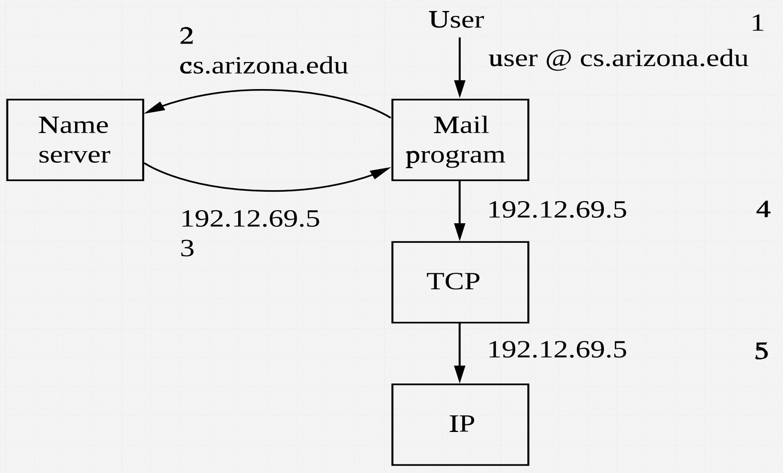
Course Overview

- Introduction and History
- Data in Wireless Cellular Systems
- Data in Wireless Local Area Networks
- Internet Protocols
- Routing and Ad-Hoc Networks
- TCP over Wireless Link
- Services and Service Discovery
- System Support for Mobile Applications

DNS: Overview

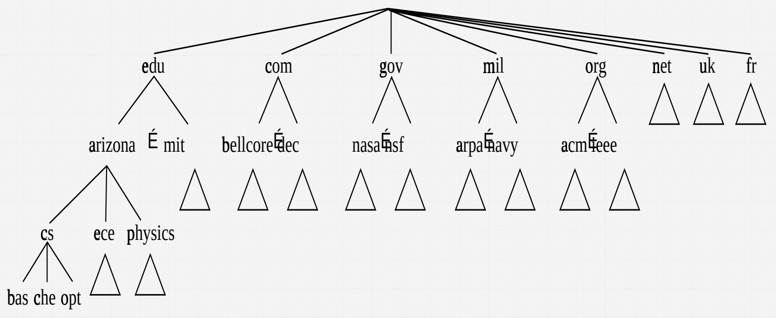
- Names versus Addresses
  - names are variable length, mnemonic, easy for humans to remember
  - addresses are fixed length, tied to routing, and easy for computers to process
- Name Space
  - defines set of possible names
  - flat versus hierarchical
  - consists of a set of name to value bindings

DNS: Overview



Domain Hierarchy

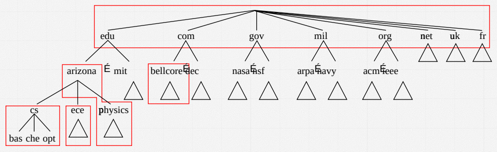
- Example hierarchy



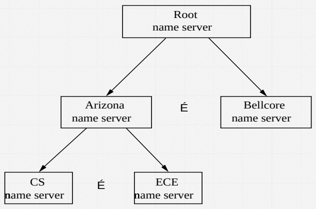
- Example name: cheltenham.cs.arizona.edu

Name Servers

- Partition hierarchy into zones



- Each zone implemented by two or more name servers



Resource Records

- Each name server maintains a collection of resource records
  - <Name, Value, Type, Class, TTL>
- Name/Value: not necessarily host names to IP addresses
- Type
  - NS: the Value field gives the domain name for a host running a name server that knows how to resolve names within the specified domain.
  - CNAME: the Value field gives the canonical name for a particular host; it is used to define aliases.
  - MX: the Value field gives the domain name for a host running a mail server that accepts messages for the specified domain.
- Class: allow other entities to define types
- TTL: how long the resource record is valid

Example

- Root server:

- <arizona.edu, telcom.arizona.edu, NS, IN>
- <telcom.arizona.edu, 128.196.128.233, A, IN>

- <bellcore.com, thumper.bellcore.com, NS, IN>
- <thumper.bellcore.com, 128.96.32.20, A, IN>

Example

- Arizona server:

- <cs.arizona.edu, optima.cs.arizona.edu, NS, IN>
- <optima.cs.arizona.edu, 192.12.69.5, A, IN>

- <ece.arizona.edu, helios.ece.arizona.edu, NS, IN>
- <helios.ece.arizona.edu, 128.196.28.166, A, IN>

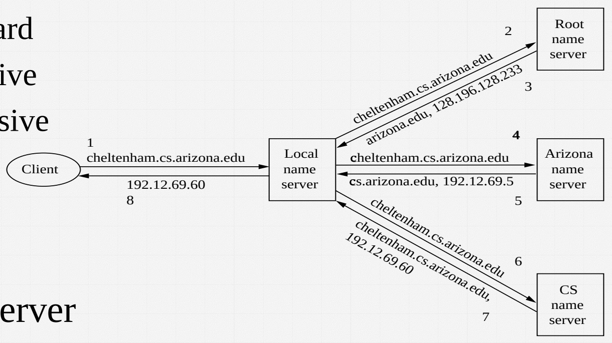
- <jupiter.physics.arizona.edu, 128.196.4.1, A, IN>
- <saturn.physics.arizona.edu, 128.196.4.2, A, IN>
- <mars.physics.arizona.edu, 128.196.4.3, A, IN>
- <venus.physics.arizona.edu, 128.196.4.4, A, IN>

Example

- CS server:
- <cs.arizona.edu, optima.cs.arizona.edu, MX, IN>
- <cheltenham.cs.arizona.edu, 192.12.69.60, A, IN>
- <che.cs.arizona.edu, cheltenham.cs.arizona.edu, CNAME, IN>
- <optima.cs.arizona.edu, 192.12.69.5, A, IN>
- <opt.cs.arizona.edu, optima.cs.arizona.edu,
- CNAME, IN>
- <baskerville.cs.arizona.edu, 192.12.69.35, A, IN>
- <bas.cs.arizona.edu, baskerville.cs.arizona.edu, CNAME, IN>

Name Resolution

- Strategies
  - forward
  - iterative
  - recursive


Client
1
cheltenham.cs.arizona.edu
192.12.69.60
8
Local name server
2
cheltenham.cs.arizona.edu
arizona.edu, 128.196.128.233
3
Root name server
4
cheltenham.cs.arizona.edu
cs.arizona.edu, 192.12.69.5
5
Arizona name server
cheltenham.cs.arizona.edu
6
cheltenham.cs.arizona.edu, 192.12.69.60
7
CS name server

- Local server
  - need to know root at only one place (not each host)
  - site-wide cache

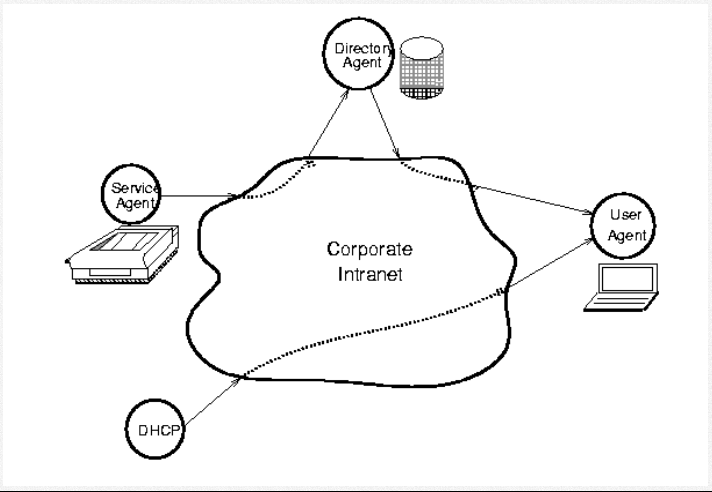
Services

- Many other standardized and non-standardized services exist. Mobile computers will need to find out about the existence of these services to provide a complete computing environment to the end-user:
  - directory services: X.500, LDAP
  - e-mail services: POP, IMAP
  - other Internet services: FTP, HTTP, NNTP, DHCP
  - non-standardized services: printers, e-commerce servers, file storage, …..
- Often, find servers in “neighborhood”, which changes over time
- Configure device to use these servers automatically

Service Location Protocol (RFC 2165)

- Clients find Servers by type and desired attributes
- Services advertise themselves
- Provide “scopes” to organize services, using arbitrary policies
- Provide low-cost administration and effortless extension to new services
- Provide decentralized and (after creation) self-administered availability
- Compatibility with browsers, existing applications, and services (using URLs)

SLP Agent Model

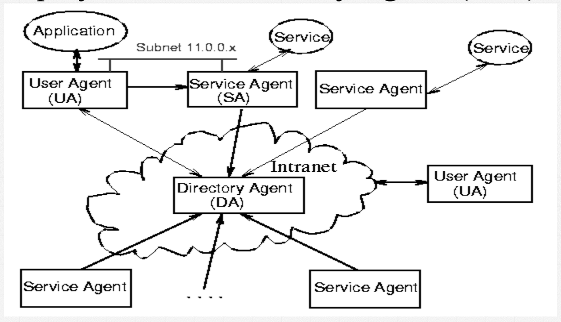


SLP Protocol Characteristics

- Distributed and self-managing, with numerous service agents
- compatible with other administrative protocols and mobile networking protocols
- string-based (reduced parser complexity)
- easily implemented
- uses existing standards where possible
- expressive query grammar (LDAP v3 query syntax)
- Scalability a prime motivation, given expected explosion of network services

Service Location Discovery Framework

- User Agents (UAs) intercede for applications
- Service Agents (SAs) intercede for services
- UAs and SAs on nearby networks can communicate
- Larger deployments use Directory Agents (DAs) transparently



Service Handles: URLs

- Standardized way to access a large variety of network resources
- General form of URLs: `<scheme>:<scheme_specific_part>`
- Often, URLs something like: `scheme://host:port/opaque`
- Some examples:
  - `nfs://slag.eng.sun.com/src/slp`
  - `service:http://www.research.sun.com/`
  - `service:lpr://motels.eng.sun.com/MPK15-214`

Service Requests

- A UA requests a service
  - by type, possibly with a naming_authority
  - from a particular scope
  - by predicate (a boolean query based on service attributes)
- Example:
  - service type = service:printer:lpr
  - scope list = Engineering, Marketing
  - predicate = (&(location-description = Printer Room)
    (duplex-mode = duplex))
- Representation: <service-type[.na],scope,[query]>

Service Request Examples

- <nfs, default, (content=slp-src)> returns only NFS servers whose content attribute have value slp-src
- <http.sun, research, homepage> returns all “homepages” within scope “research”
- <lpr, local, postscript> uses reserved value “local” for the scope and only returns printers registered with the keyword postscript

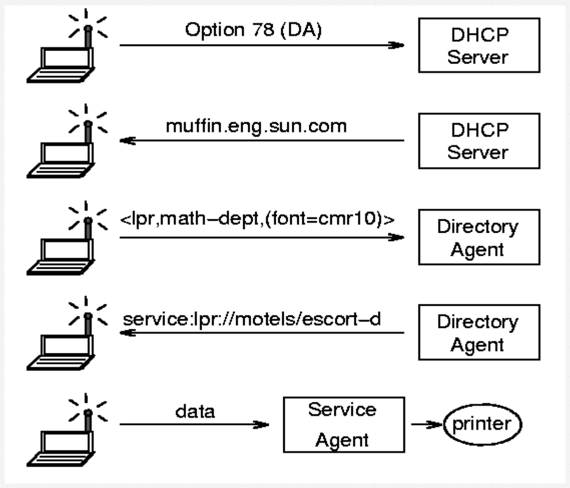
SLP: Finding Directory Agents

- There are four ways to find Directory Agents
  - listening for directory agent advertisements
  - multicast/broadcast request for small installations
  - request option 78 from DHCP
  - manual configuration
- So even in worst case (manual configuration), user does not need to configure each service manually, only SLP

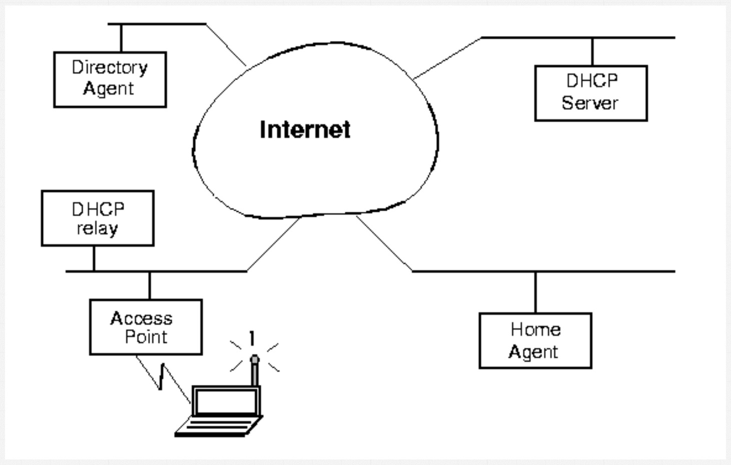
SLP: Other Messages

- Service Acknowledge
- Attribute Request
- Attribute Reply
- DA Advertisement

SLP Example: Finding a Printer



SLP Example: Mobile Computer Setup



SLP: Scalability

- Small installation: no need for DA, local broadcasts
- Large installations:
  - replicate multiple DAs
    - fault tolerance
    - load sharing
    - requires some form of "reliable broadcast"
  - scopes
    - operator-defined
    - use DHCP to configure SLP agents with non-default scopes, using SLP Service Scope Option (79)
- Very large installations: misses protocol for DA-DA interactions

SLP Status

- Standardized in RFC 2165
- Products start to appear (Novell, Apple, Cisco, …)
- Intel interested in SLP v2 as basis for their "Wired for Management" OEM initiative
- Specified as part of the MCNRS (Mobile Network Computer Reference Specification)
- Salutation Consortium has adopted SLP for service device discovery

JINI Connection Technology

- Three basic trends:
  - end user is system administrator
  - computers disappear
  - the one computer is everywhere
- Vision: When you walk up to an interaction device that is part of a system employing Jini technology, all of its services are as available to you as if they were on your own computer--and services include not only software but hardware devices as well, including disk drives, DVD players, VCRs, printers, scanners, digital cameras, and almost anything else you could imagine that passes information in and out. Adding a new device to a system employing Jini technology is simply plugging it in.

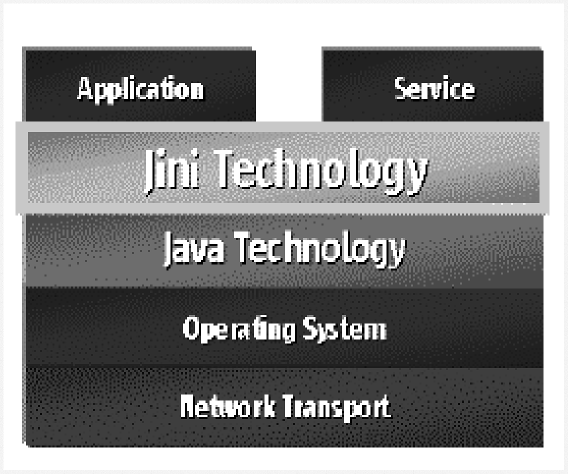
JINI Requirements

- Jini connection technology requires a few things:
  - an infrastructure which operates as a dynamically distributed system
  - a common language and implementation that enables low-overhead communication between distributed objects
  - a lookup service (which identifies objects that supply those services)
  - an add-in protocol which is implemented on each device -- the discovery/join protocol
  - a subtract-out mechanism -- providing resilience when a device is unplugged -- which is called leasing

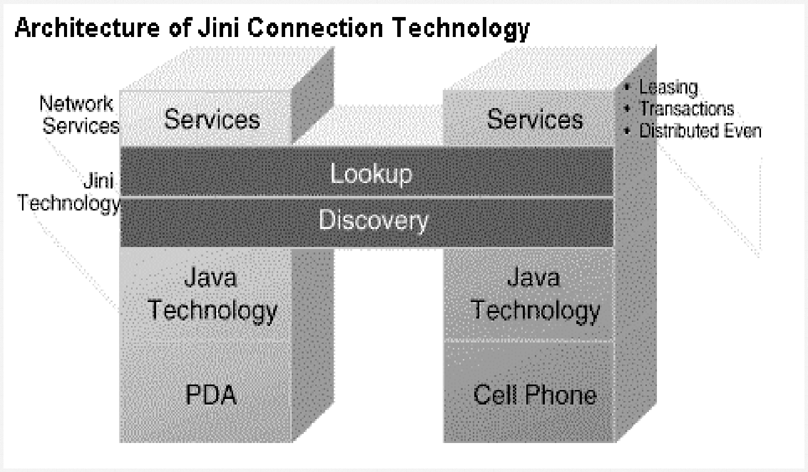
JINI's View of the World

"In a system employing Jini technology, everything is a service, everything is connected without a lot of setup trouble, and controlling them is uniformly accomplished because every service is really an object with a particular interface. Such systems will combine computer and consumer devices along with a variety of external sources -- the Net, broadcast, cable, satellite, and landline."

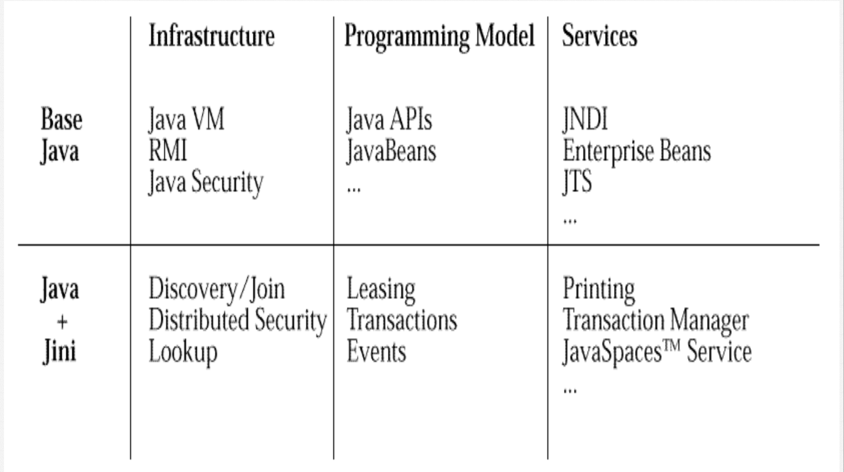
JINI Overview



JINI Overview

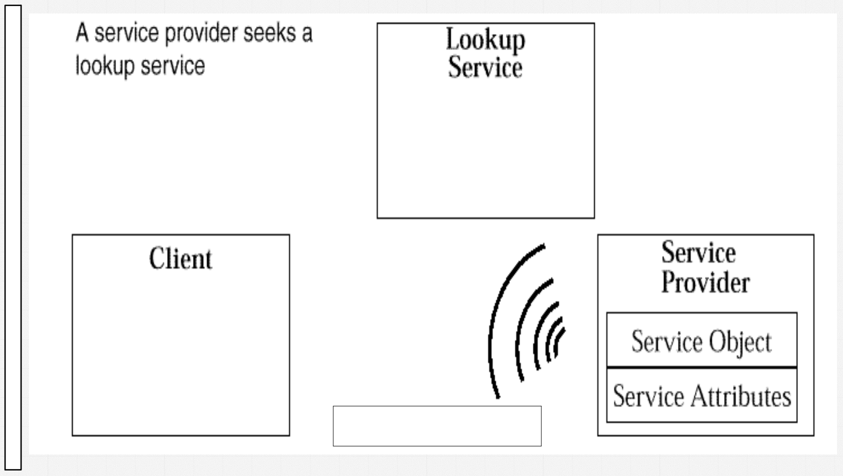


JINI Architectural Elements

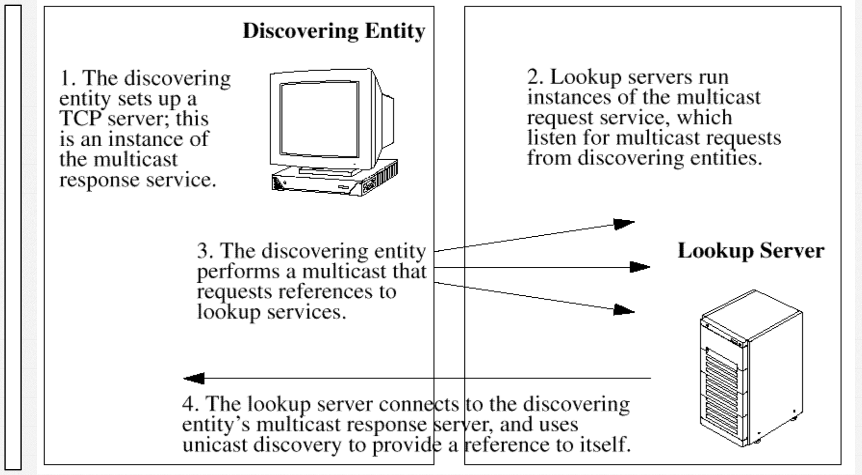


	Infrastructure	Programming Model	Services
Base Java	Java VM RMI Java Security	Java APIs JavaBeans ...	JNDI Enterprise Beans JTS ...
Java + Jini	Discovery/Join Distributed Security Lookup	Leasing Transactions Events	Printing Transaction Manager JavaSpaces™ Service ...

JINI Discovery

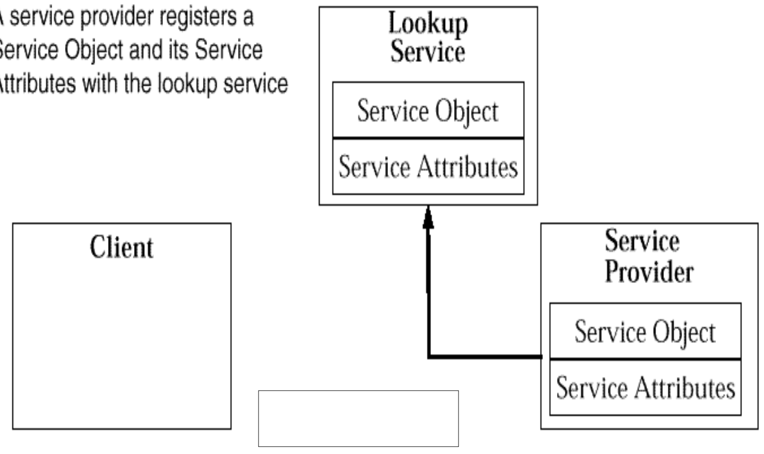


JINI Discovery Process


Discovering Entity
1. The discovering entity sets up a TCP server; this is an instance of the multicast response service.
2. Lookup servers run instances of the multicast request service, which listen for multicast requests from discovering entities.
3. The discovering entity performs a multicast that requests references to lookup services.
Lookup Server
4. The lookup server connects to the discovering entity's multicast response server, and uses unicast discovery to provide a reference to itself.

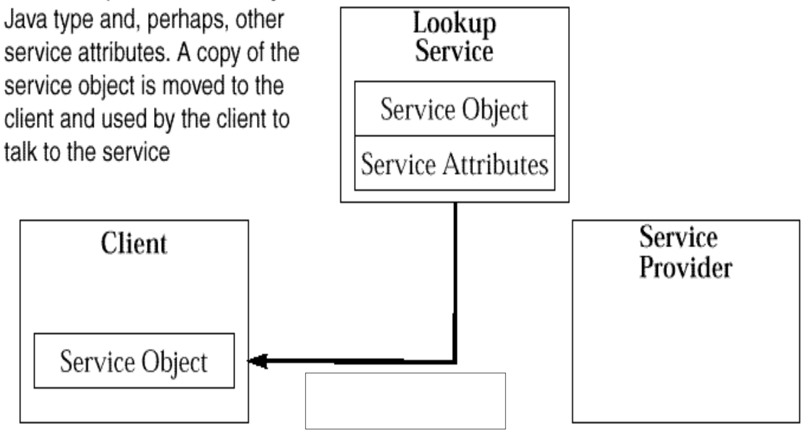
JINI Join

A service provider registers a Service Object and its Service Attributes with the lookup service

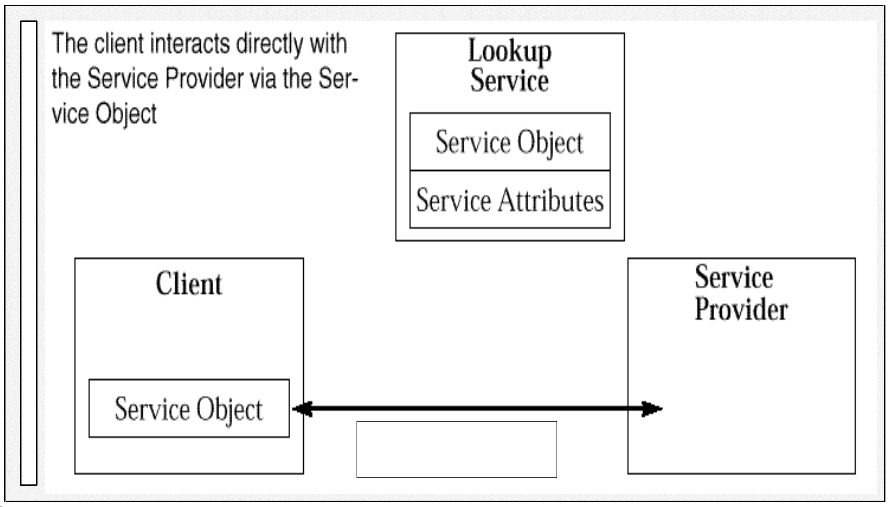


JINI Lookup

A client requests a service by Java type and, perhaps, other service attributes. A copy of the service object is moved to the client and used by the client to talk to the service



JINI Service Invocation


The client interacts directly with the Service Provider via the Service Object