Discovery

Kimmo Raatikainen

kimmo.raatikainen@cs.helsinki.fi

Lession Outline

- Bluetooth SDP
- Jini
- Salutation
- UPnP
- SLP
- Summary

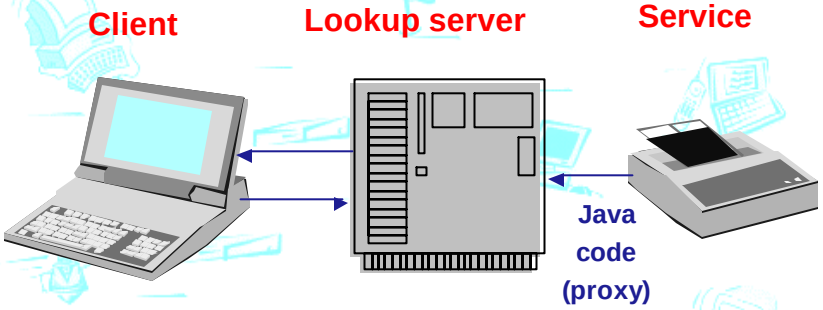
Bluetooth SDP

- The Bluetooth Service Discovery Protocol (SDP) provides a simple API for enumerating the devices in range and browsing available services.
- Stop rules that limit the duration of searches or the number of devices returned.
- Client applications use the API to search for available services either by service classes, which uniquely identify types of devices (such as printers or storage devices), or by matching attributes (such as a model number or sup-ported protocol).
- Attributes that describe the services offered by a Bluetooth device are stored as a service record and are maintained by the device's SDP server.

Jini

- Jini is a service discovery and advertisement system that relies on mobile code and leverages the platform independence of the Java language.
- Jini also requires each device either to run a Java virtual machine or to associate itself with a device that can execute a JVM on its behalf.
- Jini entities consist of services, lookup servers that catalog available services, and clients that require services.

Jini



Jini Registration

- To register service availability or to discover services, a service or client must first locate one or more lookup servers by using a multicas t request protocol.
- This request protocol terminates with the invocation of a unicast discovery protocol, which clients and services use to communicate with a specific lookup server.
- The unicast protocol culminates in the transfer of an instance of the ServiceRegistrar class, a “remote control” for the lookup server.
- A lookup server can use the multicast announcement protocol to announce its presence on the network.
- When a lookup server invokes this protocol, clients and services that have registered interest in receiving announcements of new lookup services are notified.
- These three protocols are encapsulated in a set of Jini classes.

Jini Registration

- Jini uses Java's remote method invocation (RMI) facility for all interactions between either a client or a service and the lookup server (after the initial discovery of the lookup server).
- Jini associates a proxy, or remote control object, with each service instance.
- A service advertises its availability by registering its object in one or more lookup servers via the register() method.
- The leaseduration parameter of register() specifies the service's intended lifetime.
- The service is responsible for renewing the lease within the tim e specified to maintain its listing.
- The lookup server is free to adjust the lease time, which is returned in a ServiceRegistration object.

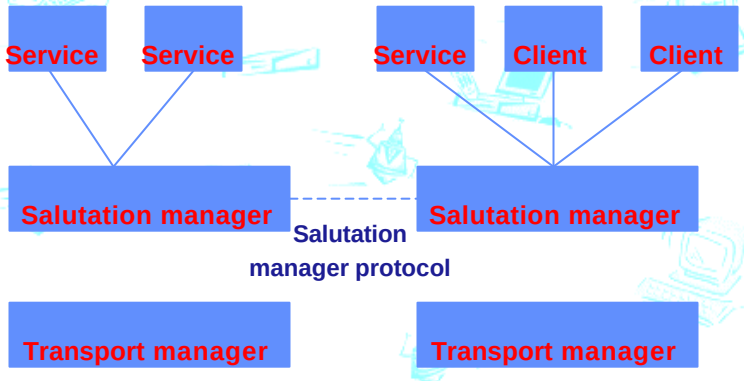
Jini Registration

- When a service first contacts a lookup server, the server generates a unique identifier for it; the service uses this ID in all future registrations. The service identifier lets clients request a specific service explicitly and recognize when services reported by different lookup servers are identical.
- Clients use the lookup() method in ServiceRegistrar to discover services. This method takes a single argument, an instance of ServiceTemplate.

Salutation

- Salutation is an architecture for service discovery under development by the Salutation Consortium.
- The consortium's goal is to build a royalty-free architecture for service advertisement and discovery that is independent of a particular network transport.
- The three fundamental components in the Salutation architecture:
  - services
  - Salutation managers are service brokers, isolated
  - by transport managers from the details of specific network trans port protocols.

Salutation Architecture



Salutation Manager

- Salutation managers function as service brokers
- Services can register and unregister functional units with the local Salutation manager.
- A client can use the slmSearchCapability() call to determine if Salutation managers have registered specific functional units.

Salutation Manager

- A Salutation manager can operate in one of three "personalities":
  - In native personality, Salutation managers are used only for discovery.
  - The emulated personality is similar to the native personality in that Salutation managers set up the connection, but in this case they transfer native data packets encapsulated in Salutation manager protocol format (bridge)
  - In Salutation personality, Salutation managers establish the connection between client and service, and they also mandate the specific format of the data transferred. The Salutation architecture defines the data formats.

Transport Manager

- Transport managers also locate the Salutation managers on their respective network segments via either multicast, static configuration, or reference to a centralized directory.
- Discovery of other Salutation managers allows a particular Salutation manager to determine which functional units have been registered and to allow clients access to these remote services.
- Communication between Salutation managers is based on remote procedure call (RPC).

Salutation-Lite

- A lightweight version (Salutation-Lite) has been developed for resource-limited devices.
- It is based primarily on IrDA to leverage the large number of infrared-capable devices.
- Salutation-Lite focuses primarily on service discovery.

UPnP

- UPnP is a proposed architecture for service advertisement and discovery supported by the UPnP Forum, headed by Microsoft.
- UPnP aims to standardize the protocols used by devices to communicate, using XML.
- UPnP's device model is hierarchical.
  - In a compound device, the root device is discoverable
  - a client (called a control point) can address the individual subdevices independently
- Devices that don't speak UPnP directly are called bridged devices.
  - They can be integrated into a UPnP network
  - A bridge maps between UPnP and device-native protocols.

UPnP Functionality

- The UPnP specification describes
  - device addressing,
  - service advertisement and discovery,
  - device control,
  - eventing, and
  - presentation.
- The eventing facility allows clients to watch for significant changes in the state of a discovered service.

Protocols in UPnP

- AutoIP a simple protocol that allows devices to dynamically claim IP addresses in the absence of a DHCP server;
- Simple service discovery protocol (SSDP), the UPnP mechanism for service discovery and advertisement;
- Simple object access protocol (SOAP), a protocol for remote procedure calls based on XML and HTTP that is used for device control after discovery; and
- Generic Event Notification Architecture (GENA), a UPnP subscription-based event notification service based on HTTP.

Items in Description Document

- A presentation URL allows entry to a device's root page, which provides a GUI for device control.
- A control URL is the entry point to the device's control server, which accepts device-specific commands to control the device.
- An event subscription URL can be used by clients to subscribe to the device's event service.
  - The client provides an event sink URL in the subscription request.
  - Significant state changes in the device result in a notification to the client's event sink URL.
- A service control protocol definition describes the protocol for interacting with the device.

Notes on Protocols

- The service control protocol (SCP) definition allows APIs to be converted to device-specific commands, shielding the application level from details of particular devices.
  - After retrieving the description document, a UPnP component on the control point called the rehydrator is "plumbed" with a definition of the device's SCP.
  - This component then sends device-specific commands via the device's control URL.
  - SOAP is used for this interaction.
- SSDP is similar to the IETF SLP, but it lacks a query facility that can search for services by attributes.

Service Location Protocol

- Service Location Protocol (SLP) is an IETF protocol for service discovery and advertisement.
- SLP comprises three entities:
  - service agents (SAs),
  - user agents (UAs), and
  - directory agents (DAs).

SLP Entities

- SAs advertise the location and attributes of available services.
- UAs discover the location and attributes of services needed by client software.
- UAs can discover services by issuing a directory-like query to the network.
- DAs cache information about available services.
  - Unlike Jini, SLP can operate without directory servers.
- The presence of one or more DAs can substantially improve performance
  - by reducing the number of multicast messages and the amount of network bandwidth used.

Interaction with Other IETF Protocols

- If DHCP is used to configure SLP agents with the location of DAs, then multicast is completely unnecessary.
- SLP also interoperates with LDAP, so services registered with an SLP DA can be automatically registered in an LDAP directory.
  - This eliminates the need to reconfigure clients that already discover services using LDAP.

Discovery Mechanisms

- Passive discovery
  - SAs and UAs listen for multicast announcements from DAs, which periodically repeat these advertisements.
- Active discovery
  - SAs and UAs multicast SLP requests or use DHCP to discover DAs.
  - When a DA is present, SAs and UAs use unicast communication to, respectively, register their services and find appropriate services.

Absence of DAs

- UAs multicast requests for service and receive unicast responses directly from the SAs that control matching services
- This tends to increase bandwidth consumption
  - but provides a simpler model, appropriate for small networks (such as a home LAN).

Summary

- Service discovery
  - Bluetooth, Jini, Salutation, UPnP, SLP
- Service announcement
  - Jini, Salutation, UPnP, SLP
- Service registry
  - Jini, Salutation, SLP
- Interoperability
  - Bluetooth, Jini, Salutation, SLP
- Security
  - Bluetooth, Jini, SLP

Further Readings

- Bluetooth: http://www.bluetooth.com/
  - membership required
- Jini: http://www.sun.com/software/jini/
- Salutation: http://www.salutation.org/
- UPnP: http://www.upnp.org/
- SLP: http://www.srvloc.org/
- UDDI: http://www.uddi.org/
  - Universal Description, Discovery and Integration of Business for the Web