Beetle: Flexible Communication For Bluetooth Low Energy

Amit Levy, James Hong, Laurynas Rilskis, Philip Levis, Keith Winstein

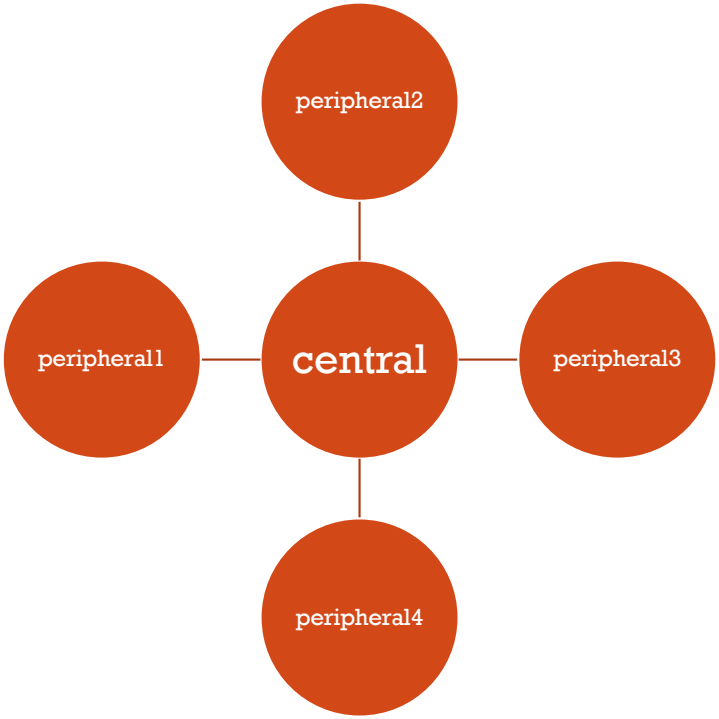
Stanford University

Presented By: Ratnadeep Bhattacharya

Abstract

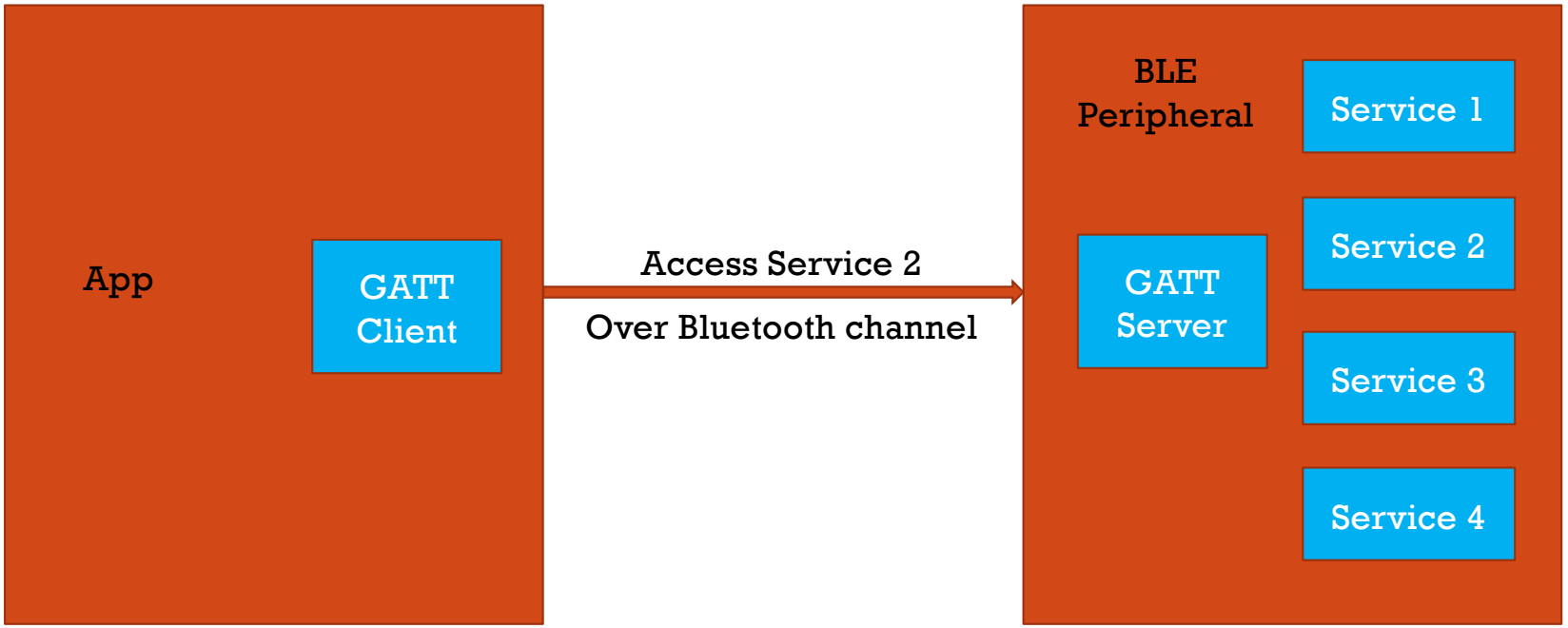
- Low power computer peripherals are becoming more and more ubiquitous.
- This paper presents Beetle, a new hardware interface that virtualizes peripherals at the application layer allowing:
  - **Safe access by multiple programs** without requiring OS to understand hardware functionality
  - **Fine-grained access control** to the peripheral devices
  - **Transparent network access** to peripherals

Background



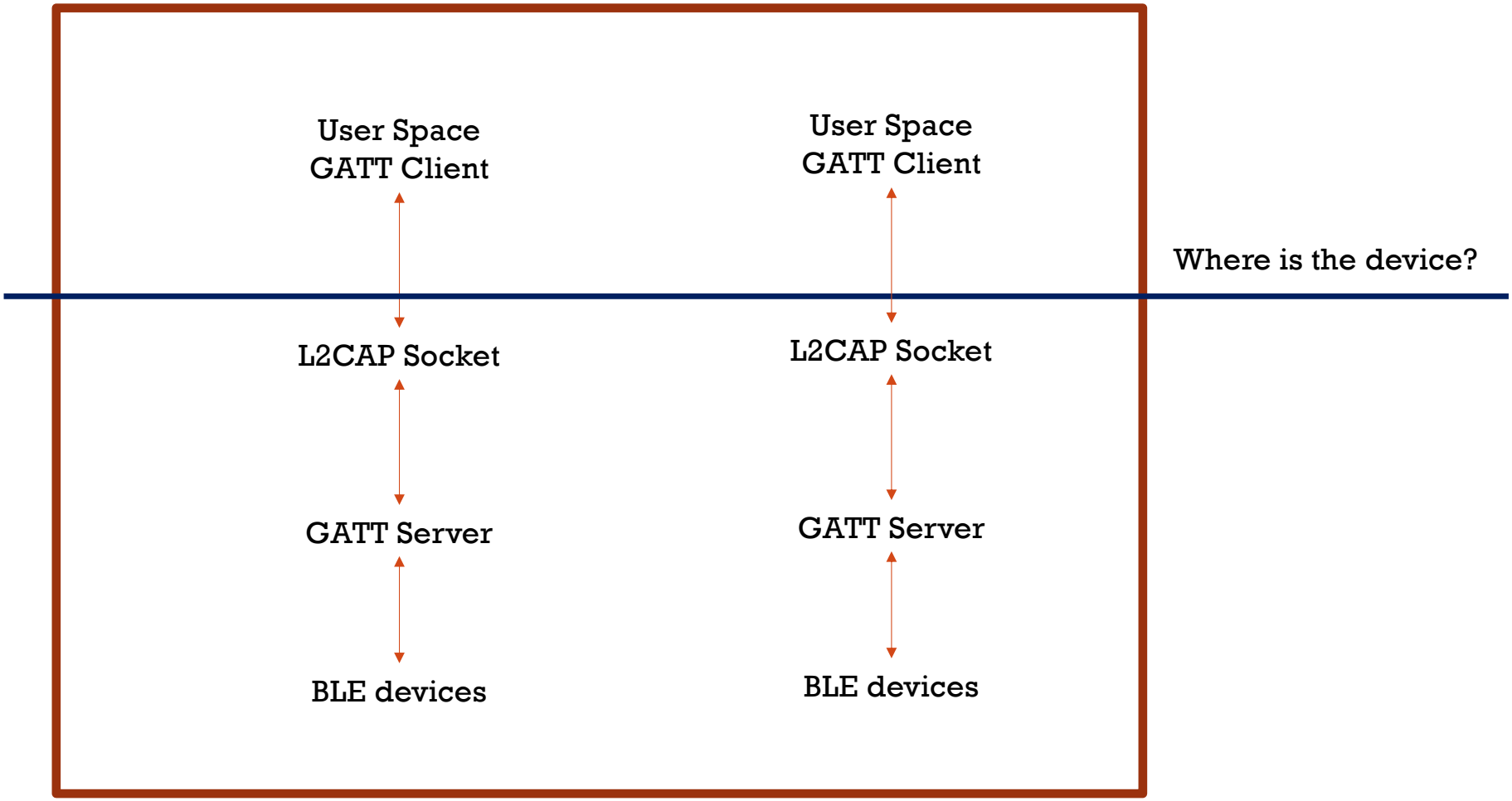
- Ultra-low power wireless protocol for single-hop networks
- Ranges of 1-10 meters.
- Avg power draw < 10μA
- Can run for weeks, months and even years
- Energy efficiency comes from tight time synchronization with the central deciding the communication schedule

How BLE Works



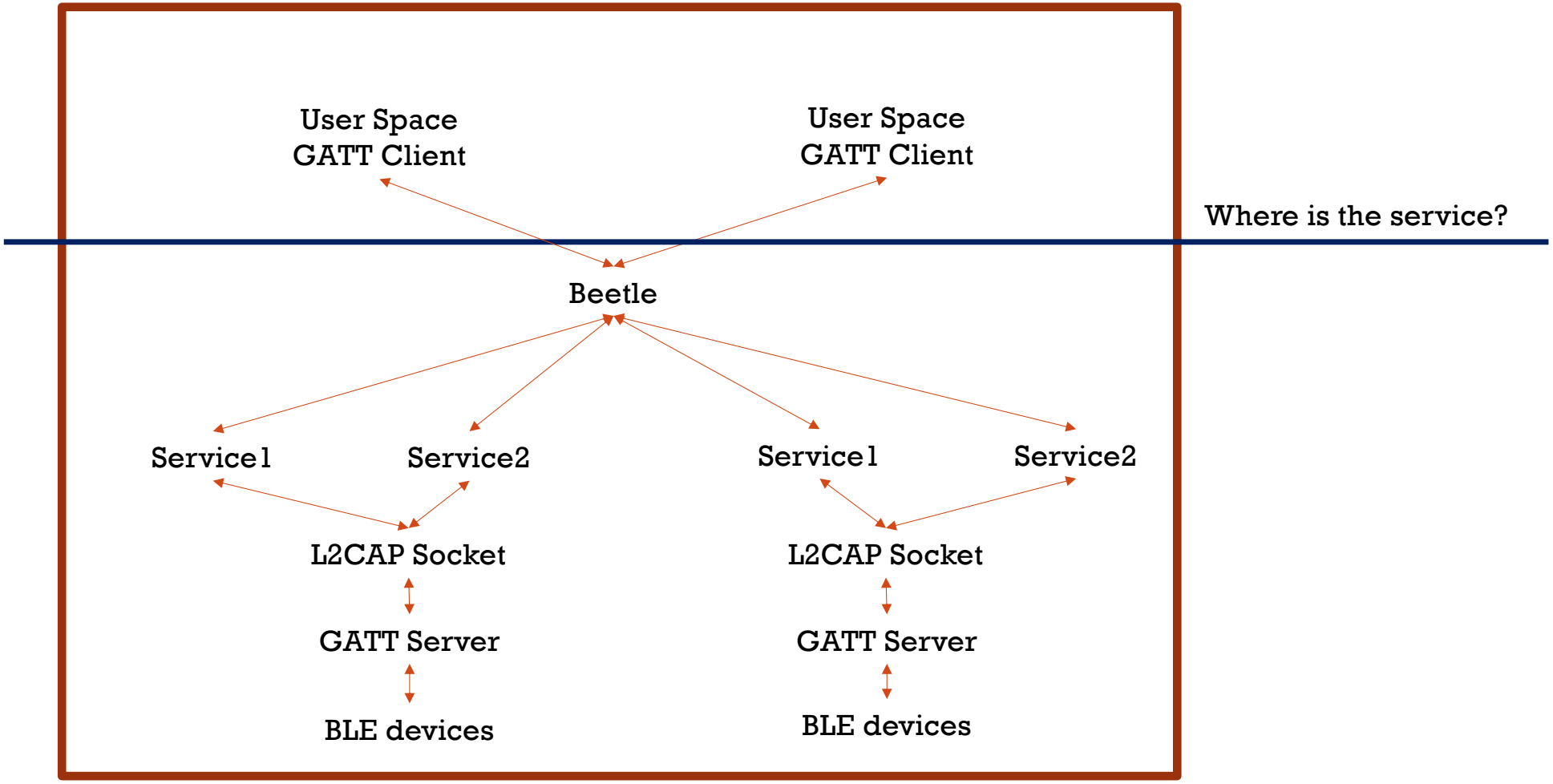
How OS handles BLE

- Currently, the way an OS provides access to BLE devices is through complete access to the hardware for individual applications.
- It's done either through sockets (Linux, Windows, OSX) or APIs and Java classes (Android). This means that once the connection is setup, the application completely owns the device.


User Space
GATT Client
User Space
GATT Client
Where is the device?
L2CAP Socket
L2CAP Socket
GATT Server
GATT Server
BLE devices
BLE devices

How Beetle handles BLE

- Beetle aims to provide access to individual functionalities (services) and transactions (READ/WRITE).
- Beetle represents every BLE resource as a virtual device backed by either:
  - A physical peripheral device
  - A user level process
  - A network service

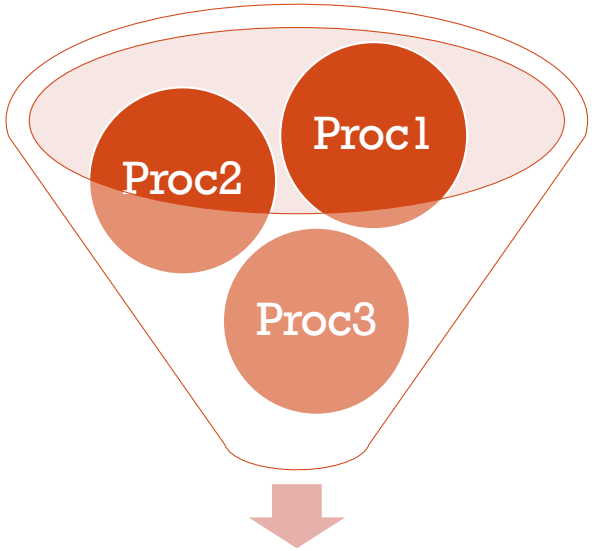

User Space
GATT Client
User Space
GATT Client
Where is the service?
Beetle
Service1
Service2
Service1
Service2
L2CAP Socket
L2CAP Socket
GATT Server
GATT Server
BLE devices
BLE devices

Introduction

- OS exposes peripherals through well defined interfaces for known functions.
- Otherwise, it exposes the raw hardware channel.
- Without knowledge of the underlying function, OS cannot provide:
  - multiplexed access
  - security.

OS restrictions on BLE peripherals

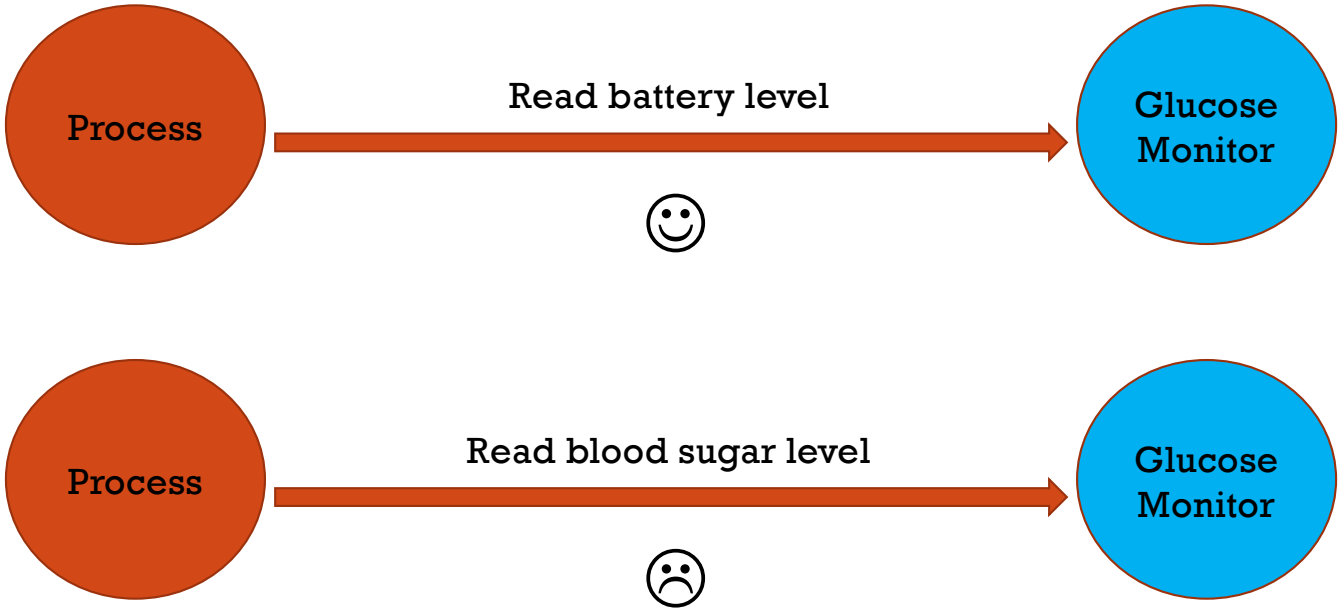
A Single Application At A Time



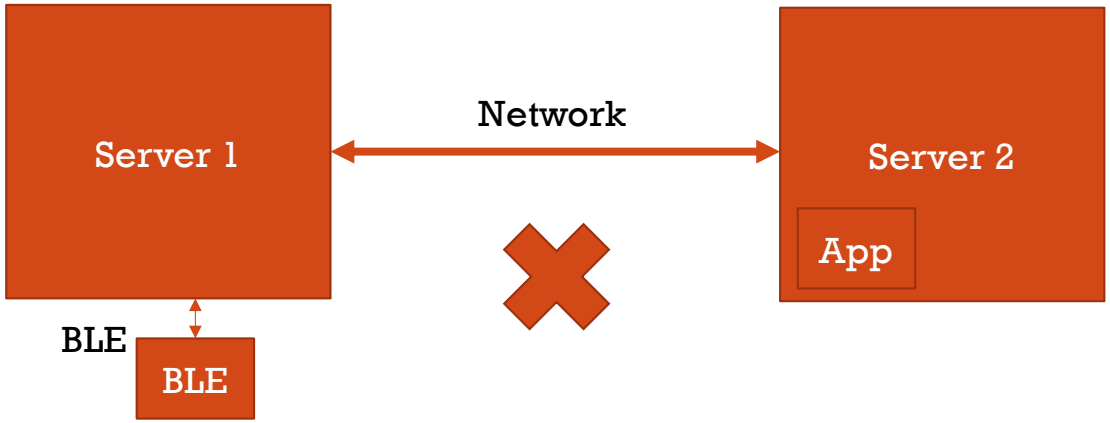
BLE Peripheral

- A vertical stack.
- One application completely owns the device
- Others must wait for the application to release the device

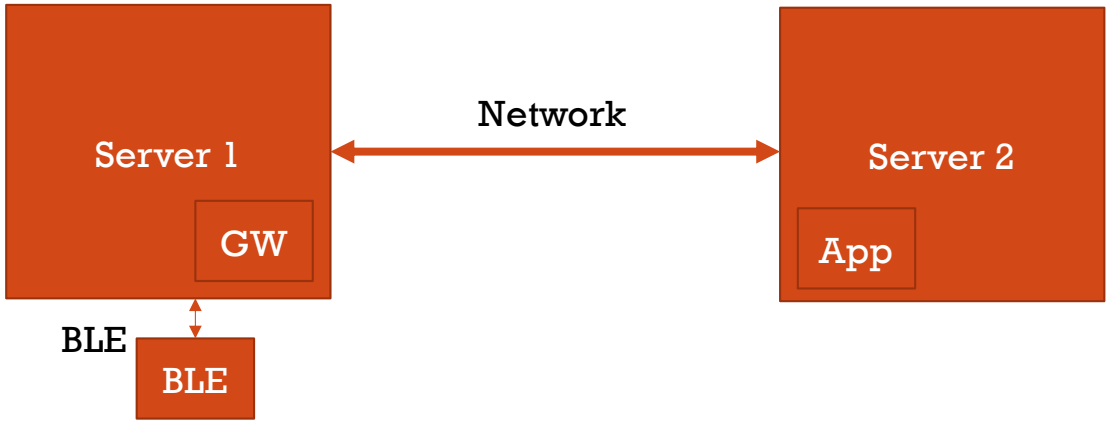
Security Implication: Application has complete control



Communication: Application on remote system can't communicate with BLE

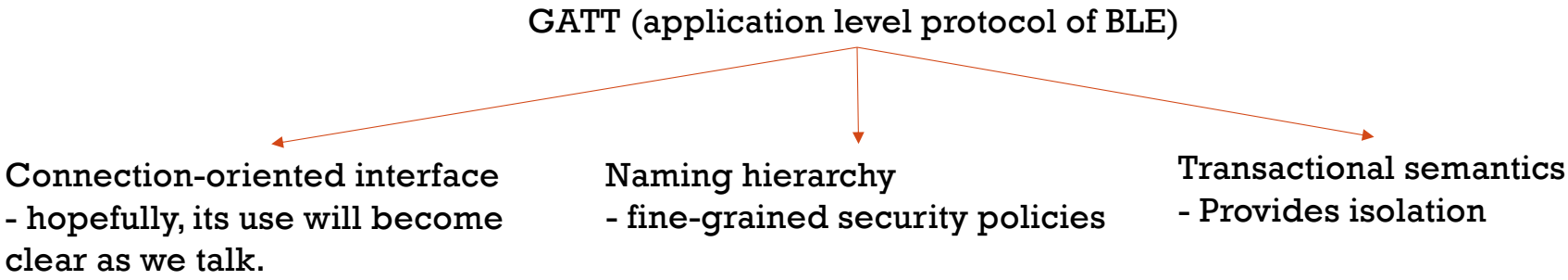


Communication: Application on remote system can't communicate with BLE



But how to talk to BLE peripherals?

Beetle uses GATT (Generic Attribute Profile)

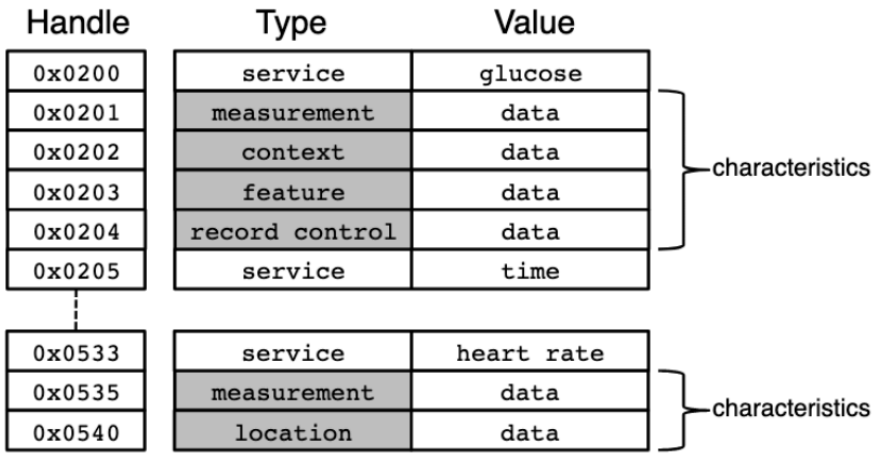


- These three GATT properties provide sufficient structure for an OS to manage and understand application behavior and properly manage access.
- Removes the need for the OS to understand the function of a peripheral.

GATT Runs On Top L2CAP (link-layer)

- Each side of a GATT connection exposes data objects as an attribute
- *Attributes*: key-type-value tuples
  - Key is 16 bits long, local to the connection and is called a *handle* (equivalent to a TCP/UDP port)
  - Type is a 128-bit UUID.
- A peripheral or central can access attributes on the other using atomic commands like:
  - **READ**
  - **WRITE**
  - **NOTIFY**

GATT Attributes



- Structured, fixed-depth, hierarchical namespace
- Any device has zero or more service attributes
- Each service attribute has a set of related data values – *characteristics*
- Typically, services and characteristics are standardized by the Bluetooth Special Interest Group.
- However, vendors also can define services for functions that have not been standardized by the Bluetooth SIG

Details of Beetle

A Brief About The Test Applications

Battery monitoring	Accessing other data from the sensors
	Preventing the main application from accessing the sensor
Sharing heart tracker	Single peripheral
	multiple applications for different purposes.
Generic gateway	Single gateway supporting multiple applications.

Access provided by OS should be able to support

Sharing: multiple apps must be able to access a peripheral while remaining isolated from each other.

Many-to-many communication: multiple peripherals should be able to talk to one another.

Access Control: fine-grained policies dictating which applications can access which services.

GATT Capabilities

GATT has a well-defined set of commands for reading, modifying and discovering attributes.

Beetle uses this layer to provide multiplexed, isolated BLE peripheral access to applications.

Beetle Transaction Layer; Command Types

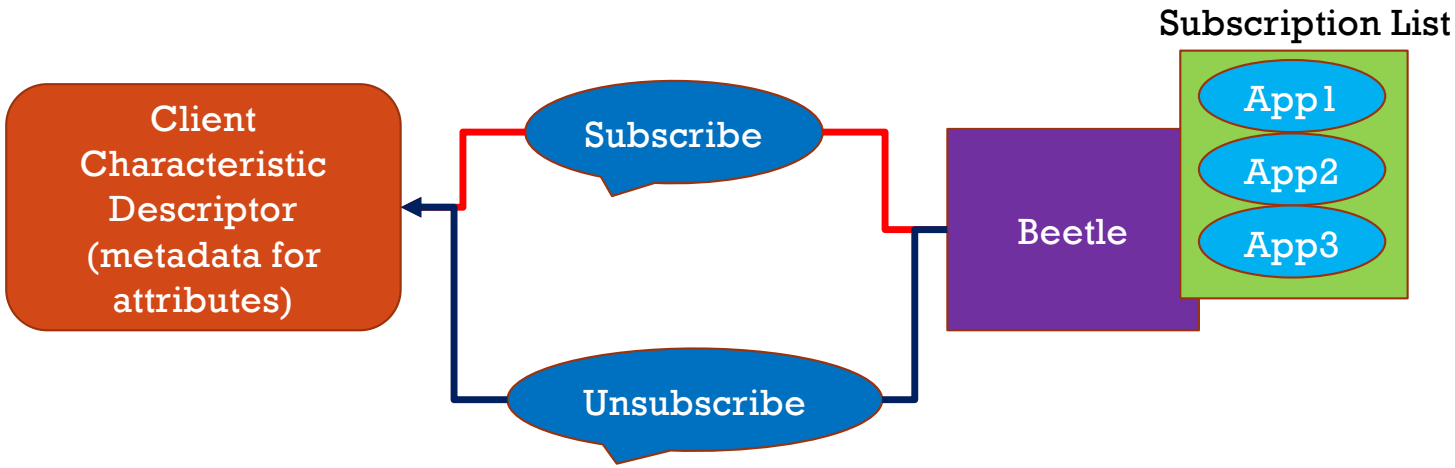
READ/WRITE

Most common command type to read/write value of an attribute

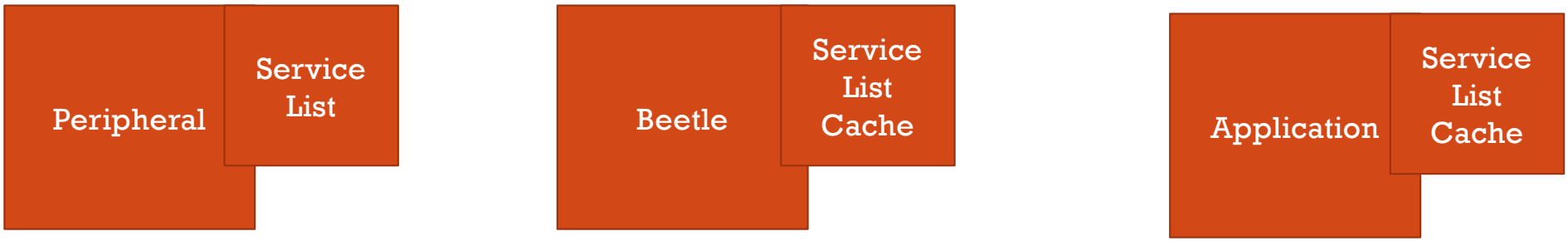
Examples:
Read current battery level
Lock/Unlock a door

Beetle simply acts as a proxy between nodes for these requests

NOTIFY


Subscription List
Client Characteristic Descriptor (metadata for attributes)
Subscribe
Unsubscribe
Beetle
App1
App2
App3

Service Discovery



Discover services for Peripheral

Service Query from application

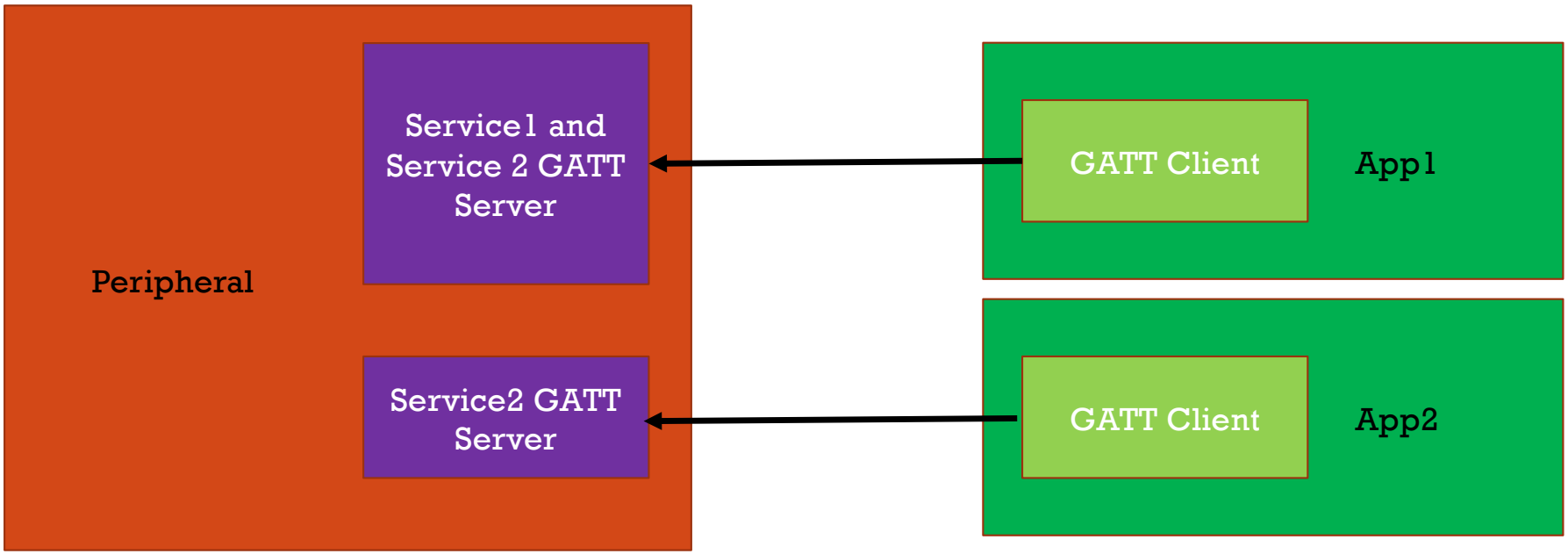
- Saves peripheral energy by limiting query to once
- Helps Beetle do notifications properly since it knows about all services

Virtual Devices

Why Are Virtual Devices Needed

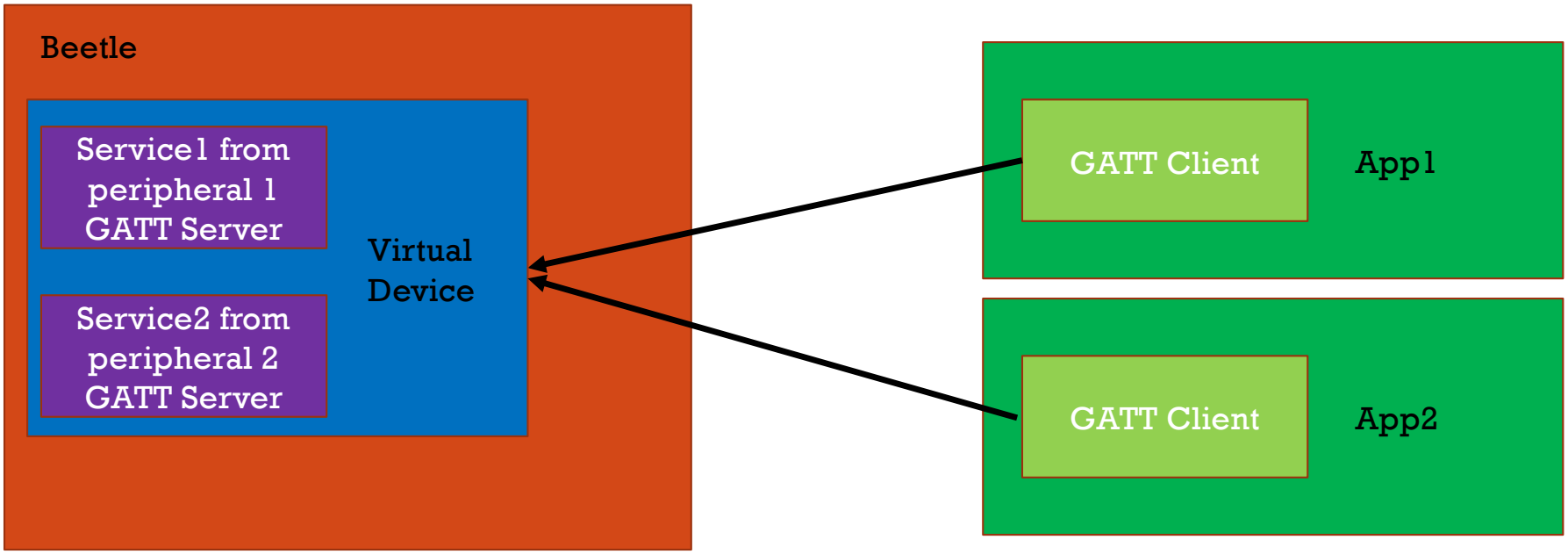
- So far, though access to the same device has been successfully provided for multiple applications, we are still dealing with the whole device.
- How do we limit an application to access only the service it was meant to access?
- How to multiplex multiple services from different devices together?
- Virtual devices provide the answer.

GATT, Again!


Peripheral
Service1 and Service 2 GATT Server
Service2 GATT Server
GATT Client
App1
GATT Client
App2

Still, why Virtual Devices!

Multiplexing Services From Different Peripherals



Access Control Is Slightly More Challenging

Peripheral

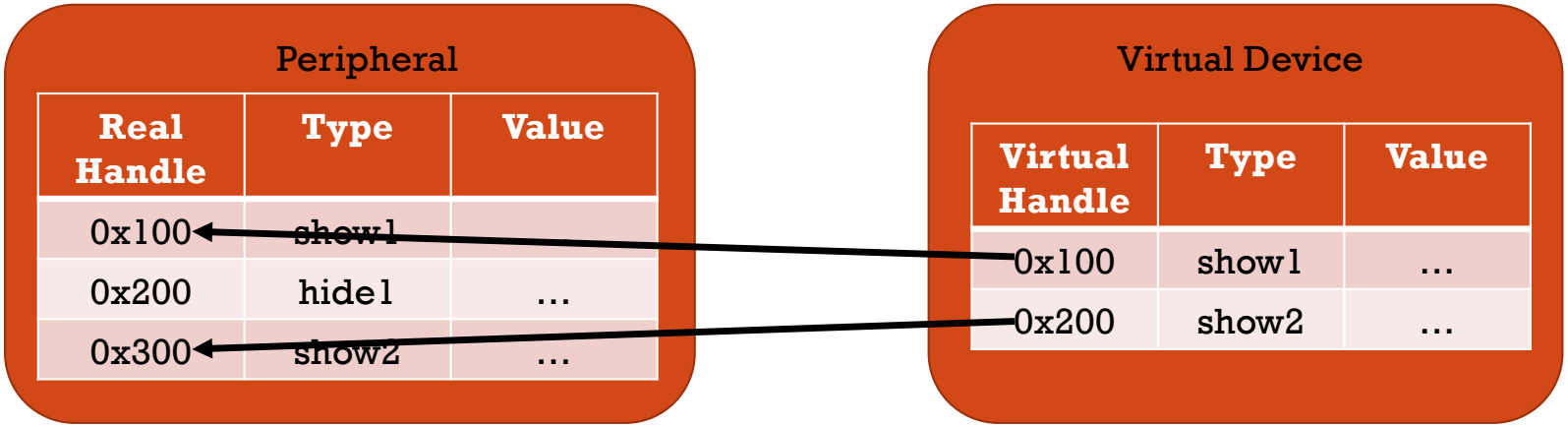
Handle	Type	Value
0x100	show1	…
0x200	hide1	…
0x300	show2	…

Virtual Device

Handle	[illegible]	Value
0x100	[illegible]	…
0x300	[illegible]	…

Handles not consecutive

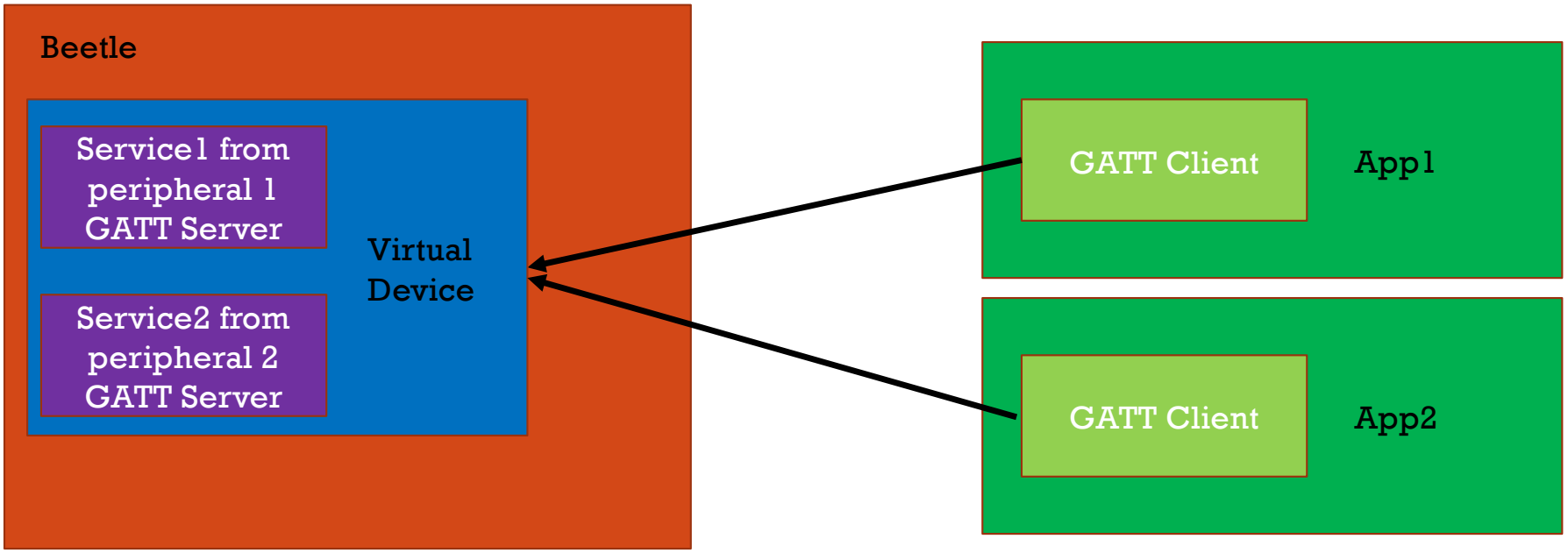
Solution



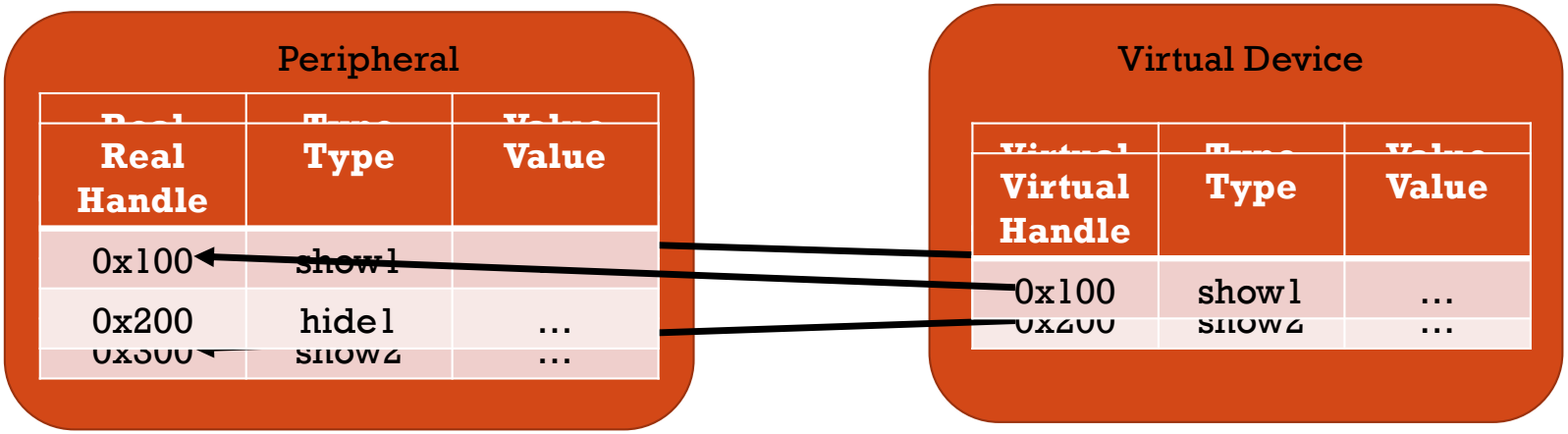
Virtual Handles consecutive

So what happens when services change?

More GATT, Service Changed Characteristics



More GATT, Service Changed Characteristics



Access Control

Exclusive Access

- What to do when writing to one attribute changes behavior of other attributes? This violates Beetle's assumption that devices can be safely multiplexed
  - Example: Changing unit or sampling rate of measurement.
- Such cases can be handled by providing exclusive access to a single application and then that application can re-expose services as it sees fit.
- Beetle can apply rules to any virtual device

Implementation

Linux

- Kernel manages BLE peripherals and exposes L2CAP socket interface to user-space.
- Application can communicate with the peripheral using GATT directly.
- With Beetle, Beetle daemon owns all L2CAP sockets
- Applications use UNIX domain sockets to access individual services through the L2CAP sockets (through files; see figure)
- UNIX domain sockets are named with the GATT UUID
- Beetle traps the GATT commands and redirects them appropriately

```
MyHRM/ ──►00:12:55:3D:2C:FF/

00:12:55:3D:2C:FF/
    ├──00001800-0000-1000-8000-00805F9B34FB=  # GAP Service
    ├──0000180D-0000-1000-8000-00805F9B34FB=  # Heart Rate
    ├──0000180F-0000-1000-8000-00805F9B34FB=  # Battery Service
    └──advertising_data
MyBike/ ──►E6:1D:1E:D0:16:0B/

E6:1D:1E:D0:16:0B/
    ├──00001800-0000-1000-8000-00805F9B34FB=  # GAP Service
    ├──00001816-0000-1000-8000-00805F9B34FB=  # Cadence Meter
    ├──0000180F-0000-1000-8000-00805F9B34FB=  # Battery Service
    └──advertising_data
VirtualFit/ ──►DF:19:D5:77:44:9B/

DF:19:D5:77:44:9B/
    ├──0000180D-0000-1000-8000-00805F9B34FB=  ──► ../MyHRM/00001800...
    ├──00001816-0000-1000-8000-00805F9B34FB=  ──► ../MyBike/00001800...
    └──advertising_data
```

Performance

Round Trip Latency

Peripheral-to-peripheral communication is currently not possible without Beetle

However, with a Beetle node connecting peripherals, latency for requests can increase by 2x.

Multi-application Throughput

For commands whose results cannot be cached (eg. WRITE), the throughput can be quite low with limited scale-up.

For cacheable commands (eg. READ), the throughput can grow linearly for quite a few devices.

SUMMARY

Beetle allows OS to provide a generic interface for BLE devices.

- Sharing access to peripherals between multiple applications
- Fine-grained access control
- Many-to-many communication between BLE peripherals