and also aspects of directory placement into a single pattern. In this paper, we consider these aspects in separate patterns.

6. SERVICES REGISTER IN DIRECTORY

Context Clients CONSULT DIRECTORY (5) to find information about services. Typically each service is listed in only one directory or few directories.

Problem The directory need to get the information about services to share with the clients.

Forces

- Manually storing the information in the directory may be feasible, but is tedious;
- Service administrators may want to control what information is stored in the directory;
- Services themselves know best where they are, what kind of services they provide, and when this information changes; and
- It's acceptable for services to become service discovery aware.

Solution Let services to register themselves to directories. The directories must provide a special service registration interface for services to provision their information.

Resulting Context The directory has the information it needs to answer clients' service discovery queries. In addition, services can update their own information, so any changes propagate in timely fashion. However, information about services that no longer exist can still remain in directories: the LEASING [8] pattern provides a solution to this.

As a result, services have to be aware of service discovery and know how to register themselves, which is likely to complicate their implementation and operation. If this is not desired, DIRECTORY FINDS SERVICES (7) instead.

Moreover, if there is more than one independent directory to register in, services may have to first find the directories (using, e.g., LISTEN TO ADVERTISEMENTS (3) or ASK LOCAL NETWORK (1)) and register with each of them. Therefore, DIRECTORY FINDS SERVICES (7) may enable the services to reach a larger audience of clients in environments with several directories.

The problem that remains unanswered is how to ensure overall validity of the service information stored in the directories? For example, can anyone register a service or are there access control mechanisms in place?

Known Uses Many service discovery protocols have a protocol and messages for registering purposes: SLP has service registration messages [15], Jini has "join protocol" [27], DNS has dynamic update messages [30] and UDDI provides a Publishing API [23].

7. DIRECTORY FINDS SERVICES

Context Clients CONSULT DIRECTORY (5) to find information about services. However, services do not require explicit control in publishing their information. In certain operational environments, there can be also a need to have multiple directories that contain the same services to facilitate even wider audience for services.

Problem The directory or the directories must obtain the information about services to share with the clients. Requiring services to register themselves in a directory places the burden on the services. Moreover, services might not possess the information of all directories, or the amount of directories may simply be overwhelming from service's point of view. Unfortunately, services that don't want, or can't, register themselves won't be listed in the directories.

Forces

- Manually storing the information in the directory may be feasible, but is simply tedious;
- Making services unaware of service discovery can simplify their implementation and deployment; and

- Directories can be more comprehensive if they do not require active cooperation from the services.

Solution Let the directories to find the services by implementing a mechanism to examine networks to the directories. The mechanism should be executed periodically. It can be manual (e.g., Google Directory or Yahoo) or automatic (e.g., Google). The more comprehensive the examination can be made, the more comprehensive directory of services can be collected. The services can now provide information about themselves in a way that is independent of the way the directories work.

Resulting Context This pattern decouples services from directories: services don't have to know where they are listed in. It is easier to have multiple directories covering the same services—service information can be more easily distributed to a larger audience. Multiple directories and even competition between them could be useful in some situations (e.g., web search engines).

Since services do not know where they are listed, timeliness of information may become problematic, since service cannot notify the directory when something has changed. The directories should execute the basic measure to improve the validity of the service contact information by using SEPARATE IDENTITY FROM LOCATION (10).

The directory has to find the services somehow, and e.g., a web search engine will not find pages that are not linked anywhere. Also, finding services by following links is especially suitable for hypertext, but may not be applicable to other types of services.

Known Uses Web search engines typically use this pattern (e.g., [14]). Even in a closed environment, such as a corporate intranet, it would be difficult to register every web page in a search engine. Instead, the search engine crawler is given a couple of starting points and it finds other pages by following hyperlinks.

In certain manually maintained directories, the directory maintainers are responsible for finding the services and sorting them to relevant categories; examples include Google Directory, Yahoo, and web-based restaurant guides. It is also possible to combine this pattern with the case where services ask to be listed in the directory, i.e. SERVICES REGISTER IN DIRECTORY (6).

8. ALIGN DIRECTORIES WITH ORGANIZATION

Context Clients CONSULT DIRECTORY (5) to find information about services on the network. Each service is usually associated with an organization or organizational unit responsible for it.

Problem Where to place the directory servers, and who should operate them?

Forces

- Availability of directory servers is critical;
- Organizational unit's information is accessed most often inside the unit;
- Organizations may want to control the contents of directories and who can access the data; and
- Organizational units want to influence the division of responsibilites and allocation of costs.

Solution Let organizational units, such as departments or divisions, establish their separate directories; then connect the directories to reflect the organization's hierarchy or other relevant static structure.

Resulting Context Organizational units gain more control to the directory contents. Local directories allow local authorization and implementation decisions, e.g. who is allowed to modify information and how modifications are done.

The overall system becomes more scalable and fault-tolerant. Moreover, as the network topology is often also aligned with organizational structures, local directories improve the overall performance of the directory system.

If static alignment of directories is impractical due to dynamics of the organization or support for dedicated directories is uncertain in the organization, it's better to PLACE DIRECTORIES DYNAMICALLY (9).

Known Uses Hierarchical directories such as DNS [21], LDAP [16], and Microsoft Active Directory [19] support deployment aligning with organization.

9. PLACE DIRECTORIES DYNAMICALLY

Context Clients CONSULT DIRECTORY (5) to find information about services on the network. No static structures exist in service discovery environment to support the categorization of nodes further to directories and regular nodes.

Problem Where to place the directory servers, and who should operate them?

Forces

- Technical differences among network nodes are insignificant in terms of providing directory functionality;

- No external restrictions or preferences are set for the directory servers—all network nodes are equally valid; and
- Technical and non-technical support dedicated directories can get is uncertain or must be minimal in the environment.

Solution Let all (or at least many) nodes implement directory functionality, and then dynamically select node(s) to assume the directory role. Depending on the context, several directory node selection algorithms may be available. For example, the selection can be based on the abilities of the nodes to operate as a directory (e.g., most processing power, best network connectivity) or if no such information is available, the choice can be even random.

Resulting Context Dynamically elected directories provide less possibilities to control the directories and their contents. Moreover, the stability properties of the directories services may be unknown—however, in certain environments the overall availability may in fact improve, since in case a failure other nodes can more easily take over.

If the directory functionality is dynamically placed, clients have to find the directory. Usually this is accomplished using either LISTEN TO ADVERTISEMENTS (3) or ASK LOCAL NETWORK (1) pattern.

Known Uses In NetBIOS networks the nodes "elect" a domain master browser among themselves [20].

10. SEPARATE IDENTITY FROM LOCATION

Context Clients and directories store information about gathered services in a dynamic environment. Directories do this as part of their normal operations—after all, they need to be able to answer service queries from clients—but clients can also store the service contact information in order to contact the same service later.

Problem What should clients and directories store as service contact information to maintain the usability of the information later in such a dynamic environment? Once again, the whole purpose of storing service contact information is to reuse it later.

Forces
- A service may be moved from a server to another server;
- A service may be running on a mobile terminal whose network attachment point constantly changes; and
- The network address (location) of a server can be dynamically assigned.

Solution Let clients and directories use an identifier, that remains valid for a long time, to identify services. In other words, clients and directories refer to services using identifiers, instead of network addresses, in their service information storages. One must provide also an additional mapping service to map these persistent identifiers, as needed, to more transient network addresses—information that is needed to locate and contact the actual services. Mapping service implementation determines the allocation method of the identifiers.

Resulting Context Clients and directories can store information for a longer time, but they also need a service to map the identifiers to network addresses (e.g., IP addresses). The mapping service may be distributed—that is, clients ASK LOCAL NETWORK (1)—or centralized if clients CONSULT DIRECTORY (5) (e.g., DNS, SIP). If the identifier resolution is based on a directory, there has to be way to update the information; usually the SERVICES REGISTER IN DIRECTORY (6) pattern is used.

There can be several layers of identifiers and service discovery. For example, a query for printers can result in a URL, which is mapped to an IP address using DNS (an example of CONSULT DIRECTORY (5)), and to an Ethernet MAC address using ARP (an example of ASK LOCAL NETWORK (1)).

Known Uses Domain names [21], Jini service identifiers [27], HIP host identities [22], SIP URIs [25], UPnP unique device names [29], and URLs [13] are all examples of identifiers that can be stored for a long time, and mapped to the service's current location (IP address, port number, etc.) when needed to obtain valid service contact information.

Related Patterns The CLIENT-DISPATCHER-SERVER pattern [4] describes a dispatcher service that knows the current location of services. Moreover, the dispatcher enables a communication channel between clients and services; clients merely identify the needed service with an service identifier while performing a service call. In this paper, however, we consider service discovery not to include service access and thus SEPARATE IDENTITY FROM LOCATION (10) contains the relevant aspects of the pattern: location—identity separation.

The IDENTIFICATION patterns [11] separate identifiers and locators in the context of distributed object middleware solutions. In the identification patterns interpretation of an object identifier is local to a server, and thus, identifiers do not translate to locators as such. In a way, SEPARATE IDENTITY FROM LOCATION (10) is an addition to these patterns; it makes a clearer functional distinction between locators and identifiers than the IDENTIFICATION patterns do.

11. CLIENT KNOWS BEST

Context A client can discover services with or without consulting directories, but it is subject to receive information about multiple services as an answer to a service query.

Problem How can a service or a directory answering a query decide which services are the most relevant for the specific client and especially for its user?

Forces

- The relevance of services depends on the context of a client and its user's preferences;
- Usually only a limited amount of context information can be sent with the query;
- Increasing the expressive power of queries increases the complexity of both clients and directories;
- Services and directories do not know how to process the context and preference information;
- The user may not know how to formulate a specific, but can pick the right service when shown the alternatives;

Solution Use a simple query language, even if that means returning several matches to the client. Let the client, or even its user, decide which of the matches is the desired one. This moves the responsibility of handling context information and user preferences to the client software.

Resulting Context The client obtains the information necessary to access a service fulfilling the current need. However, there is a risk that service or directory has too much information to transmit to the client or client has too much information to show to the user. Then it becomes essential that the SERVER DOES HEAVY WORK (12).

Known Uses The service discovery query language of UPnP has minimal expression power, namely keywords; UPnP services accept their unawareness of clients' contexts and require more participation from their clients. [29]

In many service discovery protocols, the user is presented with a list of best matching services, and the final decision about which to use is made by the user. A typical example of this would be a web search engine.

Related Patterns PROFILE-BASED SERVICE BROWSING pattern [7] recommends that when showing the user what services are available, the client software should filter the list based on user's preferences and terminal capabilities. This pattern was also inspired by PEOPLE KNOW BEST pattern in [1].

12. SERVER DOES HEAVY WORK

Context Clients CONSULT DIRECTORY (5) to discover services; the directory contains information about a large number of services.

Problem How to optimize the processing and communication requirements of the clients? It is often impractical, or even impossible, to transfer a large set of service information to the client due to limited bandwidth, storage and processing requirements.

Forces

- Clients can often better use context information and user preferences to select the right service among several alternatives: CLIENT KNOWS BEST (11);
- It is impractical to move the complete decision making process into clients due to constraints;
- Answering the query may require complex algorithms and processing large amounts of data; and
- Directory servers have better processing resources.

Solution Let the directory server perform complex search operations. This reduces the amount of information that needs to be sent to the client. However, as CLIENT KNOWS BEST (11) the directories must not be too aggressive—a careful equilibrium between minimization of communication requirements and maximizing of the client's abilities to leverage its context awareness and preferences must be obtained.

Resulting Context The client obtains the information necessary to access a service fulfilling the current need. However, the directory servers become more complex and their resource requirements increase. Then it might make sense to distribute the directory further, e.g. using ALIGN DIRECTORIES WITH ORGANIZATION (8).

Known Uses Service discovery protocols (e.g., [12], [15], [26], and [29]), web search engines (e.g., [14]), and directories in general (e.g., LDAP [16] and UDDI [23]) allow clients to define search criteria, possibly even with complex query languages. Implementing complex algorithms for ranking the results (such as Google's PageRank) is possible too.

Acknowledgments

We would like to thank our EuroPLoP 2004 shepherd, Michael Kircher, and the participants of the workshop for their encouraging comments and suggestions. Mari Korkea-aho and Sanna Liimatainen also provided valuable comments on work that lead to these patterns.

References (other patterns)

[1] Michael Adams, James Coplien, Robert Gamoke, Robert Hanmer, Fred Keeve, and Keith Nicodemus, “Fault-Tolerant Telecommunication System Patterns”, PLoP 1995.

[2] Boualem Benatallah, Marlon Dumas, Marie-Christine Fauvet, Fethi A. Rabhi, and Quan Z. Sheng, “Overview of Some Patterns for Architecting and Managing Composite Web Services”, *ACM SIGecom Exchanges* 3(3):9–16, 2002.

[3] Guy Bieber, “Service-Oriented Programming”, Jini Pattern Language workshop at OOPSLA 2000.

[4] Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, and Michael Stal, *Pattern-Oriented Software Architecture: A System of Patterns*, John Wiley & Sons 1996.

[5] Bruce Cohen, “Locate and Track”, Jini Pattern Language workshop at OOPSLA 2000.

[6] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley 1994.

[7] Martin Gitsels and Jochen Sauter, “Profile-based Service Browsing – A Pattern for Intelligent Service Discovery in Large Networks”, Jini Pattern Language workshop at OOPSLA 2000.

[8] Michael Kircher and Prashant Jain, *Pattern-Oriented Software Architecture, Volume 3: Patterns for Resource Management*, John Wiley & Sons 2004.

[9] Jörg Roth, “Patterns of Mobile Interaction”, *Personal and Ubiquitous Computing* 6(4):282–289, 2002.

[10] Markus Alexander Wischy, “Patterns in Universal Plug & Play”, Jini Pattern Language workshop at OOPSLA 2000.

[11] Uwe Zdun, Michael Kircher, and Markus Völter, “Remoting Patterns”, *Internet Computing* 8(6):60–68, 2004.

References (known uses)

[12] Apple Computer, “Apple developer connection—Rendezvous”, http://developer.apple.com/macosx/rendezvous/, 2004.

[13] Tim Berners-Lee, Larry Masinter, and Mark McCahill, “Uniform Resource Locators (URL)”, RFC 1738, IETF, 1994.

[14] Sergey Brin and Lawrence Page, “The Anatomy of a Large-Scale Hypertextual Web Search Engine”, *Computer Networks and ISDN Systems* 30(1–7):107–117, 1998.

[15] Erik Guttman, Charles Perkins, John Veizades, and Michael Day, “Service location protocol, version 2”, RFC 2608, IETF, 1999.

[16] Jeff Hodges and RL Morgan, “Lightweight Directory Access Protocol (v3): Technical Specification”, RFC 3377, IETF, 2002.

[17] Institute of Electrical and Electronics Engineers, “Information Technology—Telecommunications and Information Exchange between Systems—Local and Metropolitan Area Network—Specific Requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications”, IEEE Standard 802.11, 1999.

[18] James Kempf and Jason Goldschmidt, “Notification and Subscription for SLP”, RFC 3082, IETF, 2001.

[19] Microsoft, “Active Directory Architecture”, Microsoft TechNet, http://www.microsoft.com/technet/prodtechnol/windows2000serv/technologies/activedirectory/deploy/projplan/adarch.mspx, 2004.

[20] Microsoft, “Description of the Microsoft Computer Browser Service”, Microsoft Knowledge Base Article 188001, http://support.microsoft.com/?kbid=188001, 2004.

[21] Paul Mockapetris, “Domain names – concepts and facilities”, RFC 1034, IETF, 1987.

[22] Pekka Nikander, Jukka Ylitalo, and Jorma Wall, “Integrating security, mobility, and multi-homing in a HIP way”, Network and Distributed Systems Security Symposium (NDSS) 2003.

[23] OASIS, “UDDI API specification, version 2.04”, http://uddi.org/pubs/ProgrammersAPI-V2.04-Published-20020719.htm, 2002.

[24] Matthew Peterson et al., “OpenSLP home page”, http://www.openslp.org/, 2004.

[25] Jonathan Rosenberg et al., “SIP: Session initiation protocol”, RFC 3261, IETF, 2002.

[26] Sun Microsystems, “Jini Architecture Specification, version 1.2”, http://wwws.sun.com/software/jini/specs/jini1.2html/jini-title.html, 2001.

[27] Sun Microsystems, “Jini Technology Core Platform Specification, version 1.2”, http://wwws.sun.com/software/jini/specs/jini1.2html/core-title.html, 2001.

[28] Susan Thomson and Thomas Narten, “IPv6 stateless address autoconfiguration”, RFC 2462, IETF, 1998.

[29] UPnP Forum, “UPnP device architecture, version 1.0.1”, http://www.upnp.org/resources/documents.asp, 2003.

[30] Paul Vixie, Susan Thomson, Yakov Rekhter, and Jim Bound, “Dynamic Updates in the Domain Name System (DNS UPDATE)”, RFC 2136, IETF, 1997.

[31] Steve Waterhouse, “JXTA search: Distributed search for distributed networks”, http://search.jxta.org/JXTAsearch.pdf, 2001.