

Design Description

Java Bluetooth stack (JBS)



What is a Bluetooth stack?

- In short, to get any functionality out of a Bluetooth device, one needs to implement pretty high stack of protocol layers, quite similar to OSI referent model
- If you buy a Bluetooth device, you get that stack already implemented, and it exposes profiles(services) to the user, such as Serial port profile, LAN access, OBEX file transfer & other

Then why bother...

...coding your own Bluetooth stack?

Because you get the complete stack (with the hardware you bought), you have no other choice than to use the profiles it has exposed (most commonly used profile is Serial port profile) as virtual COM ports.

This results in a limited functionality, and inability to explore the most exciting Bluetooth features: creating pico- and scatter-nets of interoping devices running custom services.

Out there...

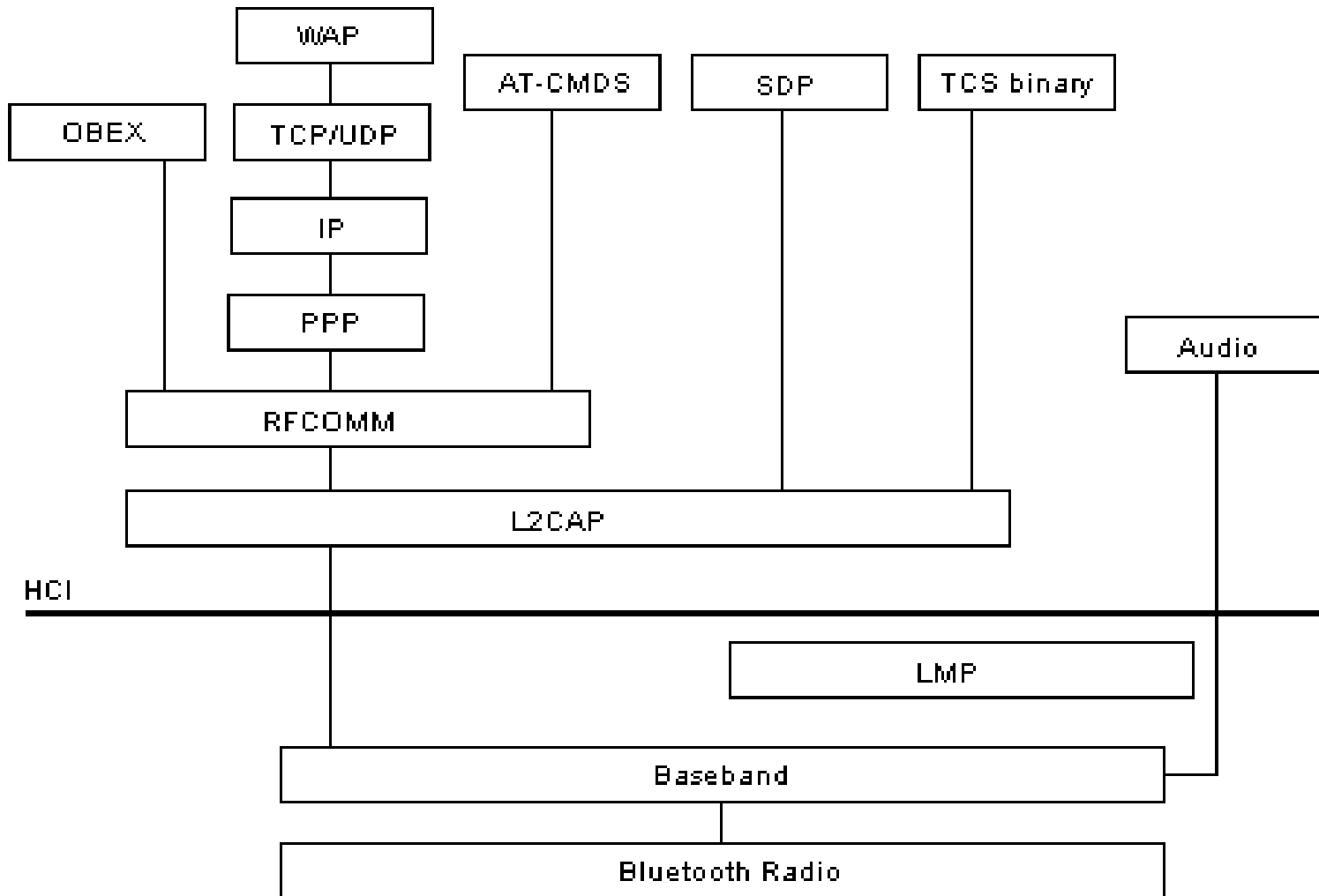
...there are few alternatives:

- Buy commercial stack (library) – you become bound to a single manufacturer, and at his mercy for updates
- Use BlueZ, a free-source Linux Bluetooth stack written in C, or it's JNI wrapped Java brother, JBlueZ
- That's about it, there is no free-source Java Bluetooth stack available*

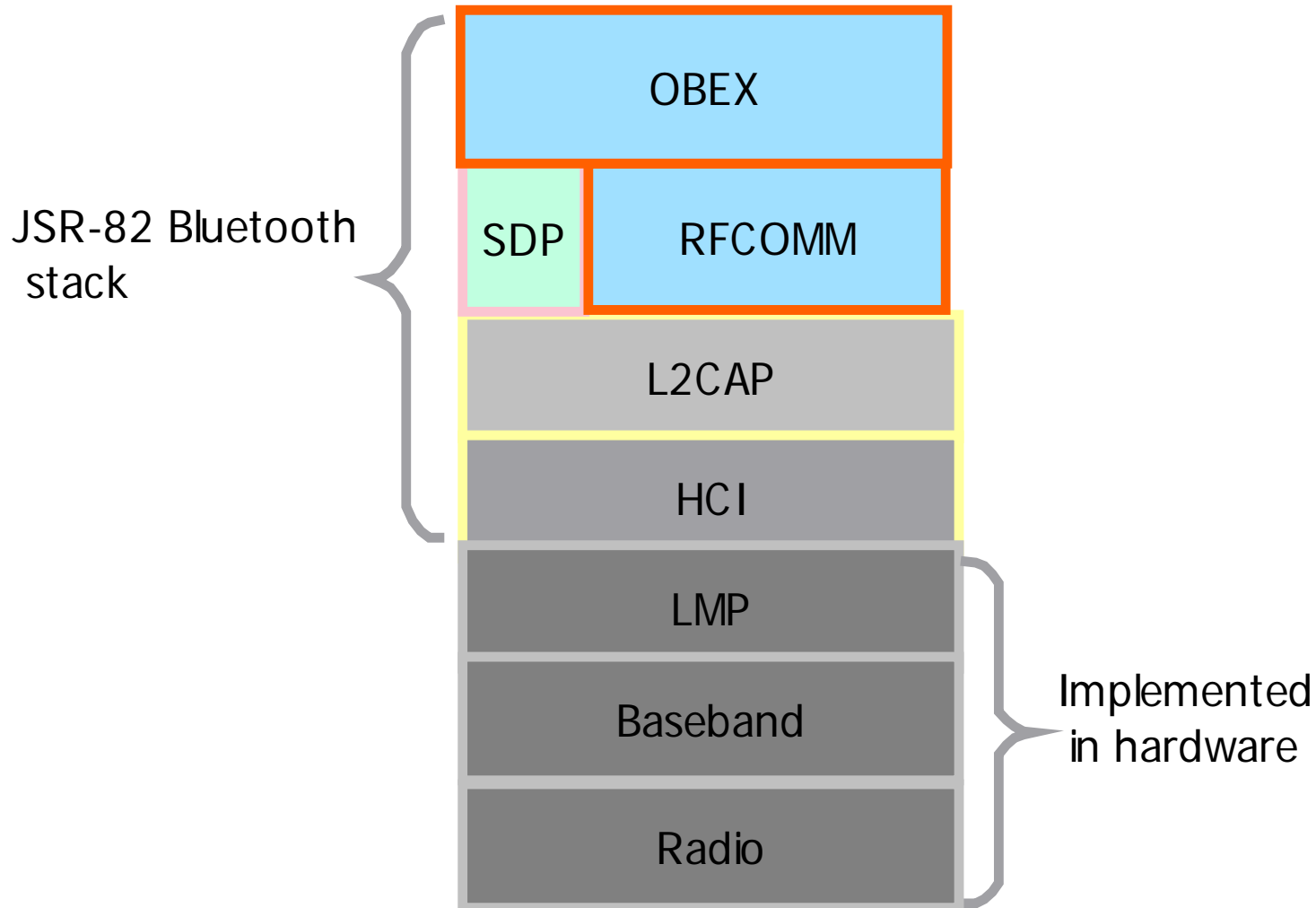
(* not quite true, read on☺)



The Big Picture



Mission JBS:Primary Objective



Goals

- Create an open-source, full JSR-82, all Java Bluetooth stack
- Build a flexible solution for accessing Bluetooth hardware and harvesting its possibilities to the fullest, leaving enough space for customization
- To provide a solution for using Bluetooth from Java with all 'Java' benefits – portability being the main one – so one could use the same stack to access a mobile phones Bluetooth abilities and a Bluetooth PC Card connected to a standard PC running Linux or Windows

What's out there?

- Harald Bluetooth stack, created by Johan Eker (<http://www.control.lth.se/~johane/harald/>)
- Javablueetooth, created by Christian Lorenz (<http://sourceforge.net/projects/javablueetooth/>)

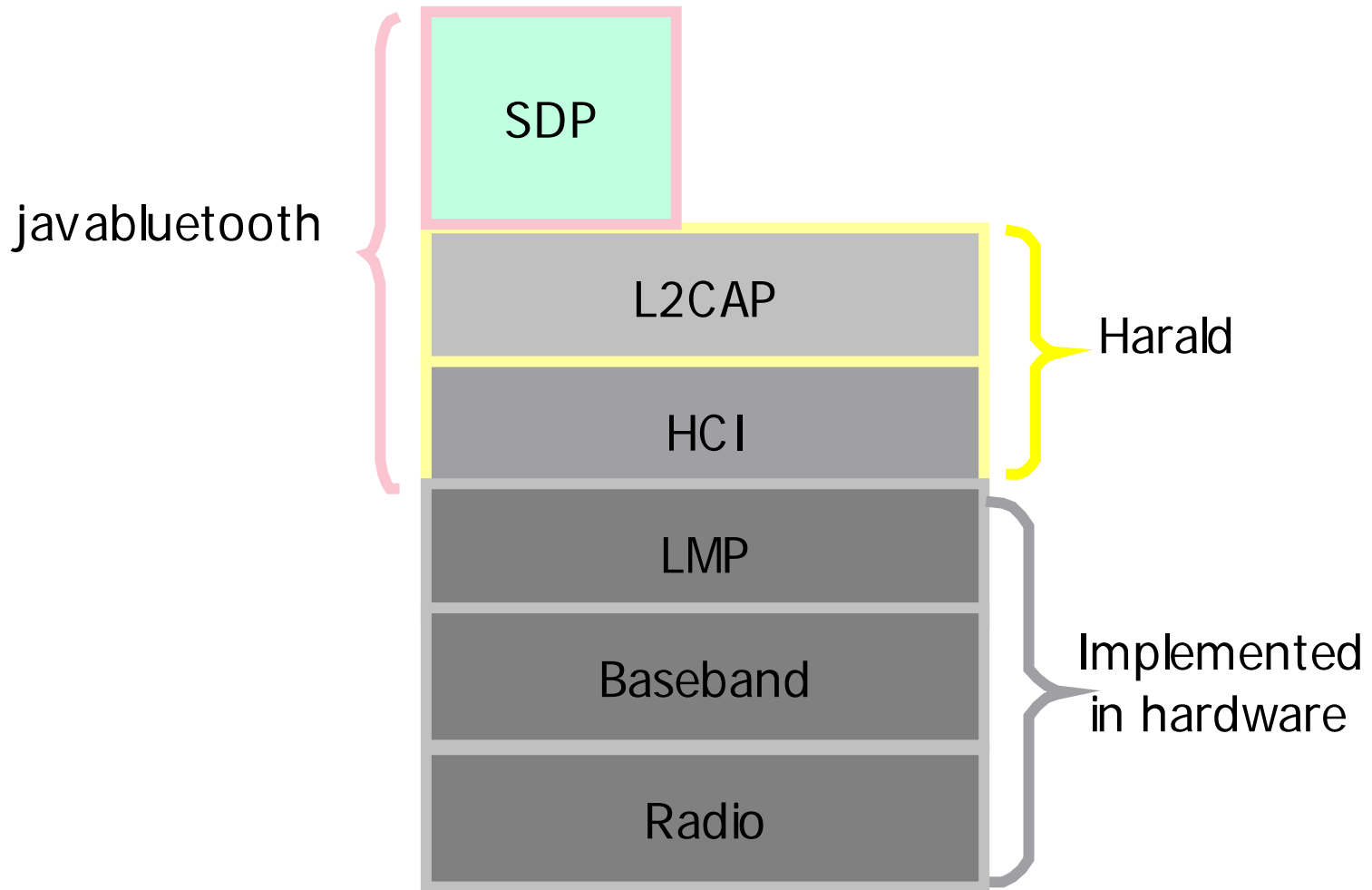
What's a Harald Bluetooth stack?

- “*a small Bluetooth stack for wireless communication between Java applications*”, author calls it
- Originally created for testing Bluetooth performance in distributed wireless control application (you can find more about it [here](#))

What about **javablue**tooth?

- distributed Bluetooth Stack for the TINI Embedded Java platform
- distributed architecture that allows a Bluetooth Chip to be controlled over TCP network

What's implemented where?



Royal blood

- Harald benefits: packet manipulating solved better, old-school coding style, understandable code
- Harald downsides: bigger in size, some things unnecessary complicated, no JSR-82 support whatsoever, no SDP
- **Extra feature:** Java object layer for passing java objects over L2CAP using serialization

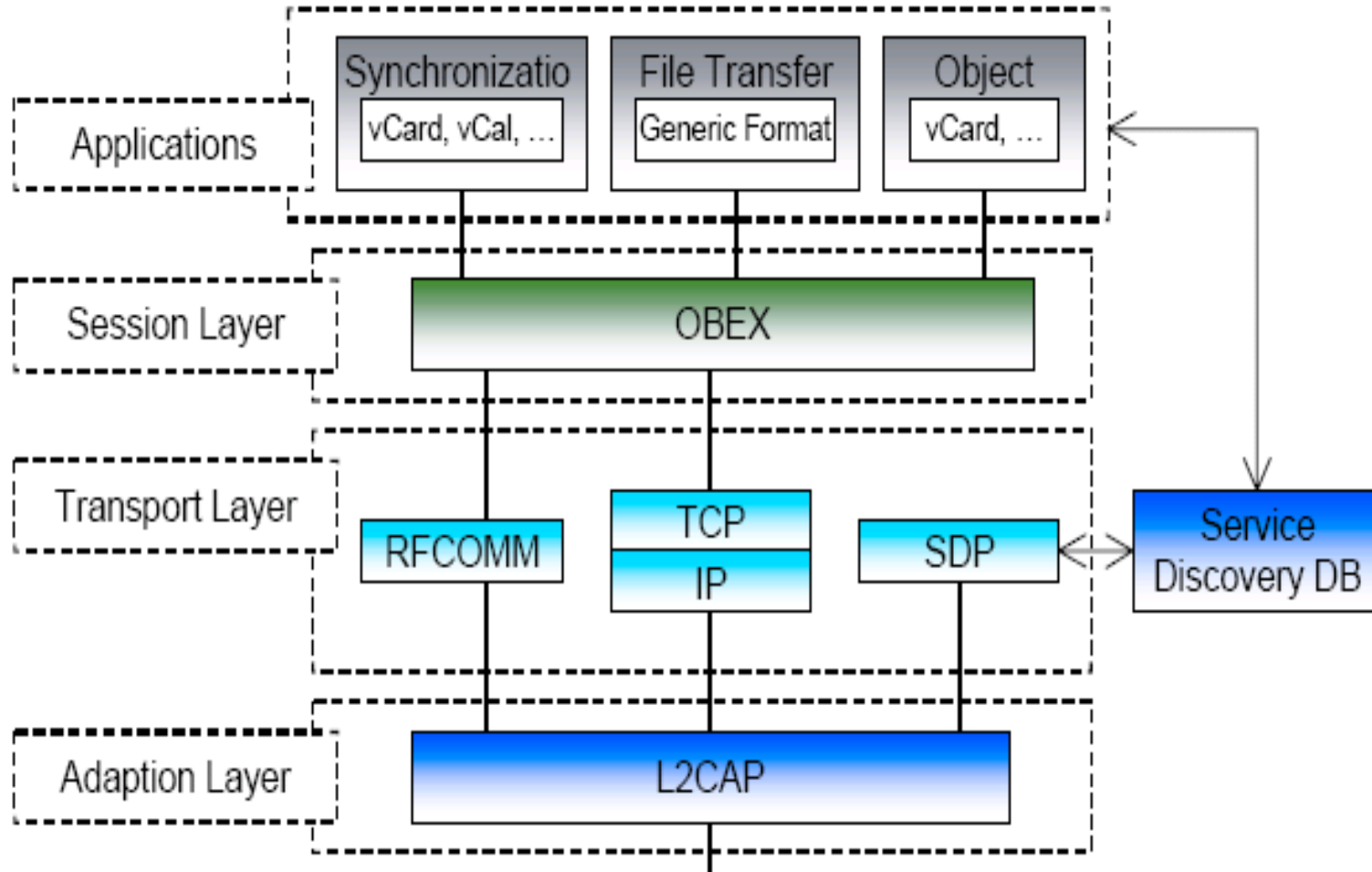
Fast & not careful

- javablueooth benefits: nicely organized code, smaller, has service discovery (SDP), some JSR-82 support
- javablueooth downsides: lots of bugs and mysterious behavior (as a result) ☹️
- **Extra feature:** distributed control over a Bluetooth device via TCP network

OBEX (OBject EXchange protocol)

- Adopted from IrDA (full name IrOBEX)
- Primary use: exchange of vCards, vCalendars, vMessages, and vNotes
- In Bluetooth: for object exchange in general
- Required by Object Push Profile, Synchronization profile, File Transfer Profile, etc.

More OBEX

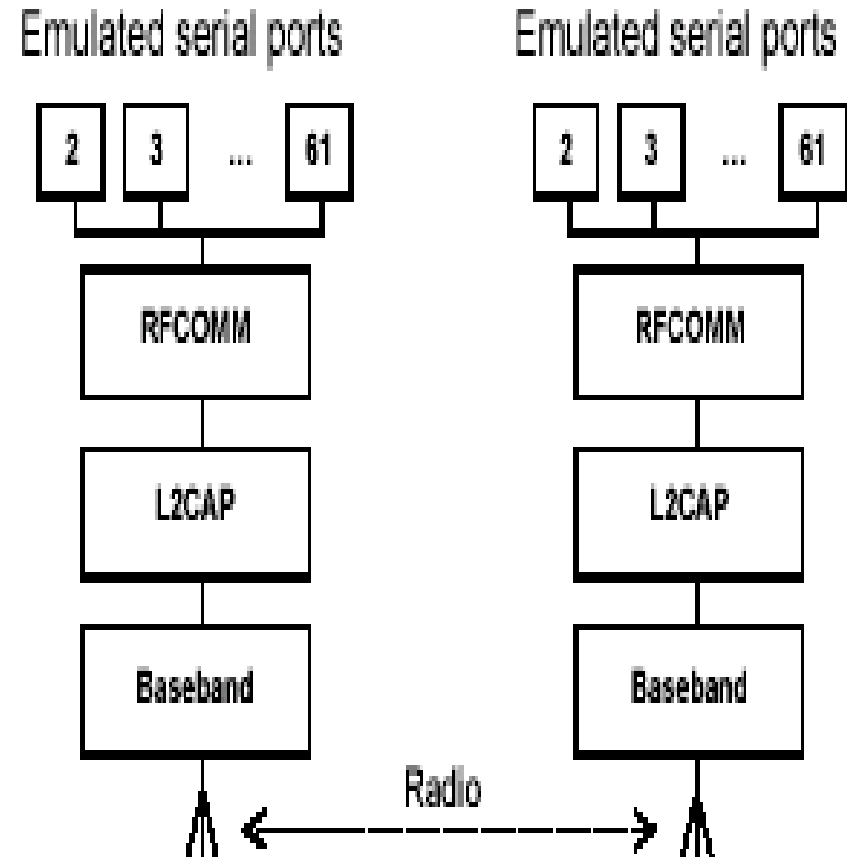


RFCOMM

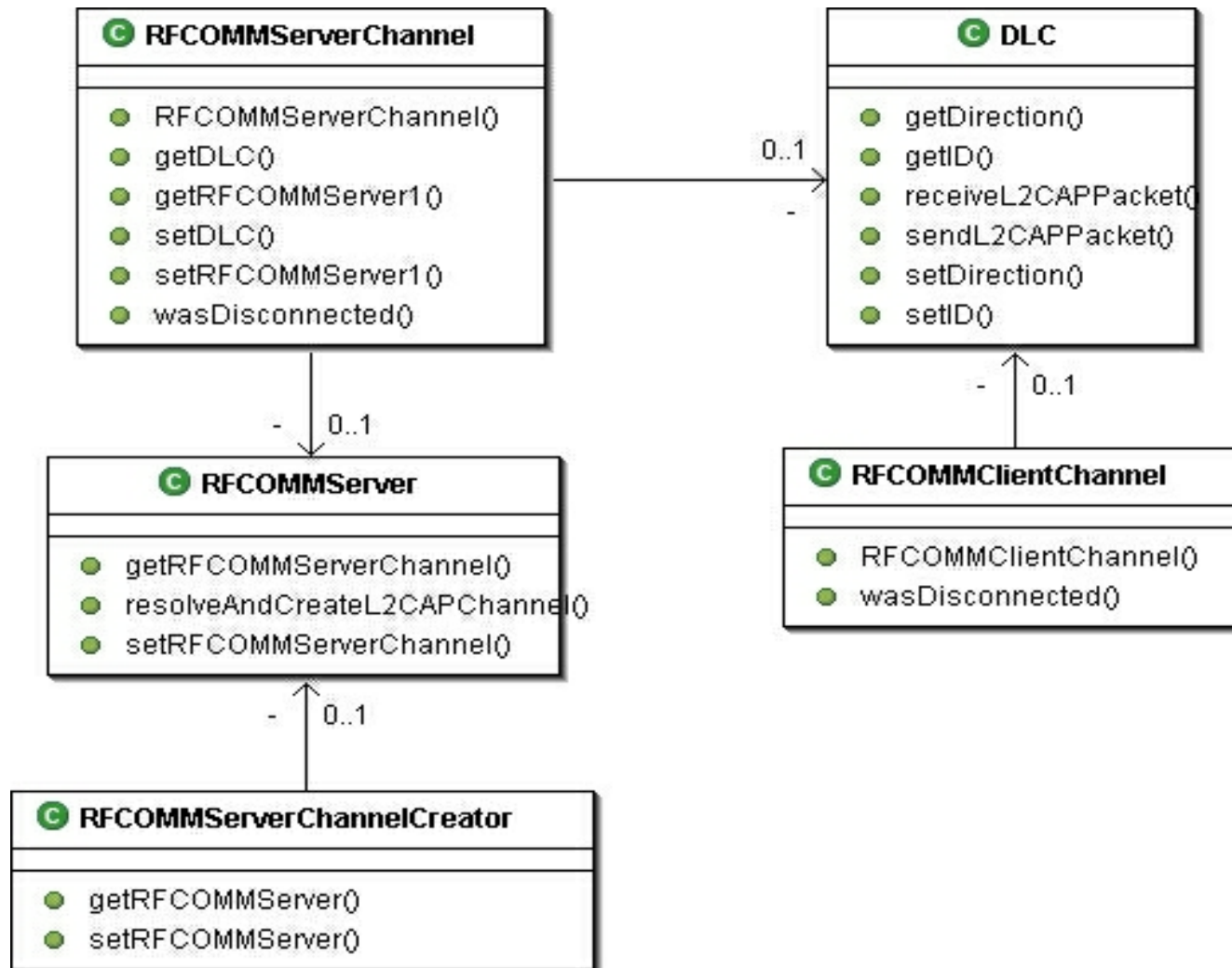
- Required by OBEX and most upper layer profiles
- Provides a transparent data stream and control channel over an L2CAP channel, and also multiplexes multiple emulated serial ports.
- Replaces a serial cable between devices

RFCOMM

- Multiple serial ports between devices (up to 60)



RFCOMM class diagram



Bluetooth security – basic stuff

- Authentication of devices(not users!)

(String url = "btspp://localhost:8128.....32;authenticate=true")

- Authorization

(String url = "btspp://localhost:8128.....32; authenticate=true
authorize=true")

- Encryption

(String url = "btspp://localhost:8128.....32;encrypt=true")

Bluetooth security in JSR-82

- Handled by BCC (Bluetooth Control Center), a centralized security manager that manages all connection requests and takes specified actions in order to protect security policy
- BCC security related responsibilities :
 - Setting the security level for the device
 - Maintaining a list of devices discovered earlier and a list of trusted devices
 - Giving a mechanism for pairing up and authorization of devices that are communicating for the first time.

Security class diagram

