Mobile Systeme
Grundlagen und Anwendungen standortbezogener Dienste

Location Based Services in the Context of Web 2.0

Department of Informatics - MIN Faculty - University of Hamburg
Lecture Summer Term 2007

Dr. Thilo Horstmann

CLDC

NMEA

MIDP

Google Earth

OpenGIS

SQL

KML

Bluetooth

Mash-Ups

Web 2.0

J2ME

Loxodrome

Euler Spaces

RDMS

GPS

PostGIS

GPX

Maps

JSR 179

Polar Coordinates

API

Threads

Today: J2ME (VII)

- Bluetooth Basics
- J2ME and Bluetooth
- Creating serial conncetions over Bluetooth

Bluetooth

- originally envisioned in 1994 by Swedish phone maker Ericsson as a way for mobile devices to communicate with each other at short ranges -- up to 30 feet, or 10 meters.
- 1998, Ericsson, IBM, Intel, Nokia, and Toshiba formed the Bluetooth Special Interest Group consortium to develop a royalty-free, open specification for short-range wireless connectivity.
- Since then, more than 2000 companies have joined the Bluetooth SIG, including virtually all manufacturers of phone, computer, and PDA equipment.

Bluetooth Features

- Bluetooth is wireless and automatic. You don't have to keep track of cables, connectors, and connections, and you don't need to do anything special to initiate communications. Devices find each other automatically and start conversing without user input, expect where authentication is required

- Bluetooth is inexpensive (about $2 per chip).

- The Industrial Scientific and Medial (ISM) band that Bluetooth uses is regulated, but unlicensed. Governments have converged on a single standard, so it's possible to use the same devices virtually wherever you travel, and you don't need to obtain legal permission in advance to begin using the technology.

- Bluetooth handles both data and voice. Its ability to handle both kinds of transmissions simultaneously makes possible such innovations as a mobile hands-free headset for voice with applications that print to fax, and that synchronize the address books on your PDA, your laptop, and your cell phone.

- Signals are omni-directional and can pass through walls. Communicating devices don't need to be aligned and don't need an unobstructed line of sight.

- Bluetooth uses frequency hopping. Its spread spectrum approach greatly reduces the risk that communications will be intercepted.

Bluetooth vs. Infrared vs. IEEE 802.11b/g/n

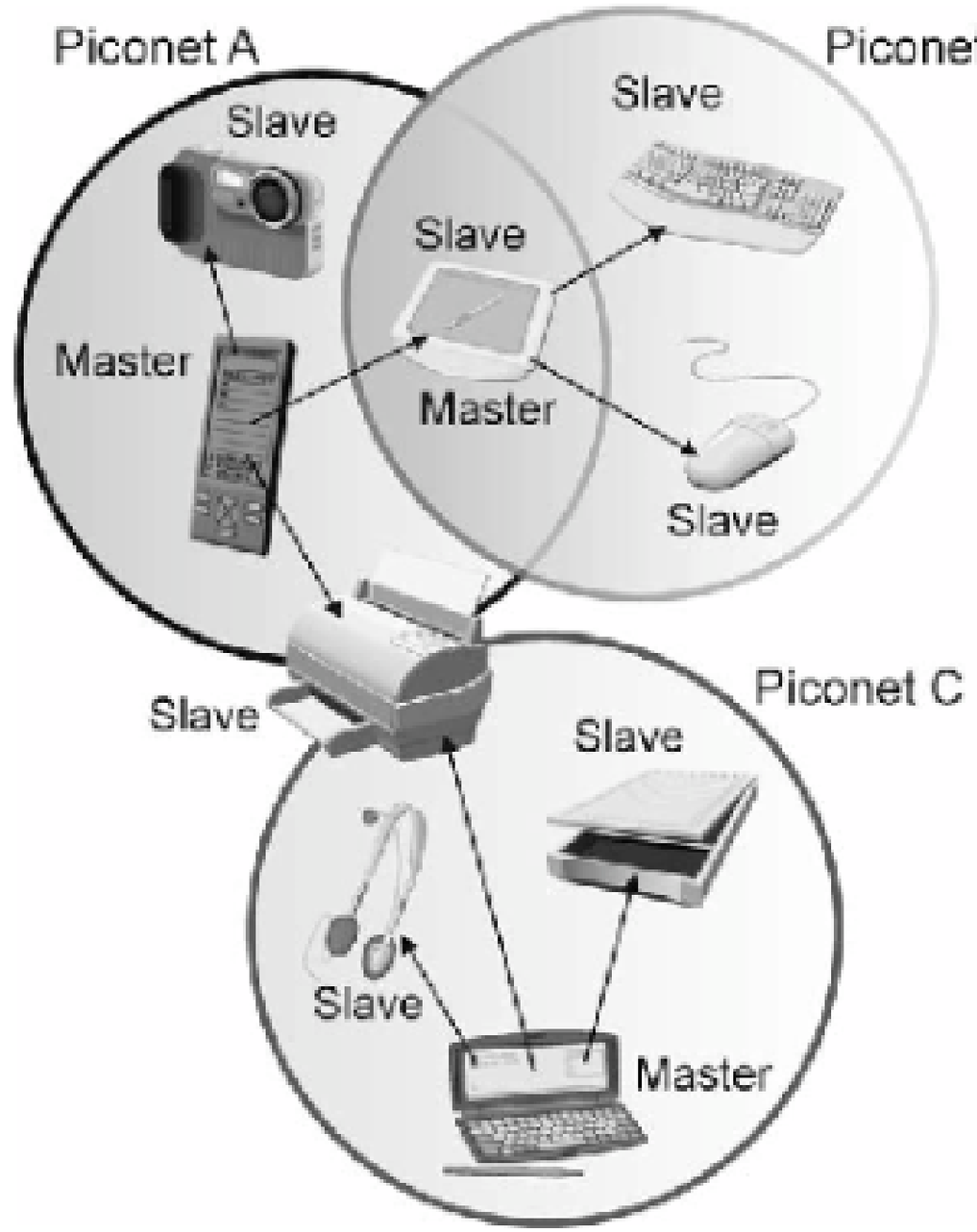
- Infrared has drawbacks:
    - It's line-of-sight, so a sender must align with its receiver.
    - It's one-to-one, so a device can't send to multiple receivers at the same time.
- and advantages:
    - Because it's line-of-sight, interference is uncommon.
    - Because it's one-to-one, message delivery is reliable: each message sent goes to the intended recipient no matter how many infrared receivers are in the room.
- IEEE 802.11b/g/n „WLAN“
    - designed to connect relatively large devices with lots of power and speed, such as desktops and laptops
    - Bluetooth offers ad hoc, resource-based opportunistic availability

Bluetooth example applications

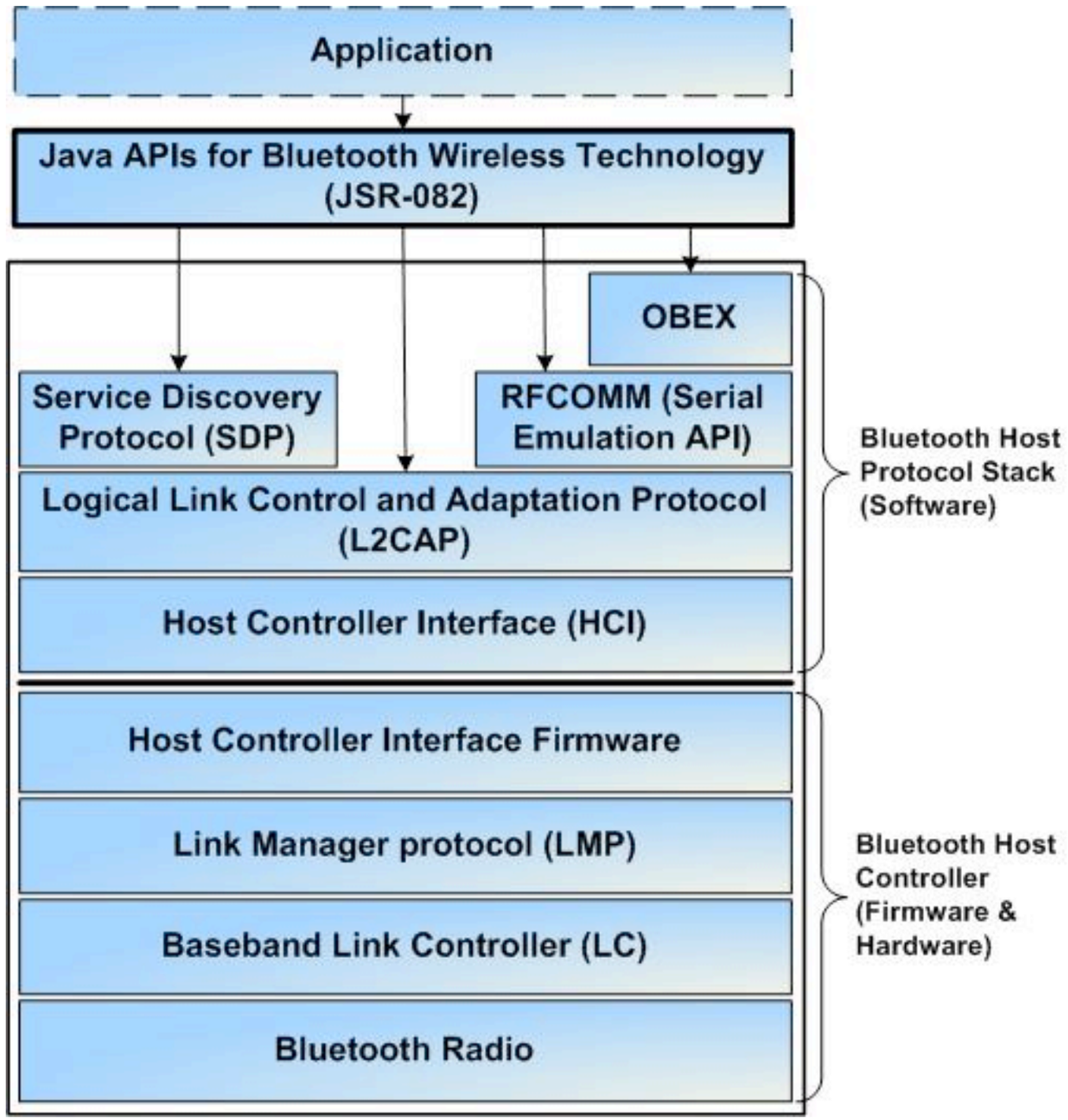
- Peripheral connectivity
- File transfer
- Ad-hoc networking: Communicating devices can spontaneously form a community of networks that persists only as long as it's needed
- Device synchronization: Seamless connectivity among PDAs, computers, and mobile phones allows applications to update information on multiple devices automatically when data on any one device changes
- Car kits:Hands-free packages enable users to access phones and other devices without taking their hands off the steering wheel
- Mobile payments: Your Bluetooth-enabled phone can communicate with a Bluetooth-enabled vending machine to buy chewing gum, and put the charge on your phone bill

Bluetooth Scatternet / Piconet

- 2 connected BT devices create a Personal Area Network
- BT devices organize themselves in Piconets: 1 master and up to 7 slaves
- Since a master can connect to a slave that is member of a different piconet, several piconets can form a scatternet



The Bluetooth Protocol Stack



Establishing a BT network connection

- Inquire: In a new environment, the device automatically initiates an inquiry to find an access point. All nearby access points respond with their addresses, and the device picks one.

- Page: The paging procedure synchronizes the device with the access point.

- Establish a link: The Link Manager Protocol establishes a link with the access point.

- Discover services: The LMP uses the Service Discovery Protocol (SDP) to find out what services are available from the access point. Here we assume that the email service is available.

- Create an L2CAP (Logical Link Control and Adaption) Channel: The LMP uses information obtained from the Service Discovery Protocol (SDP) to create an L2CAP channel to the access point. The application may use this channel directly or use a protocol like RFCOMM (Radio Frequency Communications Protocol) that might be running over L2CAP. RFCOMM emulates a serial line (Serial Cable Emulation Protocol based on ETSI TS07.10.).

- Create an RFCOMM channel: Depending on the needs of the application, an RFCOMM channel (or another channel) is created over the L2CAP channel. Creating an RFCOMM channel allows an existing application that works with serial ports to work with Bluetooth as well, without any modifications.

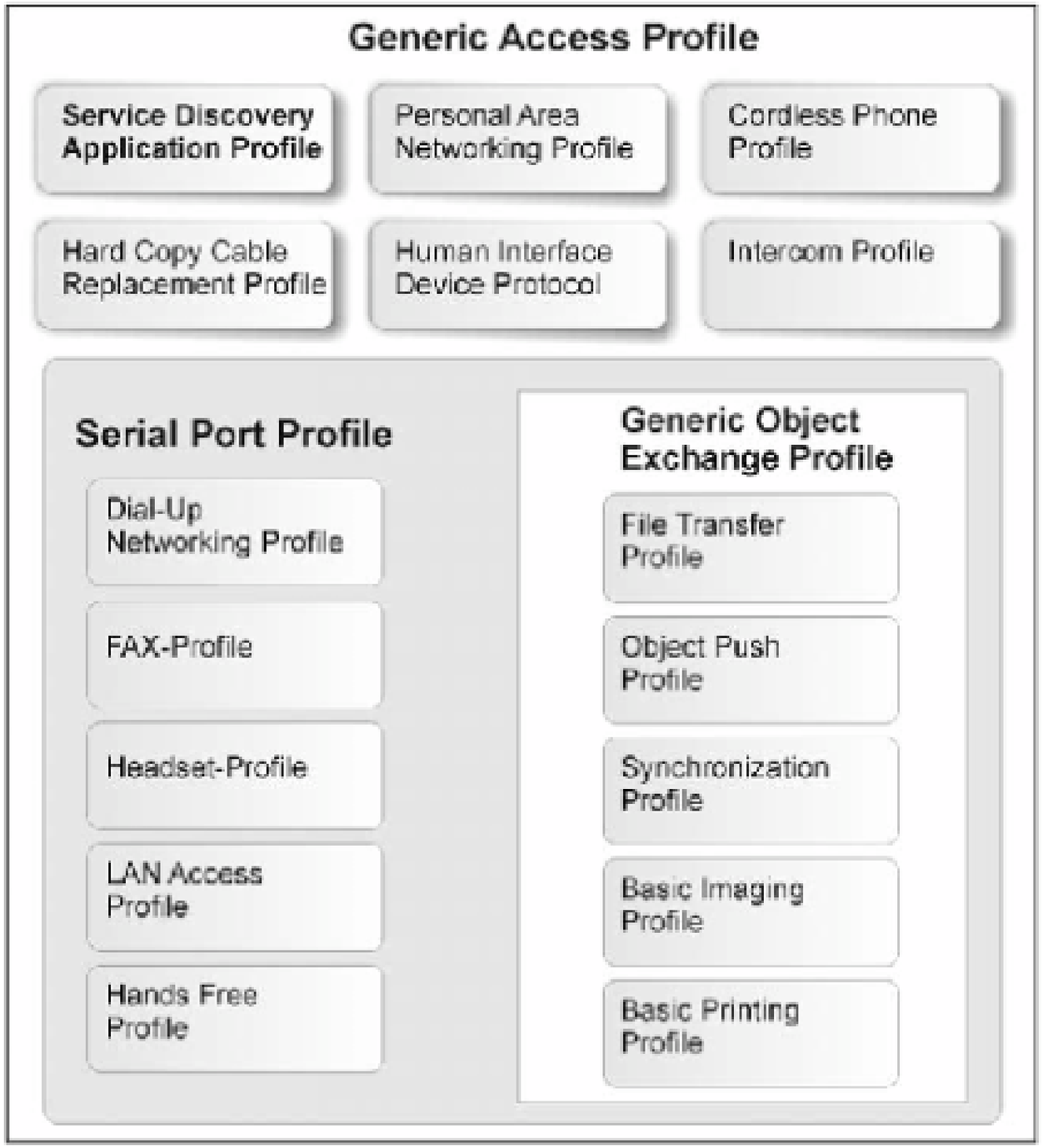
- Authenticate: This is the only step that requires input from the user. If the access point requires authentication, it will send an authentication request, and the user will be prompted to enter a PIN to access the service. For security reasons, the PIN code itself is not sent over the wireless link, but rather a key generated from it.

- Log in: If the devices use the Point-to-Point Protocol (PPP) over RFCOMM, a serial port is emulated

- Send and receive data: The email client and the access point use standard network protocols like TCP/IP to send and receive data.

Bluetooth Profiles

- A profile defines the roles and capabilities for specific types of applications
- intended to ensure interoperability among Bluetooth-enabled devices and applications from different manufacturers and vendors



BT Profiles (Examples)

- The Generic Access Profile defines connection procedures, device discovery, and link management. It also defines procedures related to use of different security models and common format requirements for parameters accessible on the user interface level. At a minimum all Bluetooth devices must support this profile.

- The Service Discovery Application Profile defines the features and procedures for an application in a Bluetooth device to discover services registered in other Bluetooth devices, and retrieves information related to the services.

- The Serial Port Profile defines the requirements for Bluetooth devices that need to set up connections that emulate serial cables and use the RFCOMM protocol.

- The LAN Access Profile defines how Bluetooth devices can access the services of a LAN using PPP, and shows how PPP mechanisms can be used to form a network consisting of Bluetooth devices.

- The Synchronization Profile defines the application requirements for Bluetooth devices that need to synchronize data on two or more devices.

Bluetooth Security

- provided in three ways: pseudo-random frequency hopping, authentication, and encryption
- three security modes:
    - Mode 1 is an insecure mode of operation. No security procedures are initiated
    - Mode 2 is known as service-level enforced security. When devices operate in this mode, no security procedures are initiated before the channel is established. This mode enables applications to have different access policies and run them in parallel
    - Mode 3 is known as link-level enforced security. In this mode, security procedures are initiated before link setup is complete

J2ME and Bluetooth

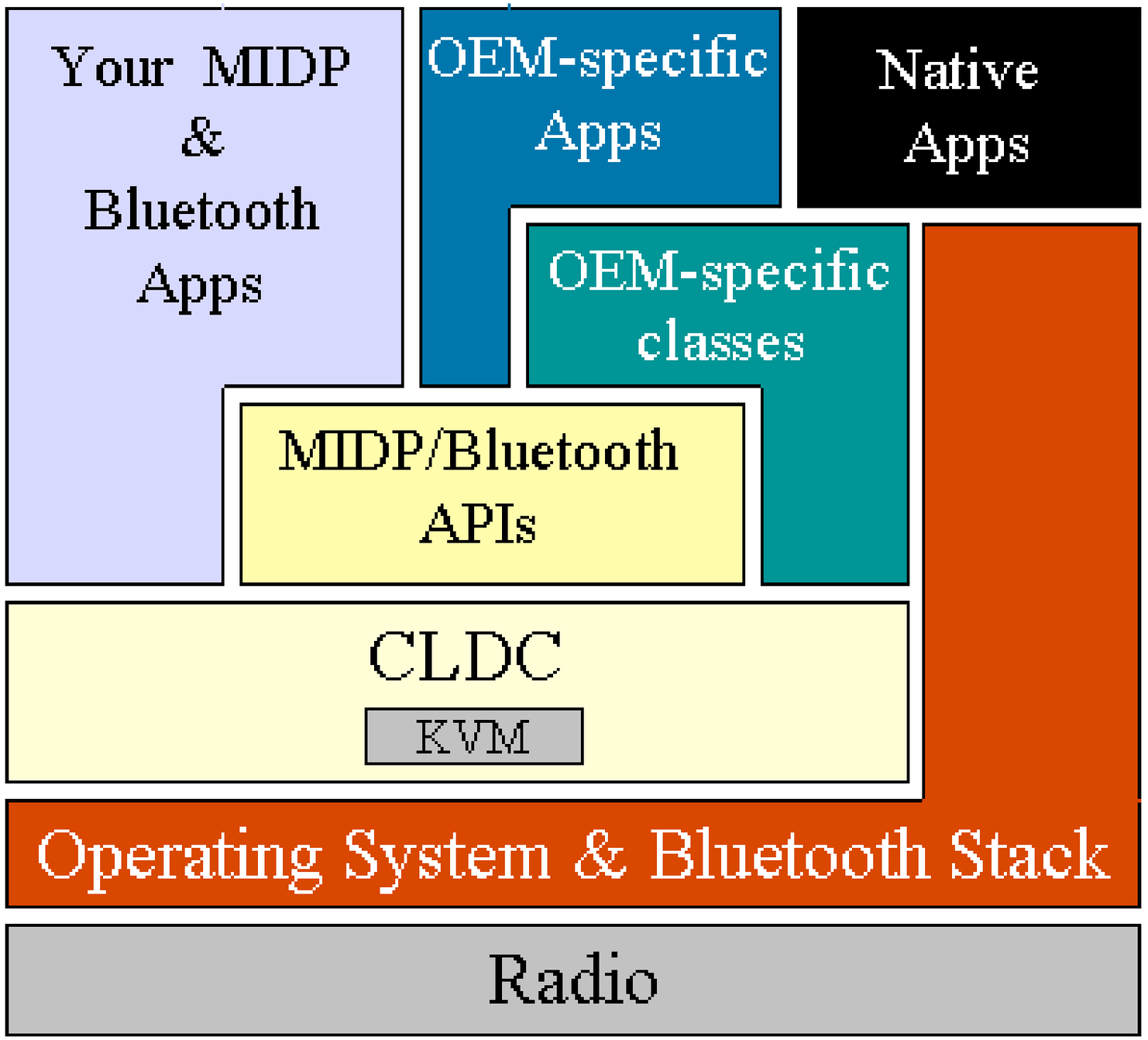
JSR 82: Java APIs for Bluetooth

- non-proprietary standard for developing Bluetooth applications using Java based on version 1.1 of the Bluetooth Specification
- consists of two optional independent packages:
  - the core Bluetooth API (javax.bluetooth)
  - the Object Exchange (OBEX) API (javax.obex)
- system in accordance with the Bluetooth Qualification Program, for at least the Generic Access Profile, Service Discovery Application Profile, and Serial Port Profile
- system must support three communication layers or protocols as defined in the 1.1 Bluetooth Specification, and the implementation of this API must have access to them: Service Discovery Protocol (SDP), Radio Frequency Communications Protocol (RFCOMM), and Logical Link Control and Adaptation Protocol (L2CAP).
- mandatory Bluetooth Control Center (BCC)

Capabilities of JSR 82

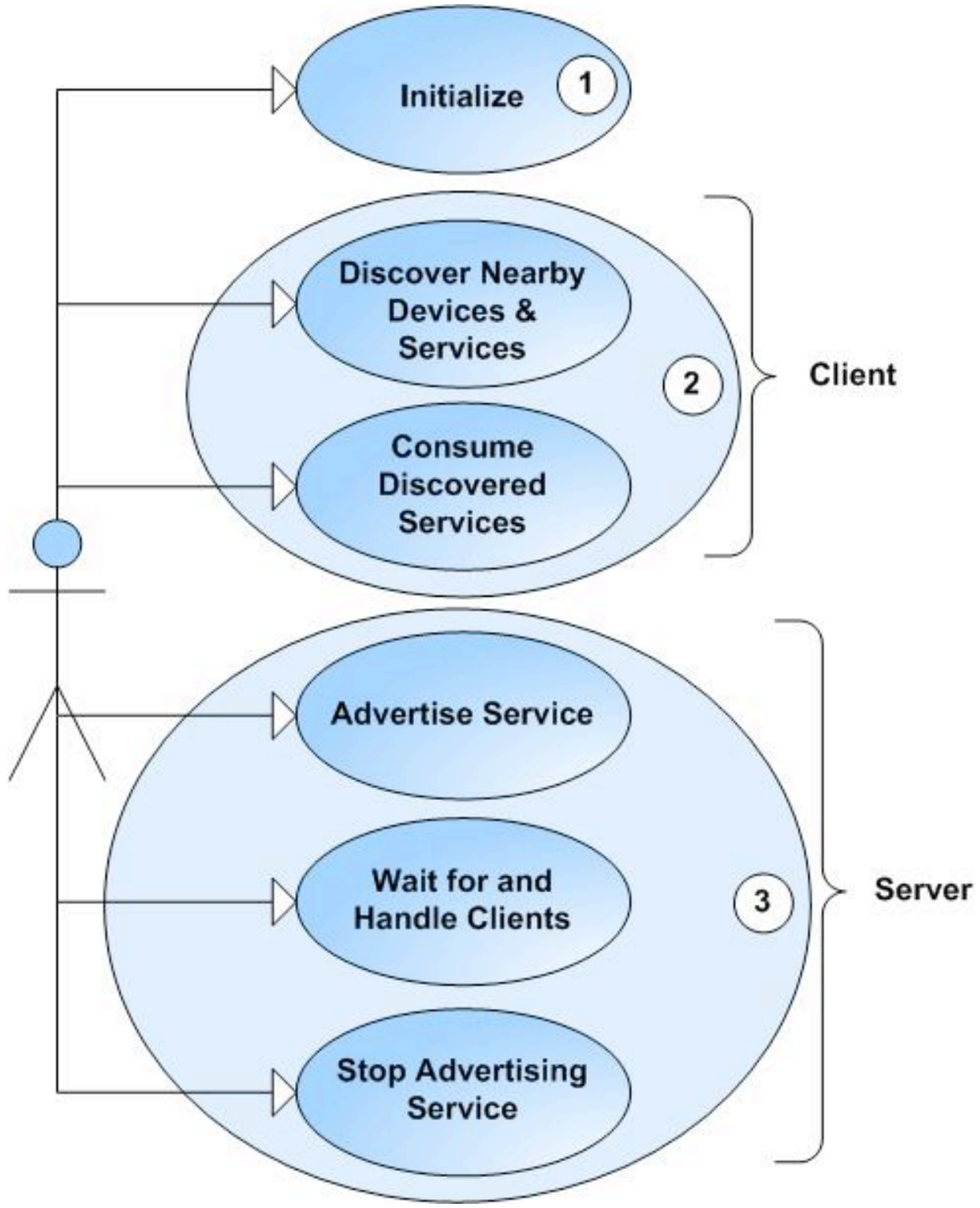
- Register services
- Discover devices and services
- Establish RFCOMM, L2CAP, and OBEX connections between devices
- Using those connections, send and receive data (voice communication not supported)
- Manage and control the communication connections
- Provide security for these activities

API Architecture

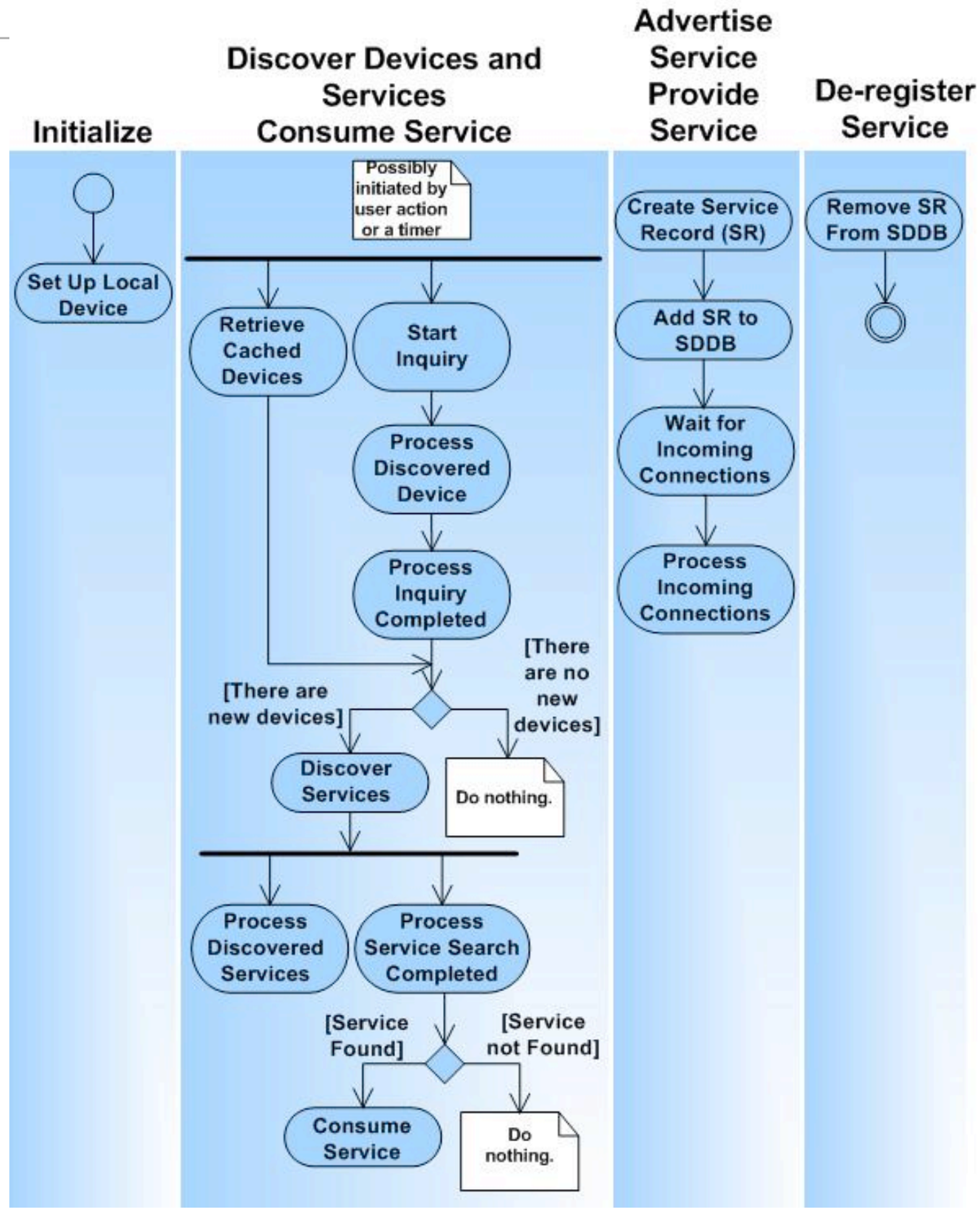


Anatomy of a J2ME BT application

- stack initialization
- device management
- device discovery
- service discovery
- communication



Activity diagram of device and service discovery



Step 1: Stack Initialization

- The initialization process comprises a number of steps whose purpose is to get the device ready for wireless communication.
- the Bluetooth specification leaves implementation of the BCC to vendors, and different vendors handle stack initialization differently :-(
- Example (not part of JSR82!):

```
...
// set the port number
BCC.setPortNumber("COM1");
// set the baud rate
BCC.setBaudRate(50000);
// set the connectable mode
BCC.setConnectable(true);
// set the discovery mode to Limited Inquiry Access Code
BCC.setDiscoverable(DiscoveryAgent.LIAC);
...
```

Step 2: Device Management

- Java Bluetooth APIs contain the classes LocalDevice and RemoteDevice providing the device-management capabilities defined in the Generic Access Profile

- Example:

```
...
// retrieve the local Bluetooth device object
LocalDevice local = LocalDevice.getLocalDevice();
// retrieve the Bluetooth address of the local device
String address = local.getBluetoothAddress();
// retrieve the name of the local Bluetooth device
String name = local.getFriendlyName();
...
```

Step 3: Device Discovery

- The core Bluetooth API's DiscoveryAgent class and DiscoveryListener interface provide the necessary discovery services to find other devices and gain access to their capabilities.
- The DiscoveryAgent.startInquiry method places the device into an inquiry mode. To take advantage of this mode, the application must specify an event listener that will respond to inquiry-related events.
- DiscoveryListener.deviceDiscovered is called each time an inquiry finds a device. When the inquiry is completed or canceled, DiscoveryListener.inquiryCompleted is invoked.
- If the device doesn't wish to wait for devices to be discovered, it can use the DiscoveryAgent.retrieveDevices method to retrieve an existing list.

Three ways to obtain a list of accessible devices

```
...
// retrieve the discovery agent
DiscoveryAgent agent = local.getDiscoveryAgent();
// place the device in inquiry mode
boolean complete = agent.startInquiry();
...
//*****************************************************//
...
// retrieve the discovery agent
DiscoveryAgent agent = local.getDiscoveryAgent();
// return an array of pre-known devices
RemoteDevice[] devices =
   agent.retrieveDevices(DiscoveryAgent.PREKNOWN);
...
//*****************************************************//
...
// retrieve the discovery agent
DiscoveryAgent agent = local.getDiscoveryAgent();
// return an array of devices found in a previous inquiry
RemoteDevice[] devices =
    agent.retrieveDevices(DiscoveryAgent.CACHED);
...
```

Class DiscoveryAgent & DiscoveryListener Interface

Step 4a: Service Discovery

- Once the local device has discovered at least one remote device, it can begin to search for available services (Bluetooth applications) it can use to accomplish useful tasks

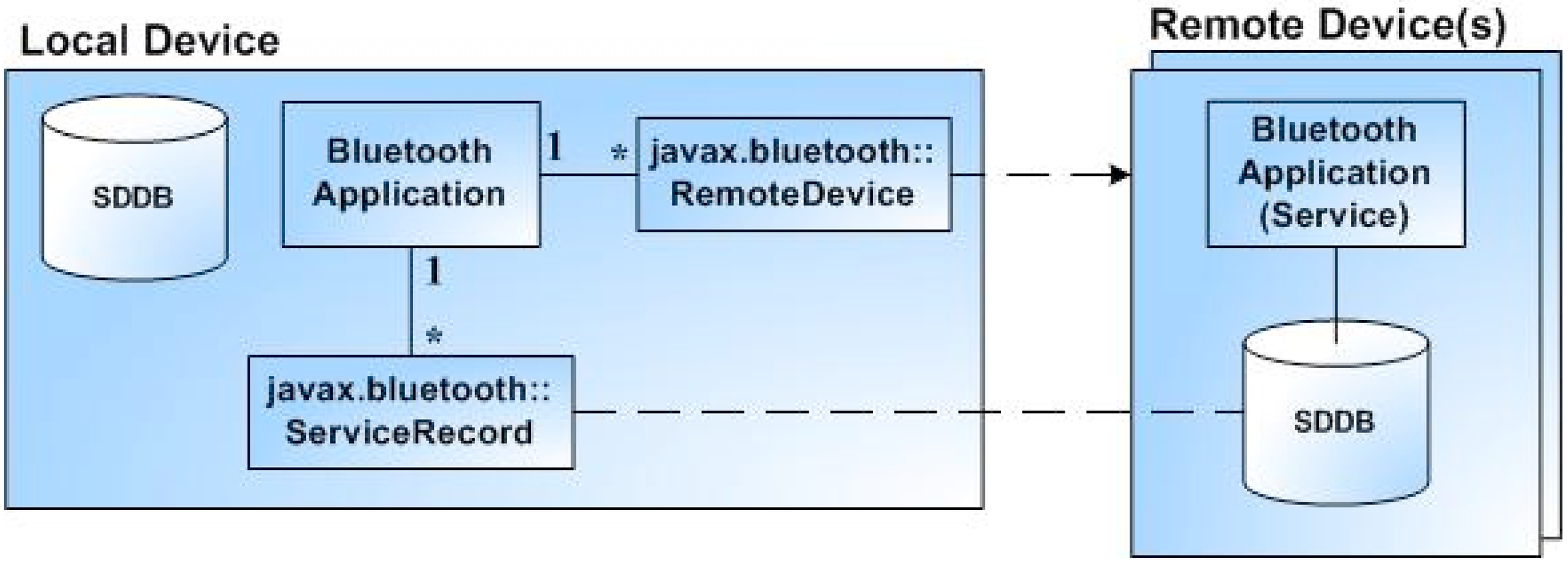
- service discovery is much like device discovery, i.e.

    - DiscoveryAgent also provides methods to discover services on a Bluetooth server device

    - and to initiate service-discovery transactions

- the API provides mechanisms to search for services on remote devices, but not for services on the local device

ServiceDiscovery Database (SDDB)



Step 4b: Service Registration (1)

- Before a service can be discovered, it must first be registered, i.e. advertised on a Bluetooth server device. The server is responsible for
- Creating a service record that describes the service offered
- Adding the service record to the server's Service Discovery DataBase (SDDB), so it's visible and available to potential clients
- Registering the Bluetooth security measures associated with the service (enforced for connections with clients)
- Accepting connections from clients
- Updating the service record in the SDDB whenever the service's attributes change
- Removing or disabling the service record in the SDDB when the service is no longer available

Attributes of a ServiceRecord (Example)

Attribute Name	Attribute ID	Attribute Value Type
ServiceInfoTimeToLive	0x0007	32-bit unsigned integer
ServiceAvailability	0x0008	8-bit unsigned integer
BluetoothProfileDescriptorList	0x0009	DATSEQ of DATSEQ pairs
DocumentationURL	0x000A	URL
ClientExecutableURL	0x000B	URL
IconURL	0x000C	URL
VersionNumberList	0x0200	DATSEQ of 16-bit unsigned integers

Step 4b: Service Registration (2)

1. To create a new service record that represents the service, invoke `Connector.open` with a server connection URL argument, and cast the result to a `StreamConnectionNotifier` that represents the service:

```
...
StreamConnectionNotifier service =
    (StreamConnectionNotifier) Connector.open("someURL");
```

2. Obtain the service record created by the server device:

```
ServiceRecord sr = local.getRecord(service);
```

3. Indicate that the service is ready to accept a client connection:

```
StreamConnection connection =
    (StreamConnection) service.acceptAndOpen();
```

   Note that `acceptAndOpen` blocks until a client connects.

4. When the server is ready to exit, close the connection and remove the service record:

```
service.close();
...
```

Step 5: Communication

- For a local device to use a service on a remote device, the two devices must share a common communications protocol.
- Java APIs for Bluetooth provide mechanisms that allow connections to any service that uses RFCOMM, L2CAP, or OBEX
- If a service uses another protocol (such as TCP/IP) layered above one of these protocols, the application can access the service, but only if it implements the additional protocol in the application, using the CLDC Generic Connection Framework.
- The OBEX protocol can be used over several different transmission media
    - JSR 82 specifies the OBEX API (javax.obex) independently of the core Bluetooth API (javax.bluetooth).

The Serial Port Profile (SPP)

- The RFCOMM protocol, which is layered over the L2CAP protocol, emulates an RS-232 serial connection.
- The Serial Port Profile (SPP) eases communication between Bluetooth devices by providing a stream-based interface to the RFCOMM protocol
- As with all GCF connection types, you create Bluetooth connections by calling the GCF connection factory javax.microedition.io.Connector. The connection URL scheme passed to Connector() determines the connection type to create.
- The URL format for an RFCOMMStreamConnection: **btspp://hostname:[CN | UUID];parameters**
  - hostname is either localhost to set up a server connection, or the Bluetooth address to create a client connection.
  - CN is the Channel Number value, used by a client connecting to a server (similar in concept to a TCP/IP port).
  - UUID is the UUID value used when setting up a service on a server.
  - parameters include name to describe the service name, and the security parameters authenticate, authorize, and encrypt.

Serial Client Server Communication via BT

The Server must

1. Construct a URL that indicates how to connect to the service, and store it in the service record
2. Make the service record available to the client
3. Accept a connection from the client
4. Send and receive data to and from the client

The Client must

1. Initiate a service discovery to retrieve the service record
2. Construct a connection URL using the service record
3. Open a connection to the server
4. Send and receive data to and from the server

SPP Server Example

```
// assuming the service UID has been retrieved
String serviceURL = "btspp://localhost:"+serviceUID.toString());
// more explicitly:
String ServiceURL = "btspp://localhost:10203040607040A1B1C1DE100;name=SPPServer1";
try {
    // create a server connection
    StreamConnectionNotifier notifier =
          (StreamConnectionNotifier) Connector.open(serviceURL);
    // accept client connections
   StreamConnection connection = notifier.acceptAndOpen();
    // prepare to send/receive data
    byte buffer[] = new byte[100];
    String msg = "hello there, client";
    InputStream is = connection.openInputStream();
    OutputStream os = connection.openOutputStream();
    // send data to the client
    os.write(msg.getBytes());
    // read data from client
    is.read(buffer);
    connection.close();
} catch(IOException e) {
  e.printStackTrace();
}
```

SPP Client Example

```
// (assuming we have the service record)
// use record to retrieve a connection URL
String url =
    record.getConnectionURL(record.NOAUTHENTICATE_NOENCRYPT, false);
// open a connection to the server
StreamConnection connection =
    (StreamConnection) Connector.open(url);
// Send/receive data
try {
    byte buffer[] = new byte[100];
    String msg = "hello there, server";
    InputStream is = connection.openInputStream();
    OutputStream os = connection.openOutputStream();
    // send data to the server
    os.write(msg.getBytes);
    // read data from the server
    is.read(buffer);
    connection.close();
} catch(IOException e) {
  e.printStackTrace();
}
```

Some Bluetooth Examples

This Lecture

- Breymann, U., Mosemann, H.: Java ME – Anwendungsentwicklung für Handys, PDA und Co., Hanser, 2006, www.java-me.de
  - Chapter 13
- J2ME and Bluetooth:
  - http://developers.sun.com/mobility/apis/articles/bluetoothintro/
  - http://developers.sun.com/techtopics/mobility/midp/articles/bluetooth1, http://developers.sun.com/techtopics/mobility/midp/articles/bluetooth2/
- JSR-82:
  - http://java.sun.com/javame/reference/apis/jsr082/

Thank you!

Dr. Thilo Horstmann

e-mail: thilo.horstmann@gmail.com
blog: http://www.das-zentralorgan.de