Providing Network Services to Mobile Users with SLP and Jini

Service Discovery and Interaction for Nomadic Computing

PROF. DR.-ING. NORBERT LUTTENBERGER
Dept. for Computer Science, Christian-Albrechts-University Kiel (FRG)
Res. Group for Communication Systems

1. Introduction: Terms and Classifications
2. Common Concepts
3. Service Location Protocol
4. Jini
5. Jini for Nomads
6. Summary and Outlook

References

[1] ohne Autorenangabe:
SLP White Paper.
http://playground.sun.com/svrloc/slp_white_paper.html

[2] ohne Autorenangabe:
An Introduction to SLP.
http://www.openslp.org/doc/html/IntroductionToSLP/index.html

[3] Guttman, E., Perkins, C., Veizades, J., Day, M.:
Service Location Protocol, Version 2.
RFC-2608, June 1999.

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

[5] Kempf, J., Guttman, E.:
An API for Service Location.
RFC-2614, June 1999.

[6] Govea, J., Babeau, M.:
Comparison of Bandwidth Usage: Service Location Protocol and Jini.
Technical Report TR-00-06, School of Computer Science, Carleton University, Oct. 2000.

[7] Bradtke, M.:
Sicherheitsdienste für das Nomadic Computing in Jini-Netzen.
Diplomarbeit am Institut für Informatik u. Prakt. Mathematik der CAU zu Kiel, Dez. 2001.

[8] Eronen, P., Lehtinen, J., Zitting, J., Nikander, P.:
Extending Jini with Decentralized Trust Management.
In: OpenArch'2000, Tel Aviv, Israel, 2000. (auch: http://citeseer.nj.nec.com/eronen00extending.html)

[9] Eronen, P., Nikander, P.:
Decentralized Jini security.
In: Network and Distributed System Security Symposium (NDSS 2001), San Diego, California, Februar 2001. (auch: http://citeseer.nj.nec.com/eronen01decentralized.html)

[10] Eronen, P.:
Security in the Jini Networking Technology: A Decentralized Trust Management Approach.
http://citeseer.nj.nec.com/eronen01security.html (2001)

[11] Kleinrock, L.:
Nomadic Computing and Communications.
In: NII 2000 Steering Committee, National Research Council (Ed.): *The Unpredictable Certainty – Information Infrastructure Through 2000. White Papers.* Washington (National Academy Press) 1997, ISBN 0-309-06036-2, pp. 335–342. (auch: http://bob.nap.edu/readingroom/books/whitepapers/)

[12] Neuman, B.C.:
Proxy–Based Authorization and Accounting for Distributed Systems.
In: Proceedings of the 13th International Conference on Distributed Computing, Pittsburgh, May 1993, pp. 283–291, 1993. (auch: http://citeseer.nj.nec.com/article/neuman91proxybased.html)

[13] Newmarch, J.:
Jan Newmarch's Guide to JINI Technologies (Version 2.08), 12 June 2001.
http://jan.netcomp.monash.edu.au/java/jini/tutorial/Jini.xml

[14] Posegga, J.:
Jini: Infrastruktur für dynamische Dienste in verteilten Systemen.
Informatik Spektrum 22, 1 (2/1999), 43–44.

[15] Schoch, Th.:
An Authentication and Authorization Architecture for Jini Services.
http://www.inf.ethz.ch/schoch/da.pdf (2000)

[16] Schoch, Th., Krone, O., Federrath, H.:
Making Jini Secure.
http://www.inf.ethz.ch/schoch/making-jini-secure.pdf (2000)

[17] Waldo, J.:
Jini Technology Architectural Overview.
http://www.sun.com/Jini/whitepapers/architecture.html

[18] Kindberg, T., Fox, A.:
System Software for Ubiquitous Computing.
Pervasive Computing, Jan-March 2002, 70–81

Introduction: Terms and Classifications

- *Nomadic Computing*

„But, in fact, most of us are nomads, moving between office, home, airplane, hotel, automobile, branch office, conference room, bedroom, etc. In so doing, we often find ourselves decoupled from our "home base" computing and communications environment. […] Nomadicity may be defined as the system support needed to provide a rich set of computing and communication capabilities and services to nomads as they move from place to place in a transparent, integrated and convenient form.“ L. Kleinrock (1997)

➔ Nomads need access to public and private services and data from different situations, locations, environments, …

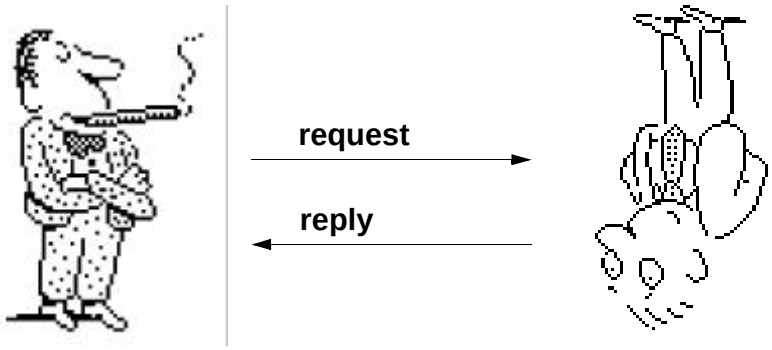
Introduction: Terms and Classifications (2)

- Four different views on “mobility”

devices are portable	small, lightweight devices (often not connected to the network) („Tragbarkeit“)
users/devices roam	changing points-of-attachment to the network („Freizügigkeit“)
users/devices are reachable	devices can be reached from other devices, independent of their current location („Erreichbarkeit“)
users/devices are in motion	devices change their point-of-attachment during an ongoing session („Beweglichkeit“)

- Four different cooperation models

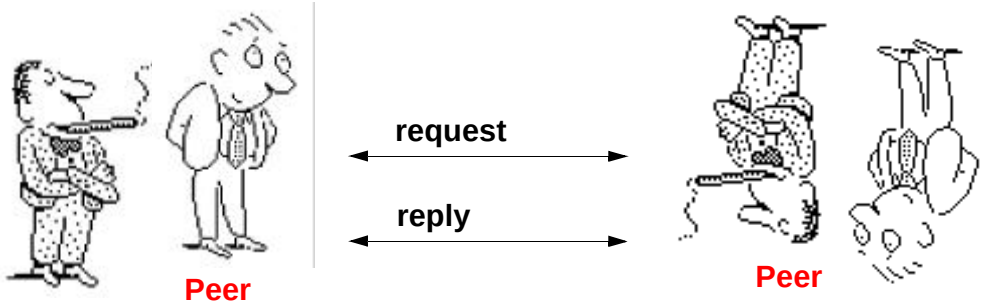
Client/Server	“classical” cooperation model; mobile device often seen as client-only host, servers hosted by fixed network hosts



— With respect to mobility: roaming clients, reachable servers!

- Four different cooperation models

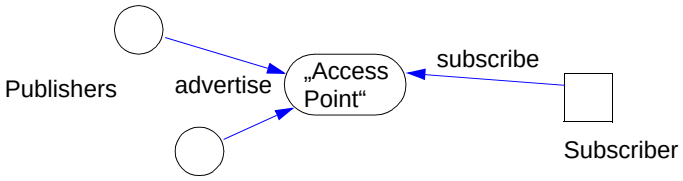
Client/Server	"classical" cooperation model; mobile device often seen as client-only host, servers hosted by fixed network hosts
Peer-to-Peer	devices host both clients and servers; sessions can be initiated from all devices; example: cellular phones



Introduction: Terms and Classifications (5)

- Four different cooperation models

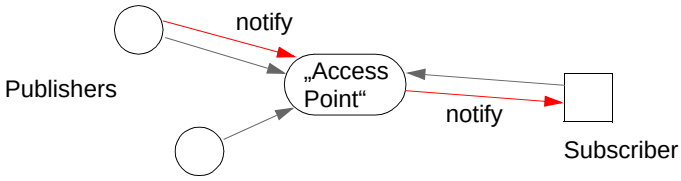
Client/Server	“classical” cooperation model;
Peer-to-Peer	devices host both clients and servers;
Publisher/Subscriber	Publisher advertises variable name and type to “Access Point”; Subscriber subscribes to variable;



Introduction: Terms and Classifications (6)

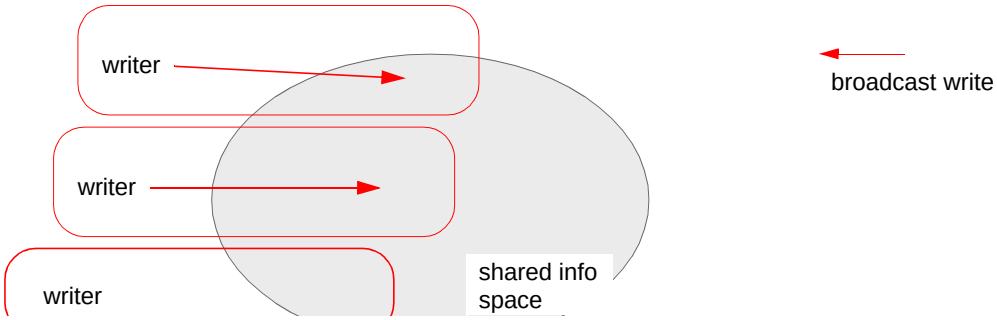
- Four different cooperation models

Client/Server	“classical” cooperation model;
Peer-to-Peer	devices host both clients and servers;
Publisher/Subscriber	Publisher advertises variable name and type to “Access Point”; Subscriber subscribes to variable; notification(s), when variable value changes



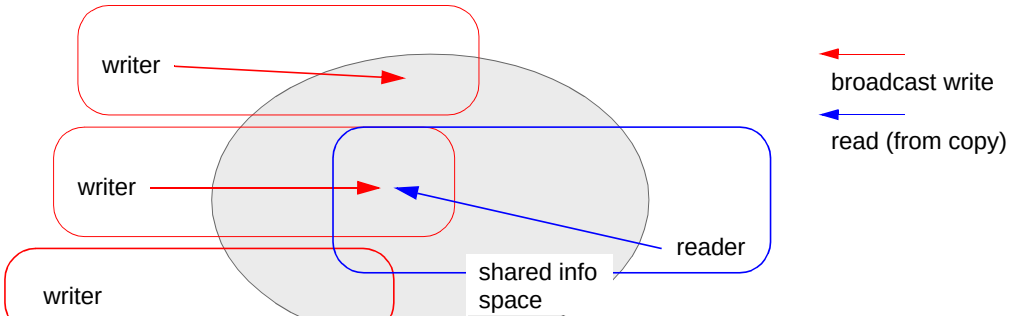
- Four different cooperation models

Client/Server	“classical” cooperation model;
Peer-to-Peer	devices host both clients and servers;
Publisher/Subscriber	publisher advertises variable name and type;
Distributed Shared Information Space	enhanced publisher/subscriber: data-centric architecture without hub; broad-/multicast communication; still experimental



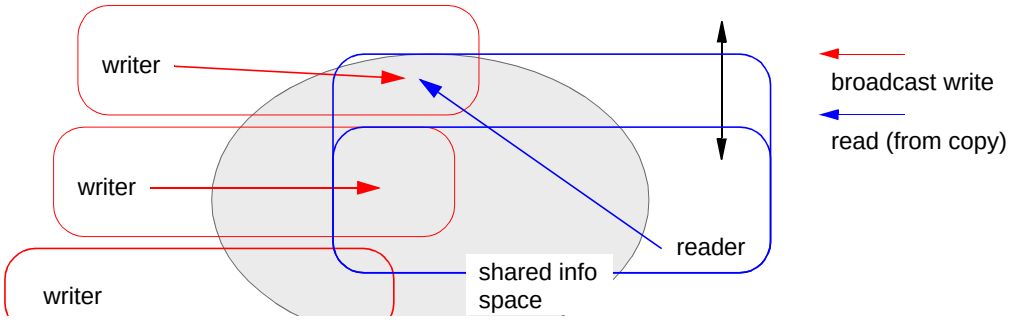
- Four different cooperation models

Client/Server	“classical” cooperation model;
Peer-to-Peer	devices host both clients and servers;
Publisher/Subscriber	publisher advertises variable name and type;
Distributed Shared Information Space	enhanced publisher/subscriber: data-centric architecture without hub; broad-/multicast communication; still experimental



- Four different cooperation models

Client/Server	“classical” cooperation model;
Peer-to-Peer	devices host both clients and servers;
Publisher/Subscriber	publisher advertises variable name and type;
Distributed Shared Information Space	enhanced publisher/subscriber: data-centric architecture without hub; broad-/multicast communication; still experimental



Introduction: Terms and Classifications (10)

- Upper Layers in the mobile computing context

  require additional functions for

Session layer	→ service discovery → service interaction (API)
Presentation layer	→ adaptation to different content presentations
Application layer	→ adaptation to different cooperation models (?)

- SLP and Jini in a box
  - Service Location Protocol

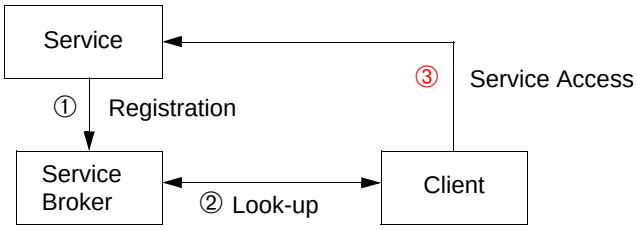
SLP	C/S	P2P	P/S	DSIS
portable	service discovery			
roaming				
reachable				
moving				

  - Jini

Jini	C/S	P2P	P/S	DSIS
portable	L5 service discovery service interaction by "polymorphic proxies" maybe → L6 content presentation (?) L7 different cooperation models (?)			
roaming				
reachable				
moving				

Common Concepts

- Support for “spontaneous interoperation”
  - UPnP, UDDI, Service Location Protocol (SLP), Jini, …
  - provision of a *service broker*:
    „Directory Agent“ in SLP, „Lookup Service“ in Jini



Service Location Protocol

- SLP Standards
  - Version 1:
    July 1997, RFC-2165
  - Version 2 (current version):
    June 1999, RFC-2608
    Authors: E. Guttman, C. Perkins, J. Veizades, M. Day

Service Location Protocol (2)

- SLP for Users

  New employee: „Hey Stan, the setup program is asking me for the name of our printer. What should I type in?“
  Stan: „Which printer do you want?“
  New employee: „I need postscript; next to me is the big printer by the copier.“
  Stan: „I’ve heard it does not work well with postscript applications. You’ll have to use the one down the hall.“
  New employee: „Ok. What should I type in?“
  Stan: „Actually, I don’t know; I use the one by the copier. You’ll probably have to call the help desk.“
  New employee: <groan> … [2]

  — User specifies
    - a service (or device): printer
    - some service attributes: printer language, location, …
  — User needs
    - a Service Access Point (SAP)

Service Location Protocol

„The Service Location Protocol (SLP) is a new IETF standards-track protocol designed to simplify the discovery and use of network resources such as printers, Web servers, fax machines, video cameras, filesystems, backup devices (tape drives), databases, directories, mail servers, calendars, and the unimaginable future variety of services coming our way. It is particularly well suited for client-server applications and establishing connections between network peers that offer or consume generic services. As the focus of interest shifts ever more onto the Internet and the World-Wide Web, network services will gain prominence in the minds of computer users everywhere. Accessing network services by manual configuration or administrative procedures will no longer be economically attractive. In the networked world of the future, interchangeable services will appear and disappear, and providing for the dynamic nature of their availability is an important accomplishment for SLP.“ [1]

„Protocols that support service location are often taken for granted, mostly because they are already included (without fanfare) in many network operating systems. For example, without Microsoft’s SMB service location facilities, „Network Neighborhood“ could not discover services available for use on the network and Novell NetWare would be unable to locate NDS trees. Nevertheless, an IETF servie location protocol was not standardized until the advent of SLP. Because it is not tied to a proprietary technology, SLP provides a service location solution that could become extremely important (especially on Unix) platforms.“ [2]

Service Location Protocol (3)

- SLP for Software Developers

  „Consider the following example of a software engineer who is writing an LDAP-enabled application. Currently, the only way to know the hostname of the LDAP server to use in the call to `ldap_init()` is to read it from a configuration file. This file must be created at install time and updated as the location of the LDAP server changes. … With SLP, the developer uses SLP to obtain the host names and attributes of LDAP servers that are displayed to the user […] if the user desires to connect to an LDAP server. … SLP does not always eliminate the need to prompt users for information. However, it does allow the software developer to present a descriptive list of services that can be understood by the user.“ [2]

  — for the developper:

    • an API for SLP

  — for the administrator:

    • a procedure for producing service descriptions

Service Location Protocol

„The Service Location Protocol provides a scalable framework for the discovery and selection of network services. Using this protocol, computers using the Internet need little or no static configuration of network services for network based applications. This is especially important as computers become more portable, and users less tolerant or able to fulfill the demands of network system administration. …

The Service Location Protocol (SLP) provides a flexible and scalable framework for providing hosts with access to information about the existence, location, and configuration of networked services. Traditionally, users have had to find services by knowing the name of a network host (a human readable text string) which is an alias for a network address. SLP eliminates the need for a user to know the name of a network host supporting a service. Rather, the user supplies the desired type of service and a set of attributes which describe the service. Based on that description, the Service Location Protocol resolves the network address of the service for the user.“ [3]

Service Location Protocol (4)

- Application areas
  - applicable for „networks under cooperative administrative control“

    „Such networks permit a policy to be implemented regarding security, multicast routing and organization of services and clients into groups which are not be feasible on the scale of the Internet as a whole.“ [3]
  - not applicable for networks „where there are hundreds of thousands of clients or tens of thousands of services“ [3]

    Keep in mind scalability:
    Does the effort grow linearly with the number of users/services or more than linear?

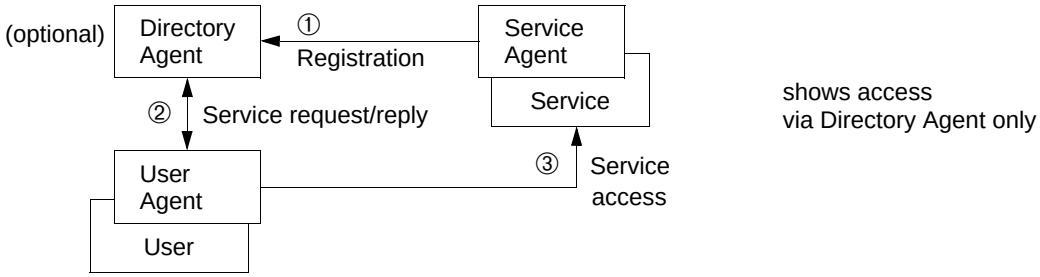
Service Location Protocol

„SLP is intended to function within networks under cooperative administrative control. Such networks permit a policy to be implemented regarding security, multicast routing and organization of services and clients into groups which are not be feasible on the scale of the Internet as a whole.

SLP has been designed to serve enterprise networks with shared services, and it may not necessarily scale for wide-area service discovery throughout the global Internet, or in networks where there are hundreds of thousands of clients or tens of thousands of services.“ [3]

Service Location Protocol (5)

- Three types of agents



— User Agent (UA):

A process working on the user's behalf to establish contact with some service.
The UA retrieves service information from the Service Agents or Directory Agents.

— Service Agent (SA):

A process working on the behalf of one or more services to advertise the services [to directory and/or user agents].

— Directory Agent (DA):

A process which collects service advertisements. … only one DA per given host. [3]

Service Location Protocol

„*Service Agent (SA)*. Services are resources that can be used by a device, a user, a program or another service. Some examples are memory space, printers, and mailboxes. In general any resource available in the network is potentially a service. An SA provides services advertising them by multicast or broadcast. An SA also intercepts and replies to queries about its services with a service access point. An SA registers its service with a DA when it is present.

Directory Agent (DA). Their primary function is to implement a repository of services where the clients can look for particular services given particular attributes. A DA catches the advertisements from the SAs, collects information from the advertisements of SAs, and replies on behalf of SAs to UAs when they request a particular service.

User Agent (UA). A UA is a client of the services. It looks for and requires services with particular characteristics by sending queries about services to the DAs or directly to the SAs.“ [6]

Service Location Protocol (6)

- Scopes
  - To improve scalability, agents have scopes.
  - Scopes are configured by system administrators.
  - Definition

    „A set of services, typically making up a logical administrative group. … Services are grouped together using 'scopes'. These are strings which identify services which are administratively identified. A scope could indicate a location, administrative grouping, proximity in a network topology or some other category. Service Agents and Directory Agents are always assigned a scope string.“
  - Request/reply messages carry scopes.

Service Location Protocol (7)

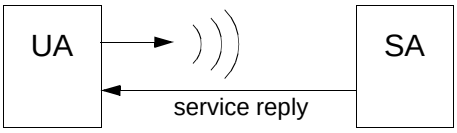
- Protocol
  - dialogues
    - How to find SAs and DAs?
    - registration of services
    - How to find a service?
  - message classes

Advertisements	DA Advertisement	DAs and SAs offer their services.
	SA Advertisement	
Service Registration (SA → DA)	Service Registration	An SA registers/deregisters a service with a DA; DA acknowledges the registration/deregistration.
	Service Deregistr.	
	Service Ack.	
Service Request (UA → DA/SA)	Service Request/Reply	UA requests a special service
	Service Type Req./Reply	UA requests some type of service
	Attribute Req./Reply	attributes of a service or service type

  - SLP multicast address 239.255.255.253
    UDP Port 427

Service Location Protocol (8)

- Dialogues
  - How to find a Service Agent? (in configurations without Directory Agent)



User Agent sends
multicast service request

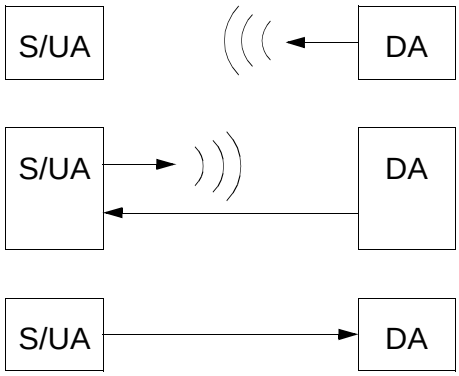
Configurations without Directory Agent(s) are discouraged:

- may generate many multicasts
- all SAs must permanently listen to multicast messages

➔ not suited for large networks

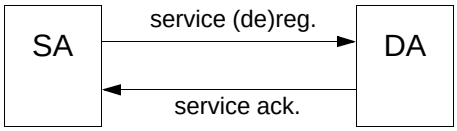
- Dialogues (cont'd.)
  - How to find a Directory Agent?

    four different ways:



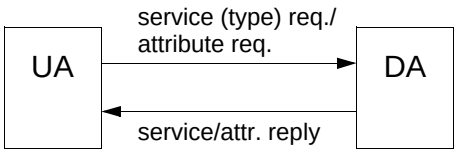
1. Service/User Agents listen to *multicast DA advertisements*
2. Service/User Agents send *multicast service type requests* with specified service type: `directory-agent`
3. static configuration
4. an option in the *Dynamic Host Configuration Protocol* (DHCP)

- Dialogues (cont'd.)
  - Registration and deregistration of services



Registration has limited lifetime; DA deletes a service registration when its lifetime expires

  - How to find a service?



User Agent asks for a service, for all available service types or for service (type) attributes.

Service Location Protocol (11)

- Modeling of network services and resources
  - Textual description of services
  - Key element is the *service type*:

    A name identifying the semantics by which the remainder of the `service:` URL is to be understood. It may denote either a particular network protocol, or an abstract service associated with a variety of protocols. If the service type denotes a particular protocol, then the service type name SHOULD either be assigned the name of a particular well known port by convention or be the Assigned Numbers name for the service. An abstract service type is a service type name which is associated with a variety of different protocols.

  - Standardization by so-called *service templates*:

    A formal description of the service attributes and service scheme associated with a particular service type.

    (from RFC-2609: Service Templates and URLs)

Service Location Protocol (12)

- Modeling of network services and resources (cont'd.)
  - URL Notation

```
service: URL   ::= "service:" service-type ":" sap ";" attrib-list
service-type   ::= abstract-type ":" url-scheme | concrete-type
abstract-type  ::= type-name [ "." naming-auth ]
concrete-type  ::= protocol [ "." naming-auth ]
```

  - Examples for an abstract type

```
service:device-drivers:ftp://x3.bean.org/drivers/diskdrivers.drv;
driver=scsi;platform=sys3.2-rs3000
```

```
service:device-drivers:tftp://x2.bean.org/vol3/disk/drivers.drv;
driver=scsi;platform=sys3.2-rs3000
```

```
service:device-drivers:http://www.bean.org/drivers/drivpak.drv;
driver=scsi;platform=sys3.2-rs3000
```

  - Example for a concrete type

```
service:ftp:/x3.bean.org/drivers/diskdrivers.drv;
driver=scsi
```

Service Location Protocol (13)

- Security
  - DA advertisements may be digitally signed.
  - SA registration messages may be digitally signed.
  - Digital signatures require system configuration:
    UAs must be enabled to verify digital signatures.

Service Location Protocol

„SLP is designed to make service information available, and it contains no mechanisms to restrict access to this information. Its only security property is authentication of the source of information, which prevents SLP from being used to maliciously propagate false information about the location of services.

Digital signatures. An SA can include a digital signature produced with public key cryptography along with its registration messages. A DA can the verify the signature before registering or deregistering any service information on the SA's behalf. These digital signatures are then forwarded in reply messages to UAs, so they can reject unsigned or incorrectly signed service information. Of course, DAs and UAs can only verify signatures, not produce them.

As an additional level of authentication, DAs can also include digital signatures with their advertisements. UAs and SAs can thus avoid DAs that have not been legitimately established by the site's administration because SLP agents that possess private keys for generating verifiable signatures are (by definition) trusted to legitimately advertise themselves."

Service Location Protocol

- SLP-API
  - defined for Java and C
  - here: definition for C

```
SLPReg()              service registration
SLPDeReg()            service deregistration
SLPFindSrvs()         locating a service
SLPFindAttrs()        service attributes
SLPFindSrvTypes()     finding out about available service types
…
```

Service Location Protocol (15)

- Comparison to DNS SRV

A recent proposal has been made to enter special resource records (SRV RRs) into the site DNS for locating services. For instance, the anonymous FTP site for an enterprise was often conventionally to be accessible by the domain. …

SLP takes care of the *intranet resource discovery* needs, and DNS SRV helps *external customers*. DNS SRV records do not offer the ability for selective acquisition of service handles. Users cannot expect to find a particular kind of service, since all services of the same type are lumped together into similar SRV records. In addition, only a service address may be obtained. Unlike the Service Location Protocol, service access information and service attributes may not be obtained with DNSSRV. Administratively, DNS, because of its critical nature, should not be used for extraneous directory operations. Resolving names into IP addresses is at the heart of the network operations of most sites, and separate functions should be kept separated from such essential operations. …Since DNS depends upon hierarchy for the structure of its namespace, and hierarchy is much less important or nonexistent in SLP, services can be more naturally administered in SLP, and directory services for SLP operations can be naturally broken into separate administrative domains in a way not naturally available to DNS. SLP agents are much smaller and easier to manage than DNS is.

Service Location Protocol (16)

- Comparison to LDAP

  Although LDAP has Lightweight in its name, SLP is still much more lightweight than LDAP. As before, SLP is better at resource management because it offers automatic directory maintenance, and because it does not require the inappropriate resource placement in a hierarchical, inflexible namespace. Adding new service types is _much_ easier than in LDAP. Queries by type are more efficient in SLP than in LDAP. Attribute descriptions for existing resources are available in SLP, but not in LDAP.

  SLP allows discovery with no a prior configuration. LDAP requires the address of the directory to start and knowledge of the directory schemas that LDAP uses.

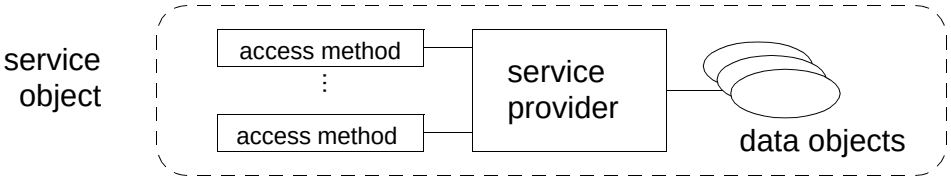
  SLP allows nonstandart attributes and new versions of the standardized service type templates, so they can evolve.

  Even with all these advantages, SLP will not likely be able to displace the current deployment of LDAP, especially for those uses which are best suited to X.500 as opposed to service location.

Jini

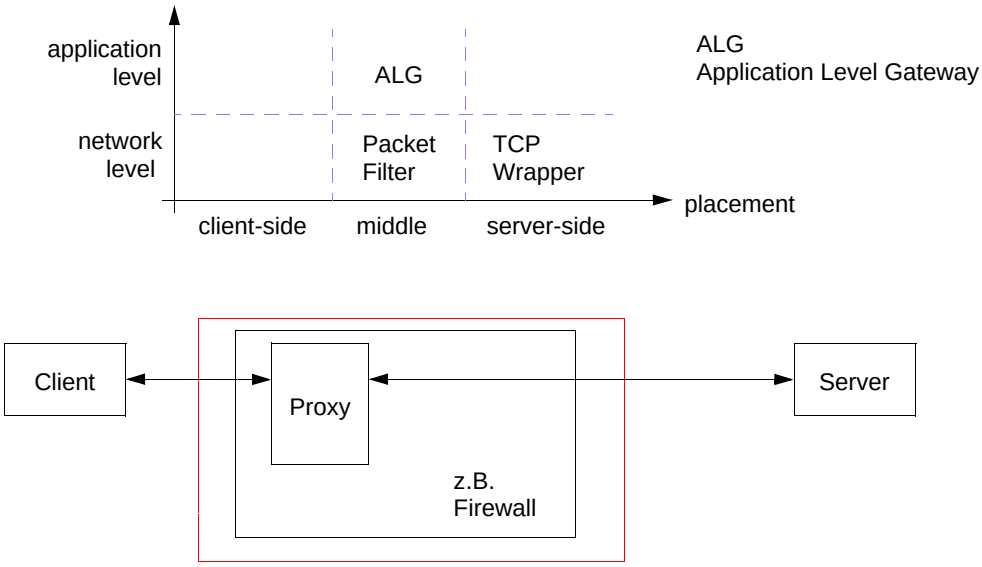
- What is Jini?
  - “Jini is not an acronym for anything, and does not have a particular meaning. (Though it gained a post hoc interpretation of ‘‘Jini Is Not Initials’’.)” [13]
  - components as provided by Sun
    - a specification of a set of middleware components
    - an API for writing services and components
    - an implementation (in pure Java) of the middleware (Java packages including source code)
    - Sun basic implementations of a number of ‘‘standard’’ services.

- Concept
  - object-oriented modeling of services/devices/systems: encapsulation in Java objects
  - access via object-provided methods (→ “proxies”) with well-defined interface



  - Proxy is mobile code: migrated to client on demand as serialized Java object (“marshalled object”)
  - Proxies may be seen as → *active capabilities*

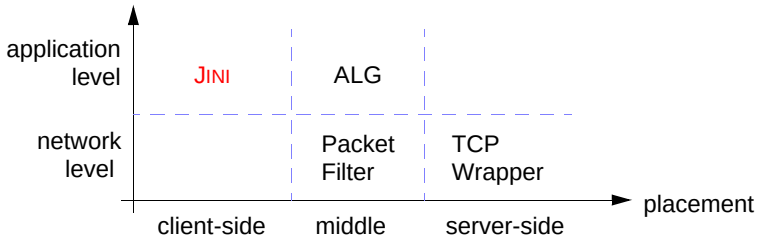
- Proxy allocation

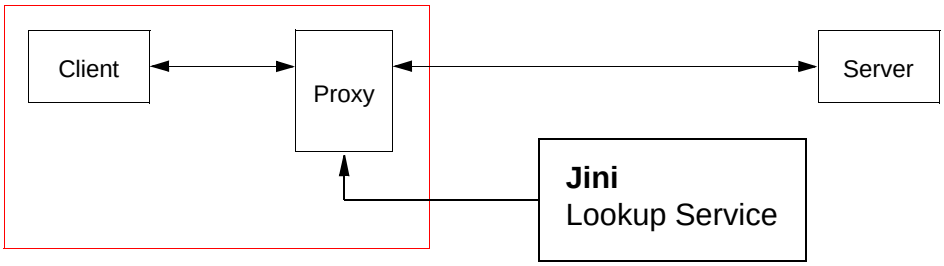


Jini

(4)

- Proxy allocation





Jini (5)

- Modelling of network services
  - an optional universally unique service identifier
  - a Java interface
  - a number of optional attributes

Jini

“A client is attempting to find one or more services that satisfy its requirements. It does so by creating a `ServiceTemplate` object and using this in a registrar’s `lookup()` call. A `ServiceTemplate` object has three fields

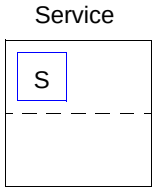
```
ServiceID           serviceID;
java.lang.Class[]   serviceTypes;
Entry[]             attributeSetTemplates;
```

An overview: firstly, if the client is repeating a request, then it may have recorded the `serviceID` from an earlier request. The `serviceID` is a ‘‘universally unique identifier’’ so can be used to identify a service unambiguously. This can be used by the service locator as a filter to quickly discard other services. Secondly, a client may want to find a service satisfying a number of interface requirements at once. For example, a client may look for a service that implements both `Toaster` and `FireAlarm` (so that it can properly handle burnt toast). Finally, the client will specify a set of attributes that must be satisfied by each service. Each attribute required by the client is taken in turn and matched against the set offered by the service. For example, in addition to requesting a `Toaster` with a `FireAlarm`, a client entry may specify a location in ‘‘GP South Building’’.This will be tried against all the variations of location offered by the service. A single match is good enough. An additional client requirement of, say, manufacturer would also have to be matched by the service.” from [13]

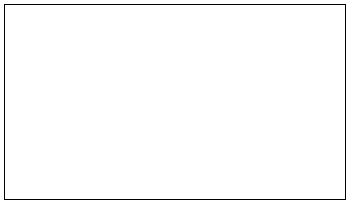
Jini

(6)

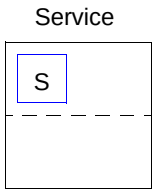
- Client/Service/Proxy interaction



Service has prepared
Proxy S

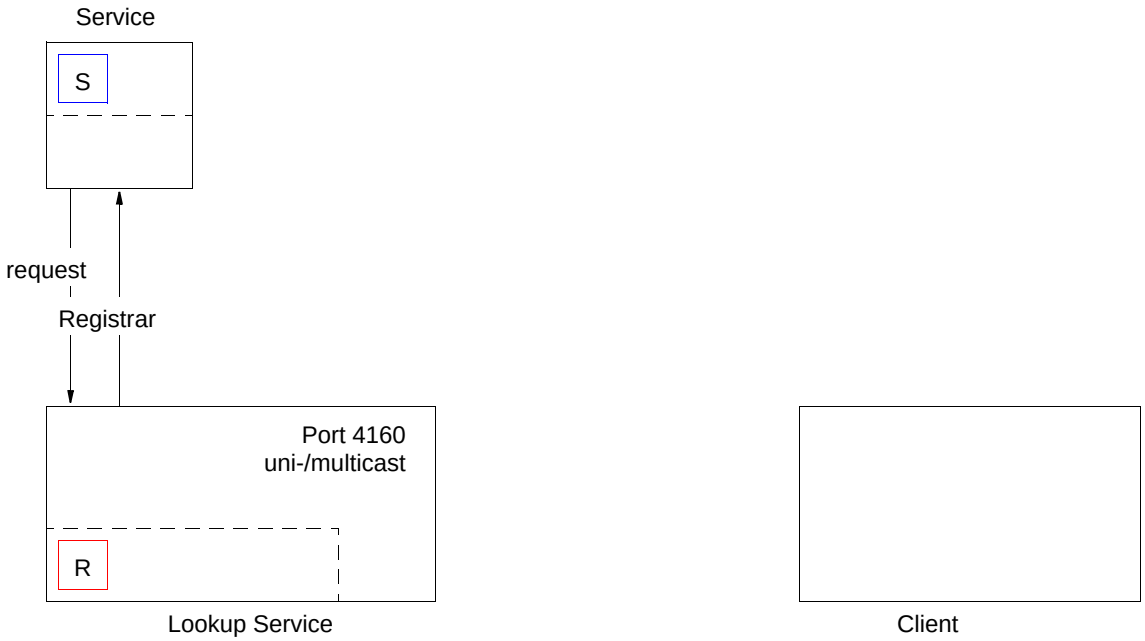


- Client/Service/Proxy interaction

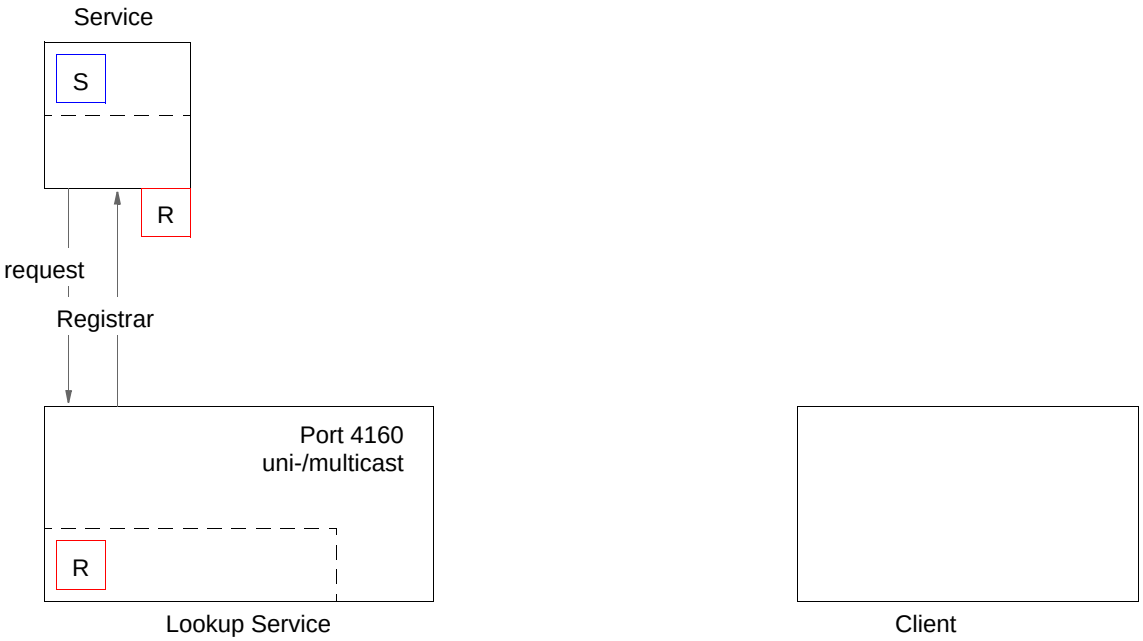


Jini Lookup Service has prepared
Proxy R: “Registrar”

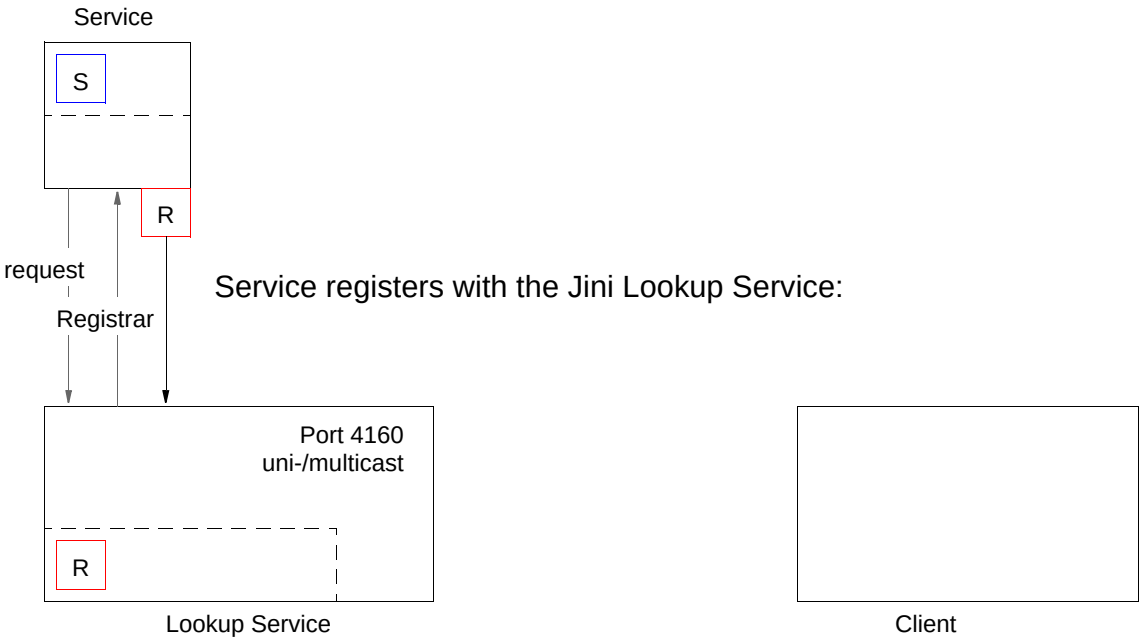
- Client/Service/Proxy interaction



- Client/Service/Proxy interaction

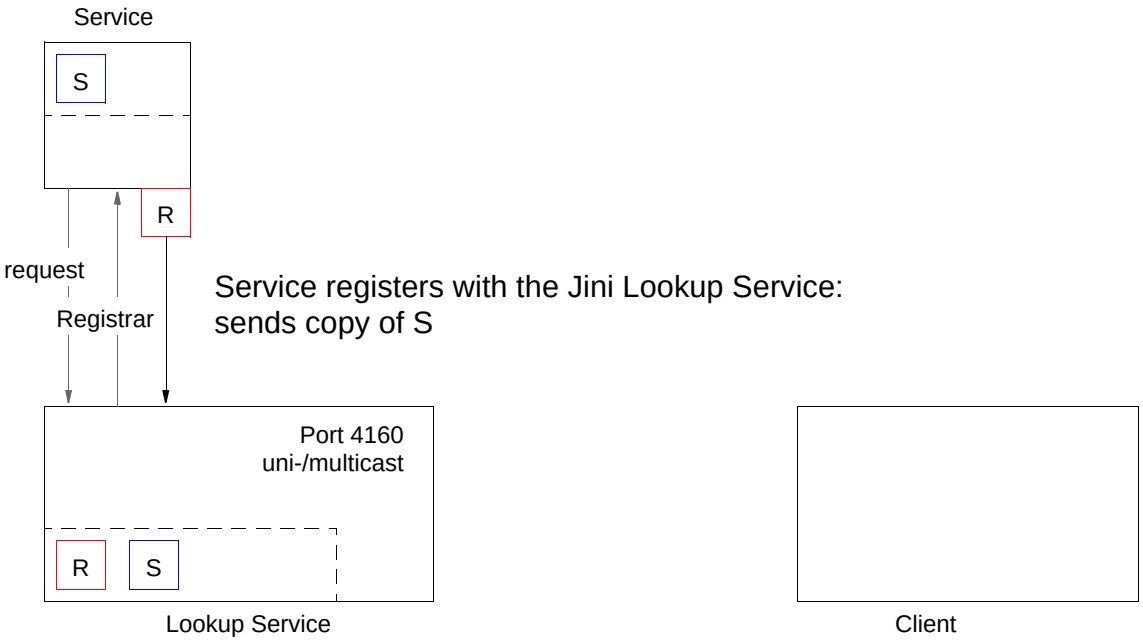


- Client/Service/Proxy interaction

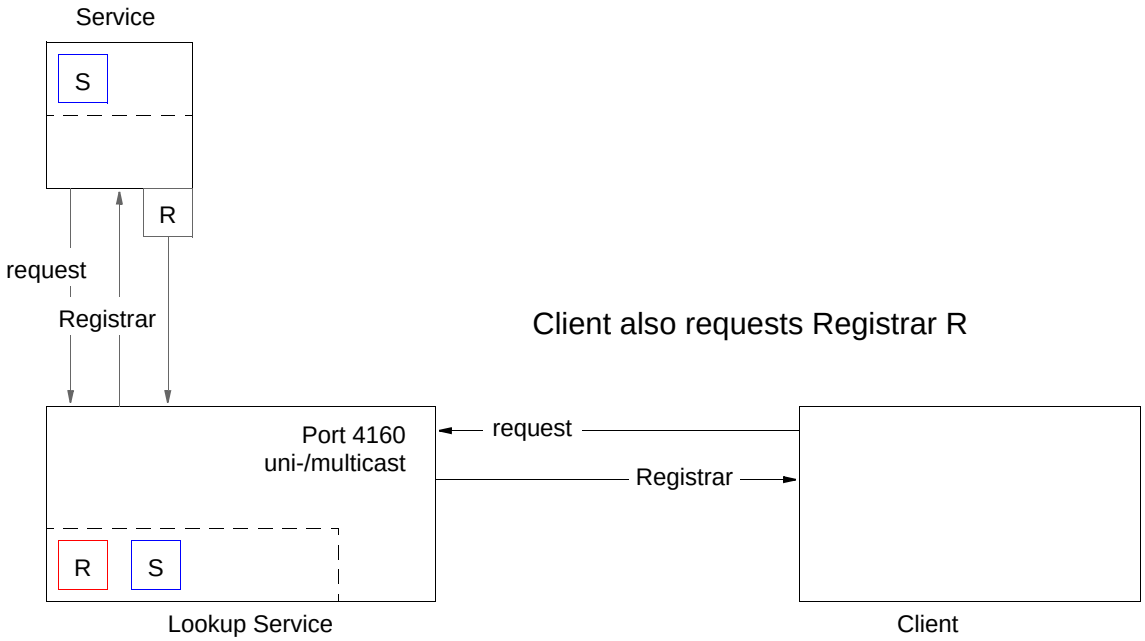


Service registers with the Jini Lookup Service:

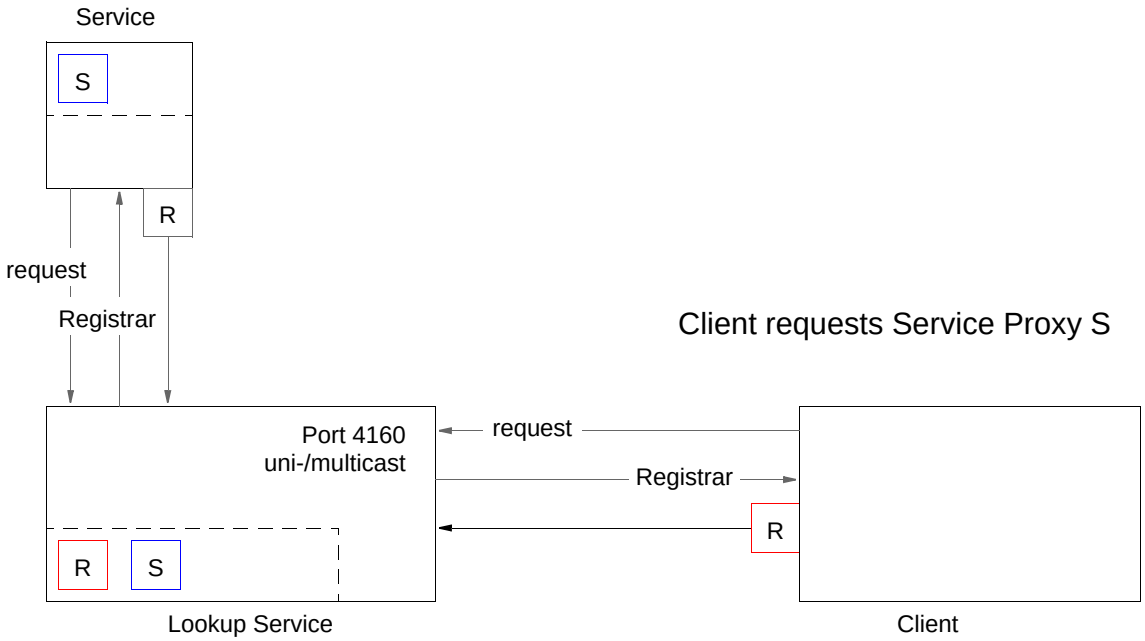
- Client/Service/Proxy interaction



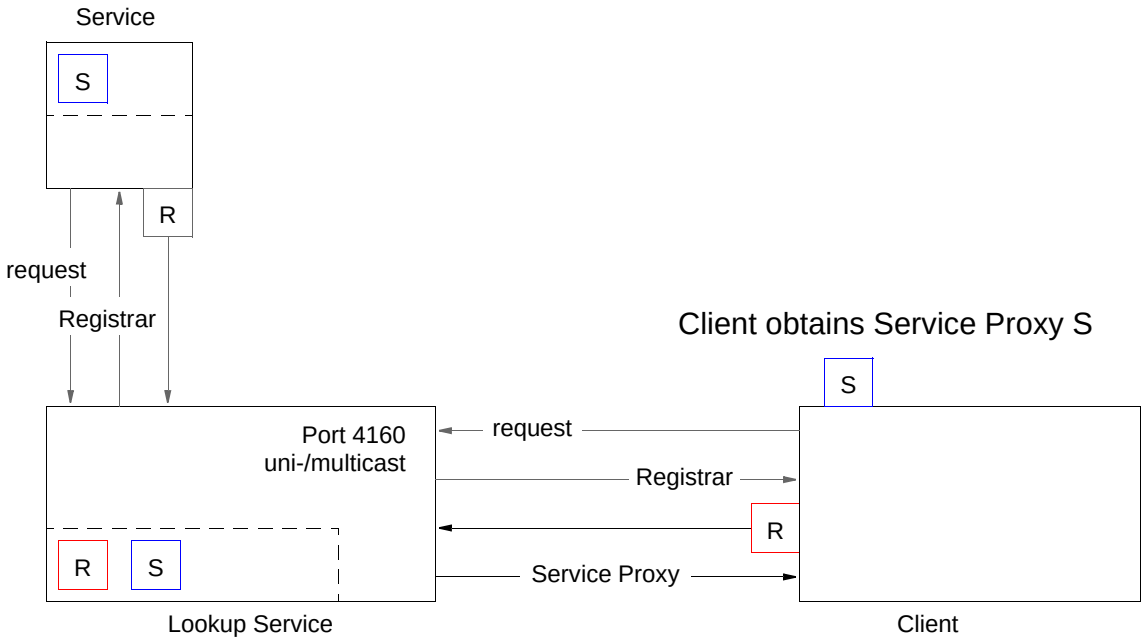
- Client/Service/Proxy interaction



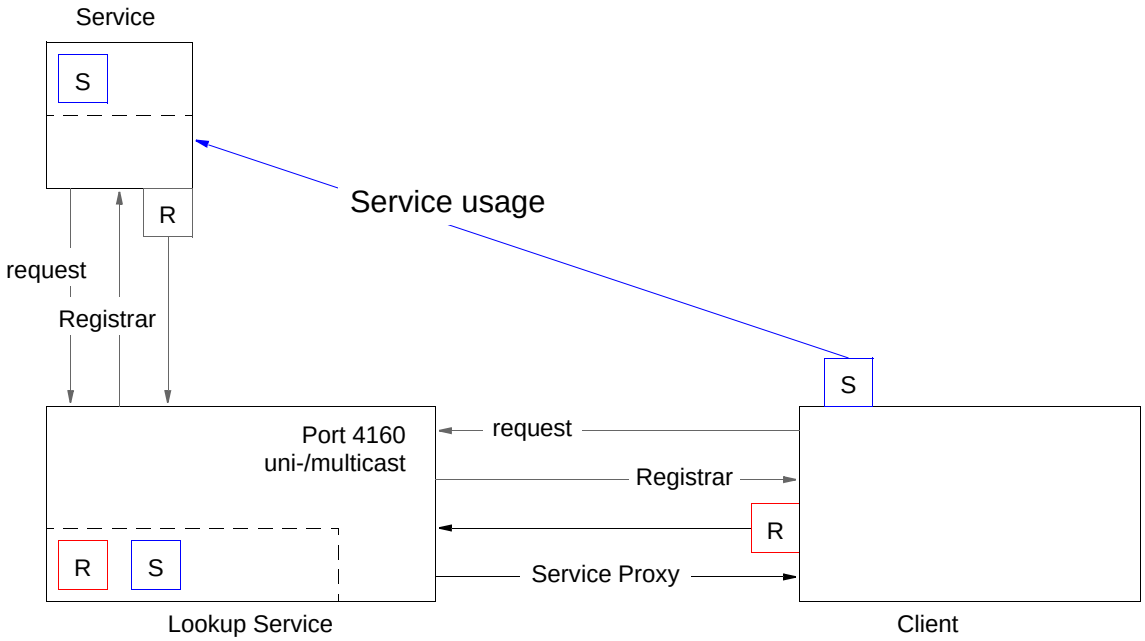
- Client/Service/Proxy interaction



- Client/Service/Proxy interaction

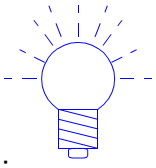


- Client/Service/Proxy interaction



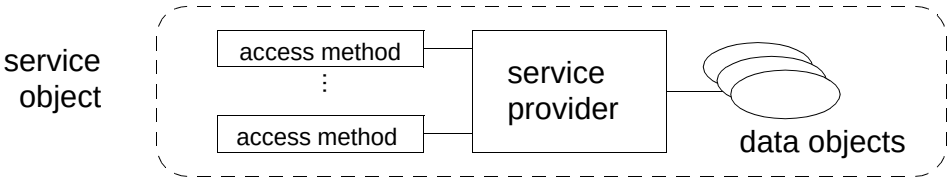
Jini for Nomads

- Concept



„**Polymorphic Services**“

➔ Provide proxies to nomads that can reflect the nomad’s current situation: home, office, vacations, business travel, internet café, …

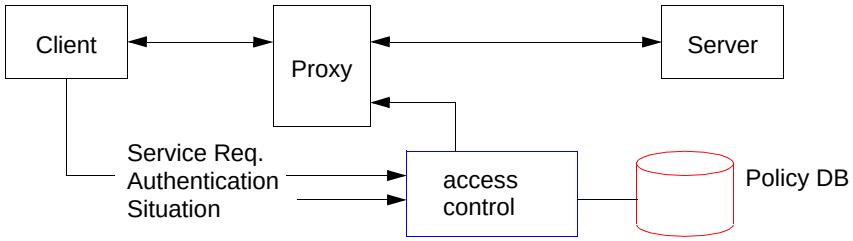


- object constraints
- timing constraints
- protocol constraints
- …

Jini for Nomads

(2)

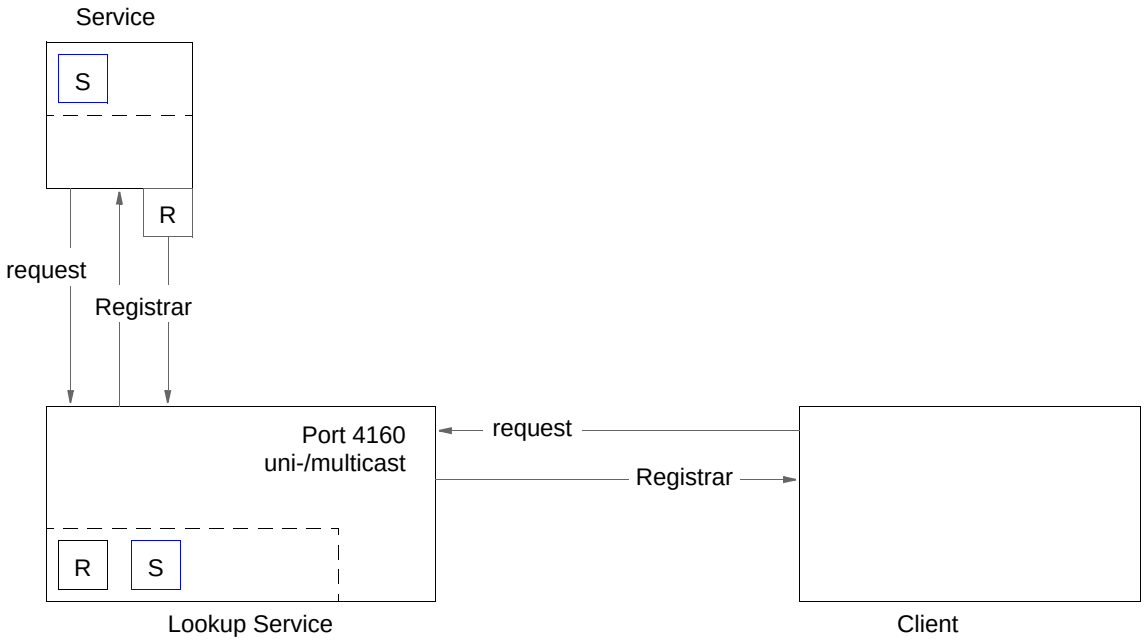
- Embedding polymorphic services in Jini (cont'd.)
  - nomad authentification
  - situation assessment
  - authorization by policy database



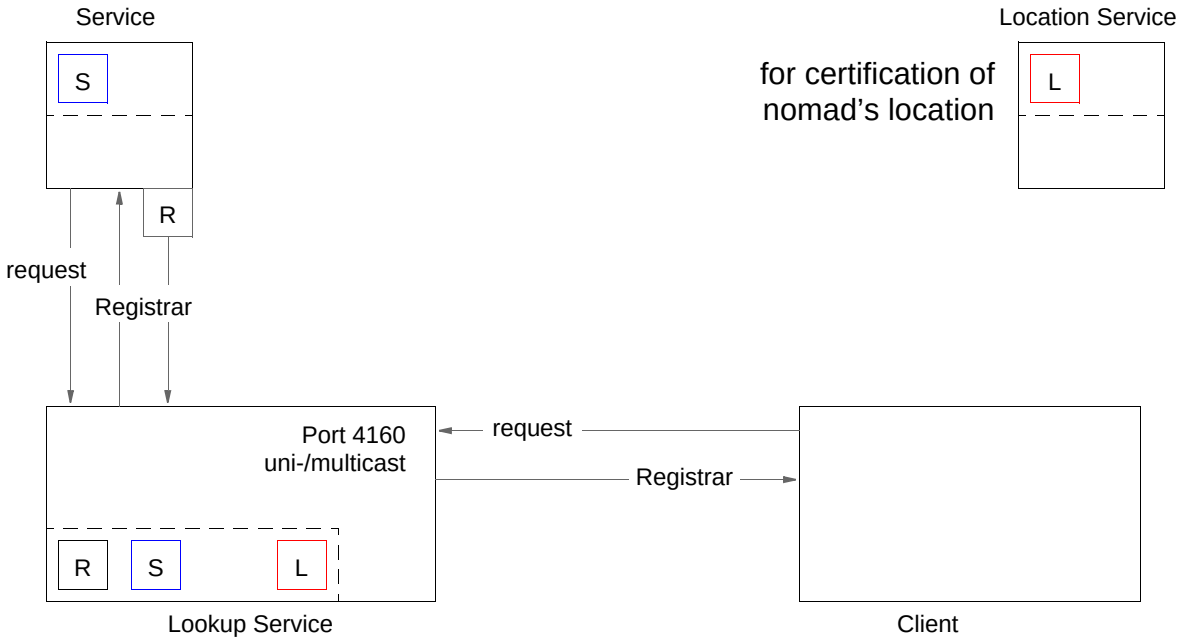
  - additional Jini security enhancements

Jini for Nomads (3)

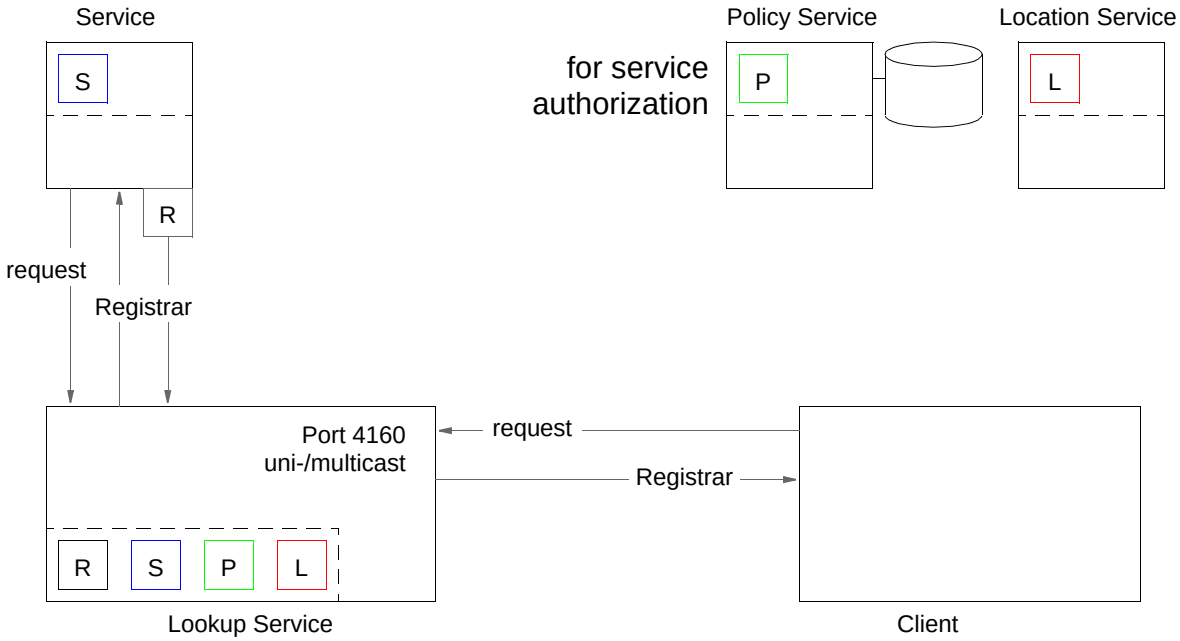
- Embedding polymorphic services in Jini (cont'd.)



- Embedding polymorphic services in Jini (cont'd.)

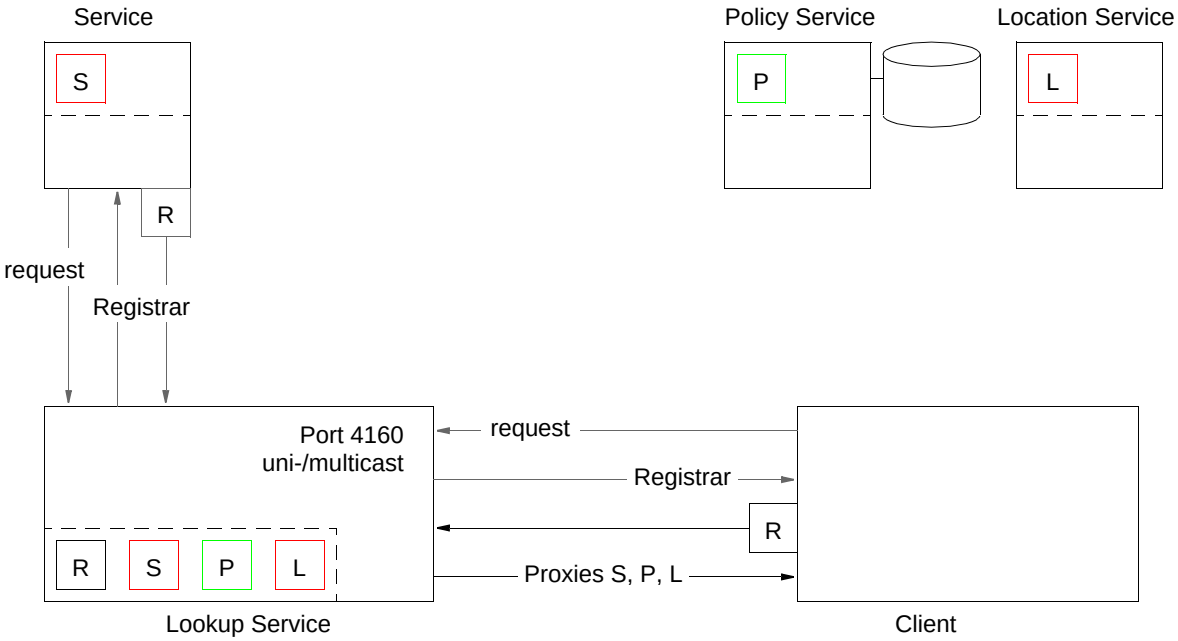


- Embedding polymorphic services in Jini (cont'd.)



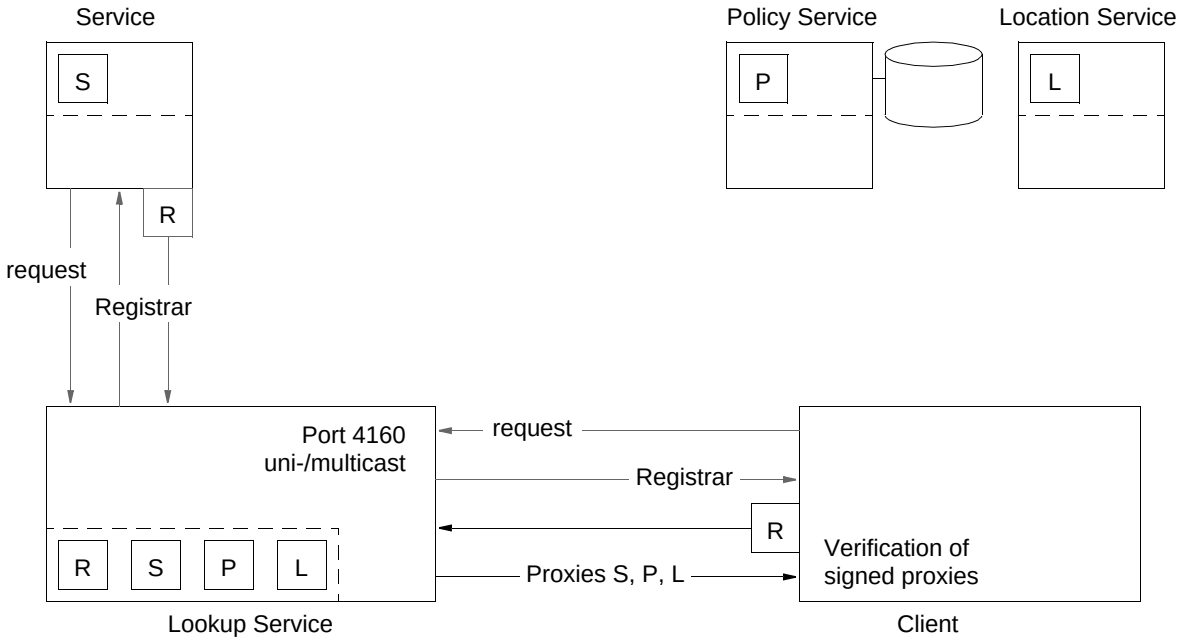
Jini for Nomads (6)

- Embedding polymorphic services in Jini (cont'd.)



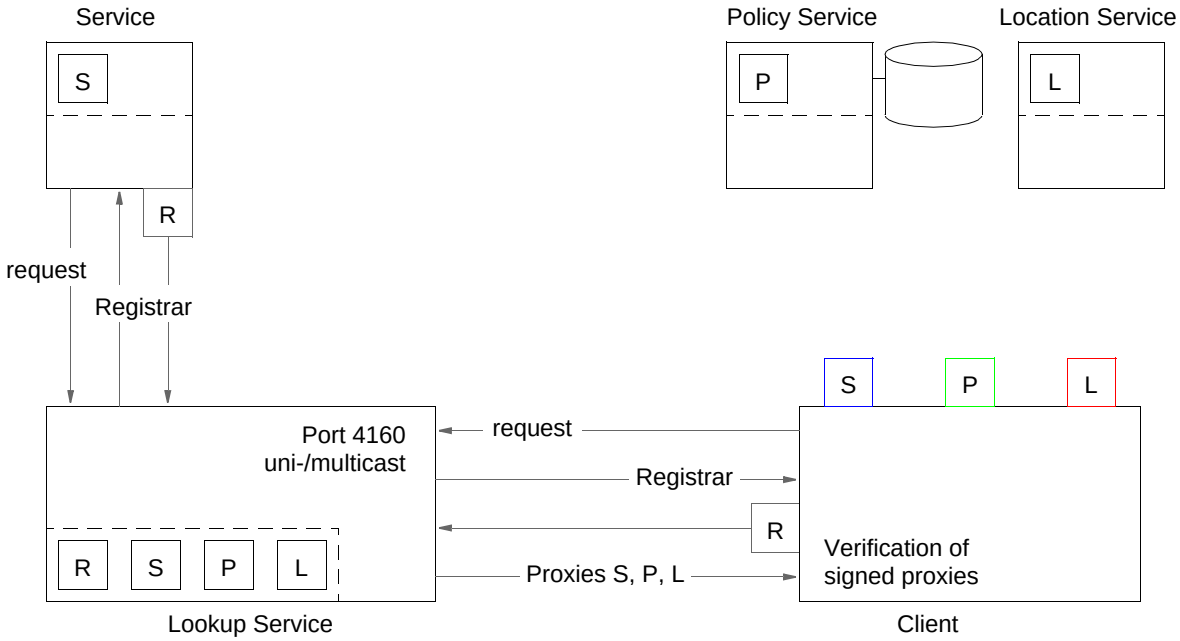
Jini for Nomads (7)

- Embedding polymorphic services in Jini (cont'd.)

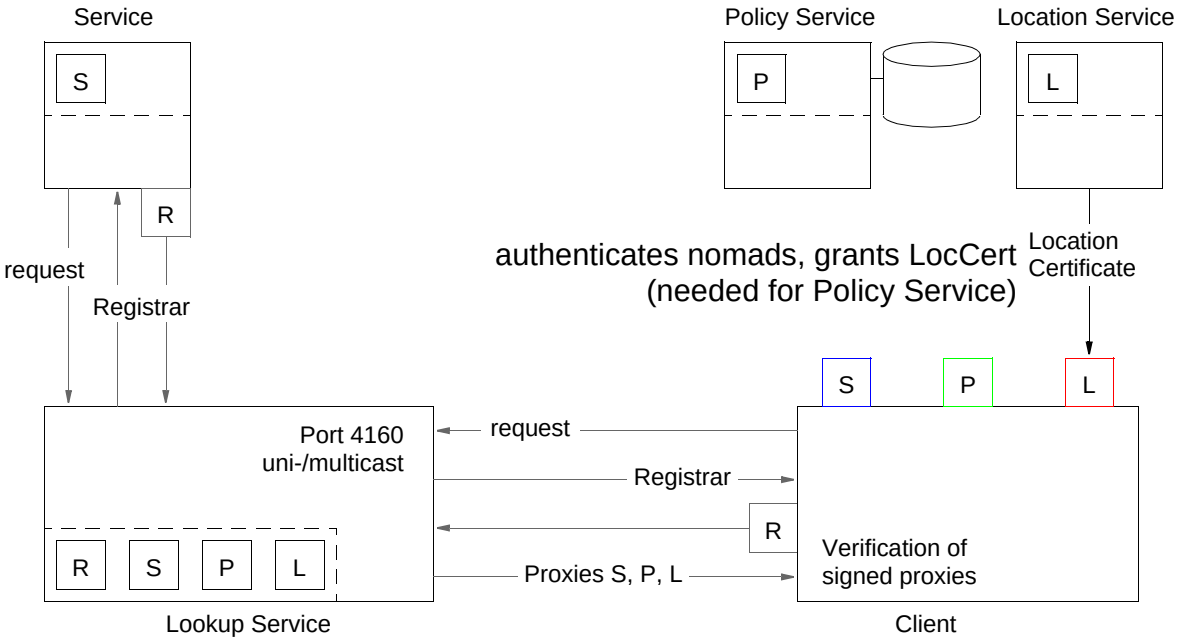


Jini for Nomads (8)

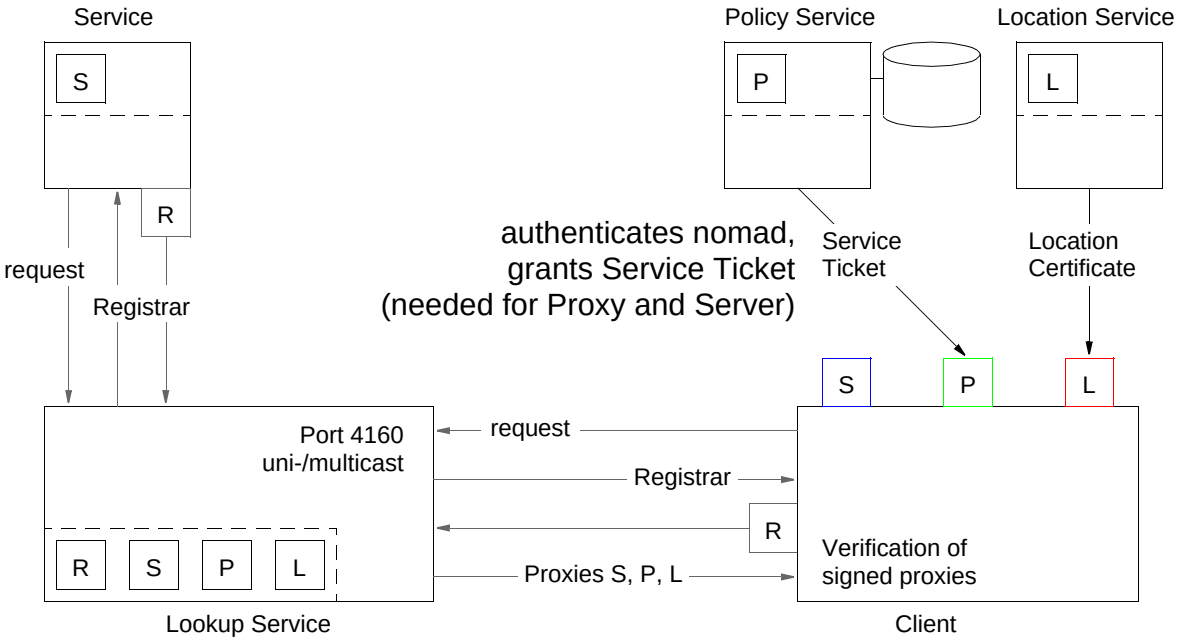
- Embedding polymorphic services in Jini (cont'd.)



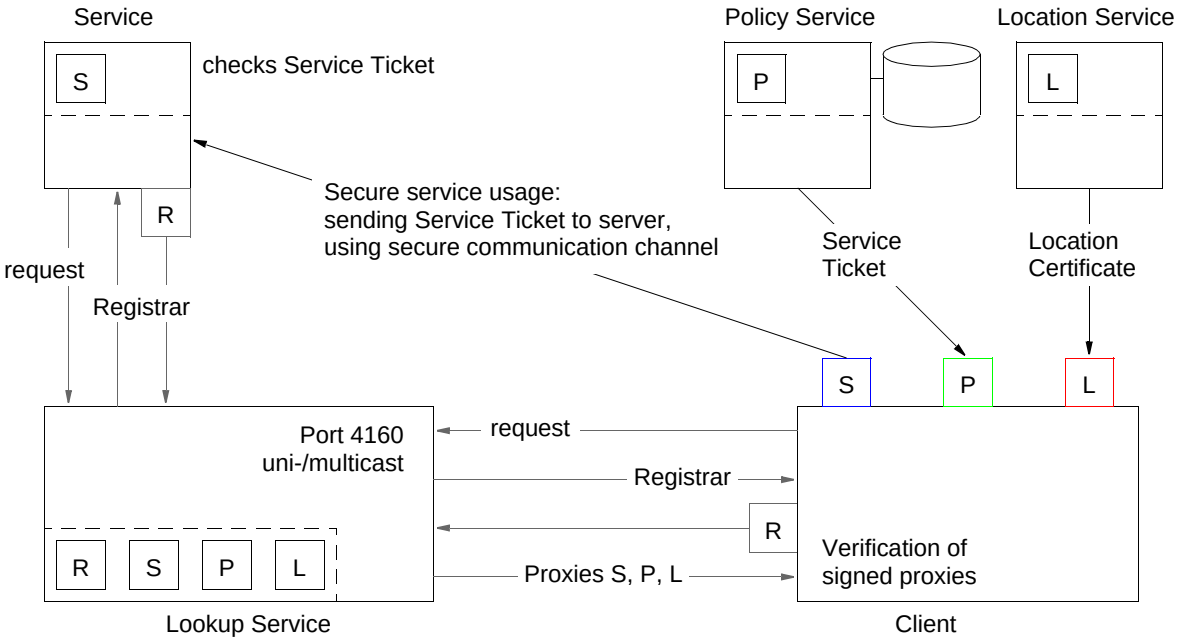
- Embedding polymorphic services in Jini (cont'd.)



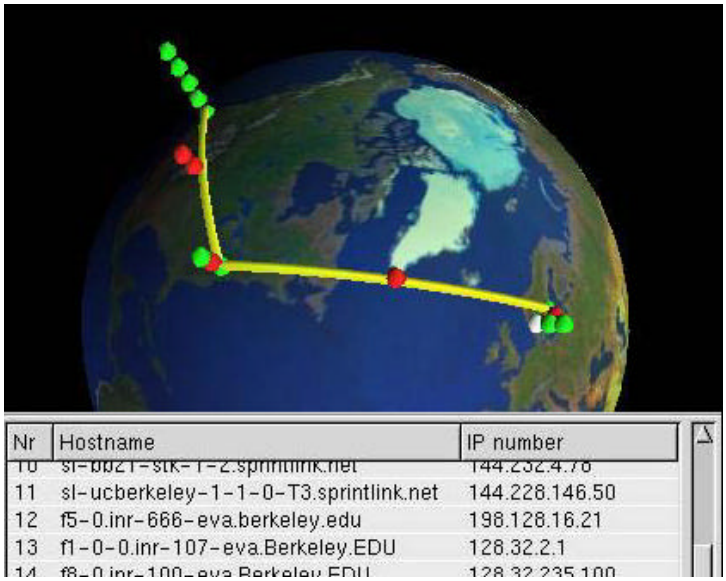
- Embedding polymorphic services in Jini (cont'd.)



- Embedding polymorphic services in Jini (cont'd.)



- Location information in different “coordinates”



Nr	Hostname	IP number
10	sl-bb21-stk-1-2.sprintlink.net	144.232.4.76
11	sl-ucberkeley-1-1-0-T3.sprintlink.net	144.228.146.50
12	f5-0.inr-666-eva.berkeley.edu	198.128.16.21
13	f1-0-0.inr-107-eva.Berkeley.EDU	128.32.2.1
14	f8-0.inr-100-eva.Berkeley.EDU	128.32.235.100

- geographical: *longitude, latitude, city, river etc.*
- network-related: IP addresses for known locations

  „home network“, *„foreign network“*
- neighbourhood in network: *Location Service* with limited message spread-out area

➔ *Location Service* generates *Location Certificates* with expiration time.

Jini for Nomads

- Sample Policy for nomad authorization

```
pattern wochentag (
   weekday in (mon, tue, wed, thur)
   or (weekday equals fri and time before 16:00) )

pattern wochenende (
   weekday equals sat or weekday equals sun
   or weekday equals fri and time after 16:00 )

group mitarbeiter@komsys (user1@komsys, user2@komsys)

fileserver://komsys (
   for mitarbeiter@komsys
   from komsys::intranet ) auth

fileserver://komsys (
   for user1@komsys
   on wochentag
   from partner::intranet ) authcrypt

fileserver://komsys (
   for mitarbeiter@komsys
   on wochenende ) authcrypt

ftp-archive-get://komsys permit
```

Jini for Nomads

- Complex Implementation, required …
  - Java `keytool` extension: PKI integration
  - modification of `java.rmi.MarshalledObject` :
    verification of digitally signed proxies: „verification before deserialization“
  - implementation of sockets for authenticated and encrypted communication
- JINI Problems
  - Service modeling/specification is overly precise: full Java Interface

    > “A challenge in ubicomp service description is avoiding overspecification. Consider a wireless-enabled camera brought into a home. If the camera contains a printing service description, a user could take a picture and send it from the camera to a matching printer without using a PC. But one issue is brittleness: Both camera and printer must obey the same vocabulary and syntax. Association can’t take place if the camera requests `serviceType=printing` when the printer has `service=print`.” [18]
  - difficult integration of “Legacy Services”
  - no integrated solution for communication across firewalls with Jini deployed proxies

Summary and Outlook

- Research for *Nomadic Computing* support
  - finished: development of a JINI-based prototype
  - Polymorphic services for *Virtual Service Networks* (VSNs), (including the development of a *Wireless VSN Access Hub*)
  - Secure Web Services
- Additional projects
  - Business process-based security analysis
  - Privacy of medical data: cryptography and certification
  - Evaluation of Web Services for Ubiquitous Computing