BLUETOOTH LOW ENERGY TECHNOLOGY TRAINING

Agenda	Coverage
High Level View	What is low energy? Optimizations? Comparisons with Bluetooth, New Models, Gateways, Application Stores
	morning break
Core Spec : Controller	Physical Layer, Direct Test Mode, Link Layer, Host Controller Interface
	lunch break
Core Spec : Host	L2CAP, Security Manager, Attribute Protocol, Generic Attribute Profile, Generic Access Profile
	afternoon break
Applications / Use Cases	Client / Server, State & Behavior, Remote, Proximity, Automation, Medical, Qualification

WHAT IS LOW ENERGY?

IT IS NEW TECHNOLOGY

blank sheet of paper design

optimized for ultra low power

different to classic *Bluetooth* technology

NEW TECHNOLOGY?

Yes

- efficient discovery / connection procedures
- very short packets
- asymmetric design for peripherals
- client server architecture

No

- reuse existing BR radio architecture
- reuse existing HCI logical and physical transports
- reuse existing L2CAP packets

BASIC CONCEPTS

Optimize *EVERYTHING* for lowest power consumption

from the physics to the users

Why?

button cells will be main power source for peripherals

< 15 mA peak current

< 1 µA average current

BASIC CONCEPTS

Receiving more expensive, Transmitting cheap

- best optimize to reduce Rx time as much possible

Advertising Channels – discovery / connections

- just use 3 channels – requires modulation index change
- lowest power device advertises
  - "Peripheral"

BASIC CONCEPTS

Memory is expensive

- Memory requires silicon area – costs money
- Memory increases leakage current – costs battery life

Reduce dynamic memory footprint for specs

- keep packets short – less buffer memory
- keep protocol simple – less state information
- keep services simple – one protocol / defined behavior

BASIC CONCEPTS

Keep packets short

- when Tx – short packets don't need constant calibration
  - reduces peak current during Tx

- when Rx – radio on for less time
  - reduces total current usage

Optimized for low power consumption

BASIC CONCEPTS

Peripherals are simple – very resource constrained

- optimize peripherals first

Central devices are complex – lots of memory & battery

- not critical to optimize here

Asymmetric is good

- Link Layer, Security, Application Architecture

BASIC CONCEPTS

Design for success

- able to discover thousands of devices in local area
- unlimited number of slaves connected to a master
- unlimited number of masters
- state of the art encryption
- security including privacy / authentication / authorization
- class leading robustness, data integrity
- future proof

BASIC CONCEPTS

Everything has STATE
- devices expose their state
- these are servers

Clients can use the state exposed on servers
- read it – get current temperature
- write it – increase set point temperature for room

Servers can tell clients when state updates
- notify it – temperature up to set point

BASIC CONCEPTS

Client Server Architecture

- proven architecture for web-infrastructure

Gateways allow interconnect of internet & low energy

- weighing scales send reports to doctor
- home security web site shows all windows closed
- assisted living for your parents allows low cost monitoring
- sports data immediately uploaded via cellular phone

BASIC CONCEPTS

Modular architecture allows gradual innovation

emerging markets well served

Phase 1: Smart Meters

Meters publish rates / usage to home

Phase 2: Smart Appliances

Appliance Power controlled from web site

Phase 3: Smart Energy Brokers

Automatically schedule appliances depending on costs

BASIC CONCEPTS

Cellular phones will create billion unit market in 2 years

- Dual mode BR/EDR & low energy devices go into phones
  - and computers
  - and televisions
  - and anything else that has classic *Bluetooth* now

- Single mode low energy devices will connect to these
  - huge pull from market for low energy devices

COMPARISON WITH CLASSIC BLUETOOTH

Feature	BR/EDR	LE	Notes
RF Channels	79	40	2 MHz spacing in LE
Modulation	GFSK	GFSK	Simple and effective
Modulation Index	0.25 to 0.35	0.45 to 0.55	Wider signal – more robust
Max Tx Power	+20 dBm (class 1) +4 dBm (class 2)	+10 dBm	No “class” structure +10 dBm regulatory limit
Rx Sensitivity (typical)	-85 dBm	-85 dBm	Pathloss = 90 dB for BR Pathloss = 95 dB for LE
Range (typical)	30 meters	50 meters	Modulation Index, increased power for class 2

COMPARISON WITH CLASSIC BLUETOOTH

Feature	BR/EDR	LE	Notes
Packet Format	6 (BR / EDR)	2 (LE)	ID, FHS, DM, DH, 2-DH, 3-DH - Advertising / Data
Ack Packet Len	126 µs	80 µs	63% shorter
8 octet Packet	214 µs	144 µs	67% shorter
Max Packet Size	2875 µs = 1021 octets	328 µs = 27 octets	LE very short
Max Data Rate	2178.1 kb/s	305 kb/s	EDR much faster
Time to transfer 1Mbyte	DH1 = 18.2 s, DH5 = 8.8 s, 3-DH5 = 2.9 s	13.9 s (LE)	LE less efficient for large packets
CRC Strength	16	24	LE stronger
Encryption	Safer+	AES-128	LE stronger

COMPARISON WITH CLASSIC BLUETOOTH

Feature	BR/EDR	LE	Notes
Authentication	once	every packet	more secure
Acknowledge	immediate	sliding window	lower power
Topology	flexible	pure star	LE simpler
Discoverable	Inquiry Scanning 11.25 ms / 1.25 s	Advertising 1.25 ms / 1.25 s	10x lower power
Connectable	Page Scanning 11.25 ms / 1.25 s	Advertising 1.25 ms / 1.25 s	10x lower power
Discoverable + Connectable	Inquiry + Page Scan 22.5 ms / 1.25 s	Advertising 1.25 ms / 1.25 s	20x lower power
LMP PDUs	75	14	5x simpler
Feature Bits	59	1	59x simpler

COMPARISON WITH CLASSIC BLUETOOTH

Feature	BR/EDR	LE	Notes
Connection time	20 ms (R0 Page Scan)	2.5 ms	8x quicker
LMP negotiation time	min 5 ms ~ 50 ms	no negotiation required	instant
L2CAP connection setup time	min 5 ms ~ 50 ms	uses fixed channel no negotiation required	instant
Time to send application data	30 ms ~ 120 ms	3 ms	10x quicker
Time to AFH	1.25 ms	instant	no penalty for coexistence
Time to Sniff Subrating	2.5 ms	instant	no penalty for low power

COMPARISON WITH CLASSIC BLUETOOTH

Feature	BR/EDR	LE	Notes
Protocols Supported by Host	14	3	Only ATT required for all plain text applications
Min protocols for application	3 (SDP, L2CAP, App Protocol)	2 (ATT, L2CAP)	LE uses ATT for service discovery and apps
1 MB using BR	8.81 s	13.93 s	BR 60% faster for data
1 MB using EDR	2.93 s	13.93 s	EDR ~5x faster
1 MB using HS	< 1s	13.93 s	LE very slow
L2CAP overhead	4 to 12 octets	4 octets	LE basic headers only
L2CAP configuration options	7	0	Nothing configured in LE
L2CAP commands	17	1	LE very simple

COMPARISON WITH CLASSIC BLUETOOTH

Uses 2.4 GHz ISM Band

- Industrial Scientific Medical band
- License Free – with certain rules
- 2400 MHz to 2483.5 MHz

Used by many other standards

- IEEE 802.11, IEEE 802.15
- and many proprietary radios

NEW CONNECTION MODELS

Bluetooth BR/EDR is largely cable replacement:

- Headset Cables
- Mouse Cables
- Keyboard Cables

Bluetooth low energy is application enabling:

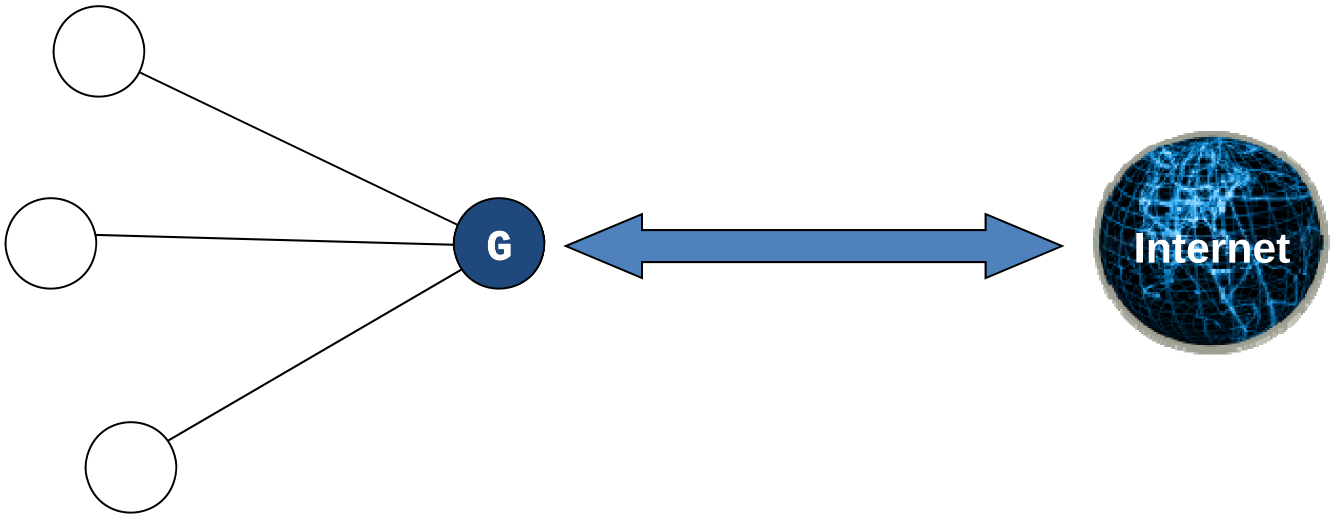
- Accessories for smartphone apps
- Internet connected devices
- New billion unit markets

NEW MARKETS NEED NEW TOPOLOGIES

	TAM	Topology
Phone accessories	> 10 billion	P2P / Gateway
Smart Energy (meters & displays).	~ 1 billion	Gateway Hub
Home Automation	> 5 billion	Gateway / Mesh
Health, Wellness, Sports & Fitness	> 10 billion	P2P / Gateway
Assisted Living	> 5 billion	Gateway
Animal Tagging	~ 3 billion	P2P
Intelligent Transport Systems	> 1 billion	?
M2M (Internet connected devices)	> 10 billion	Gateway

Over 90% of the next 50 billion devices may use a gateway topology

GATEWAYS, END TO END & APPS



Devices talk through gateways to web apps.
They can cause specific apps to be loaded on a gateway device.
Internet apps communicate directly with the device – the gateway is just a tunnel.
Gateways are generic.
Bluetooth low energy supports this better than any other standard.

GATEWAYS ARE GENERIC

A Gateway application has limited functionality

It finds out what the device wants to connect to.

It provides a secure, transparent tunnel to that destination.

Handsets can ship with generic gateways apps

The connect ANY device to its internet host

Users don’t need to load any drivers or software.

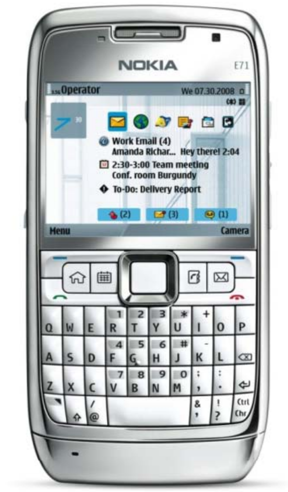
Gateways enable Internet connected devices.

DEVICES SHIP WITH A WEB ADDRESS

DEVICES SHIP WITH A WEB ADDRESS

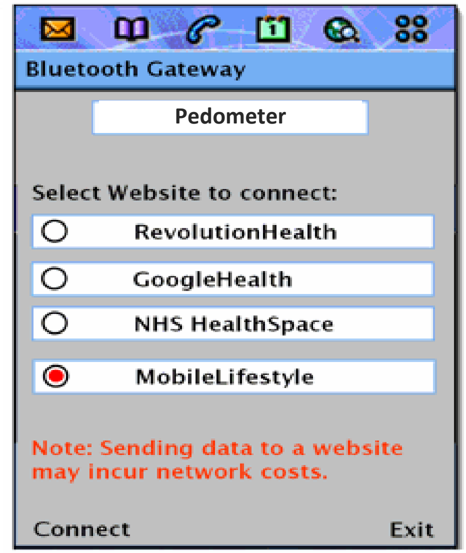
www.patientslikeme.com

THEY CONNECT TO A GENERIC GATEWAY APP



THEY CONNECT TO A GENERIC GATEWAY APP

THEY CONNECT TO A GENERIC GATEWAY APP



THENCE TO THE WEB

AND AUTOMATICALLY TRANSFER DATA

The phone is just a pipe, running a generic application.

AND AUTOMATICALLY TRANSFER DATA

The phone is just a pipe, running a generic application.

THE APPS STORE MODEL

The user paradigm has changed

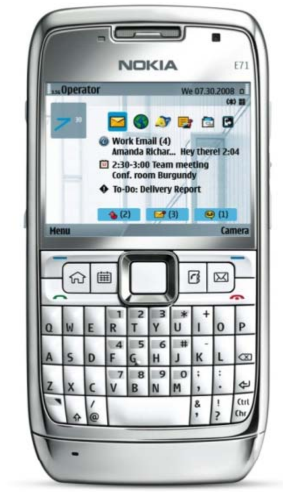
Smartphones are enabling a new application market

Today that stops at the phone

Bluetooth low energy allows it to extend to devices

- Operators can make revenue without the keyboard being touched.
- Connected devices mean many new applications
- low energy devices can make Apps selection easy

DEVICES TELL THE PHONE WHAT THEY ARE



DEVICES TELL THE PHONE WHAT THEY ARE

DEVICES TELL THE PHONE WHAT THEY ARE

Pedometer
Acme Model XYZ
Steps per Minute
Total Steps
Calories Used
Find me an APP…

MAKING APPS DOWNLOADS EASY

A low energy device can tell a client device exactly what it can do, by having its services read.

This can include the location of preferred apps for a smartphone.

A generic application on a handset can use this information to interrogate an Apps store to provide a list of applications that are know to work with the device.

The user experience is significantly improved, as choosing a compatible app is made much easier.

TAILORING THE APPLICATION STORE CHOICE

Easy to buy

=

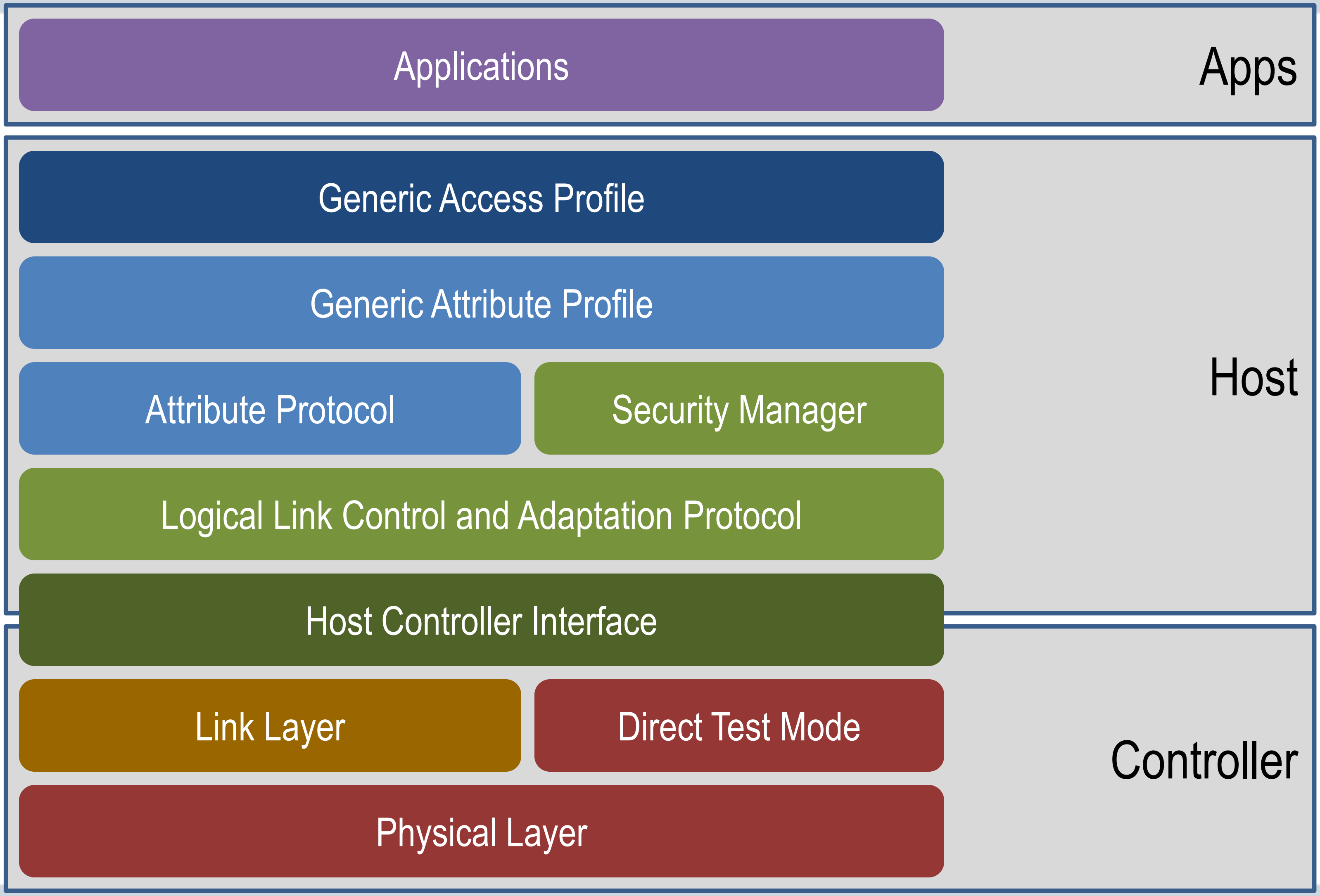
More revenue

Guaranteed to work = More downloads

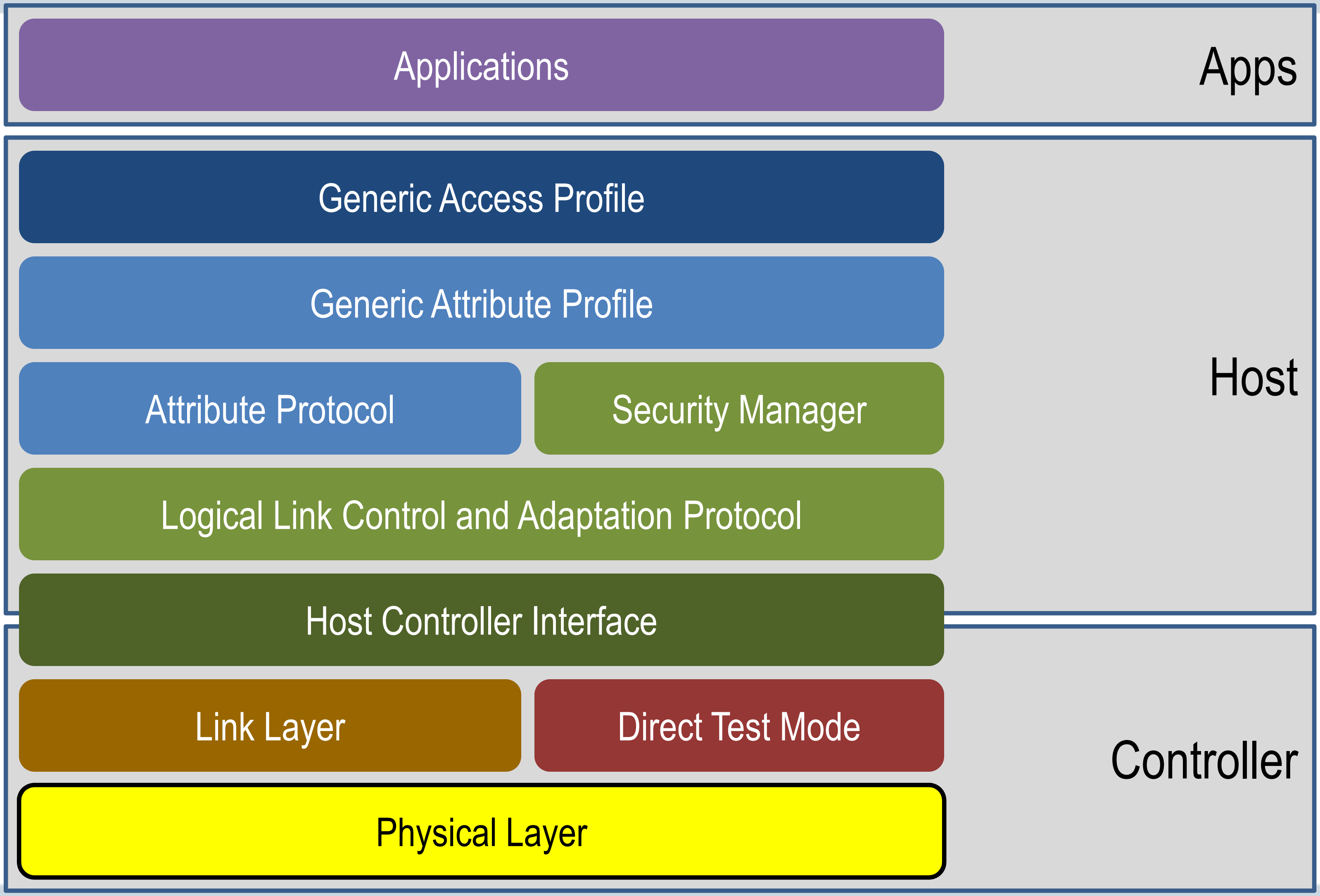
MORNING BREAK

return at 11:00

STACK ARCHITECTURE



PHYSICAL LAYER



PHYSICAL LAYER

Uses 2.4 GHz ISM Band

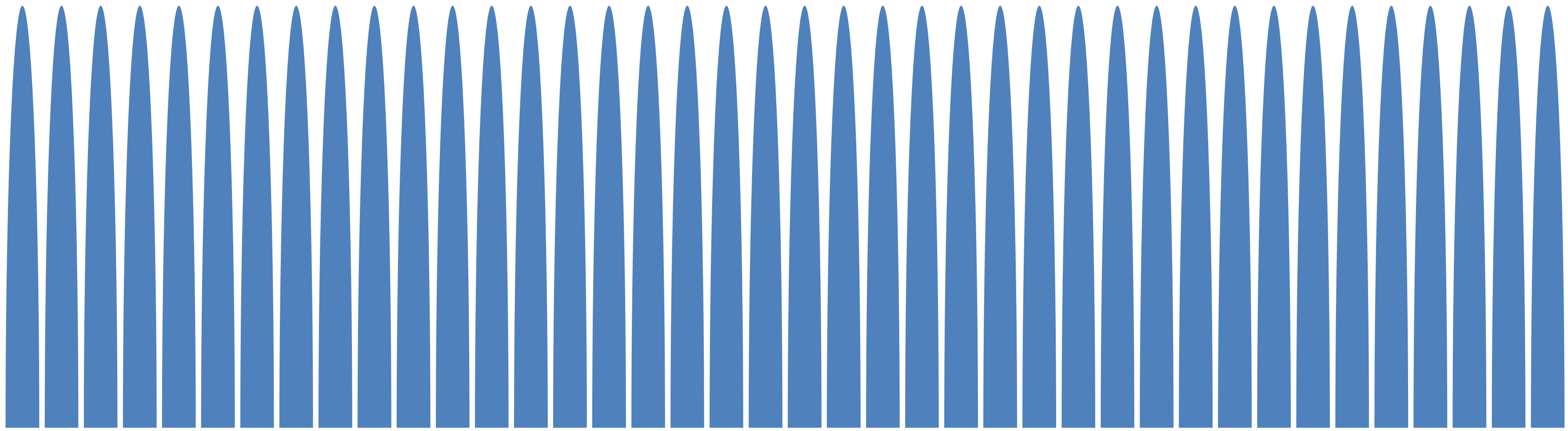
- Industrial Scientific Medical band
- License Free – with certain rules
- 2400 MHz to 2483.5 MHz

Used by many other standards

- IEEE 802.11, IEEE 802.15
- and many proprietary radios

40 PHYSICAL CHANNELS

$f = 2402 + 2 \cdot k$ MHz



RF	Frequency
0	2402 MHz
1	2404 MHz
2	2406 MHz
3	2408 MHz
4	2410 MHz
5	2412 MHz
6	2414 MHz
7	2416 MHz
8	2418 MHz
9	2420 MHz
10	2422 MHz
11	2424 MHz
12	2426 MHz
13	2428 MHz
14	2430 MHz
15	2432 MHz
16	2434 MHz
17	2436 MHz
18	2438 MHz
19	2440 MHz
20	2442 MHz
21	2444 MHz
22	2446 MHz
23	2448 MHz
24	2450 MHz
25	2452 MHz
26	2454 MHz
27	2456 MHz
28	2458 MHz
29	2460 MHz
30	2462 MHz
31	2464 MHz
32	2466 MHz
33	2468 MHz
34	2470 MHz
35	2472 MHz
36	2474 MHz
37	2476 MHz
38	2478 MHz
39	2480 MHz

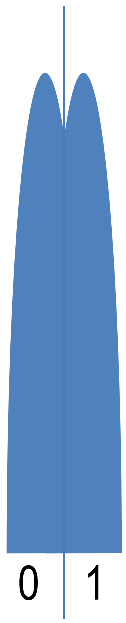
MODULATION

GFSK Modulation

bit period product BT = 0.5

modulation index = 0.5 ± 0.05

PHY Bandwidth = 1 million bits / seconds



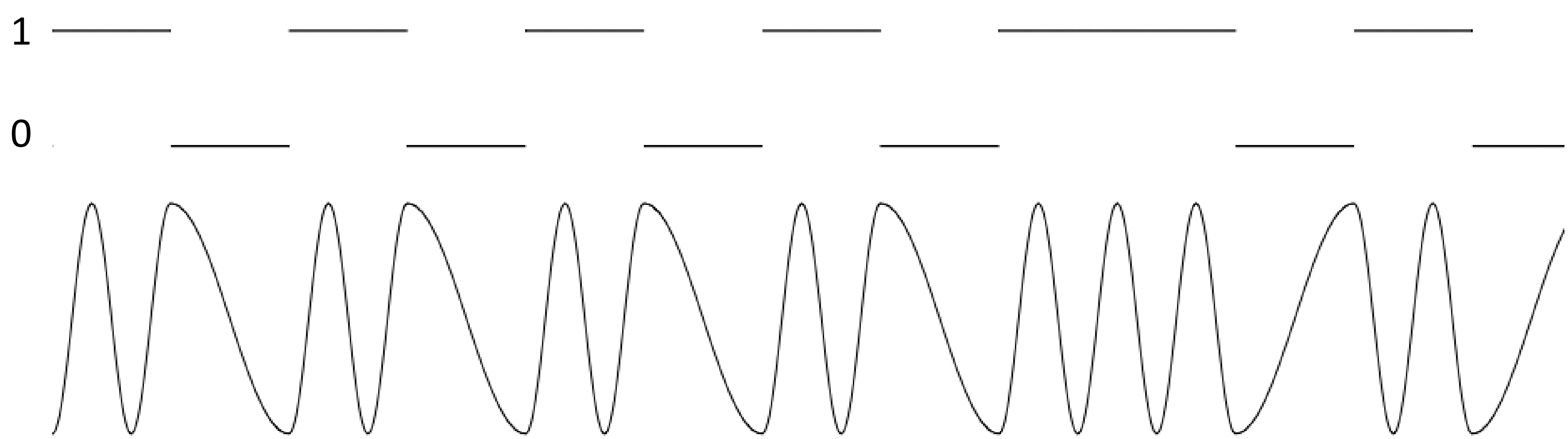
Why GFSK?

"pulse shaping"

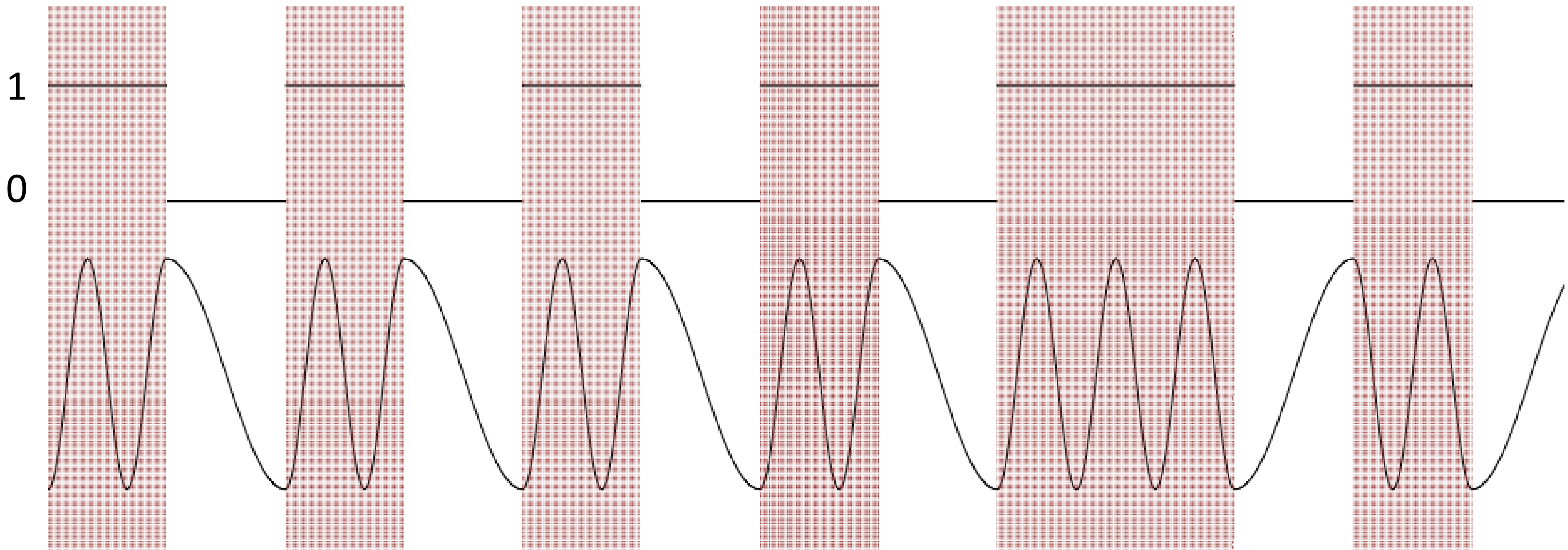
Gaussian filter smoothes transitions from zero to one

reduces spectral width

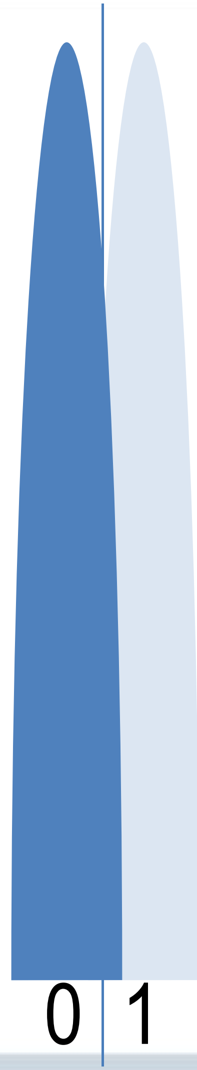
GFSK MODULATION EXAMPLE



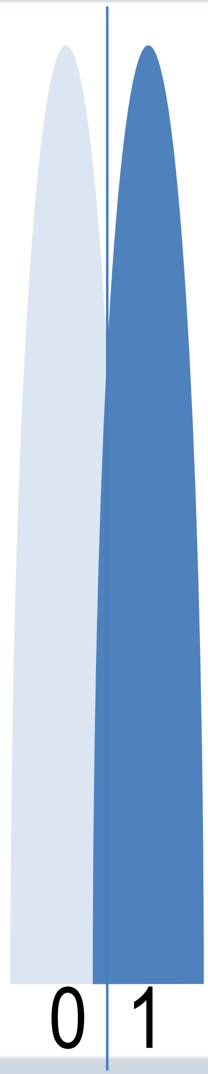
GFSK MODULATION EXAMPLE



WHEN TRANSMITTING A '0'



WHEN TRANSMITTING A '1'



POWER

Minimum Output Power

0.01 mW (-20 dBm)

Maximum Output Power

10 mW (10 dBm)

Receiver sensitivity < -70 dBm

When BER of 0.1%

RANGE

With Tx = 0 dBm and Rx = -70 dBm

Range ~ 30m

With Tx = 10 dBm and Rx = -90 dBm

Range > 100m

PHYSICAL LAYER SUMMARY

2.4 GHz GFSK

- Modulation Index = ~0.5

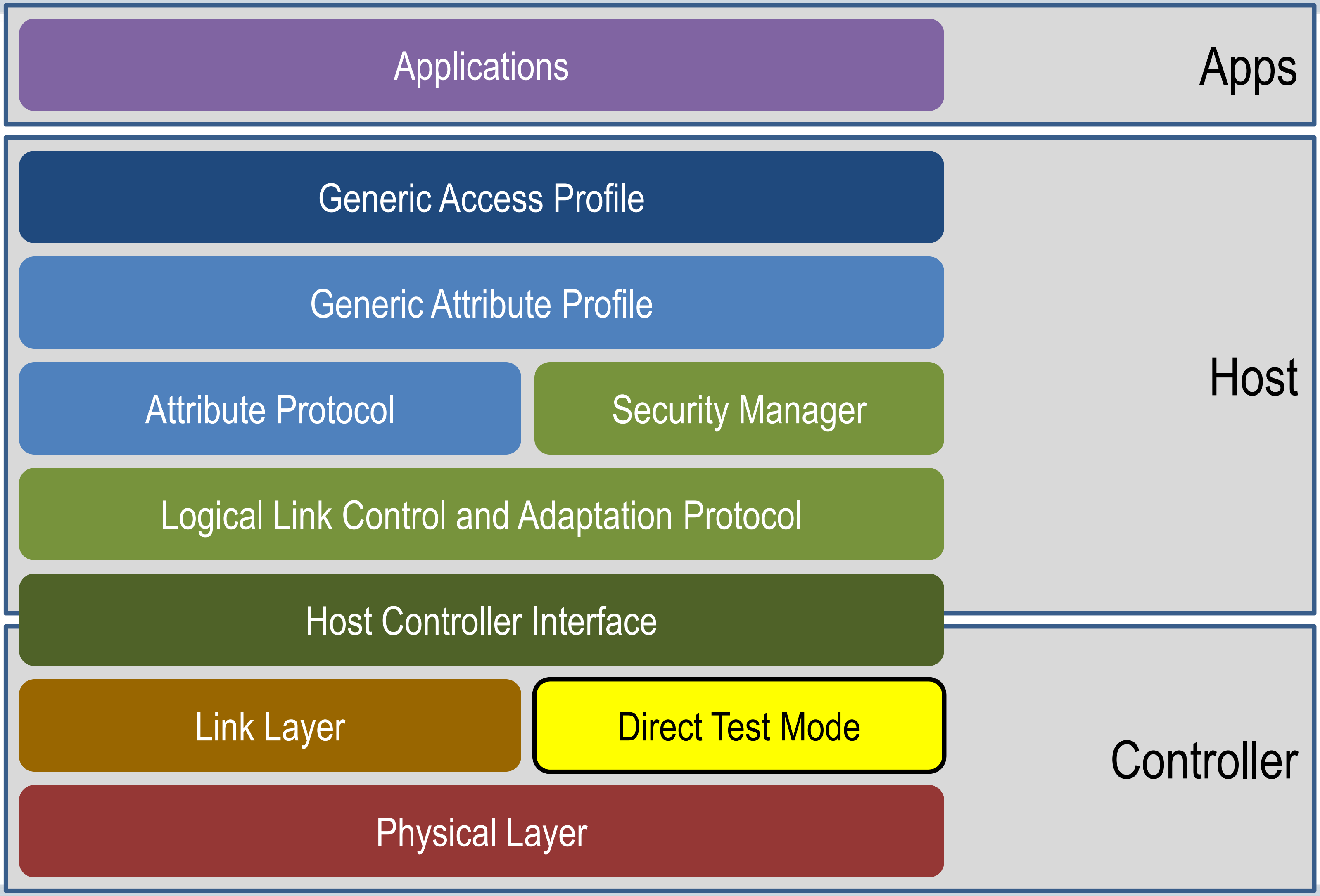
40 channels

- 2 MHz channel spacing
- 2402 MHz to 2480 MHz

Range

- 30m to further than 100 m

DIRECT TEST MODE



DIRECT TEST MODE

Used to test Physical Layer

- by commanding a device to transmit or receive test packets

Used for PHY RF qualification tests

- Can also be used to production line tests

Can run over standard HCI interface

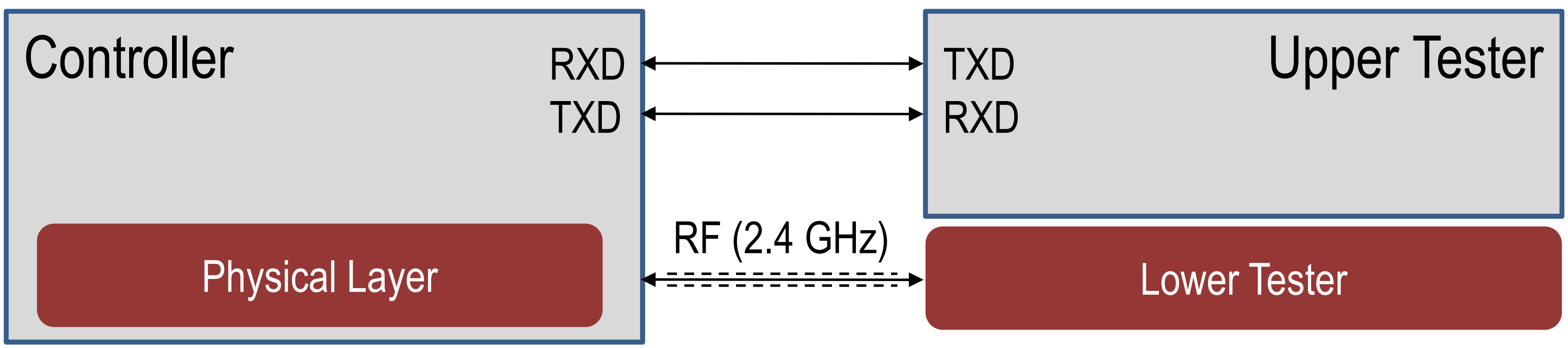
- can also use specific 2-wire UART interface

DIRECT TEST MODE

Can use 2-wire UART interface

- Uses very simple commands

Allows all controllers to be tested simply

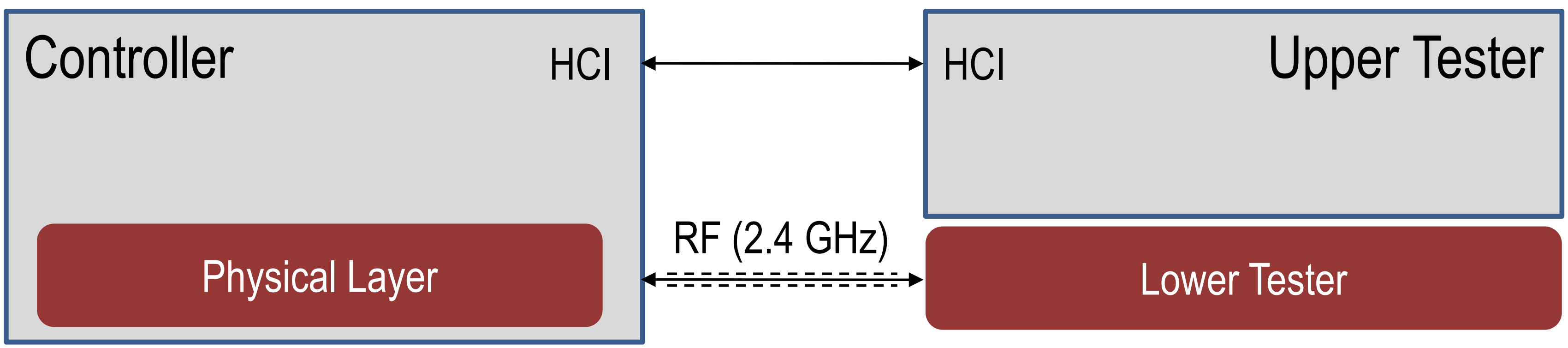


DIRECT TEST MODE

Can also use standard HCI

- Uses very simple HCI commands / events

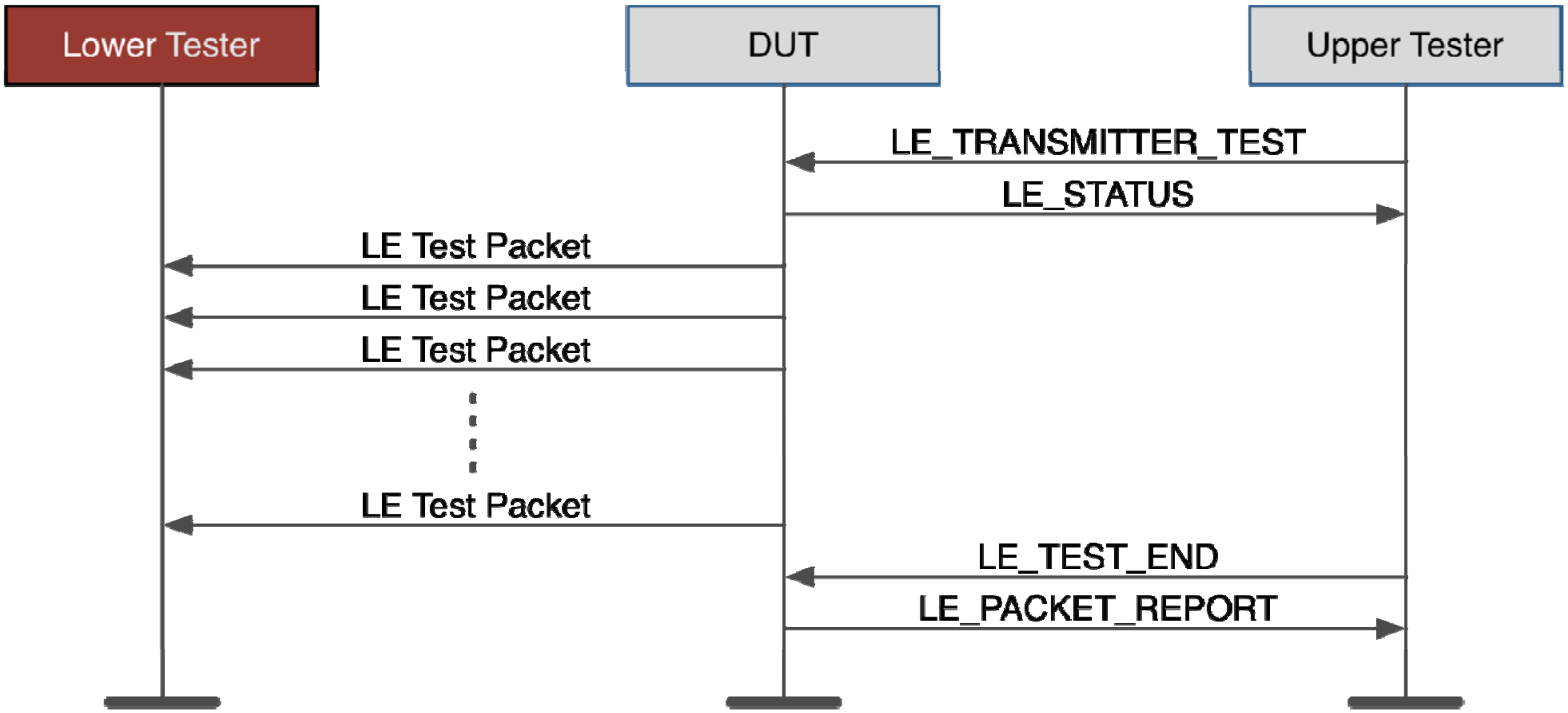
Can use any transport: 3-wire, UART, USB, SDIO, etc…



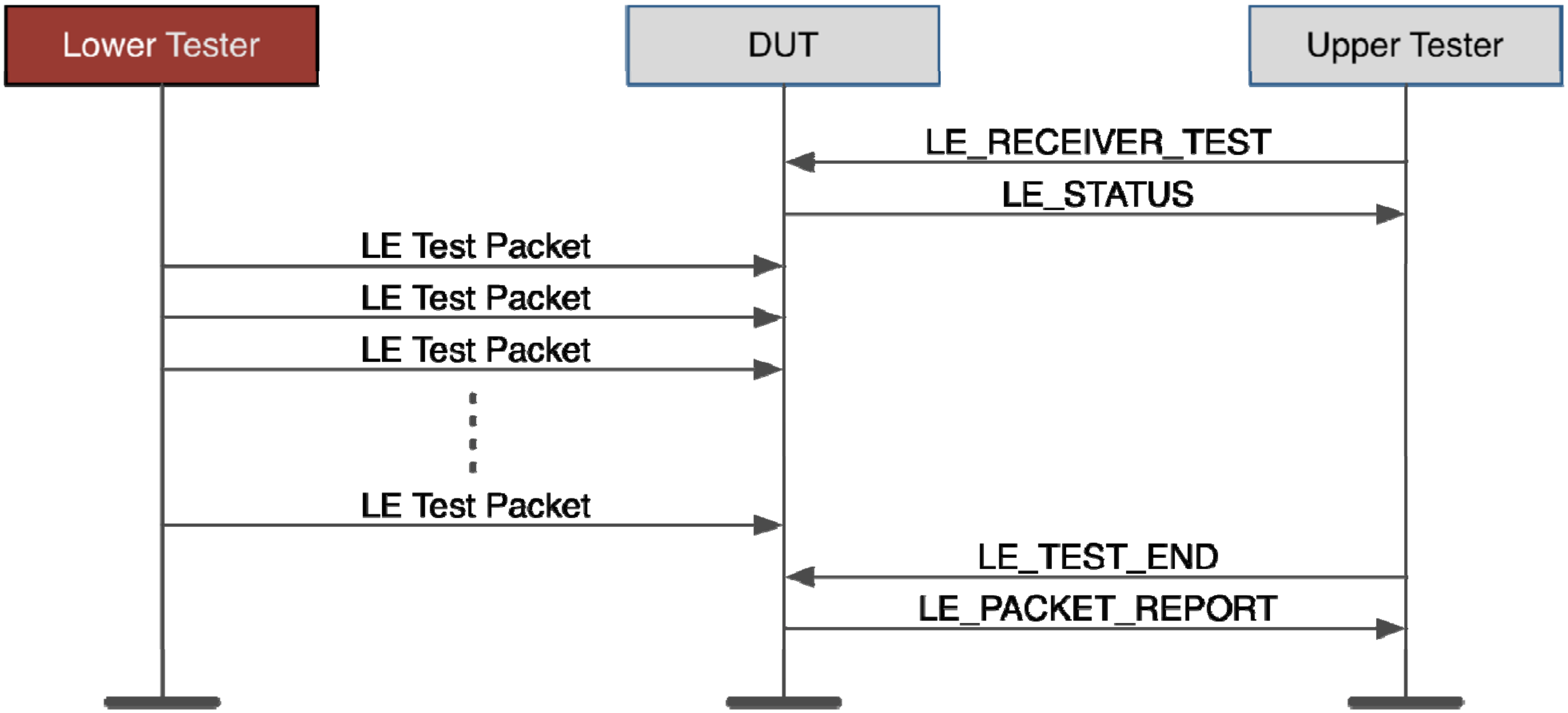
TEST SEQUENCES

RF Test Command / Event	Description
LE_RESET	Resets controller to known state
LE_TRANSMITTER_TEST	Commands DUT to transmit LE Test Packets until told to stop with LE_TEST_END
LE_RECEVIER_TEST	Commands DUT to receive LE Test Packets until told to stop with LE_TEST_END
LE_TEST_END	The test has ended, stop transmitting or receiving, send LE_PACKET_REPORT to Tester
LE_STATUS	Command received, starting test
LE_PACKET_REPORT	Command received, test ended, number of packets received

TRANSMITTER TEST



RECEIVER TEST



DIRECT TEST MODE SUMMARY

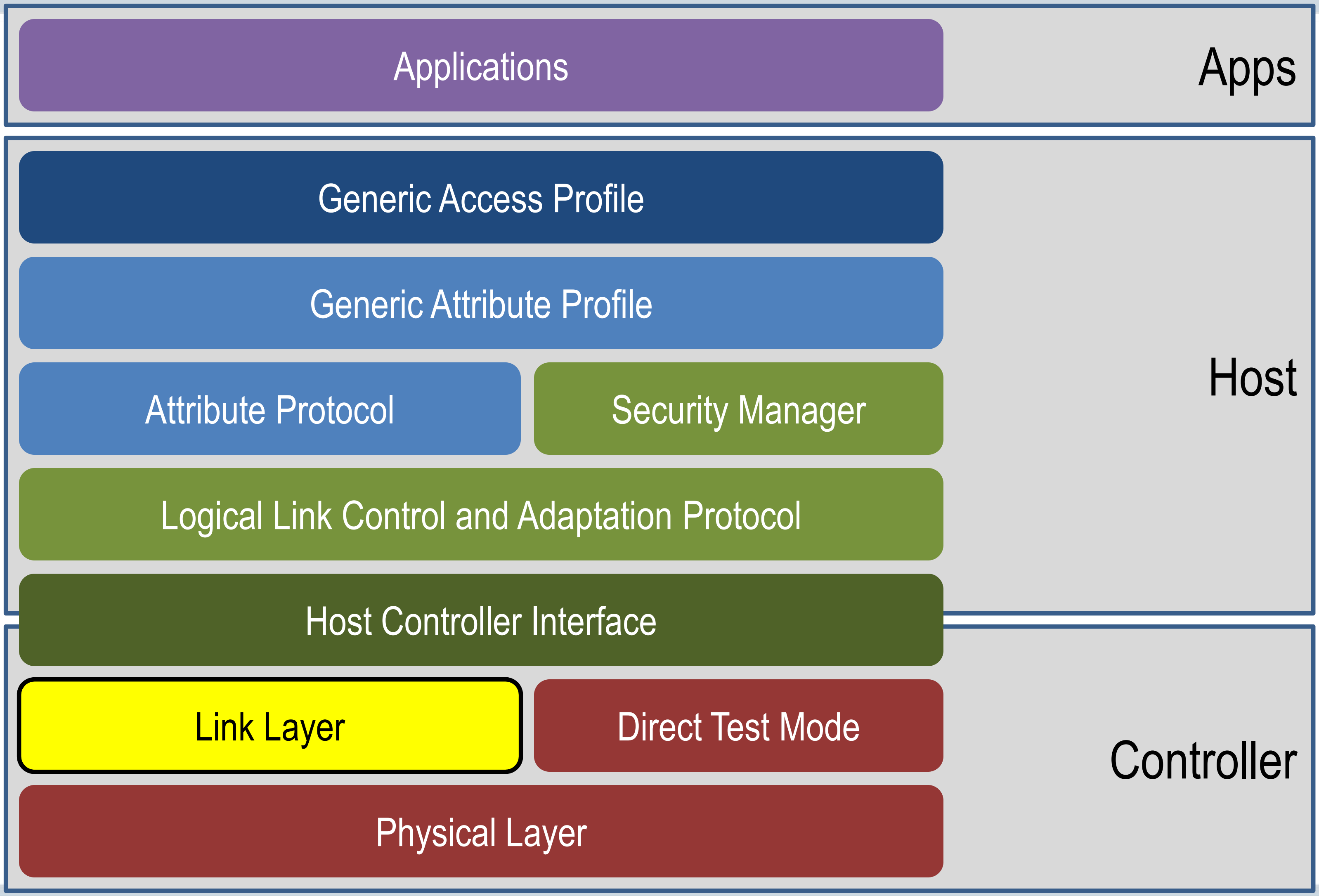
Used to test Physical Layer RF

- for qualification tests
- or for production line tests

Works over

- standard HCI transport layer
- special 2-wire transport

LINK LAYER


Applications
Apps
Generic Access Profile
Generic Attribute Profile
Attribute Protocol
Security Manager
Host
Logical Link Control and Adaptation Protocol
Host Controller Interface
Link Layer
Direct Test Mode
Controller
Physical Layer

LINK LAYER (LL)

Link Layer State Machine

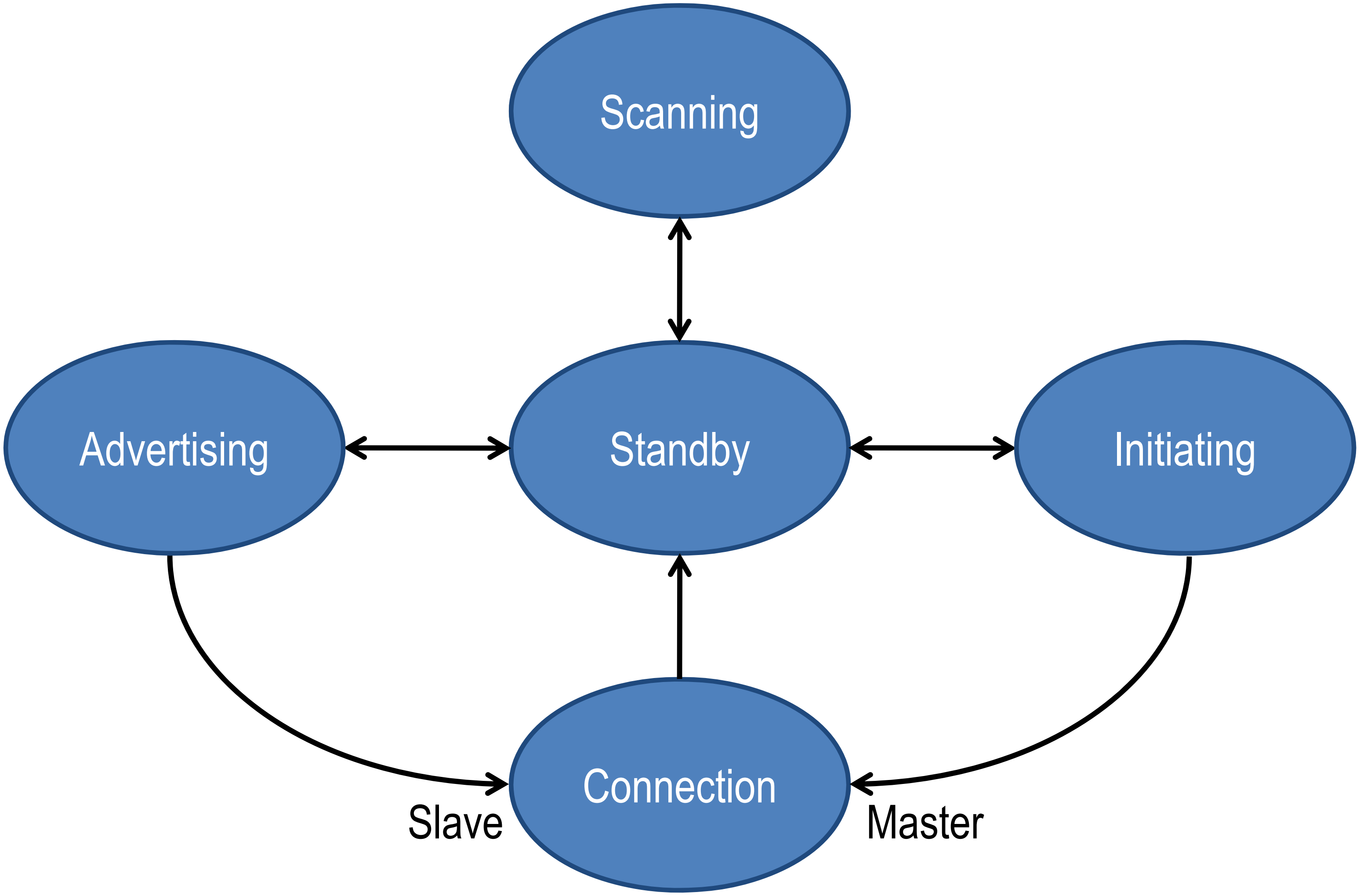
- can have multiple state machines active in device

Link Layer Channels

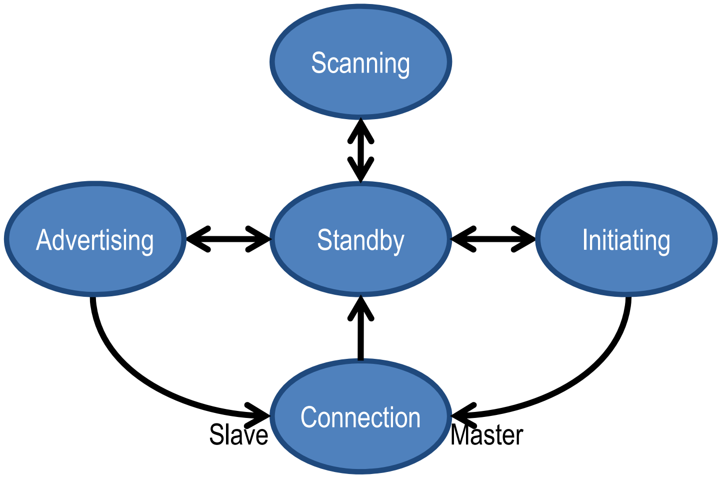
- Advertising Channels & Data Channels
- Advertising Packets & Data Packets

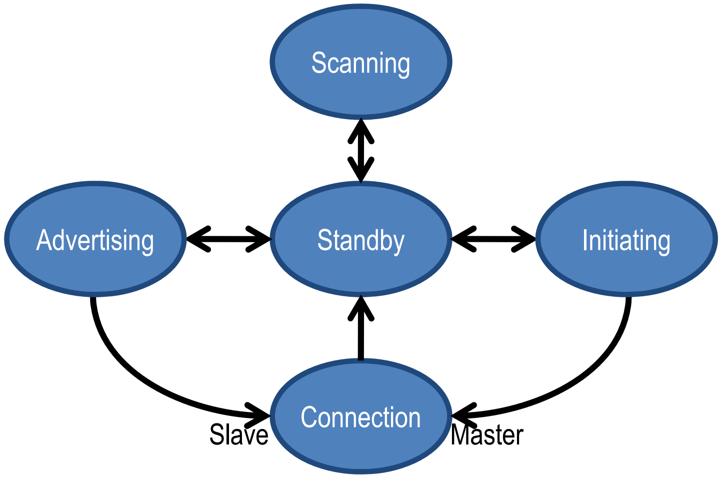
Link Layer Control Procedures

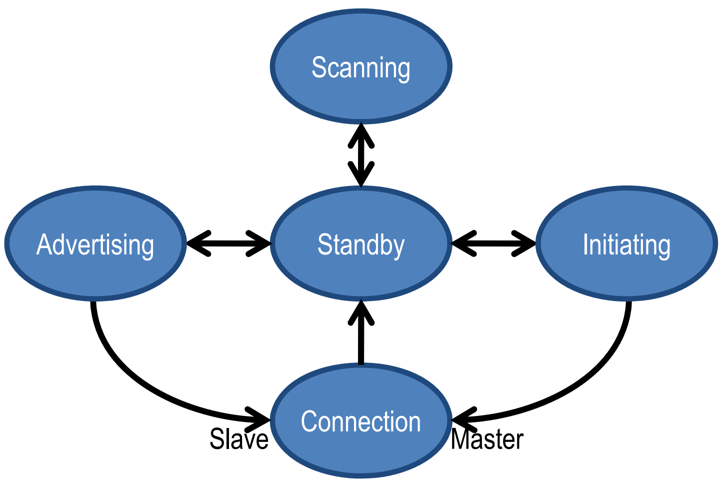
LINK LAYER STATE MACHINE

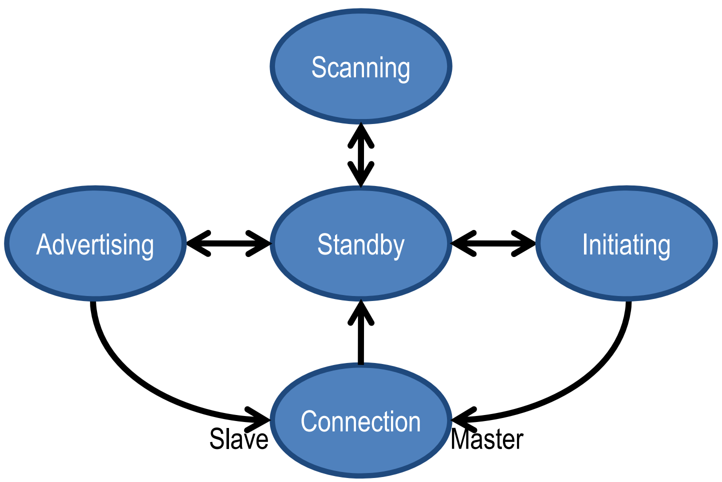


LINK LAYER STATE MACHINE

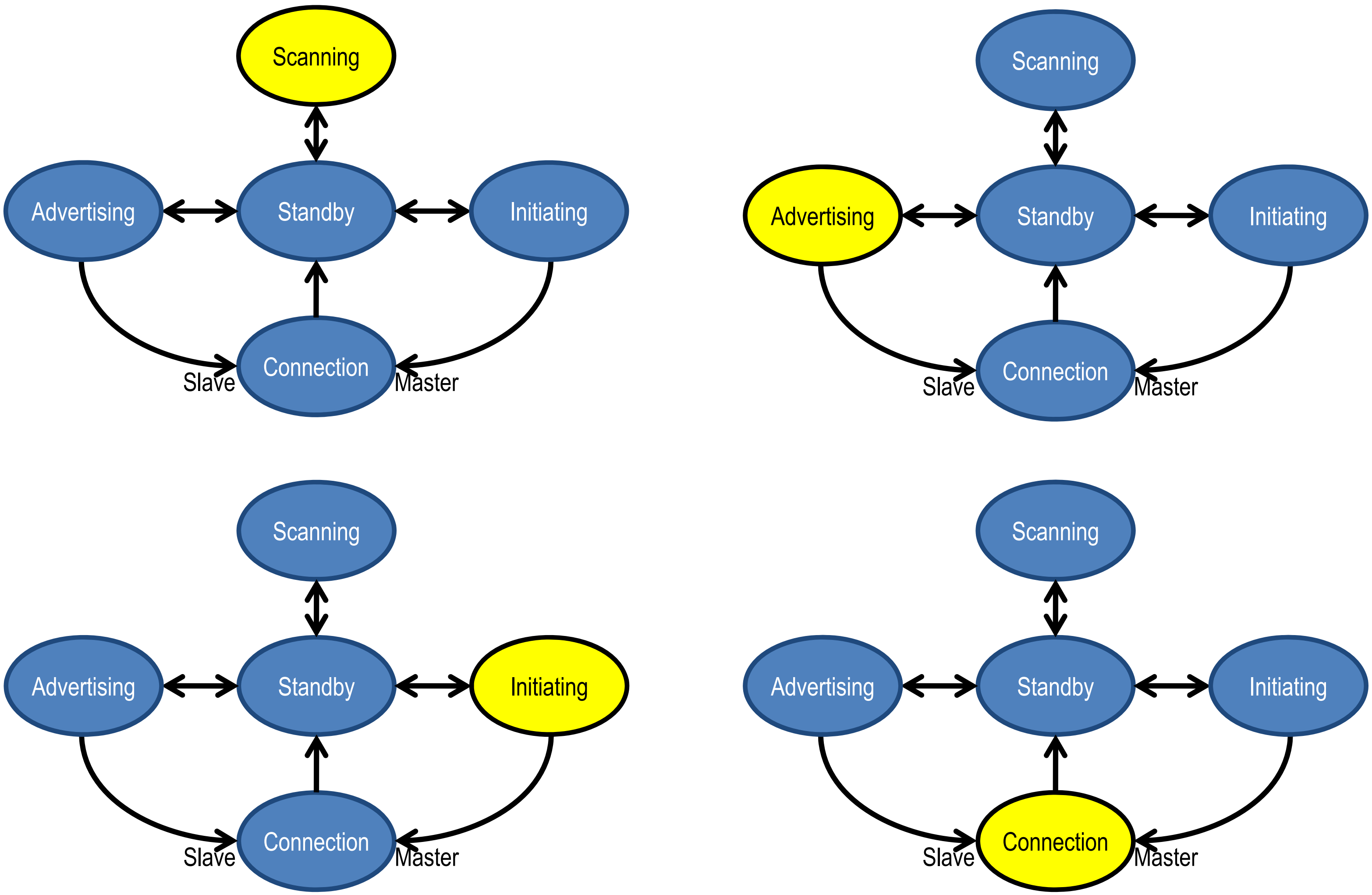








LINK LAYER STATE MACHINE



TWO TYPES OF CHANNELS

Advertising Channels

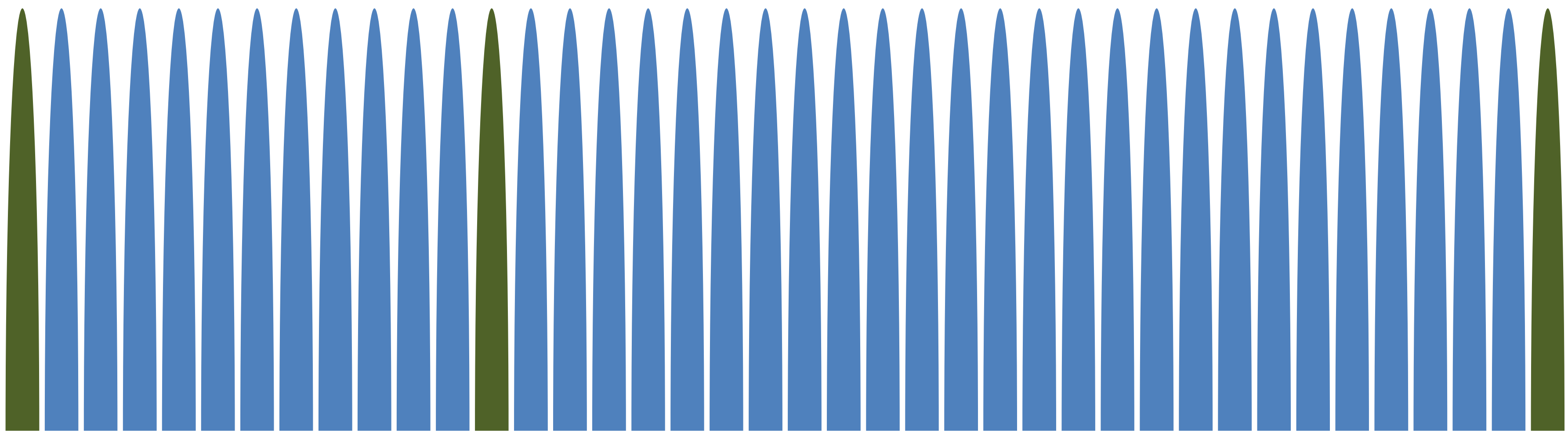
- Advertising Channel Packets
- Used for Discoverability / Connectability
- Used for Broadcasting / Observing

Data Channels

- Data Channel Packets
- Used to send application data in Connection

LINK LAYER CHANNELS

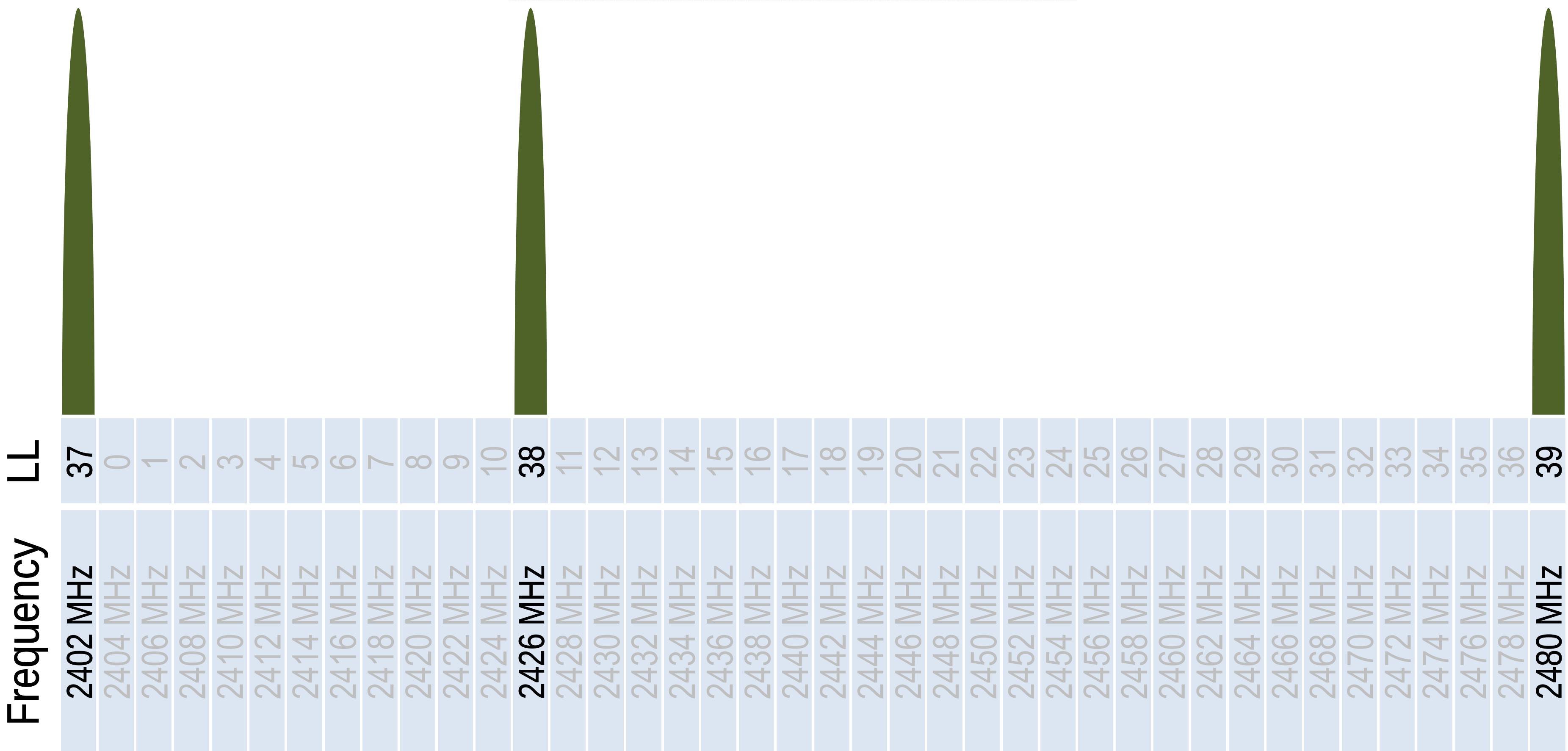
3 Advertising Channels and 37 Data Channels



LL	Frequency
37	2402 MHz
0	2404 MHz
1	2406 MHz
2	2408 MHz
3	2410 MHz
4	2412 MHz
5	2414 MHz
6	2416 MHz
7	2418 MHz
8	2420 MHz
9	2422 MHz
10	2424 MHz
38	2426 MHz
11	2428 MHz
12	2430 MHz
13	2432 MHz
14	2434 MHz
15	2436 MHz
16	2438 MHz
17	2440 MHz
18	2442 MHz
19	2444 MHz
20	2446 MHz
21	2448 MHz
22	2450 MHz
23	2452 MHz
24	2454 MHz
25	2456 MHz
26	2458 MHz
27	2460 MHz
28	2462 MHz
29	2464 MHz
30	2466 MHz
31	2468 MHz
32	2470 MHz
33	2472 MHz
34	2474 MHz
35	2476 MHz
36	2478 MHz
39	2480 MHz

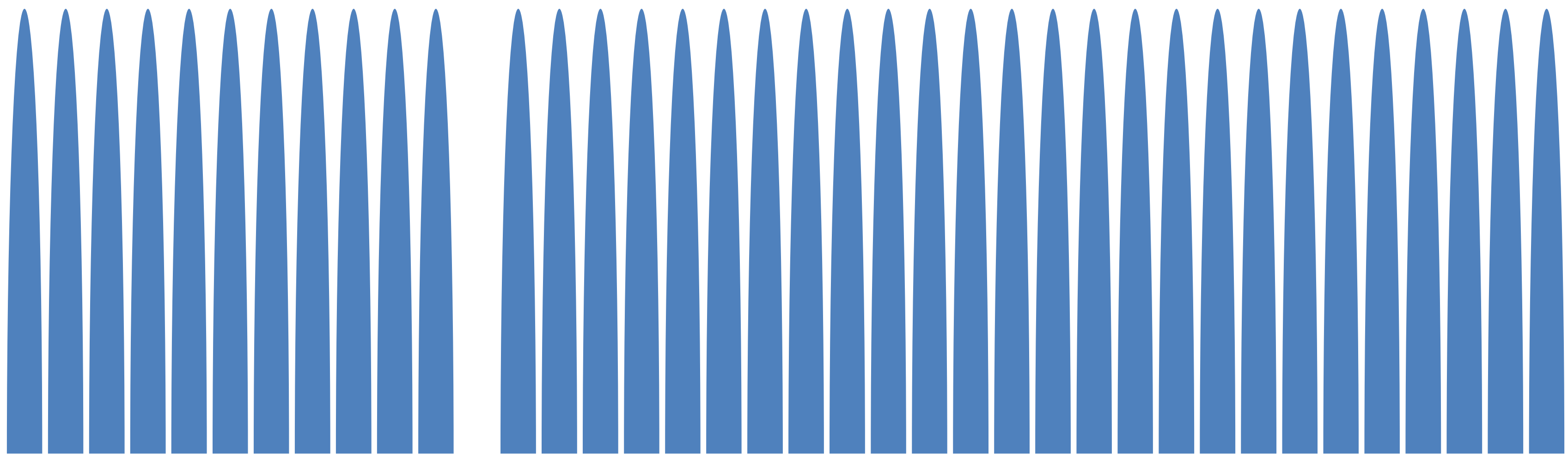
LINK LAYER CHANNELS

3 Advertising Channels



LINK LAYER CHANNELS

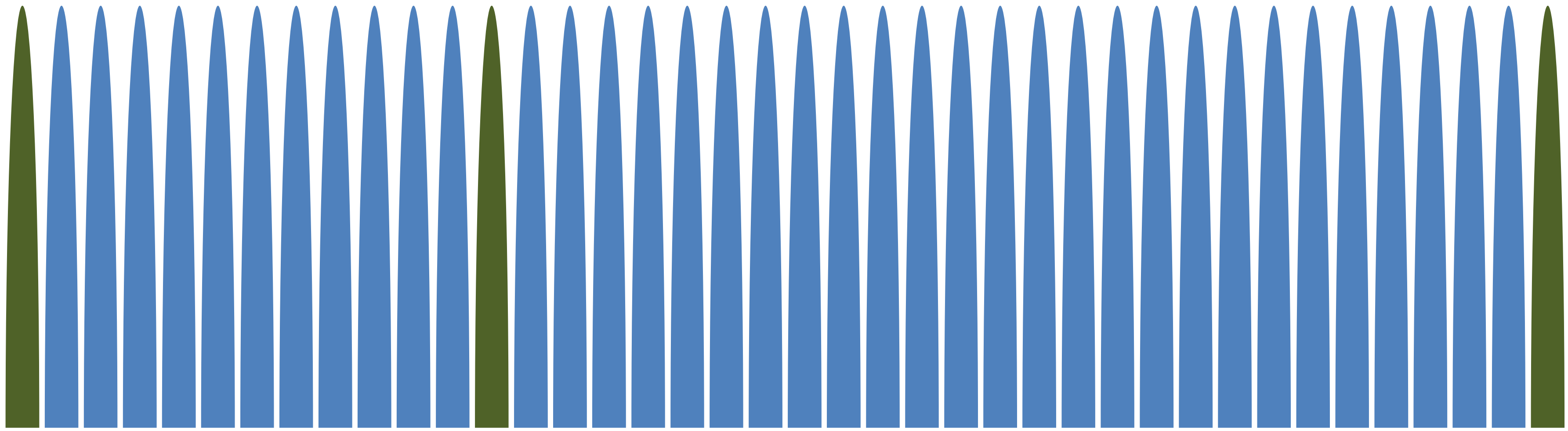
37 Data Channels



LL	Frequency
37	2402 MHz
0	2404 MHz
1	2406 MHz
2	2408 MHz
3	2410 MHz
4	2412 MHz
5	2414 MHz
6	2416 MHz
7	2418 MHz
8	2420 MHz
9	2422 MHz
10	2424 MHz
38	2426 MHz
11	2428 MHz
12	2430 MHz
13	2432 MHz
14	2434 MHz
15	2436 MHz
16	2438 MHz
17	2440 MHz
18	2442 MHz
19	2444 MHz
20	2446 MHz
21	2448 MHz
22	2450 MHz
23	2452 MHz
24	2454 MHz
25	2456 MHz
26	2458 MHz
27	2460 MHz
28	2462 MHz
29	2464 MHz
30	2466 MHz
31	2468 MHz
32	2470 MHz
33	2472 MHz
34	2474 MHz
35	2476 MHz
36	2478 MHz
39	2480 MHz

LINK LAYER CHANNELS

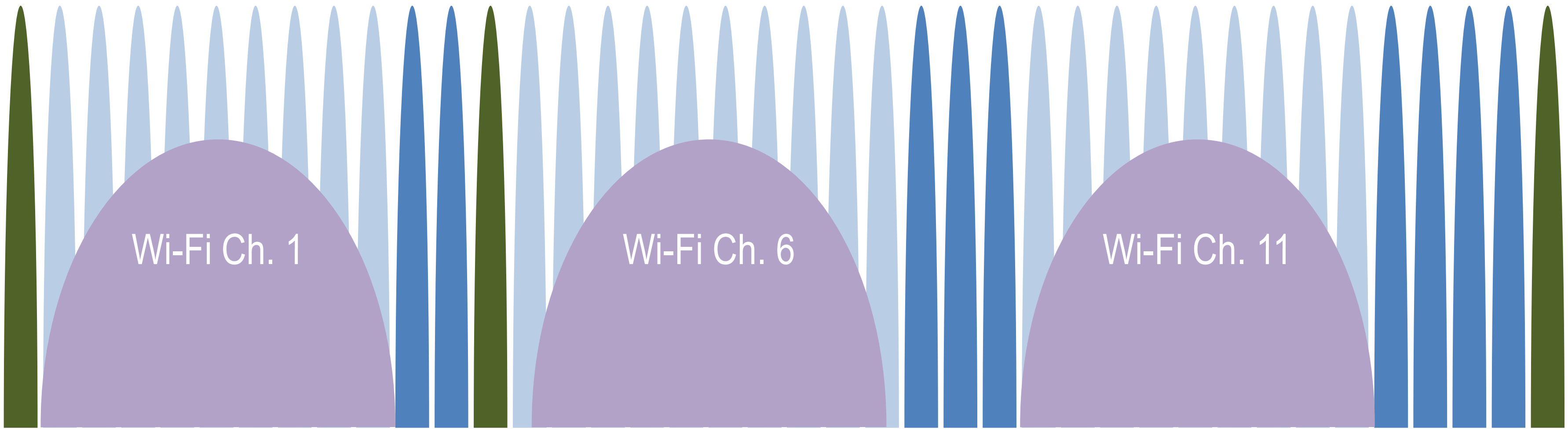
3 Advertising Channels and 37 Data Channels



LL	Frequency
37	2402 MHz
0	2404 MHz
1	2406 MHz
2	2408 MHz
3	2410 MHz
4	2412 MHz
5	2414 MHz
6	2416 MHz
7	2418 MHz
8	2420 MHz
9	2422 MHz
10	2424 MHz
38	2426 MHz
11	2428 MHz
12	2430 MHz
13	2432 MHz
14	2434 MHz
15	2436 MHz
16	2438 MHz
17	2440 MHz
18	2442 MHz
19	2444 MHz
20	2446 MHz
21	2448 MHz
22	2450 MHz
23	2452 MHz
24	2454 MHz
25	2456 MHz
26	2458 MHz
27	2460 MHz
28	2462 MHz
29	2464 MHz
30	2466 MHz
31	2468 MHz
32	2470 MHz
33	2472 MHz
34	2474 MHz
35	2476 MHz
36	2478 MHz
39	2480 MHz

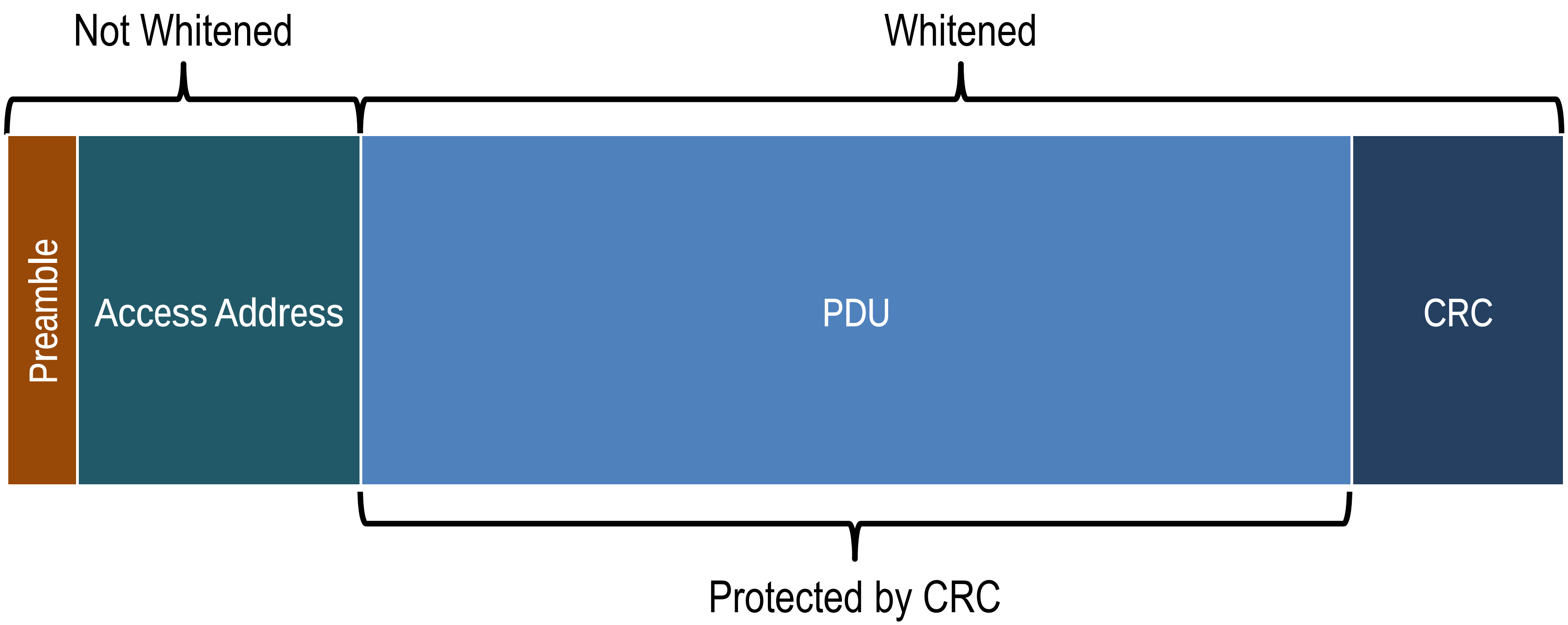
LINK LAYER CHANNELS

9 LL Data Channels still available

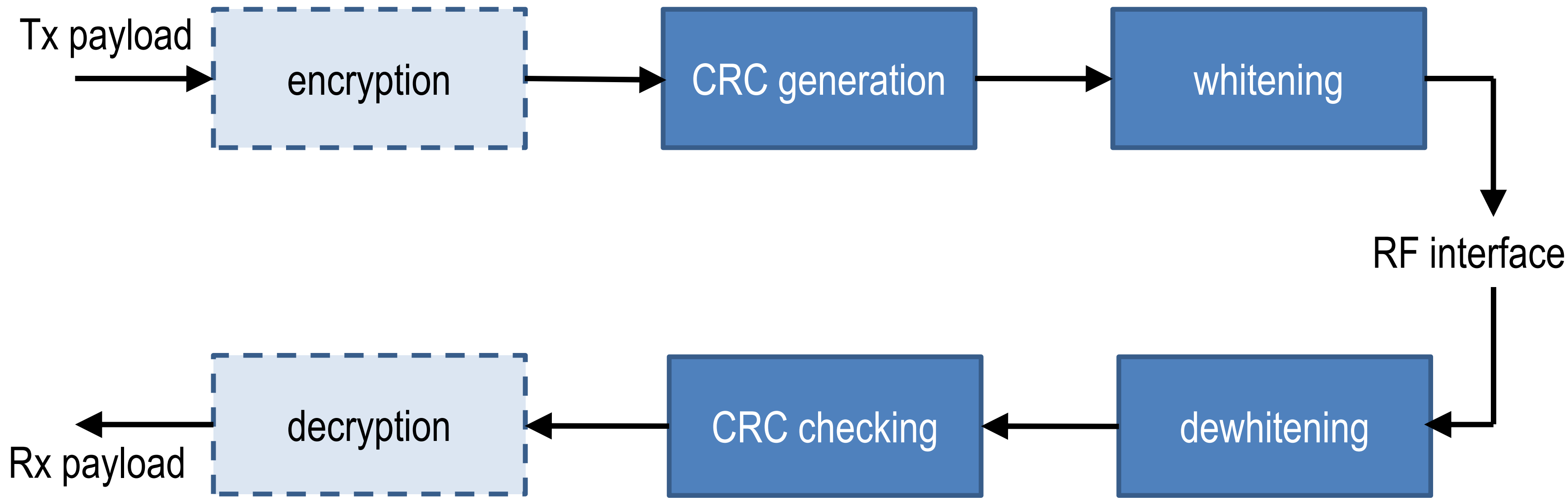


LL	Frequency
37	2402 MHz
0	2404 MHz
1	2406 MHz
2	2408 MHz
3	2410 MHz
4	2412 MHz
5	2414 MHz
6	2416 MHz
7	2418 MHz
8	2420 MHz
9	2422 MHz
10	2424 MHz
38	2426 MHz
11	2428 MHz
12	2430 MHz
13	2432 MHz
14	2434 MHz
15	2436 MHz
16	2438 MHz
17	2440 MHz
18	2442 MHz
19	2444 MHz
20	2446 MHz
21	2448 MHz
22	2450 MHz
23	2452 MHz
24	2454 MHz
25	2456 MHz
26	2458 MHz
27	2460 MHz
28	2462 MHz
29	2464 MHz
30	2466 MHz
31	2468 MHz
32	2470 MHz
33	2472 MHz
34	2474 MHz
35	2476 MHz
36	2478 MHz
39	2480 MHz

ONE PACKET FORMAT



BIT STREAM PROCESSING

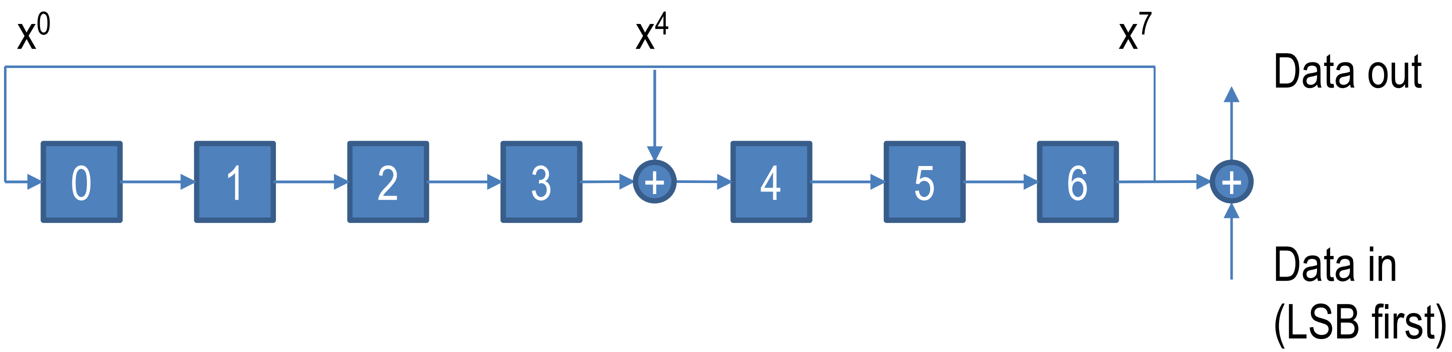


WHITENING

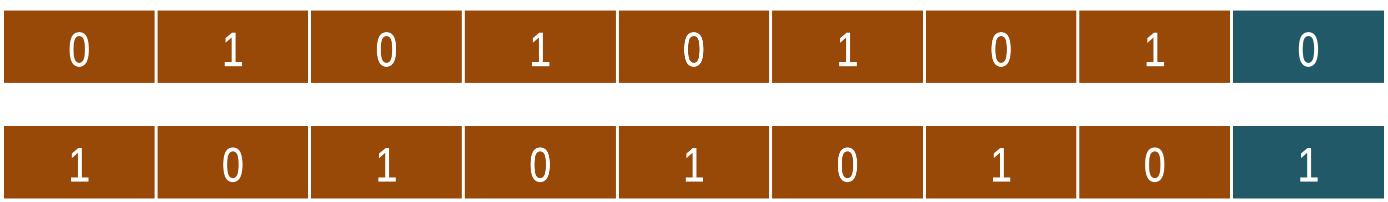
Avoids long sequences of ‘0’s or ‘1’s

Uses same 7 bit LFSR as BR/EDR

- LFSR defined as: $x^7+x^4+x^0$
- initialized with LL Channel Index



PREAMBLE

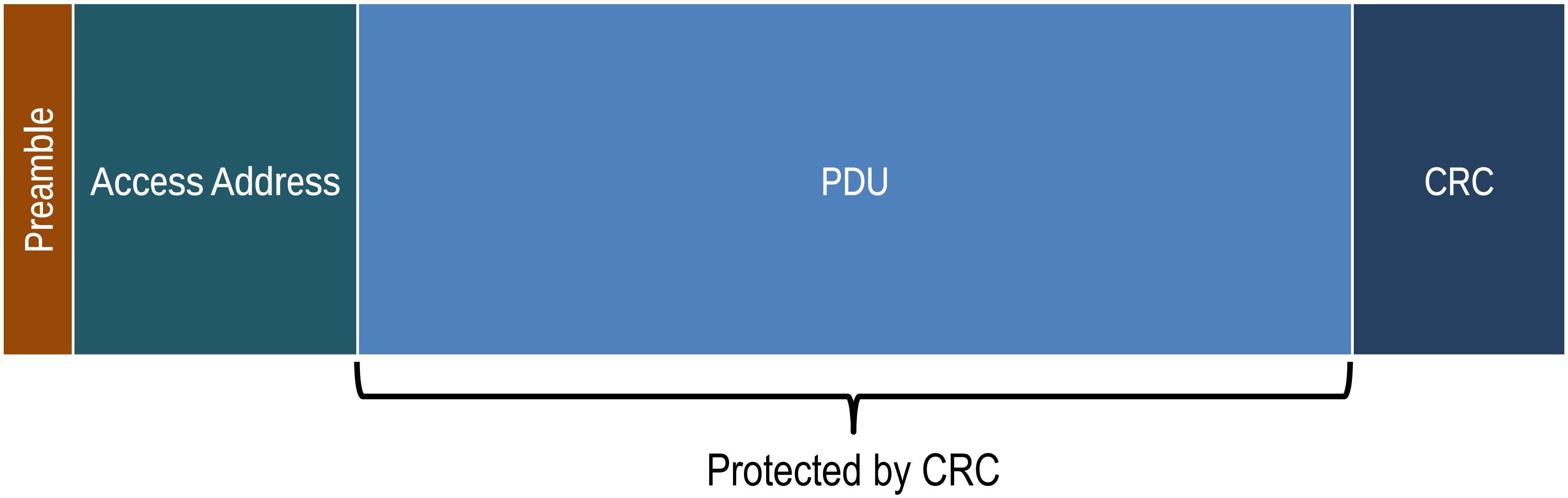


ACCESS ADDRESS

Access Address (32 bits)

- Advertising Packets = 0x8E89BED6
- Data Packets = random value per connection

CRC

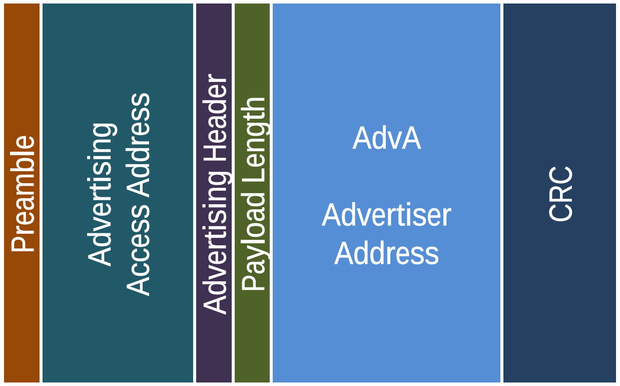


Cyclic Redundancy Check (24 bits)

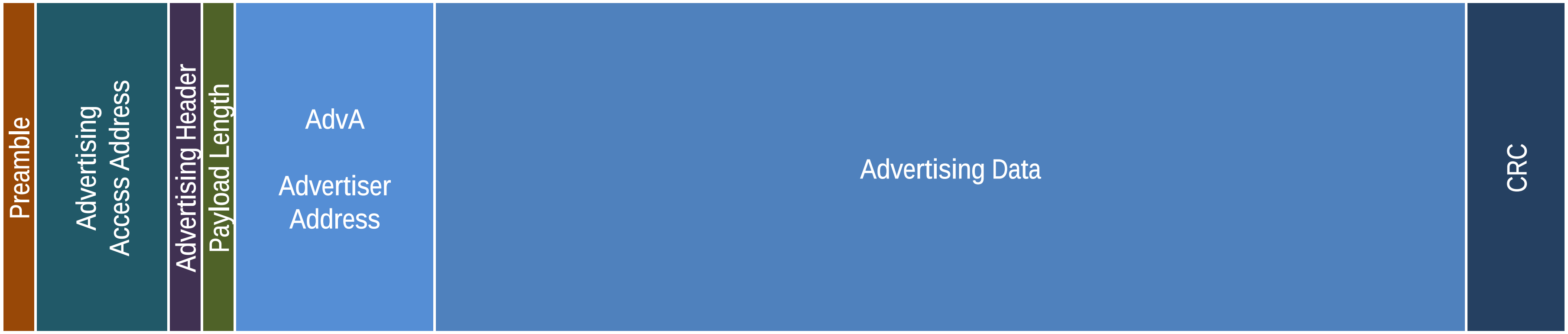
$x^{24}+x^{10}+x^{9}+x^{6}+x^{4}+x^{3}+x^{1}+x^{0}$

Detects all 1,2,3,4,5,odd bit errors in PDU

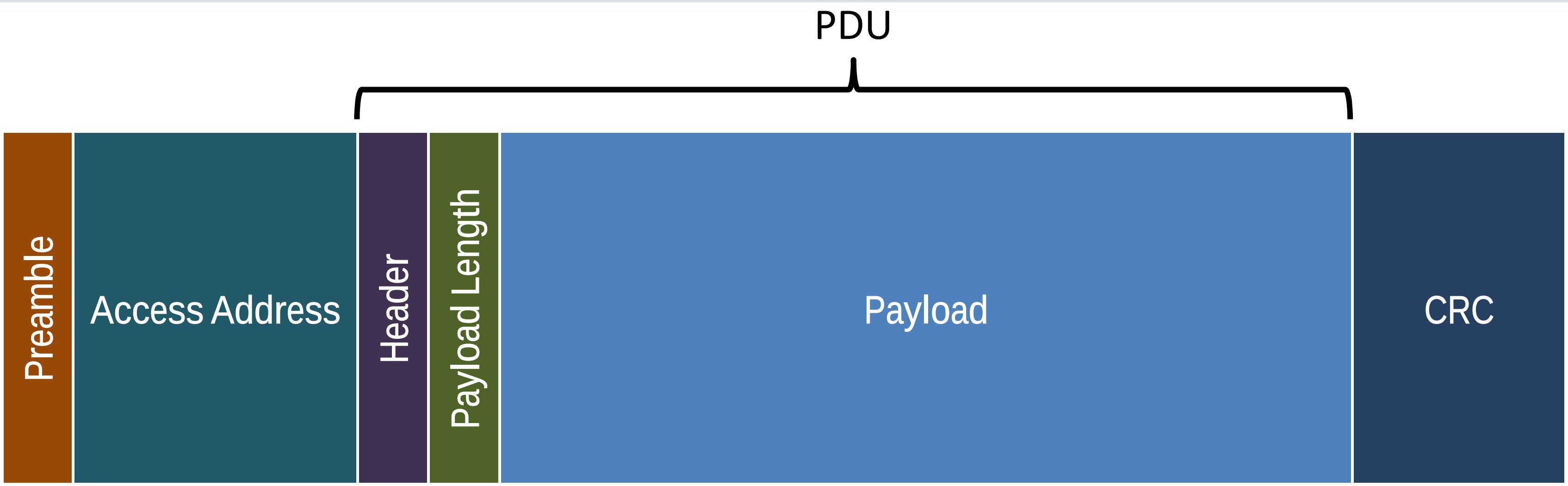
ADVERTISING PACKET



0 to 31 bytes of Advertising data



PDU HEADERS



Advertising channel PDU Header / Payload Length

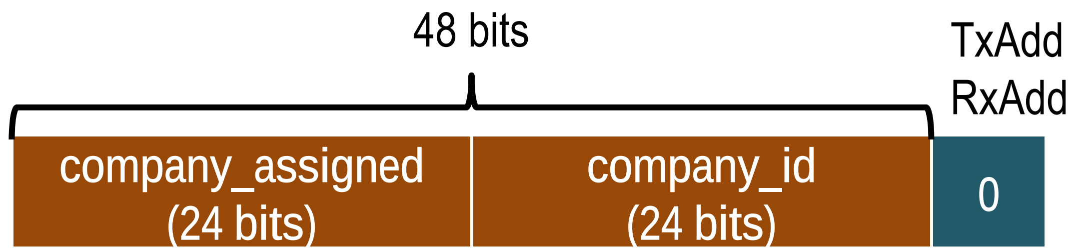
PDU Type	RFU	TxAdd	RxAdd
Length (6 – 37)		RFU	

ADVERTISING PACKET PDU TYPES

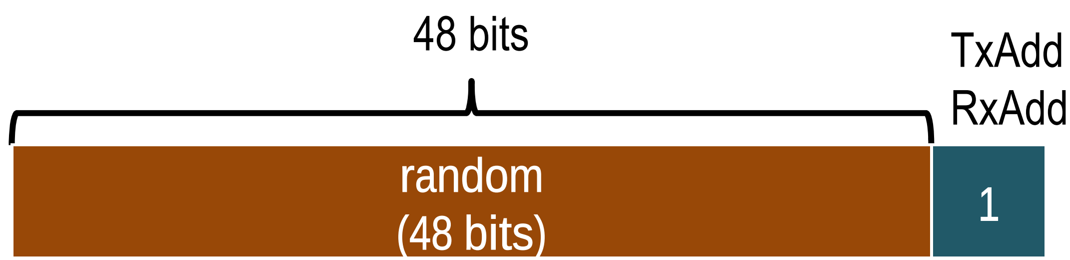
PDU Type	Packet Name	Description
0000	ADV_IND	connectable undirected advertising event
0001	ADV_DIRECT_IND	connectable directed advertising event
0010	ADV_NONCONN_IND	non-connectable undirected advertising event
0011	SCAN_REQ	Scanner wants information from Advertiser
0100	SCAN_RSP	Advertiser gives more information to Scanner
0101	CONNECT_REQ	Initiator wants to connect to Advertiser
0110	ADV_DISCOVER_IND	non-connectable undirected advertising event

DEVICE ADDRESS

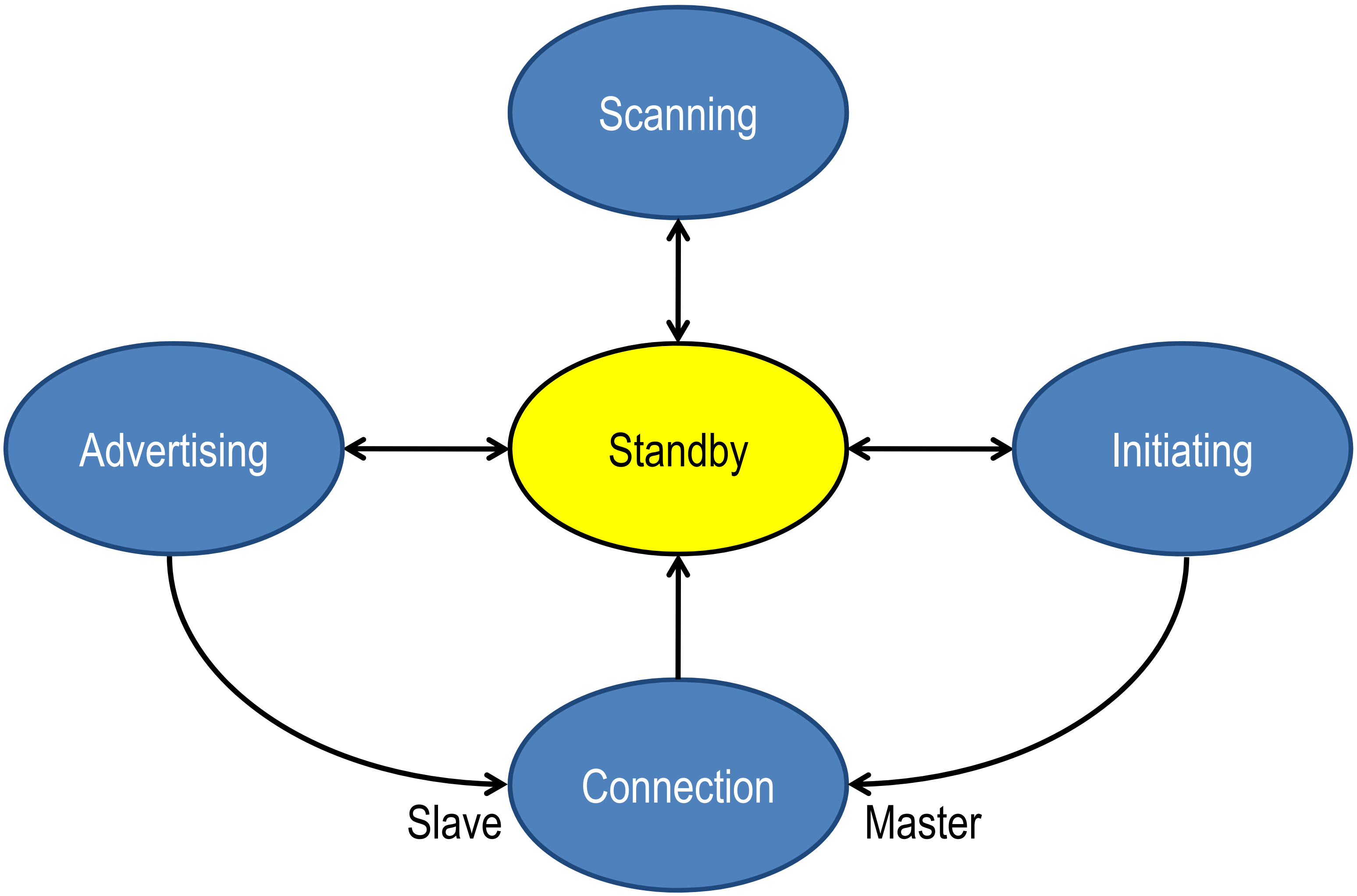
Public Device Address



Random Device Address

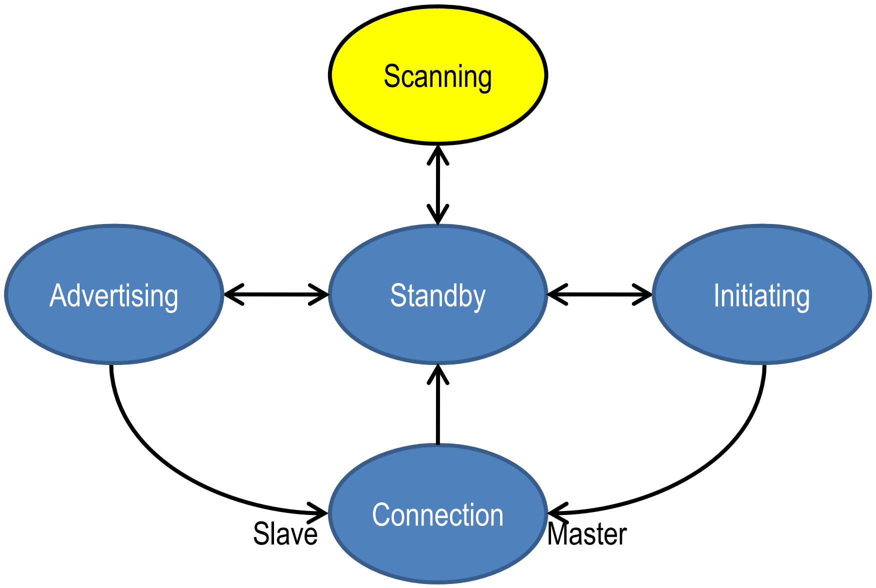


STANDBY

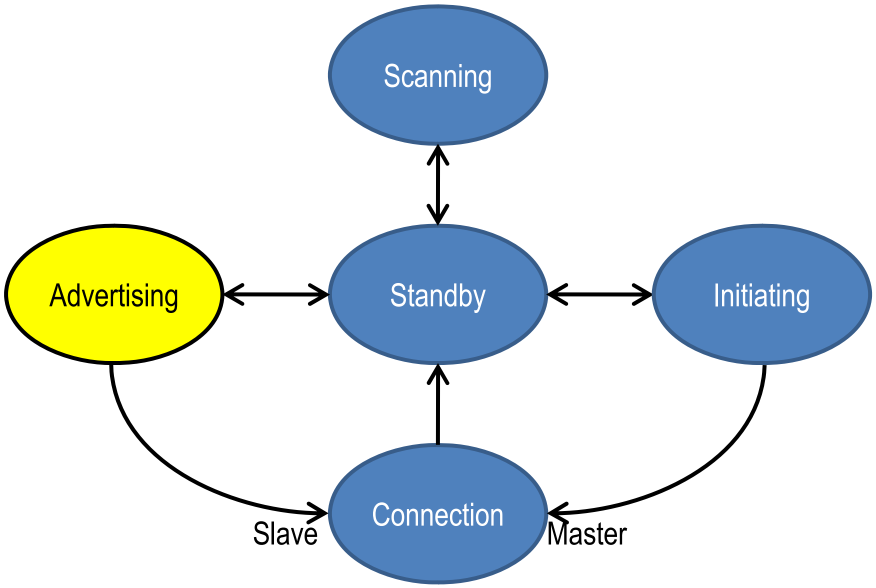

Scanning
Advertising
Standby
Initiating
Connection
Slave
Master

ADVERTISING

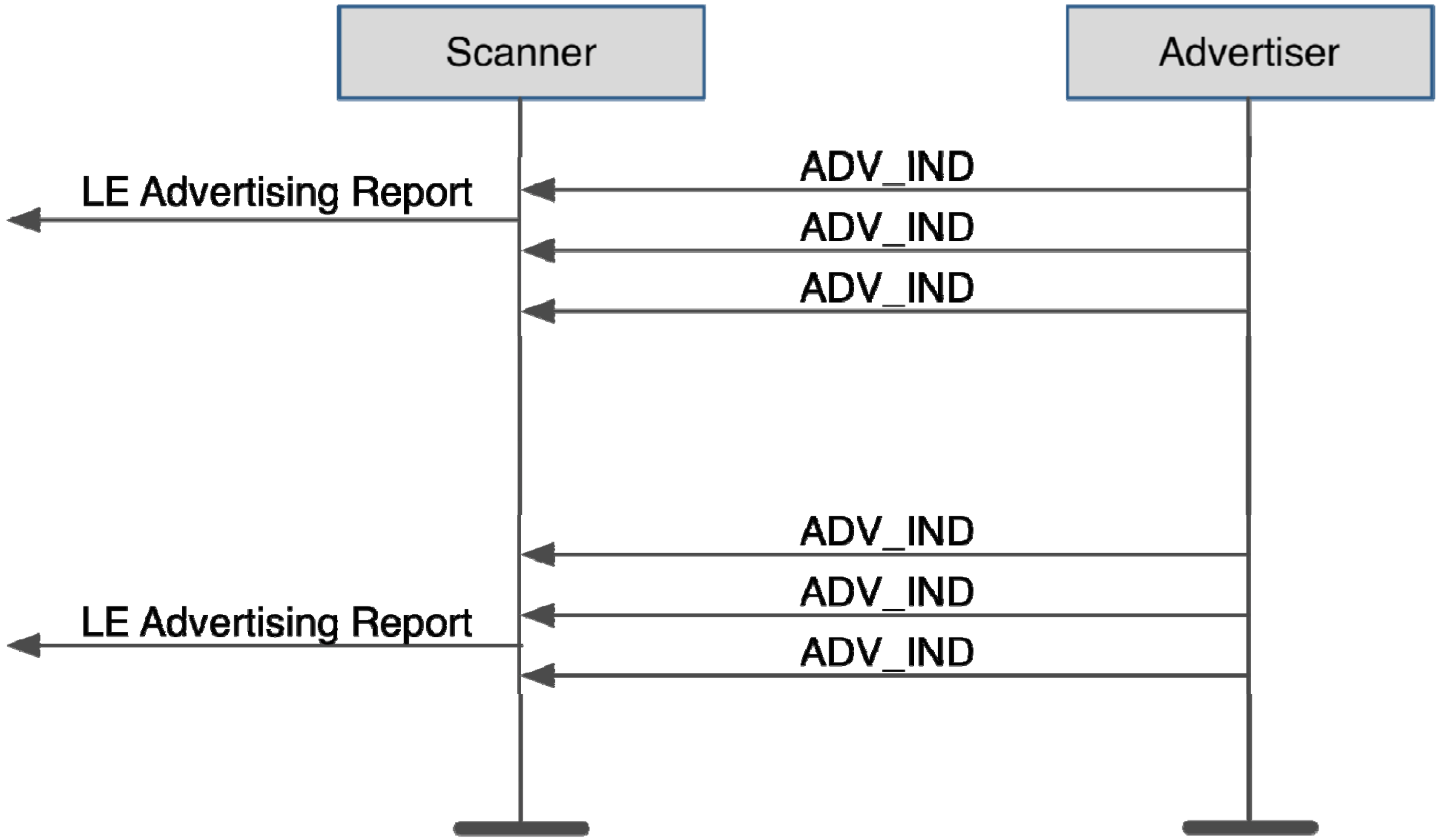
Passive Scanner



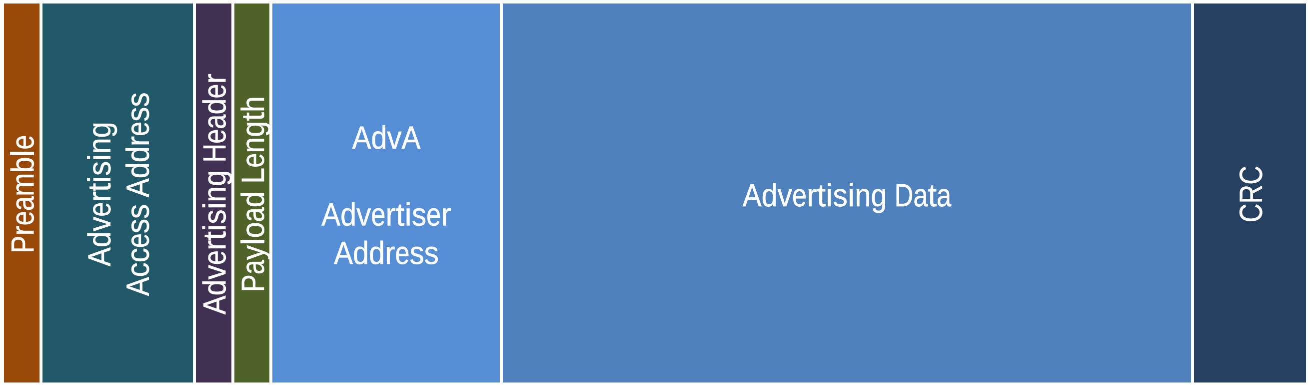
Advertiser



PASSIVE SCANNING



ADVERTISING DATA



Advertising Data

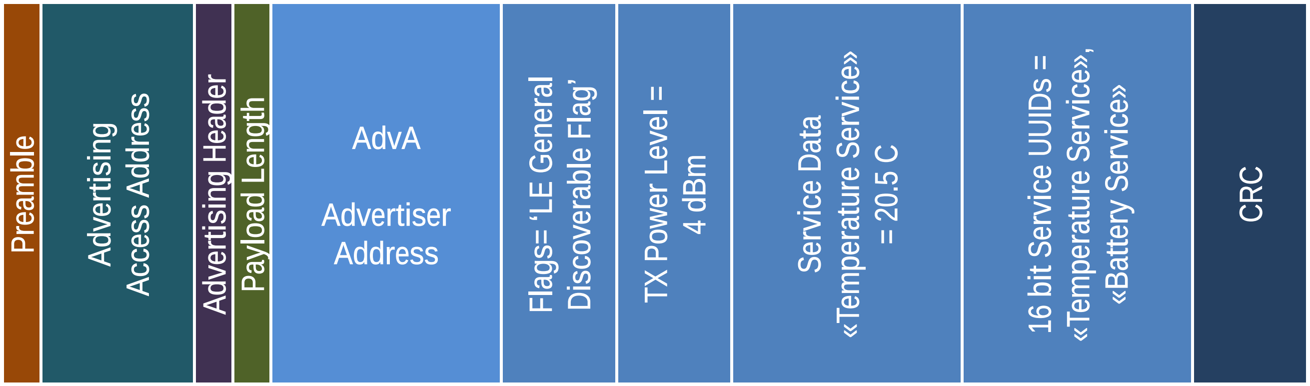
“I’m generally discoverable”,

transmitting at +4 dBm,

the temperature is 20.5 C,

I support Battery and Temperature services

ADVERTISING DATA



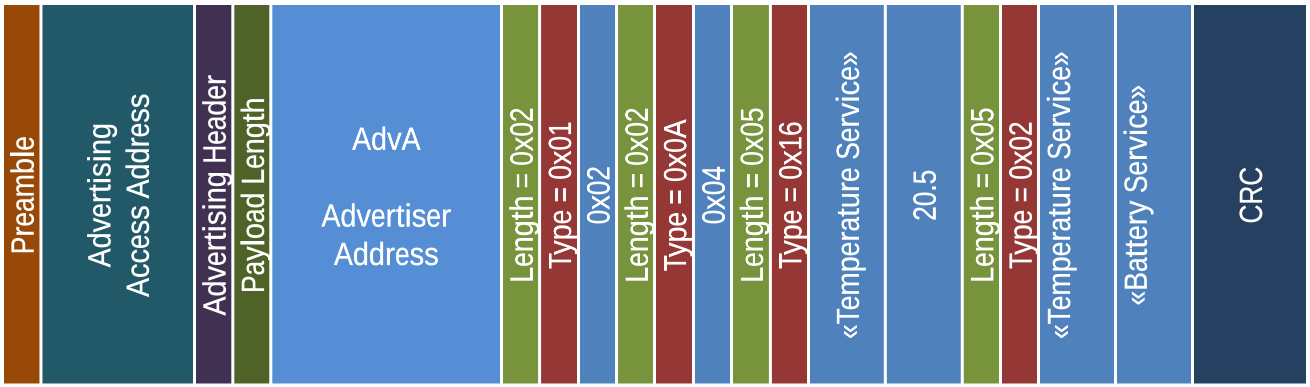
LE General Discoverable

Tx Power Level = 4 dBm

Service Data «Temperature Service» = 20.5 C

Services Supports = «Temperature Service», «Battery Service»

ADVERTISING DATA



Length, Type, Data formatted

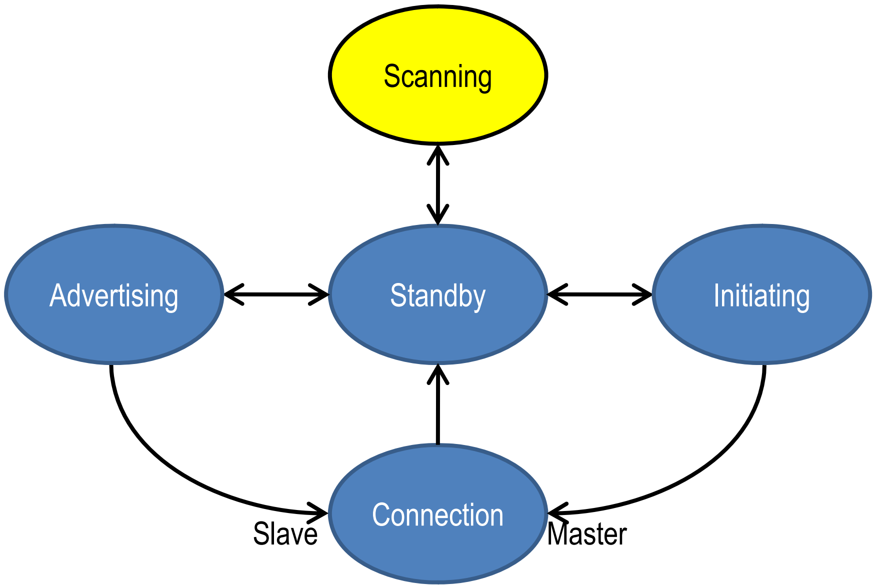
Length = length of (Type | Data)

Type = assigned number defined in GAP

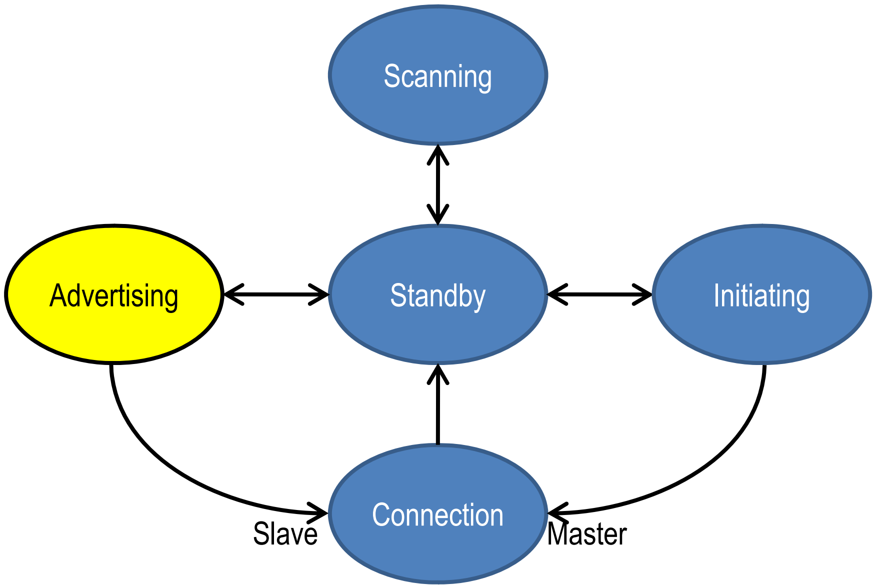
Data = typed data defined by GAP or Service

ACTIVE SCANNING

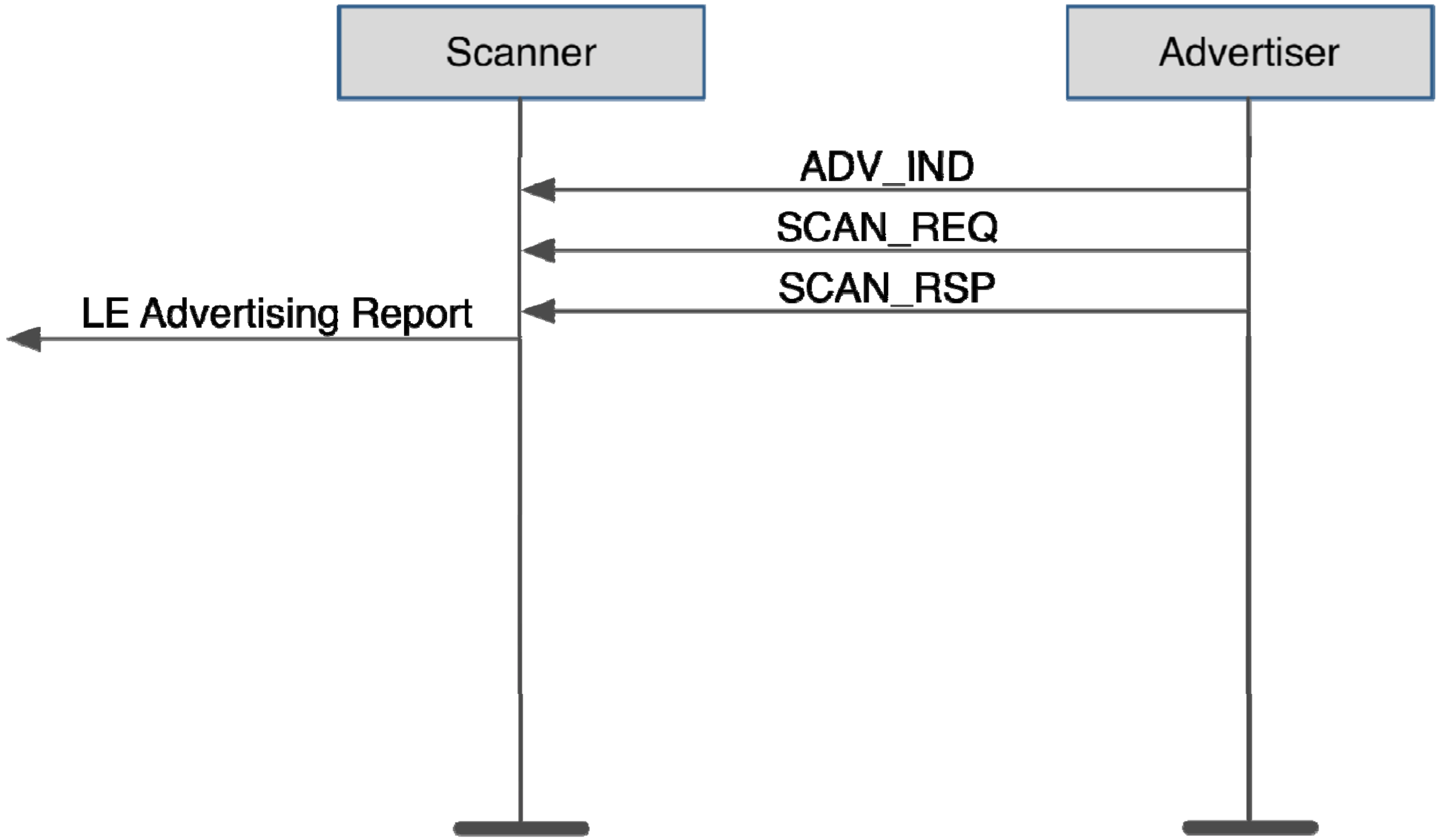
Active Scanner



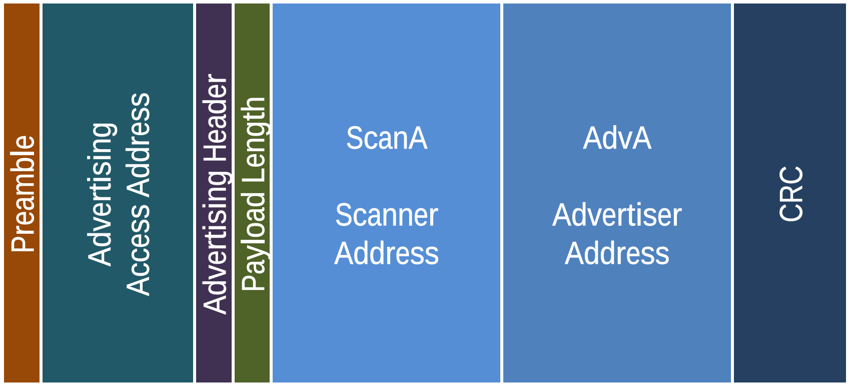
Discoverable Advertising



ACTIVE SCANNING

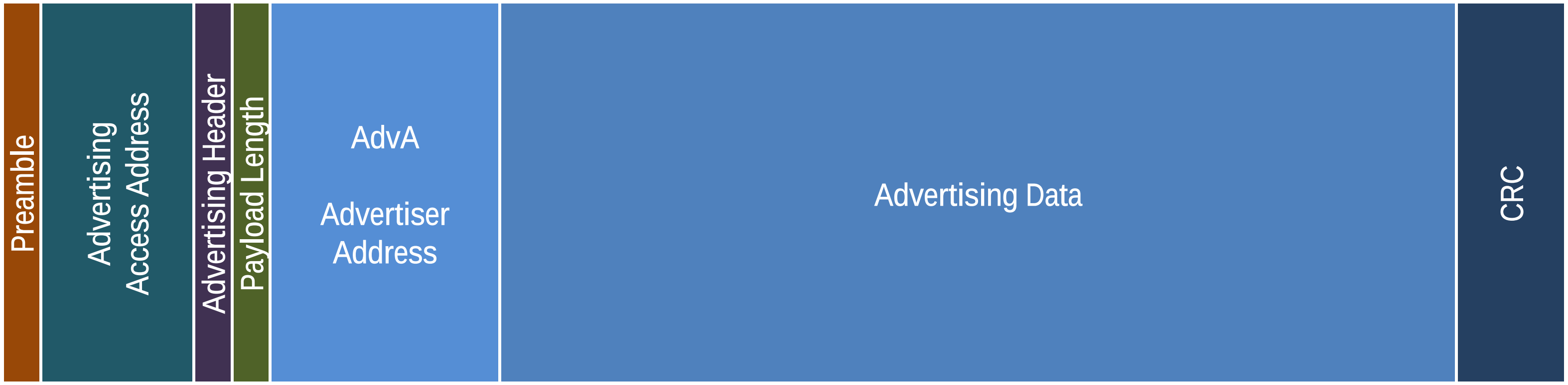


SCAN_REQ



Sent from scanner to advertiser to ask for additional data

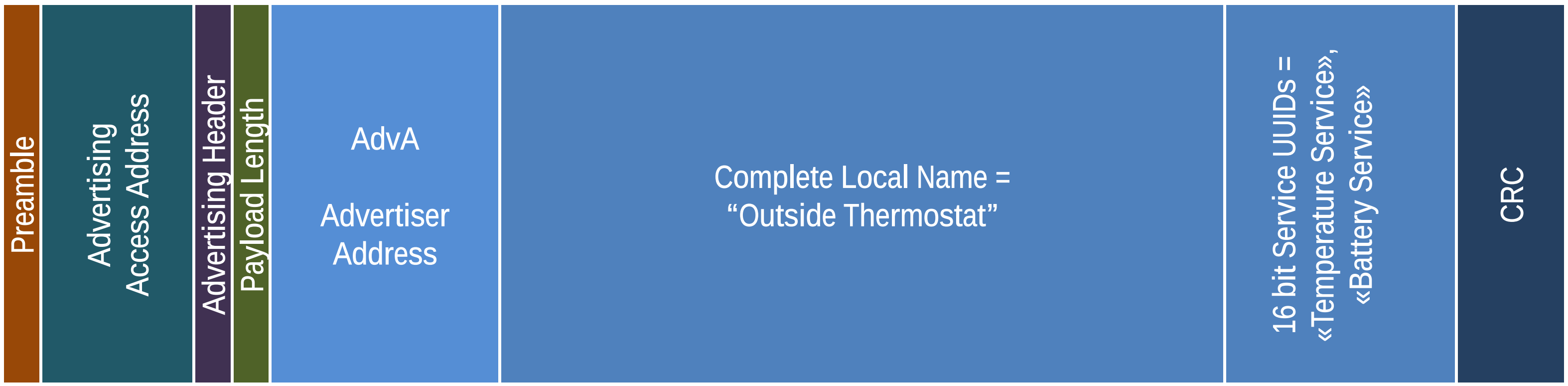
SCAN_RSP



Scan Response Data

- Device name is "Outside Thermostat"
- I support Battery and Temperature services

SCAN_RSP



Complete Local Name = “Outside Thermostat”

Services Supports = «Temperature Service», «Battery Service»

SCAN_RSP

Length, Type, Data formatted

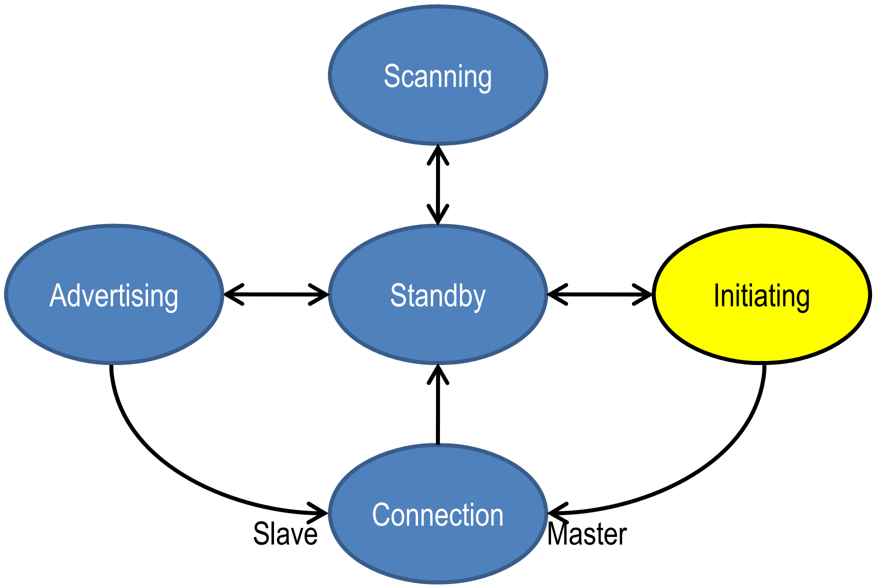
Length = length of (Type | Data)

Type = assigned number defined in GAP

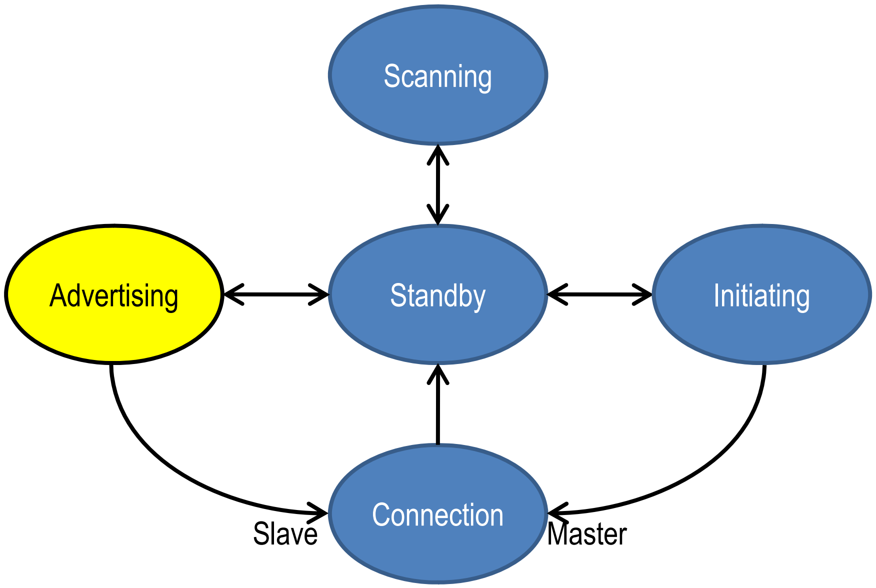
Data = typed data defined by GAP or Service

INITIATING CONNECTIONS

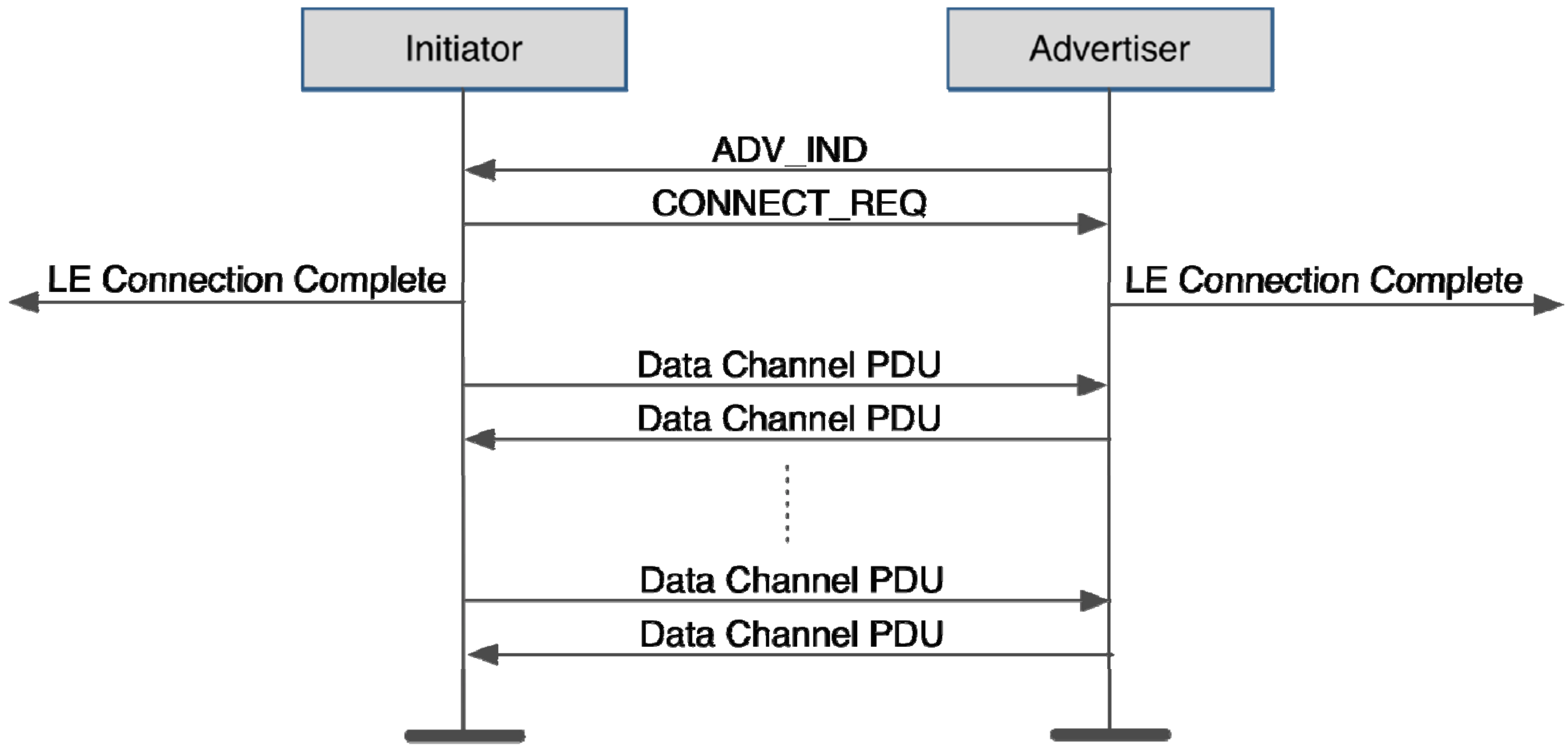
Initiator



Connectable Advertiser



INITIATING CONNECTIONS



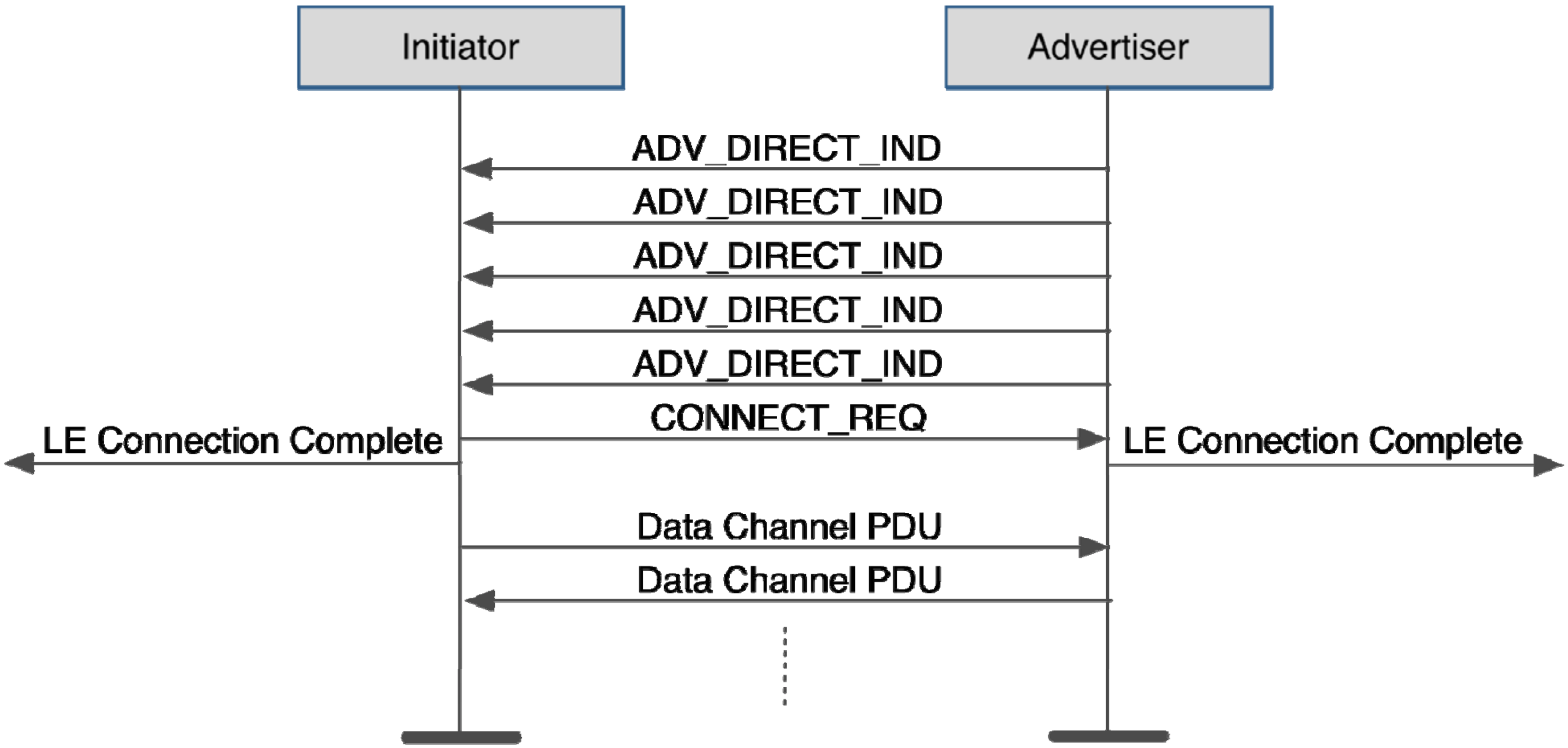
CONNECT_REQ

Sent from Initiator to Advertiser

Contains all information for link, including:

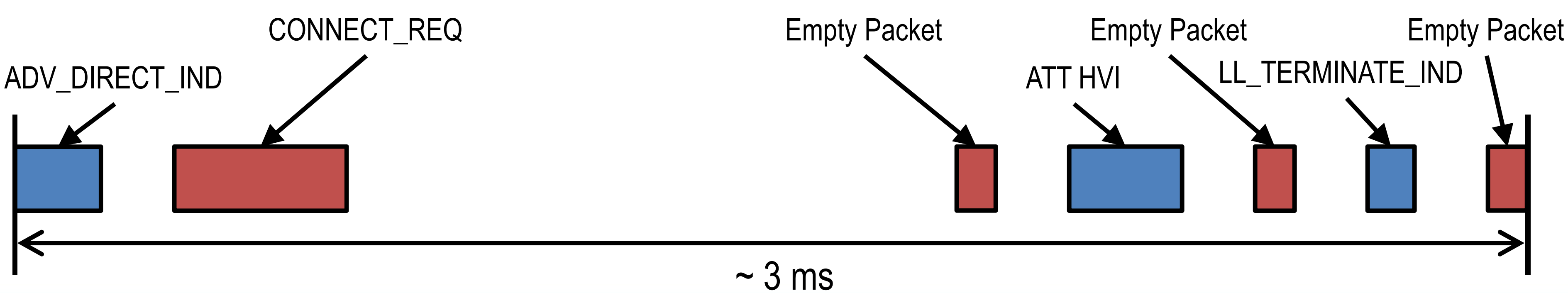
- Access Address, CRCInit, WinSize, WinOffset, Connection Interval, Latency, Timeout, Channel Map, Hop Interval, Sleep Clock Accuracy

DIRECTED CONNECTIONS

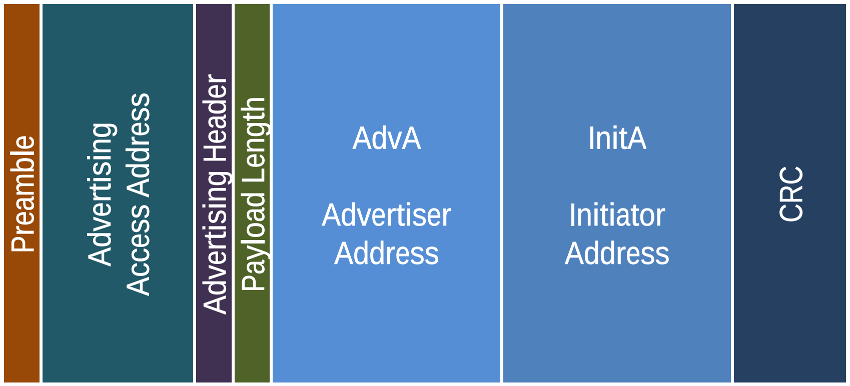


TIME FROM DISCONNECTED TO DATA ~ 3MS (RADIO ACTIVE ~ 1MS)

Time (us)	Master Tx	Radio Active (us)	Slave Tx
0		176	ADV_DIRECT_IND
326	CONNECT_REQ	352	
1928	Empty Packet	80	
2158		144	Attribute Protocol Handle Value Indication
2452	Empty Packet (Acknowledgement)	80	
2682		96	LL_TERMINATE_IND
2928	Empty Packet (Acknowledgement)	80	



DIRECTED ADVERTISING



Sent from Advertiser to Initiator

says “I want you to connect to me”

WHITE LISTS

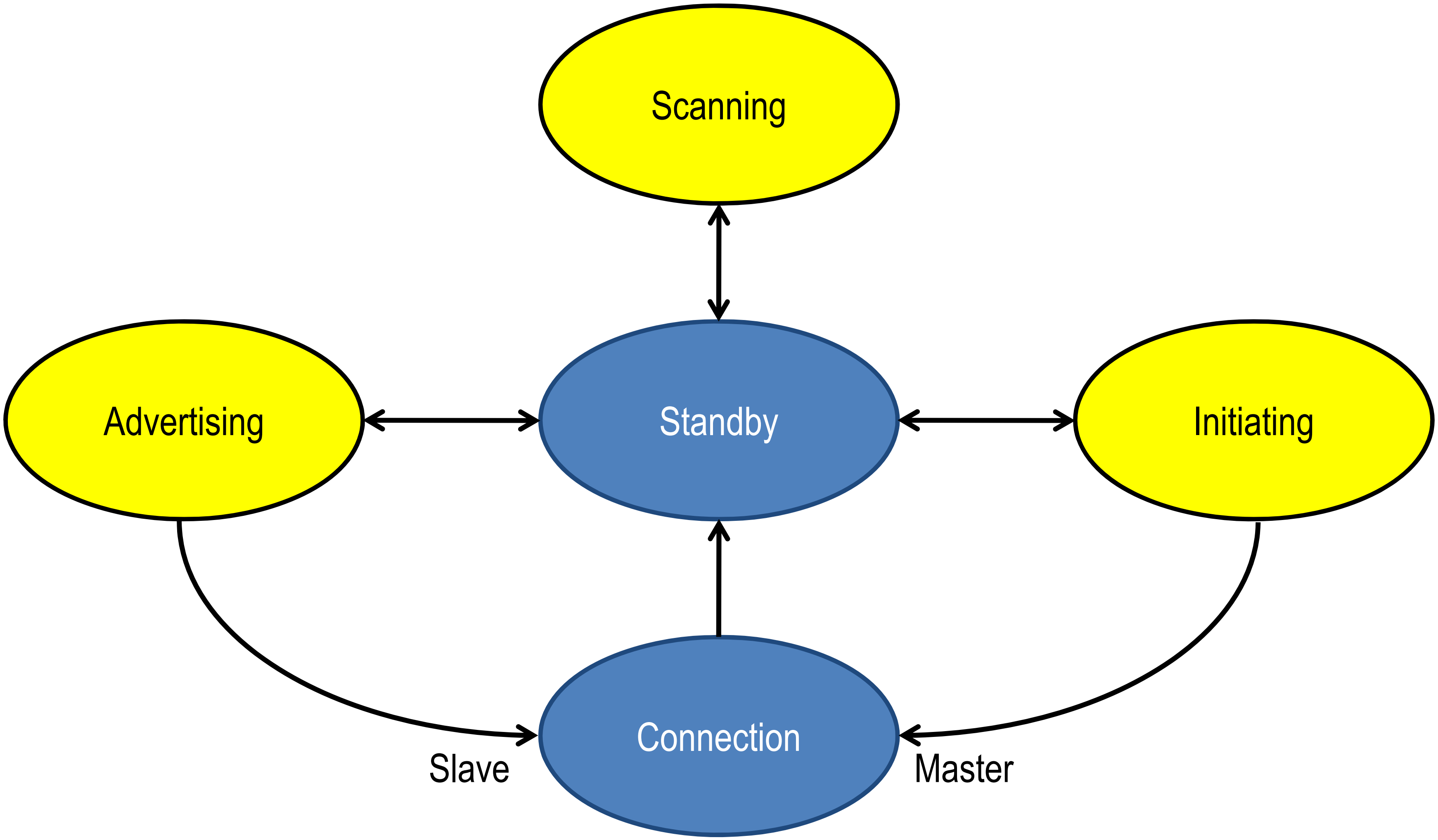
Because advertising channels will be “busy”

- sometimes useful to filter out devices you don’t care about

White List

- set of devices in advertising packets that will be processed
- consist of an array of device addresses of a known size

WHITE LISTS AND LINK LAYER STATES



ADVERTISING FILTER POLICY

Determines how controller will process scan and/or connection requests when as advertiser

- The Link Layer shall process scan and connection requests only from devices in the White List
- The Link Layer shall process scan and connection requests from all devices (i.e. the White List is not in use). This is the default on reset
- The Link Layer shall process scan requests from all devices and shall only process connection requests from devices that are in the White List
- The Link Layer shall process connection requests from all devices and shall only process scan requests from devices that are in the White List

SCANNER FILTER POLICY

Determines how controller will process advertising packets when scanning:

- The Link Layer shall process advertising packets only from devices in the White List
- The Link Layer shall process all advertising packets (i.e., the White List is not used). This is the default on reset

INITIATOR FILTER POLICY

Determines how controller will process advertising packets when initiating:

- The Link Layer shall process connectable advertising packets from all devices in the White List
- The Link Layer shall ignore the White List and process connectable advertising packets from a specific single device specified by the Host

WHITE LISTS

No need to send every advertising packet to Host

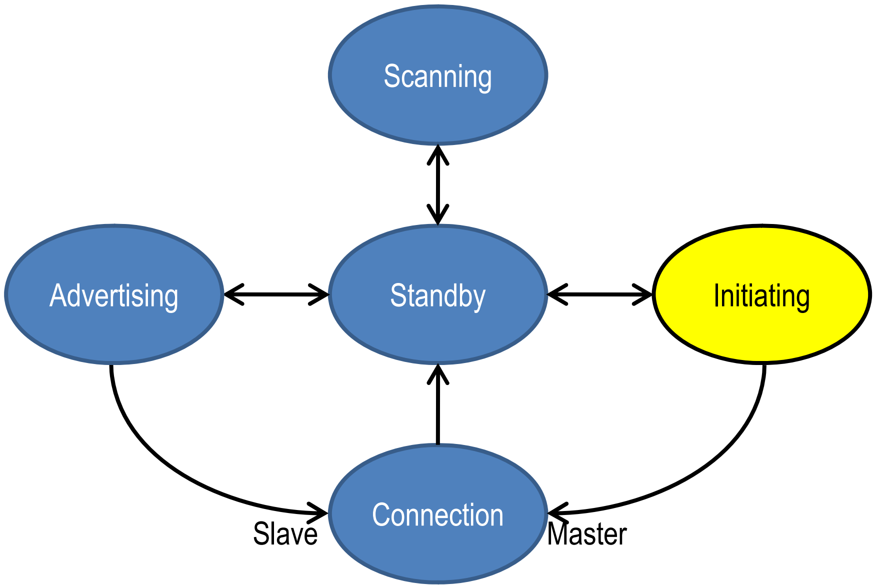
- only send information from devices in white list

Allows "connect to white list" semantics

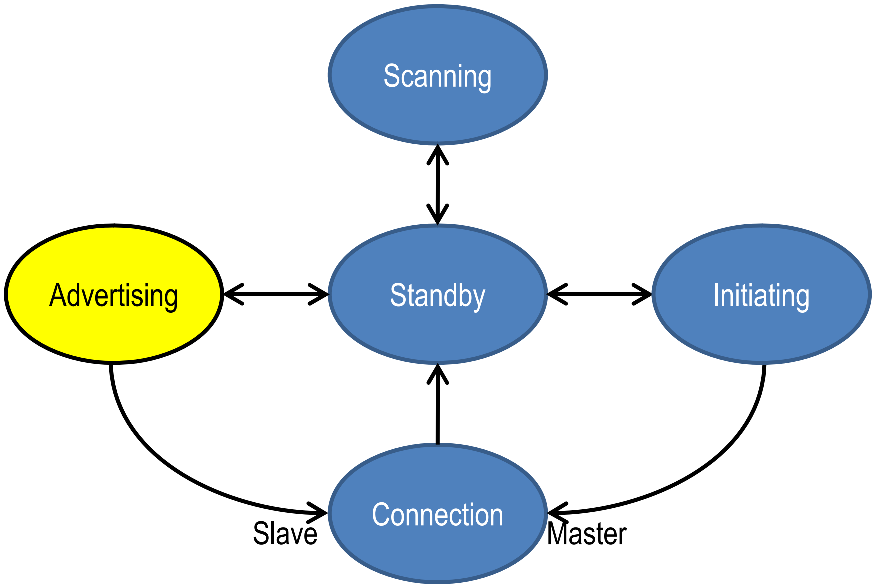
- a master can automatically connect to a set of devices
- will connect when sees adverts from these devices
- allows very fast connections from many devices

INITIATING CONNECTIONS

Initiator

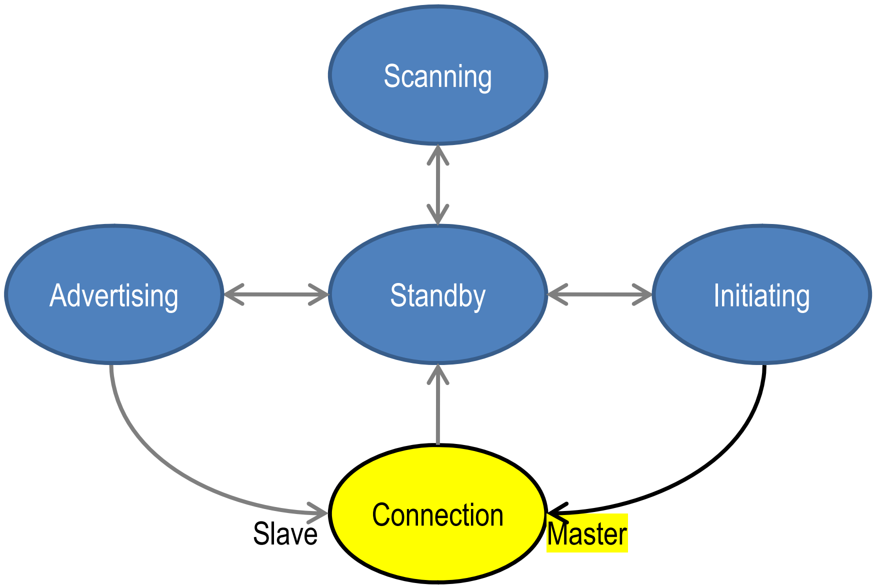


Connectable Advertiser

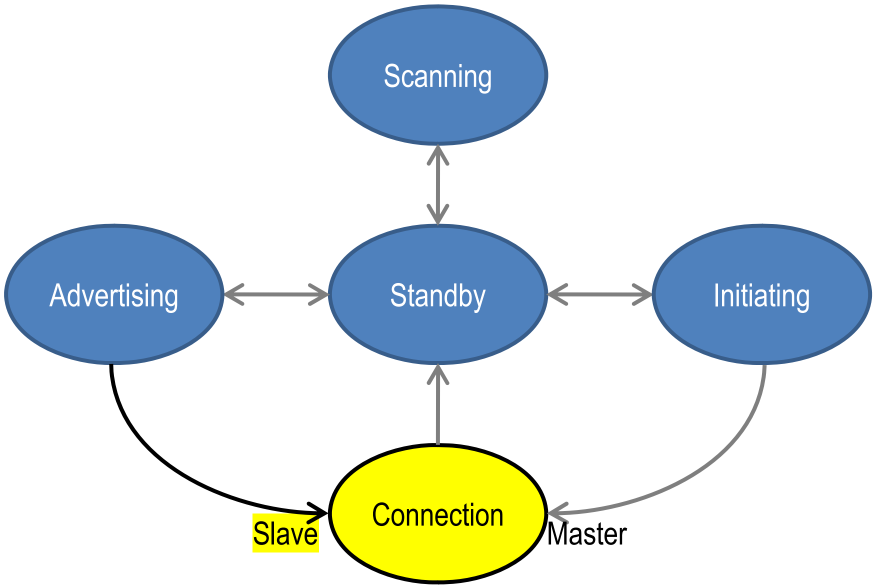


CONNECTIONS

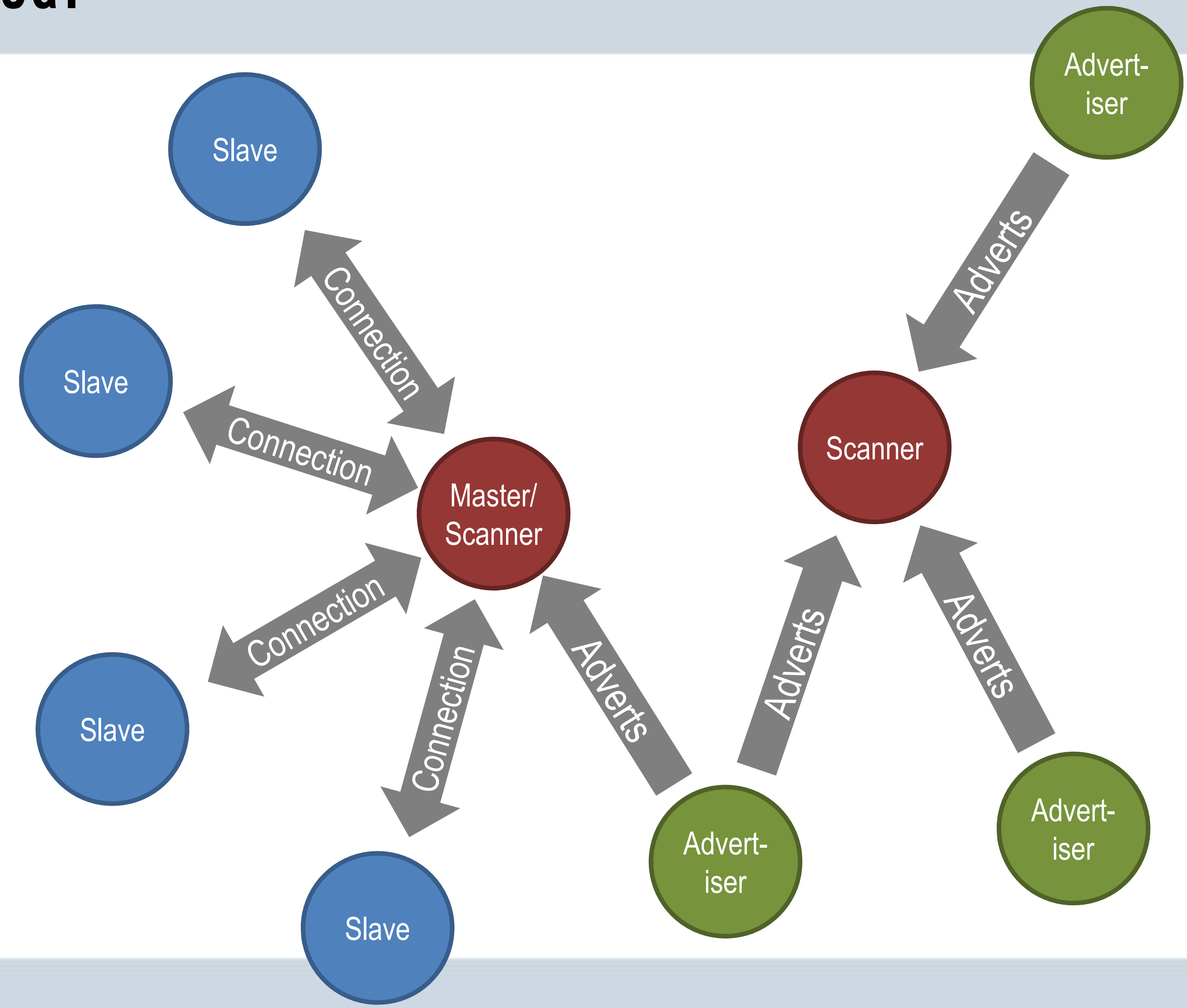
Master



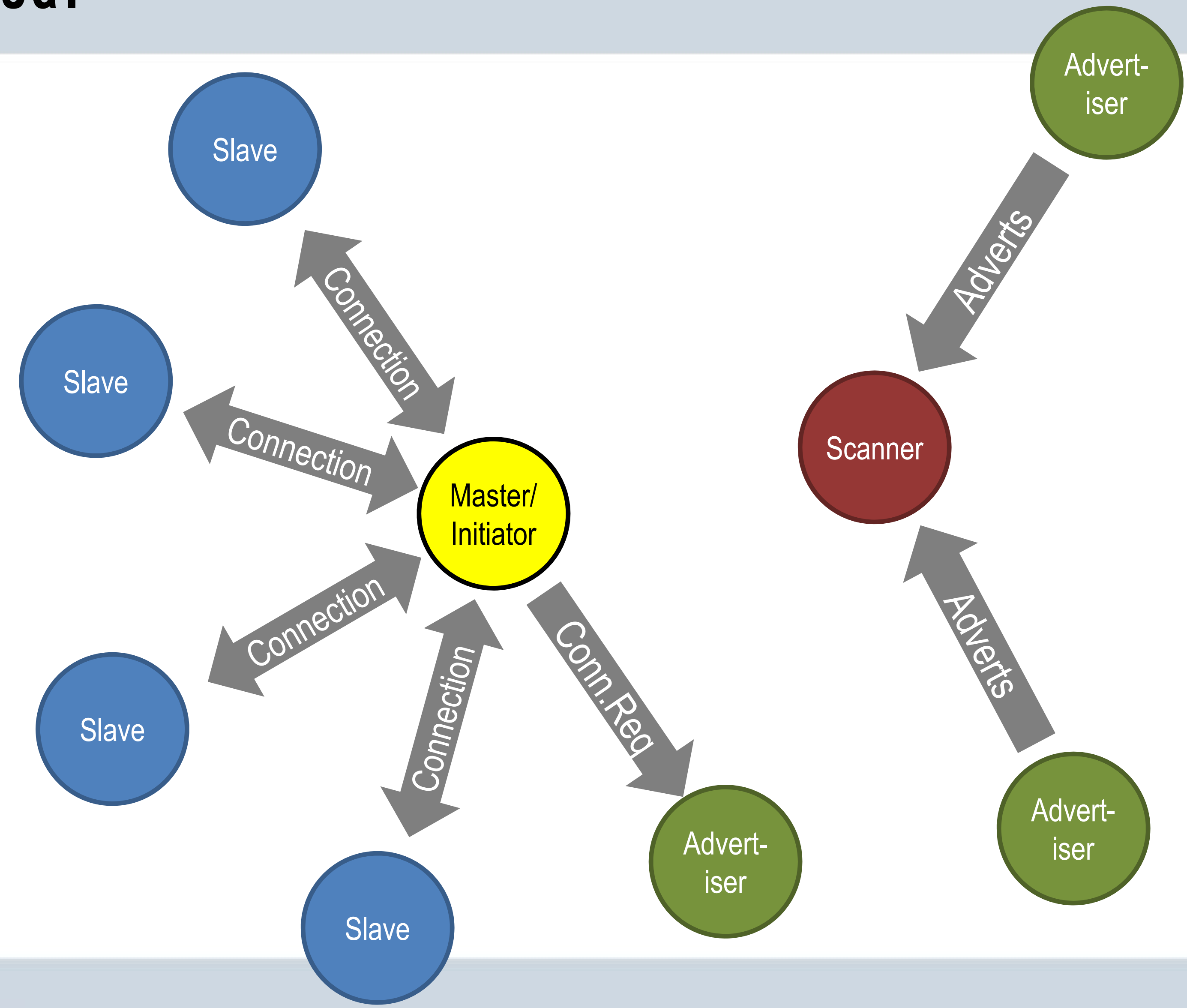
Slave



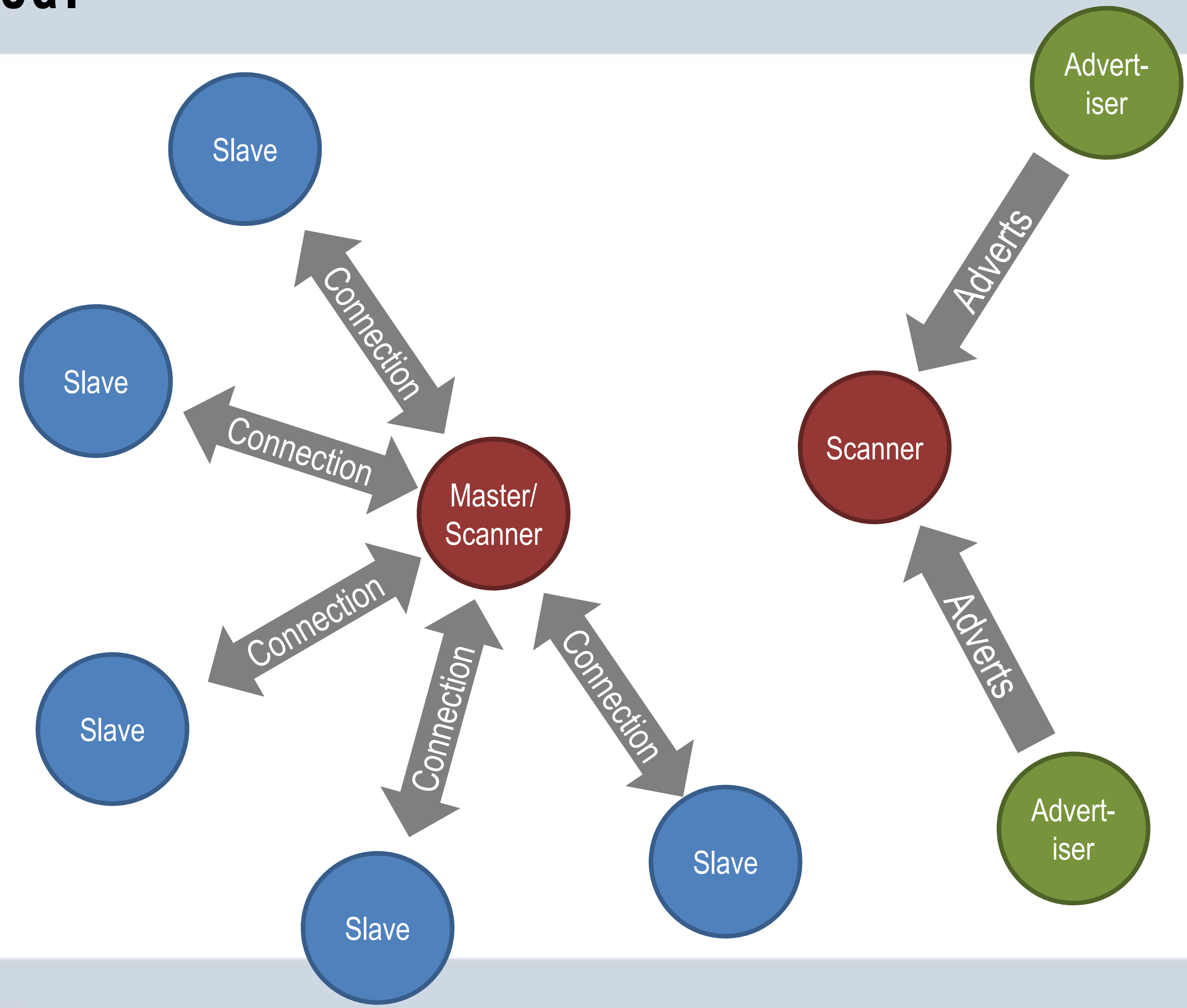
TOPOLOGY


Slave
Slave
Slave
Slave
Connection
Connection
Connection
Connection
Master/ Scanner
Scanner
Adverts
Adverts
Adverts
Adverts
Advert- iser
Advert- iser
Advert- iser

TOPOLOGY


Slave
Slave
Slave
Slave
Connection
Connection
Connection
Connection
Master/ Initiator
Conn.Req
Advert- iser
Scanner
Adverts
Adverts
Advert- iser
Advert- iser

TOPOLOGY


Slave
Slave
Slave
Slave
Slave
Connection
Connection
Connection
Connection
Connection
Master/
Scanner
Scanner
Advert-
iser
Advert-
iser
Adverts
Adverts

LIMITS

A single master can address ~2^{31} slaves

~ 2 billion addressable slaves per master

Max Connection Interval = 4.0 seconds

Can address a slave every ~ 5 ms (assuming 250 ppm clocks)

~ 800 active slaves per master

CONNECTIONS

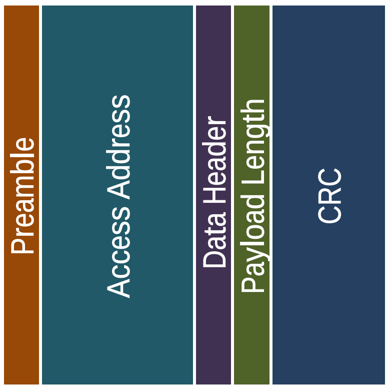
Used to send application data

- reliably, robustly

Includes

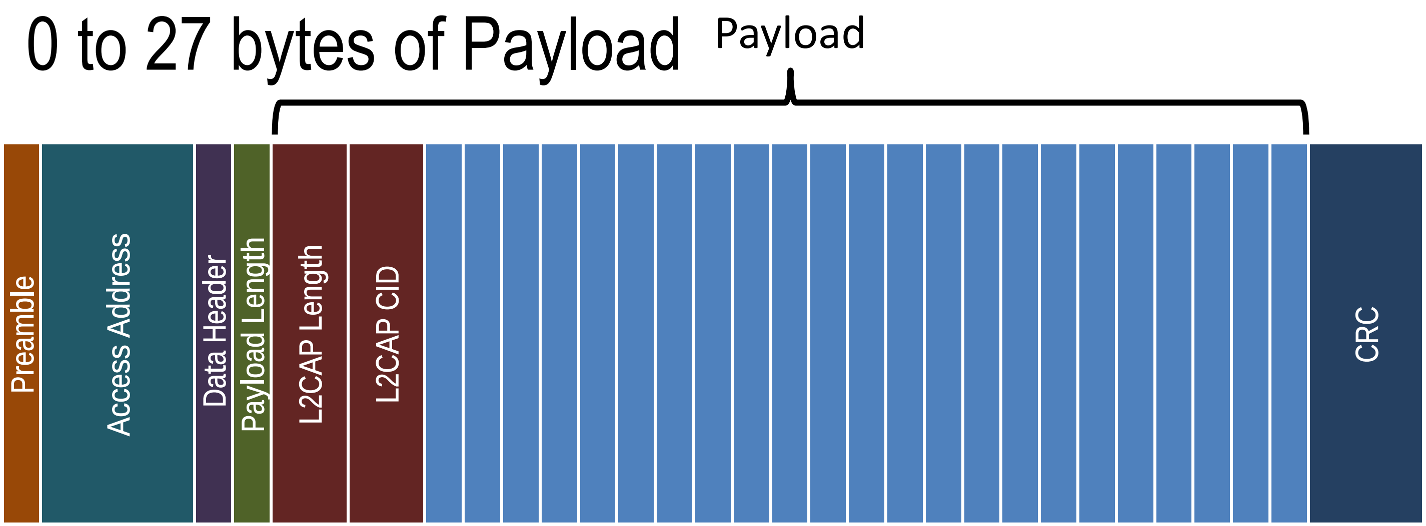
- ultra low power connection mode
- adaptive frequency hopping
- connection supervision timeout

DATA PACKET



Empty Packet

0 to 27 bytes of Payload

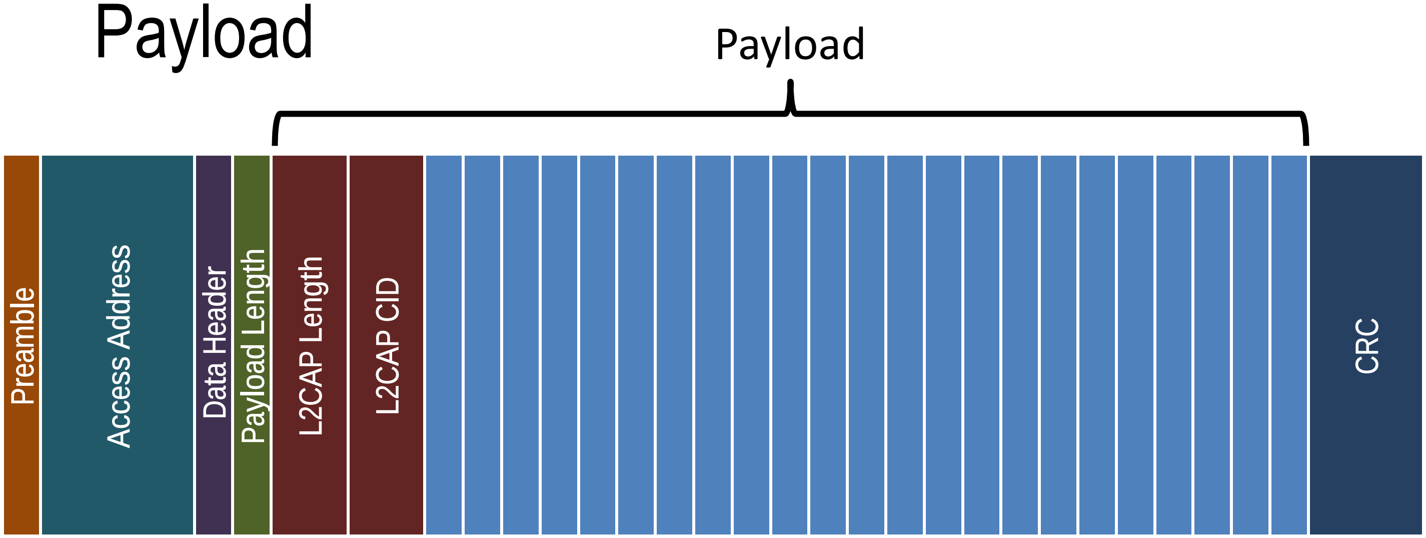


DATA PACKET

0 to 27 bytes of Payload (unencrypted)

CRC protects

- Data Header
- Payload Length
- Payload

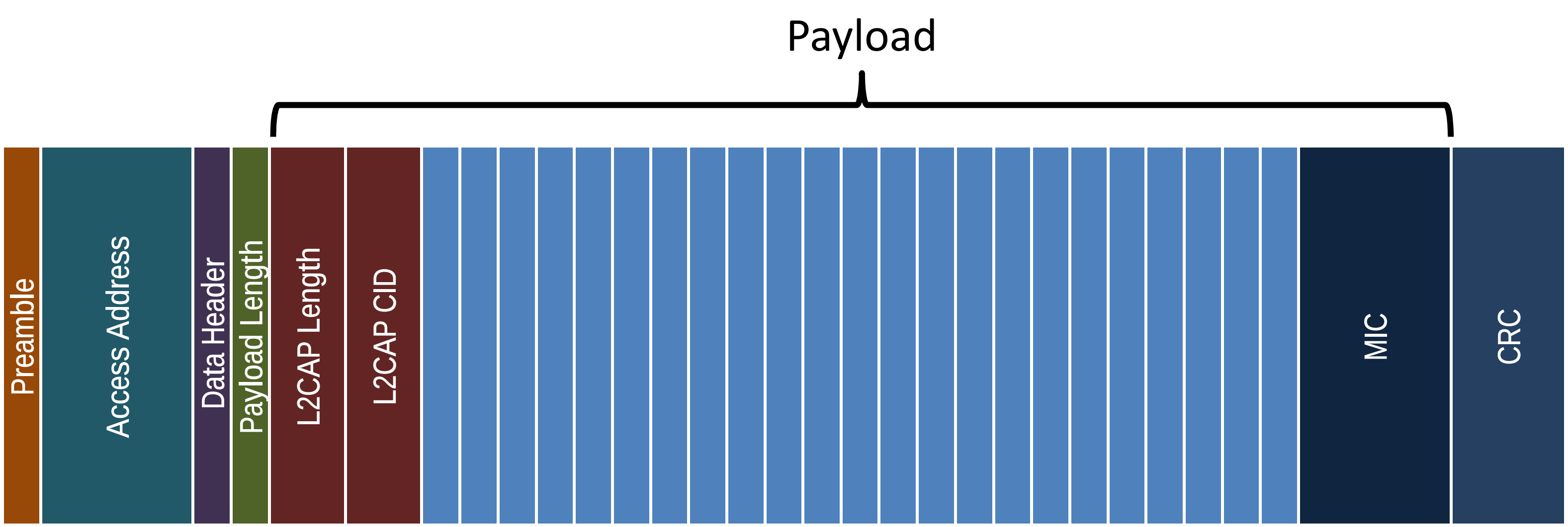


ENCRYPTED DATA PACKET

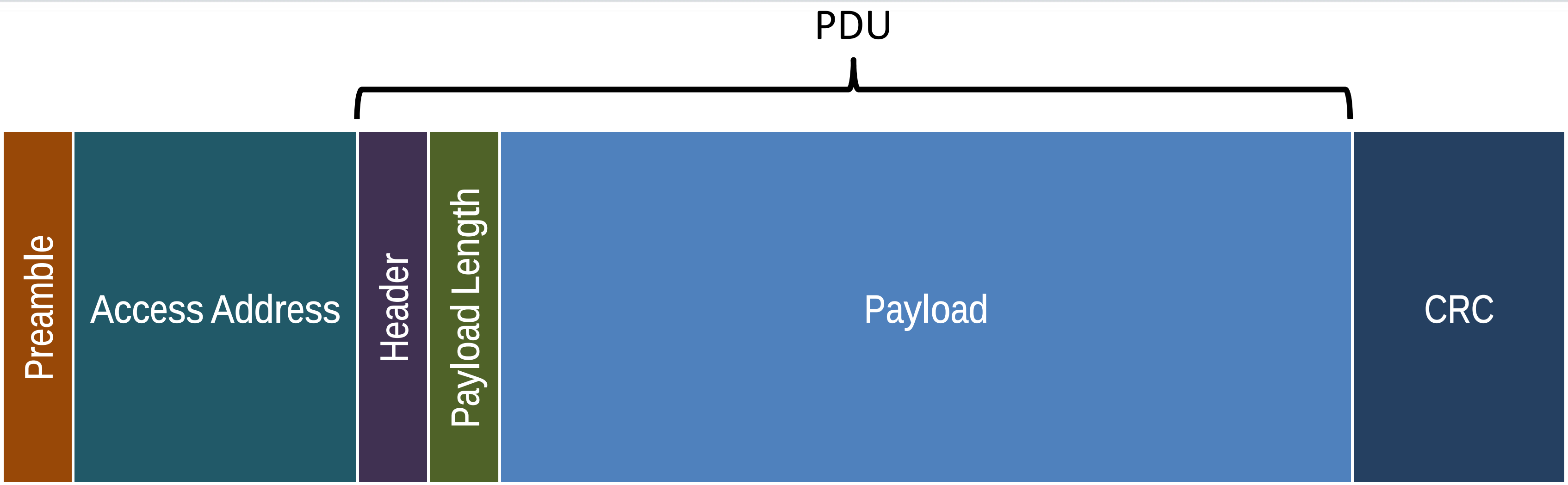
4 to 31 bytes of payload length

MIC is part of “Payload”, CRC protects it

MIC can be computed / checked in background



PDU HEADERS



Data channel PDU Header / Payload Length

LLID	NESN	SN	MD	RFU
Length (0 – 31)				RFU

LOGICAL LINK IDENTIFIER

LLID	Description
00	Reserved
01	LL Data PDU Continuation of an L2CAP message or an Empty PDU
10	LL Data PDU Start of an L2CAP message Complete L2CAP message
11	LL Control PDU

SEQUENCE NUMBERS

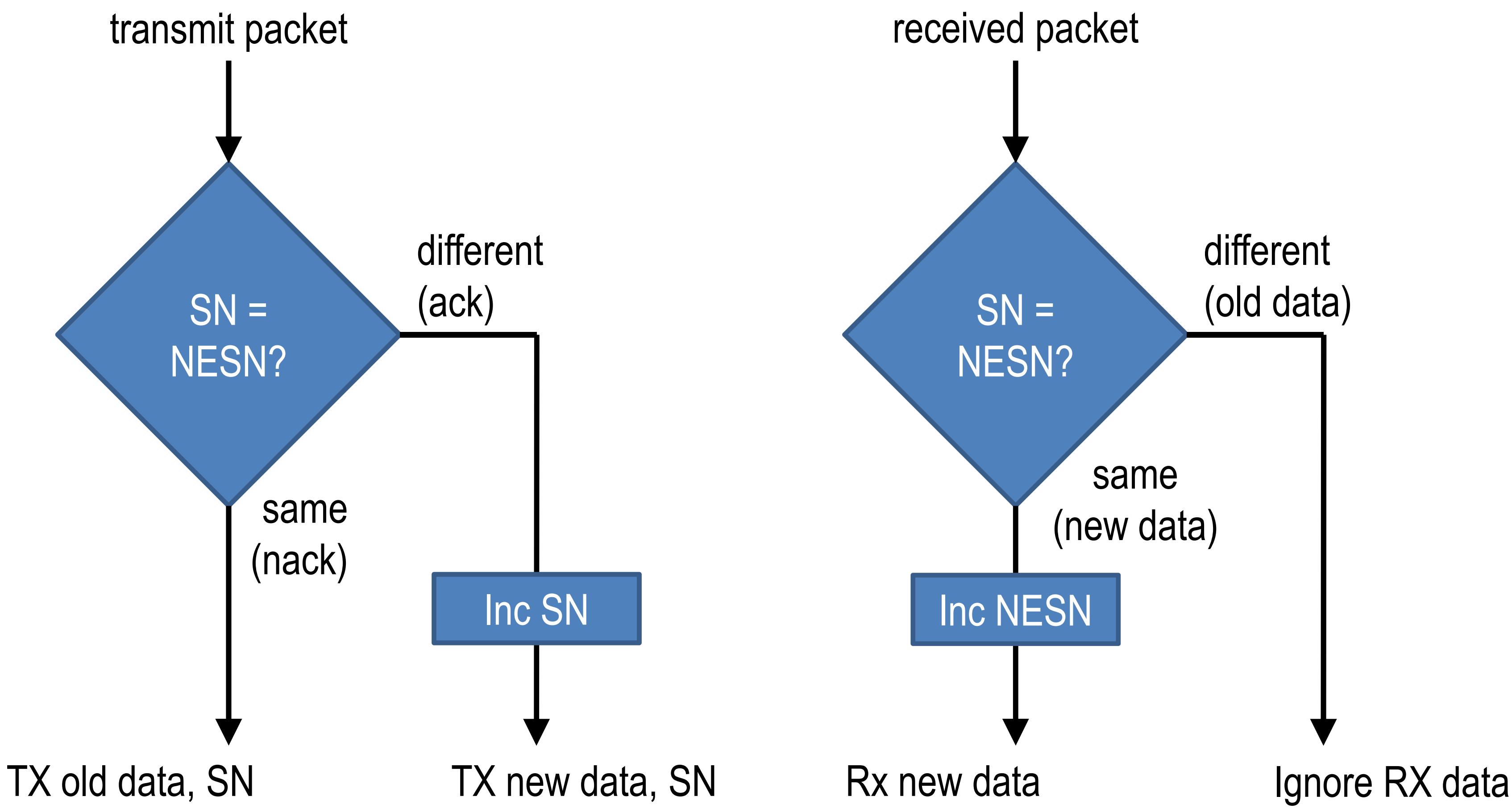
SN = Sequence Number

NESN = Next Expected Sequence Number

Sliding Window Algorithm

- window size of 1
- lazy acknowledgement possible – saves power

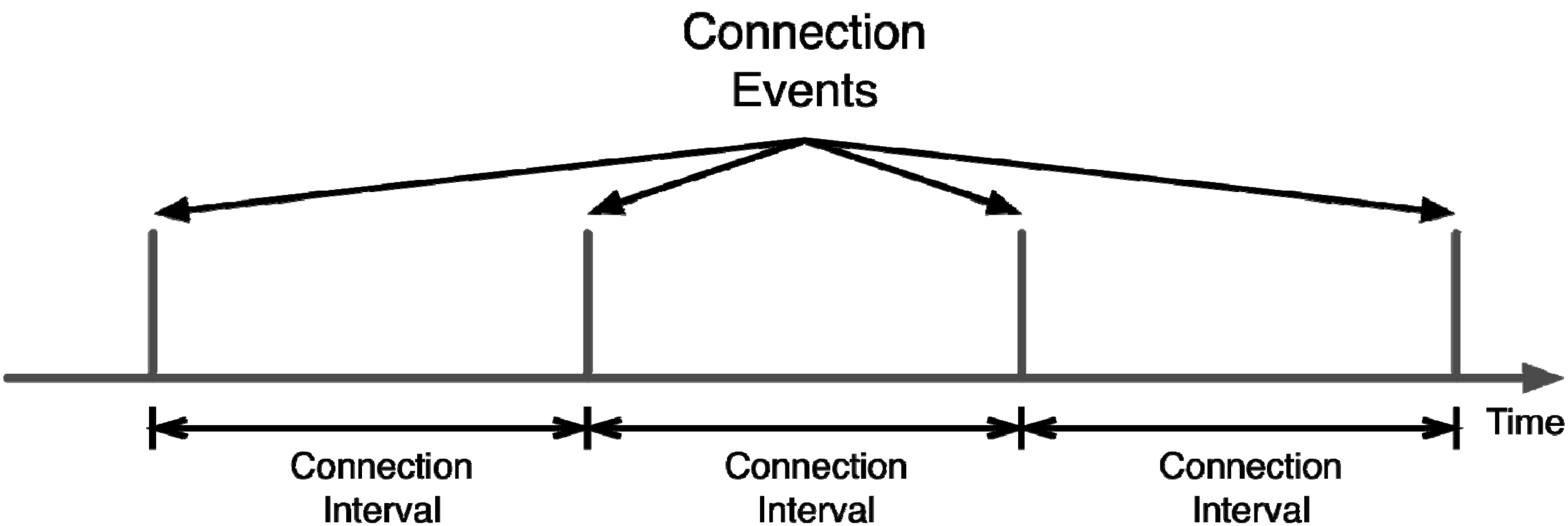
TRANSMITTING DATA



CONNECTION EVENTS

Masters transmits periodically at a connection events

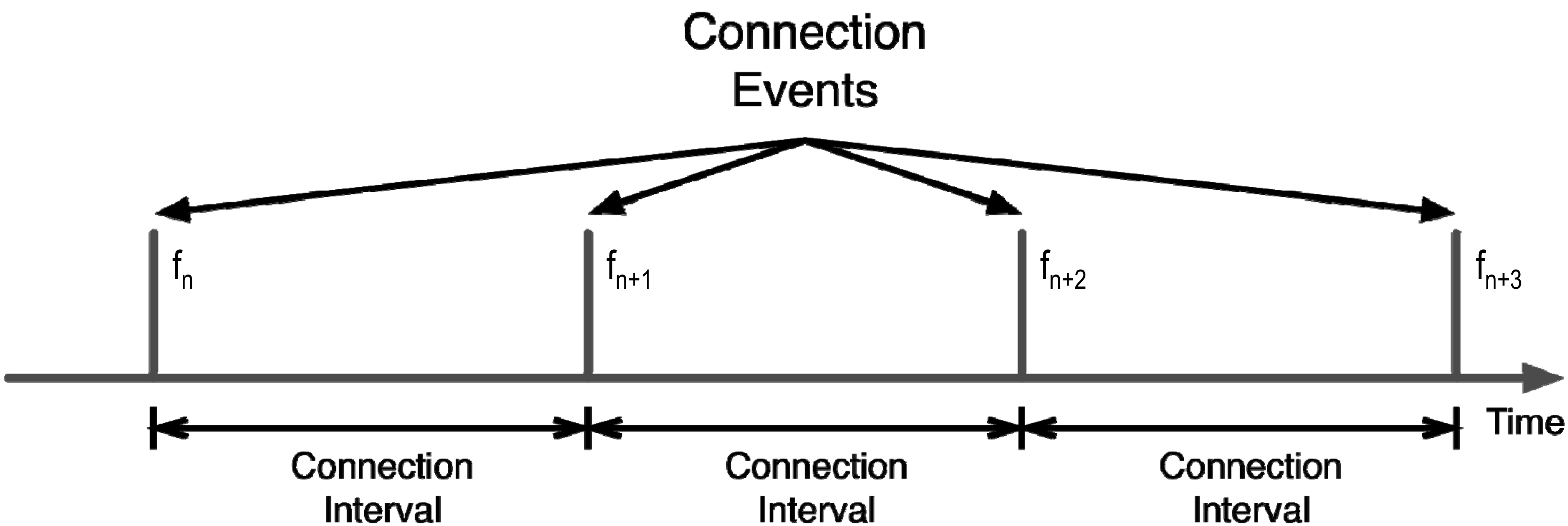
- Connection Interval sent in CONNECT_REQ
- Connection events continue until MD = 0



CONNECTION EVENTS

Each connection event uses a different channel

$f_{n+1} = (f_n + hop) \bmod 37$



Bluetooth®
SPECIAL INTEREST GROUP

LATENCY

Master Latency

how often the master will transmit to slave

Slave Latency

how often the slave will listen to master

The two latencies don't have to be the same

Master Latency = Connection Interval (7.5 ms to 4.0 s)

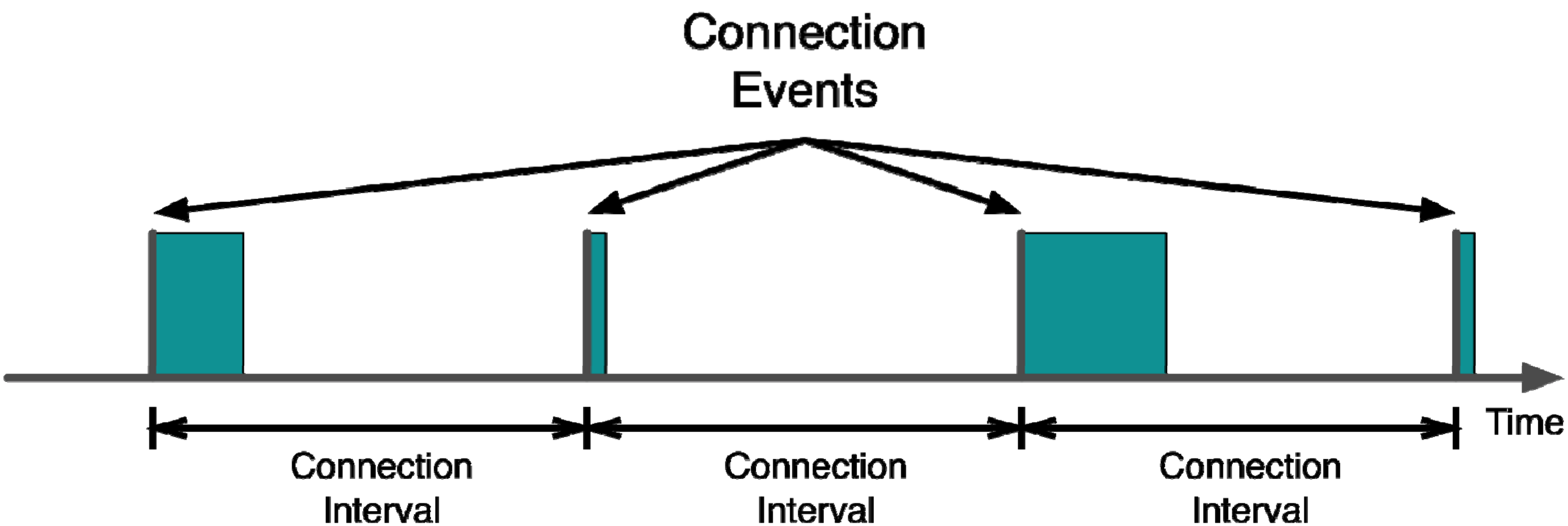
Slave Latency = Connection Interval * Slave Latency

MORE DATA

		Master: MD = 0	Master: MD = 1
Slave	MD = 0	Neither device has more data to send. Connection event closed	Master has more data, Slave has no more data. Master may continue, Slave should listen
Slave	MD = 1	Slave has more data, Master has no more data. Master may continue, Slave should listen	Both devices have more data. Master may continue, Slave should listen.

CONNECTION EVENTS

More Data bit automatically extends connection events



PACKET TIMINGS

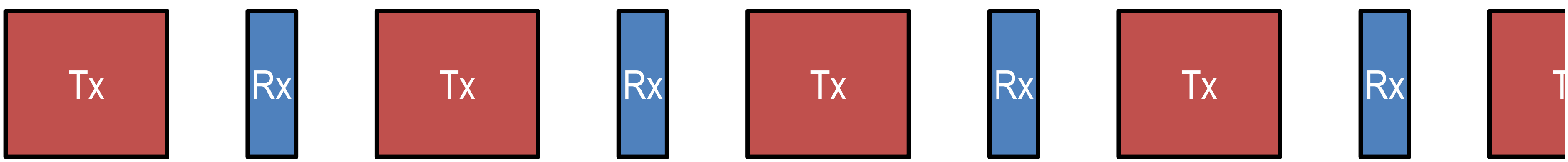
Peer device transmits 150 µs after last packet

Minimum size packet = 80 µs

(Preamble + Access Address + Header + CRC)

Maximum size packet = 328 µs

(Preamble + Access Address + Header + Payload + MIC + CRC)



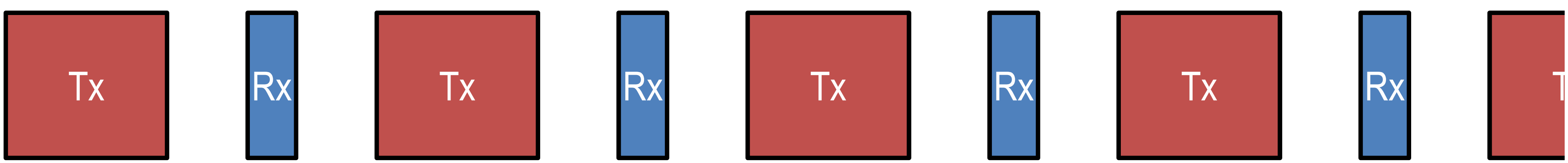
MAXIMUM DATA RATE

Asymmetric Tx/Rx Packet Sequence

328 + 150 + 80 + 150 = 708 μs

Transmitting 27 octets of application data

~305 kbps



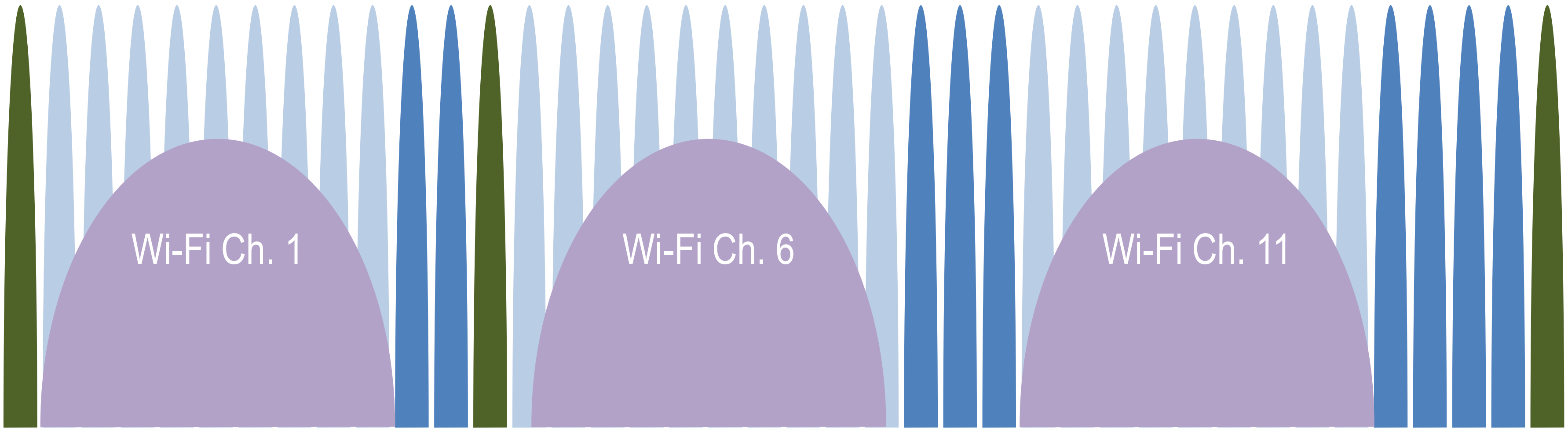
ADAPTIVE FREQUENCY HOPPING

Frequency Hopping algorithm is very simple

$f_{n+1} = (f_n + hop) \bmod 37$

If fn is a “used” channel, use as is

If fn is an “unused” channel, remap to set of good channels

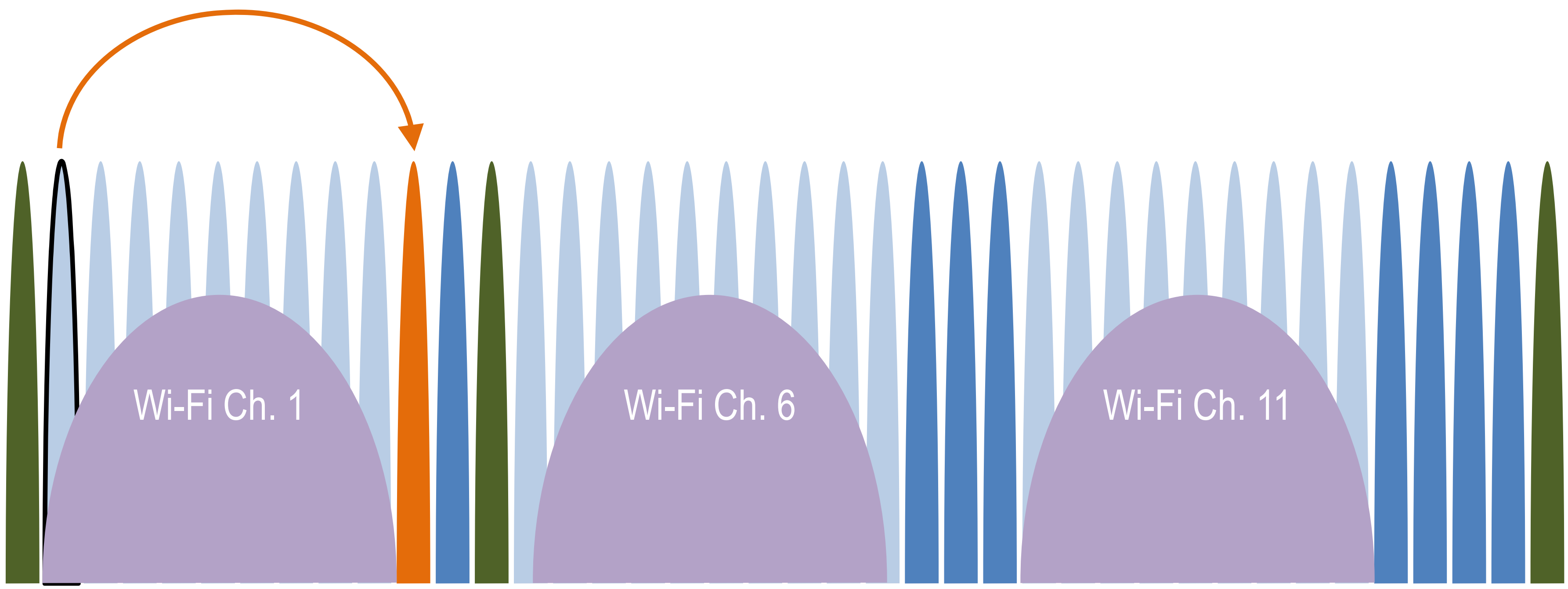


ADAPTIVE FREQUENCY HOPPING

$f_n = 0$, hop = 7, used = [9, 10, 21, 22, 23, 33, 34, 35, 36]

"unused"; $f_{n+1} = f_n + 7 \bmod 37 = 7 \bmod 37 \rightarrow 7$

$0 \bmod 9 \rightarrow 0$; used[0] $\rightarrow 9$

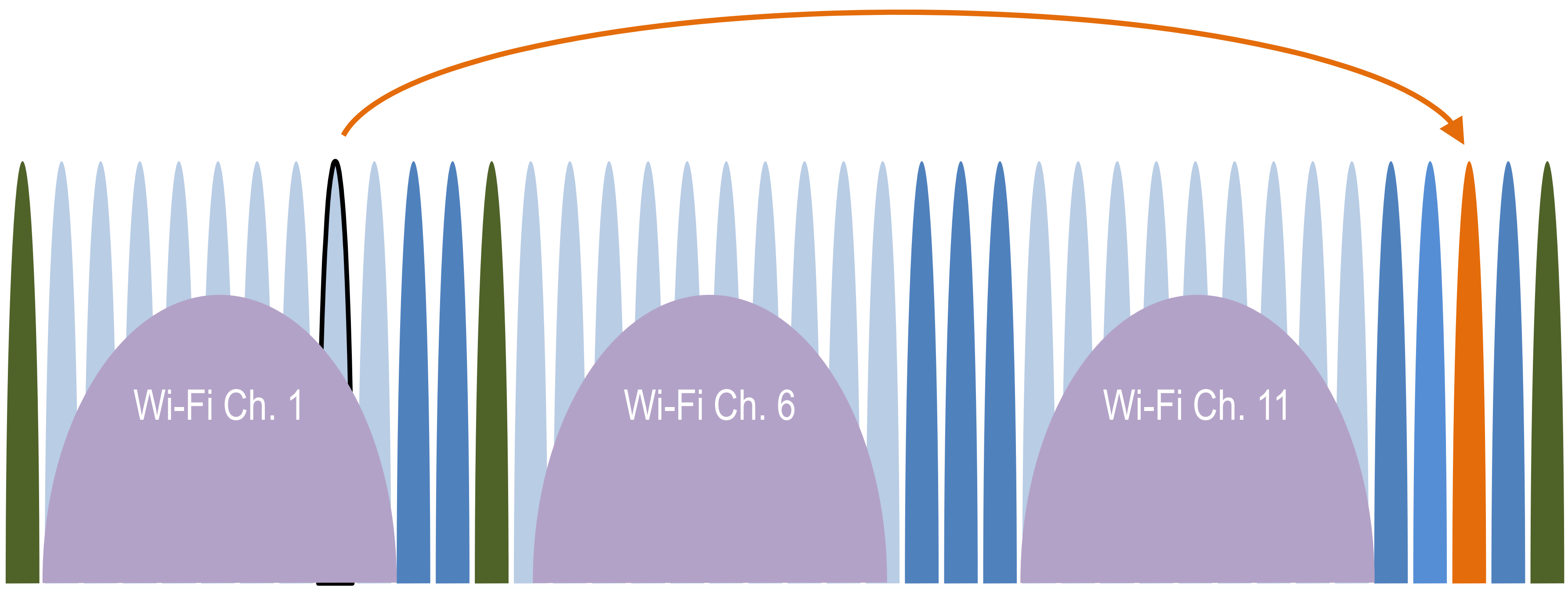


ADAPTIVE FREQUENCY HOPPING

f_n = 7, hop = 7, used = [9, 10, 21, 22, 23, 33, 34, 35, 36]

"unused"; $f_{n+1} = f_n + 7 \bmod 37 = 14 \bmod 37 \rightarrow 14$

$7 \bmod 9 \rightarrow 7$; used[7] $\rightarrow$ 35

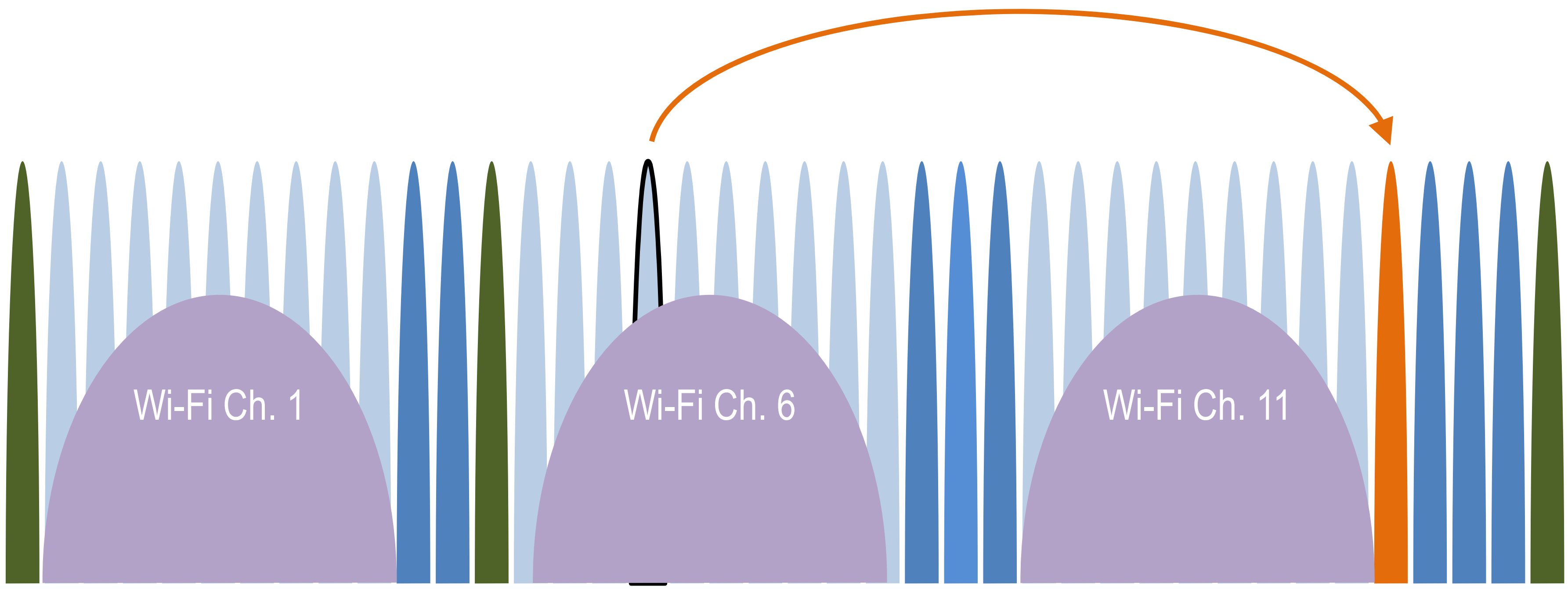


ADAPTIVE FREQUENCY HOPPING

f_n = 14, hop = 7, used = [9, 10, 21, 22, 23, 33, 34, 35, 36]

"unused"; $f_{n+1} = f_n + 7 \bmod 37 = 21 \bmod 37 \rightarrow 21$

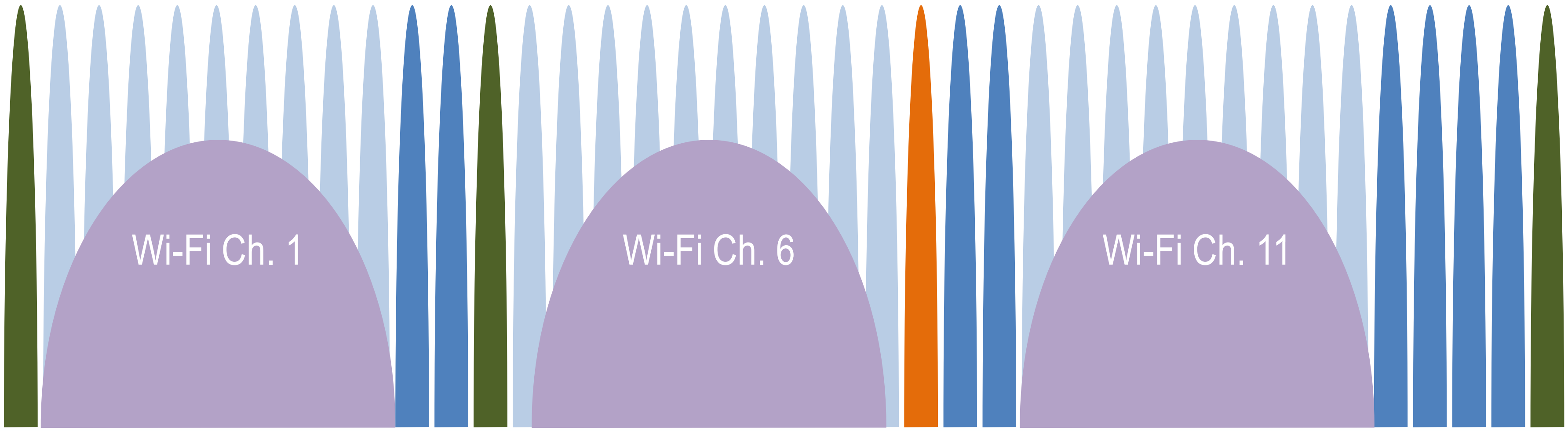
$14 \bmod 9 \rightarrow 5$; used[5] $\rightarrow$ 33



ADAPTIVE FREQUENCY HOPPING

fn = 21, hop = 7, used = [9, 10, 21, 22, 23, 33, 34, 35, 36]

“used”; $f_{n+1} = f_n + 7 \bmod 37 = 28 \bmod 37 \rightarrow 28$

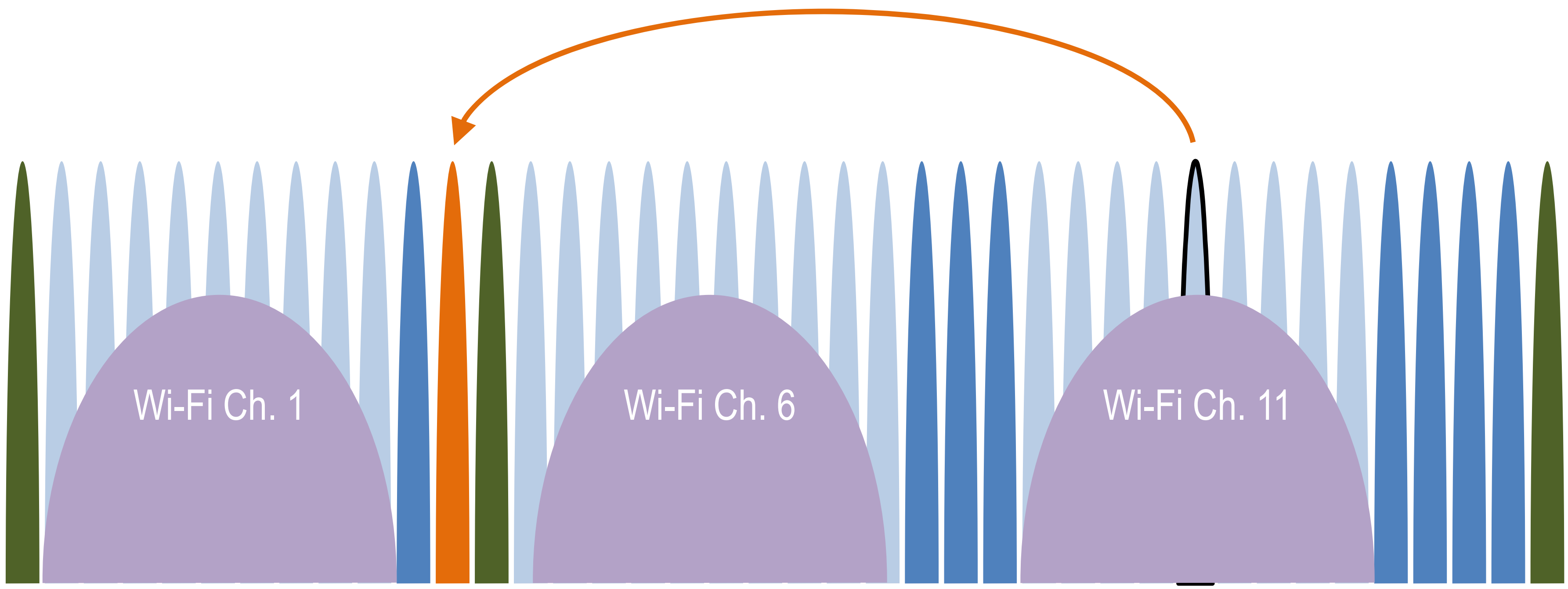


ADAPTIVE FREQUENCY HOPPING

fn = 28, hop = 7, used = [9, 10, 21, 22, 23, 33, 34, 35, 36]

"unused"; $f_{n+1} = f_n + 7 \bmod 37 = 35 \bmod 37 \rightarrow 35$

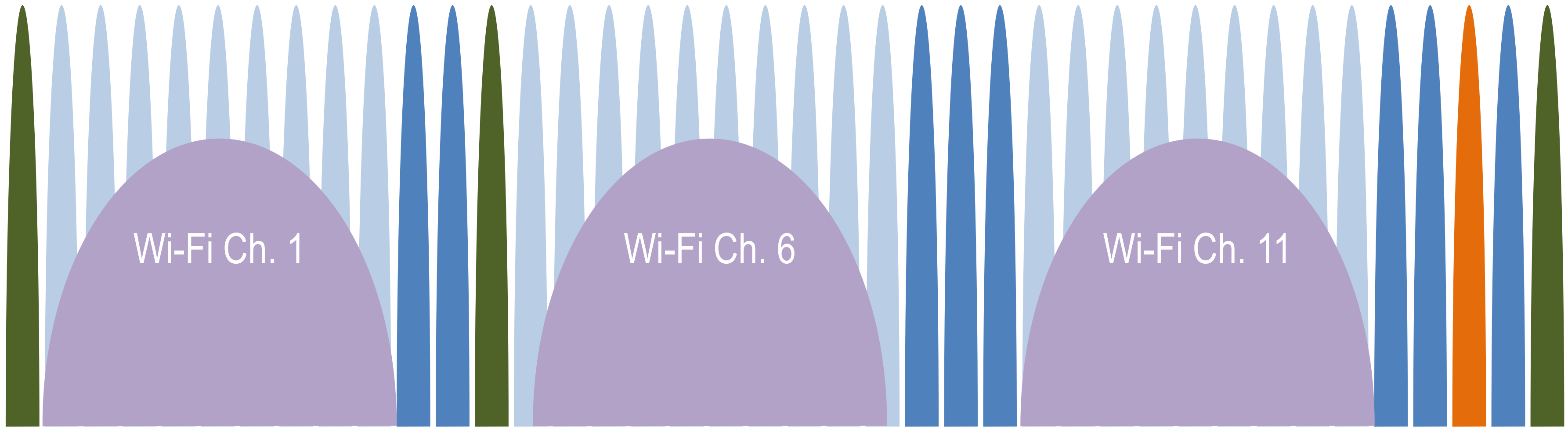
$28 \bmod 9 \rightarrow 1$; used[1] $\rightarrow 10$



ADAPTIVE FREQUENCY HOPPING

fn = 35, hop = 7, used = [9, 10, 21, 22, 23, 33, 34, 35, 36]

“used”; $f_{n+1} = f_n + 7 \bmod 37 = 42 \bmod 37 \rightarrow 5$

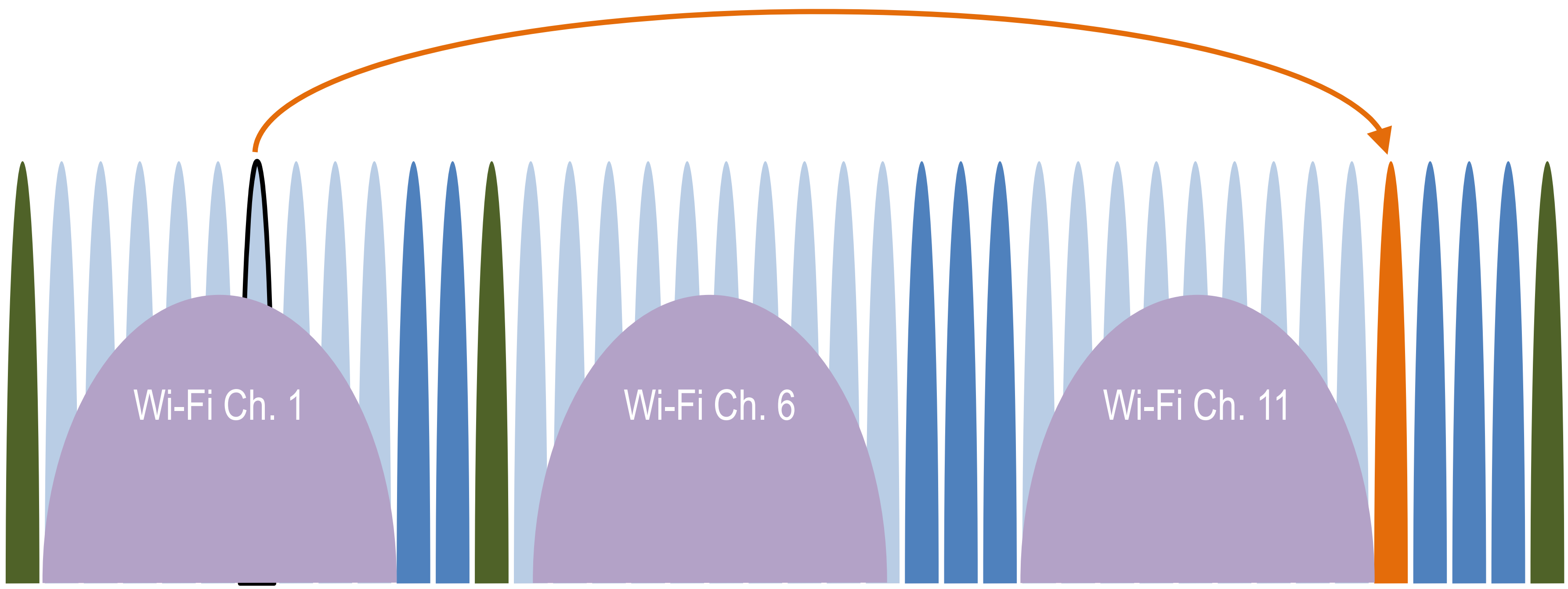


ADAPTIVE FREQUENCY HOPPING

fn = 5, hop = 7, used = [9, 10, 21, 22, 23, 33, 34, 35, 36]

"unused"; $f_{n+1} = f_n + 7 \bmod 37 = 12 \bmod 37 \rightarrow 12$

$5 \bmod 9 \rightarrow 5$; used[5] $\rightarrow 33$

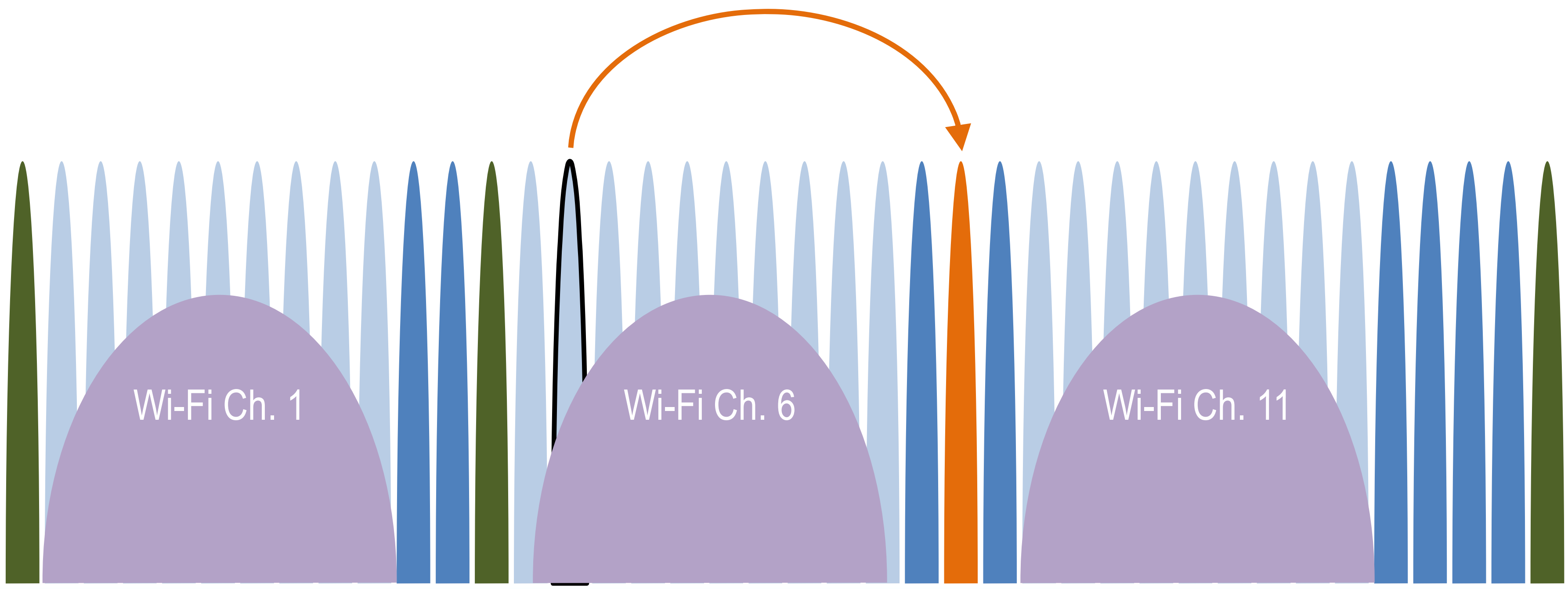


ADAPTIVE FREQUENCY HOPPING

fn = 12, hop = 7, used = [9, 10, 21, 22, 23, 33, 34, 35, 36]

"unused"; $f_{n+1} = f_n + 7 \bmod 37 = 19 \bmod 37 \rightarrow 19$

$12 \bmod 9 \rightarrow 3$; used[3] $\rightarrow 22$

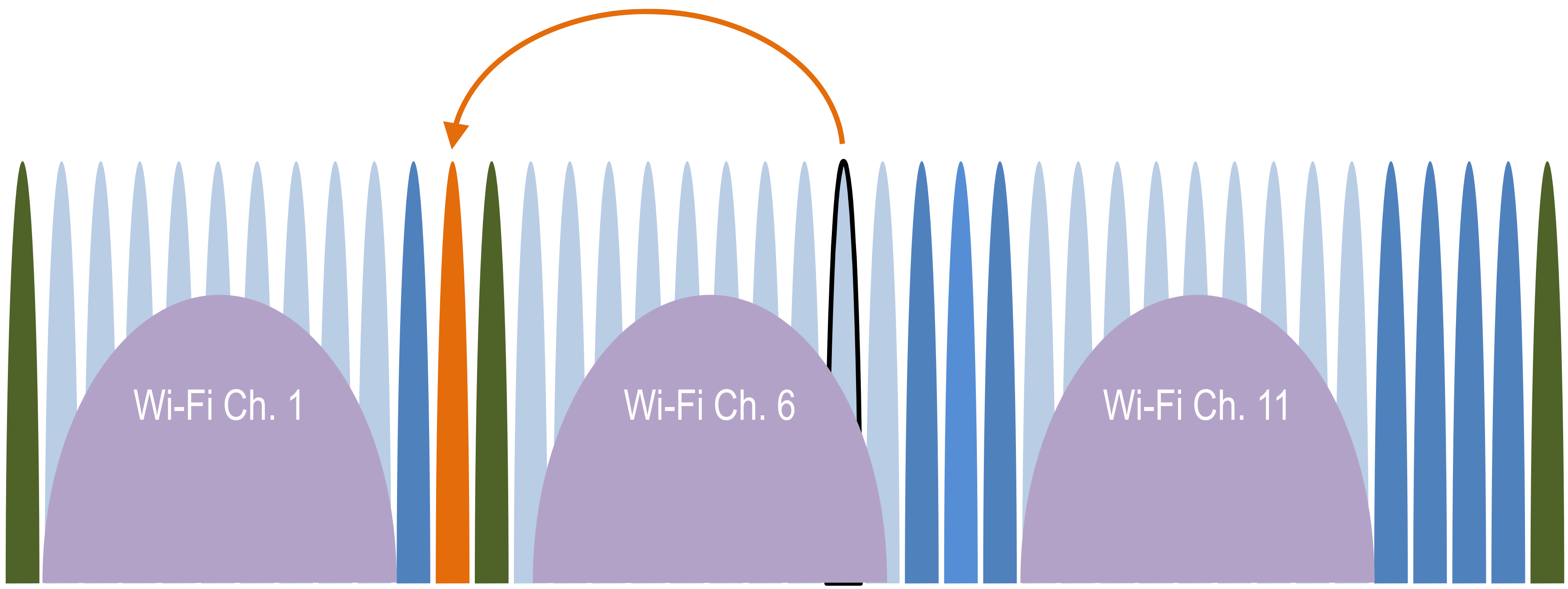


ADAPTIVE FREQUENCY HOPPING

fn = 19, hop = 7, used = [9, 10, 21, 22, 23, 33, 34, 35, 36]

“unused”; $f_{n+1} = f_n + 7 \bmod 37 = 26 \bmod 37 \rightarrow 26$

$19 \bmod 9 \rightarrow 1$; used[1] $\rightarrow 10$

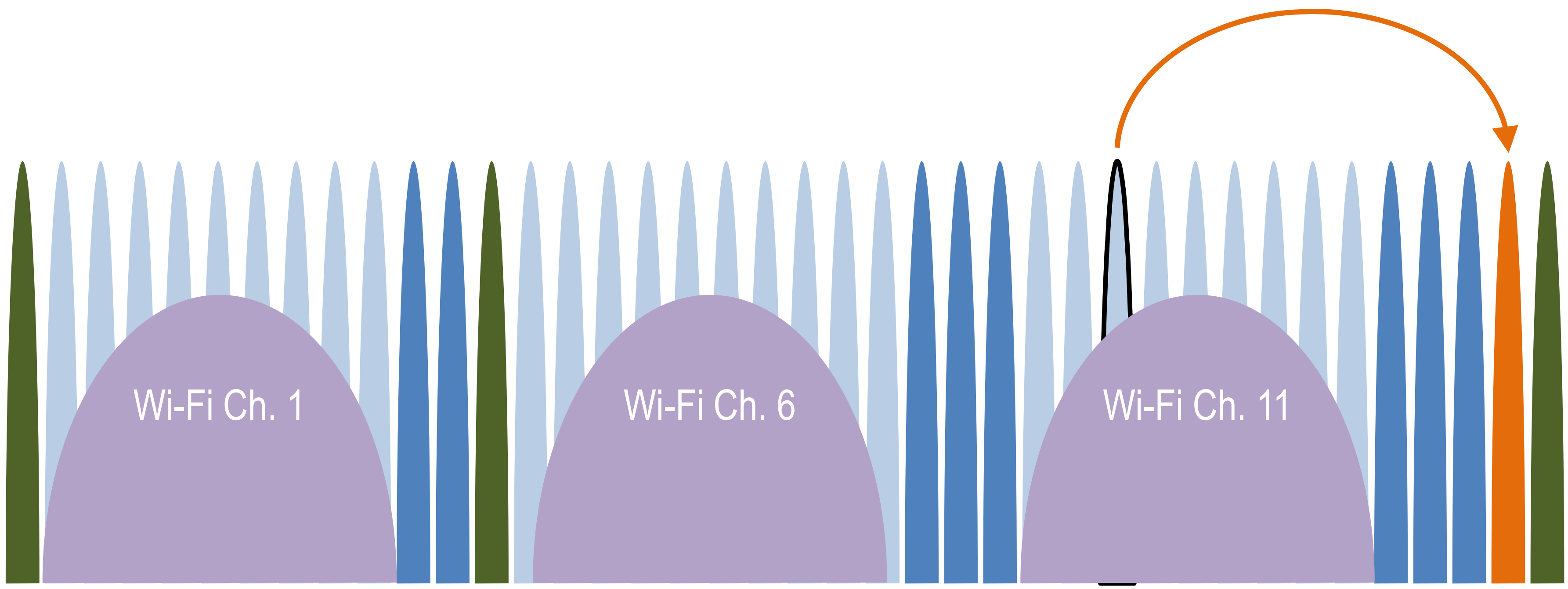


ADAPTIVE FREQUENCY HOPPING

fn = 26, hop = 7, used = [9, 10, 21, 22, 23, 33, 34, 35, 36]

“unused”; $f_{n+1} = f_n + 7 \bmod 37 = 33 \bmod 37 \rightarrow 33$

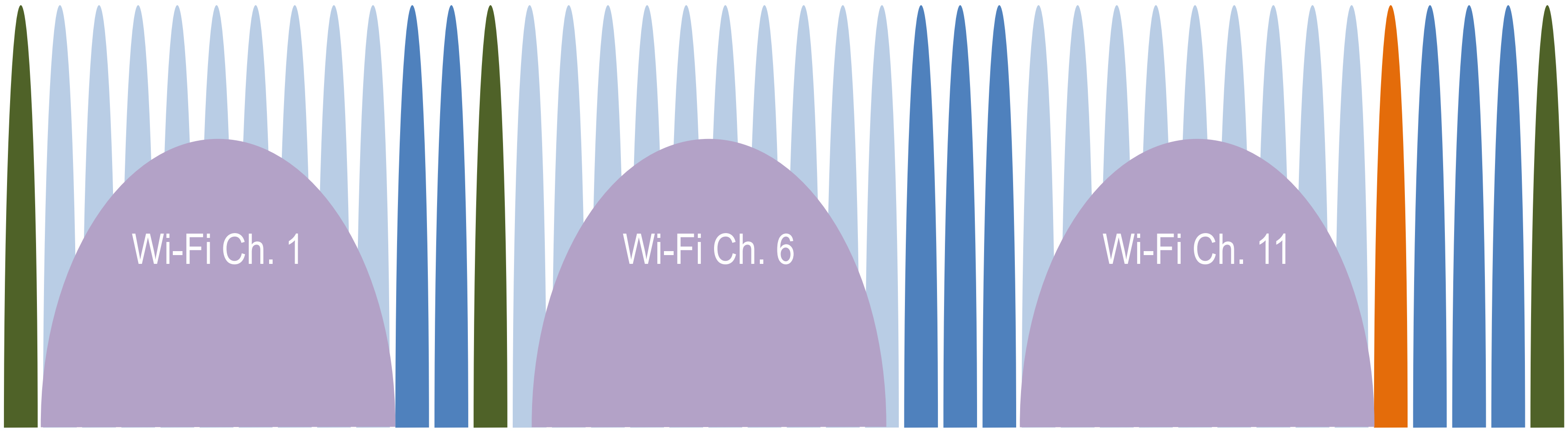
$26 \bmod 9 \rightarrow 8$; used[8] $\rightarrow 36$



ADAPTIVE FREQUENCY HOPPING

fn = 33, hop = 7, used = [9, 10, 21, 22, 23, 33, 34, 35, 36]

“used”; $f_{n+1} = f_n + 7 \bmod 37 = 40 \bmod 37 \rightarrow 3$

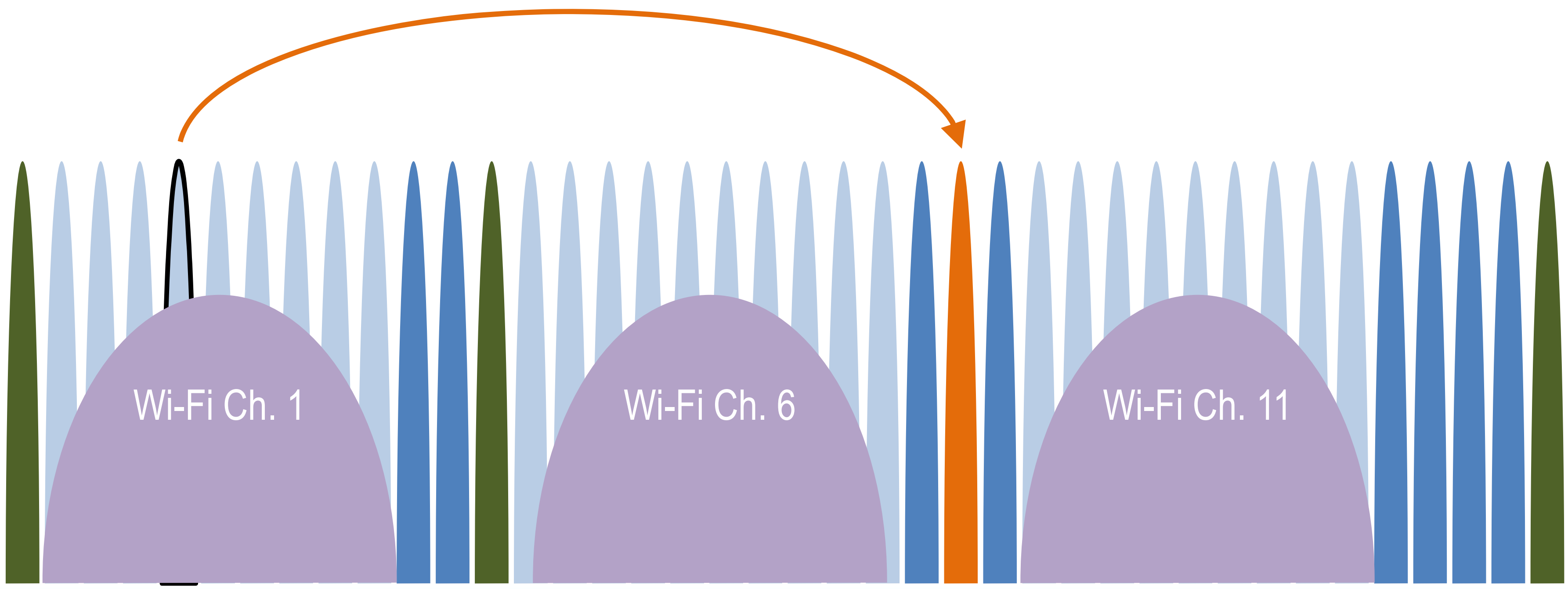


ADAPTIVE FREQUENCY HOPPING

fn = 3, hop = 7, used = [9, 10, 21, 22, 23, 33, 34, 35, 36]

"unused"; $f_{n+1} = f_n + 7 \bmod 37 = 10 \bmod 37 \rightarrow 10$

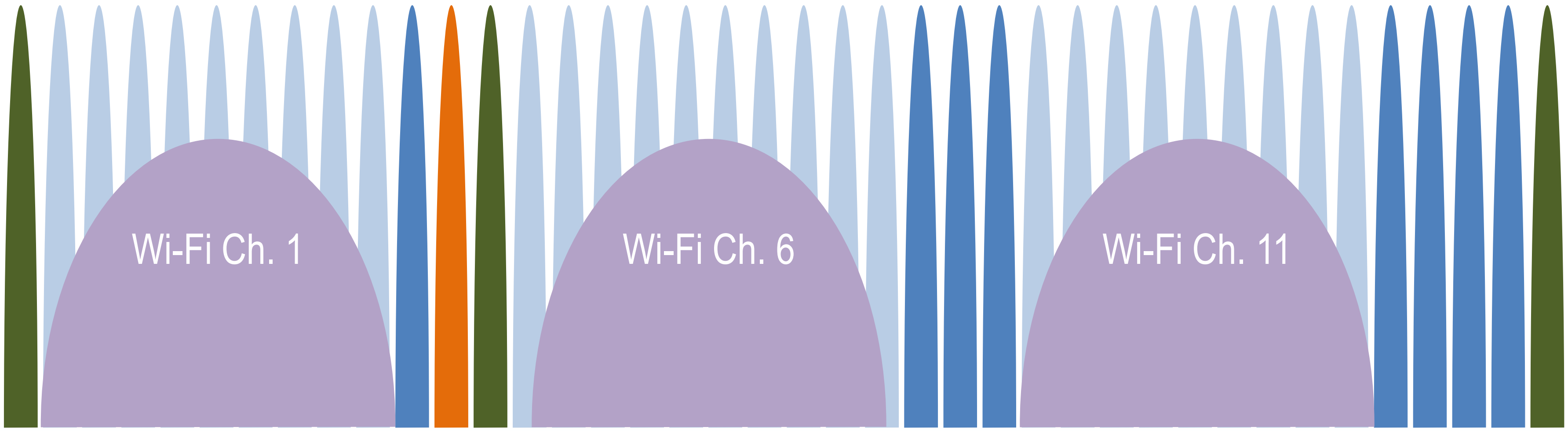
$3 \bmod 9 \rightarrow 3$; used[3] $\rightarrow 22$

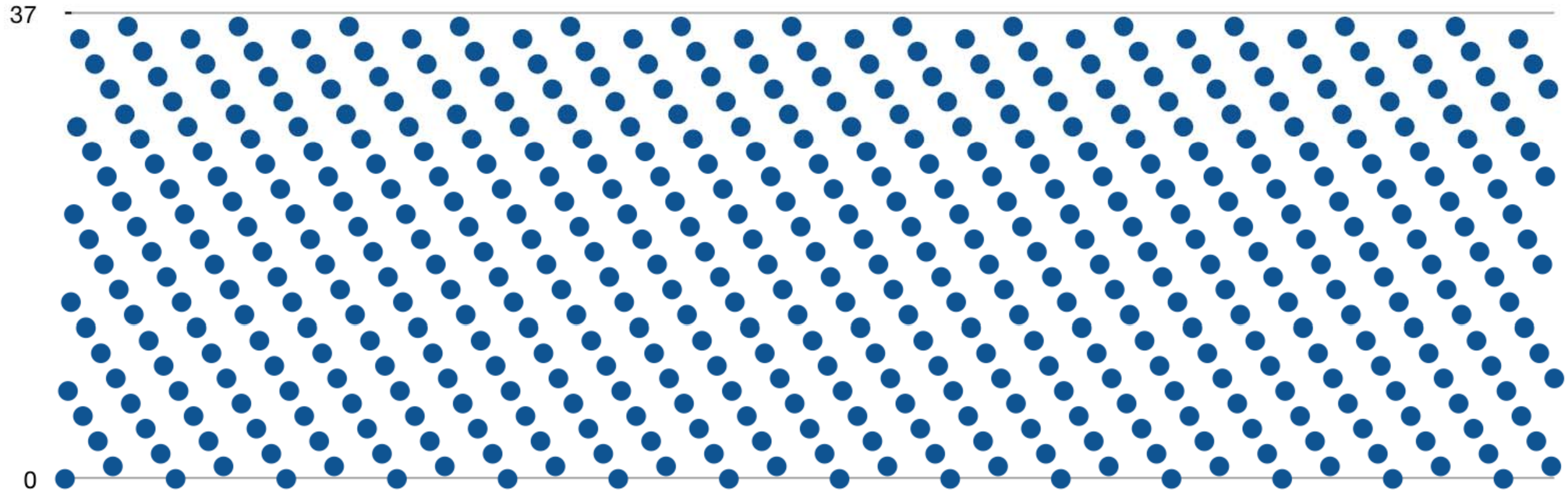


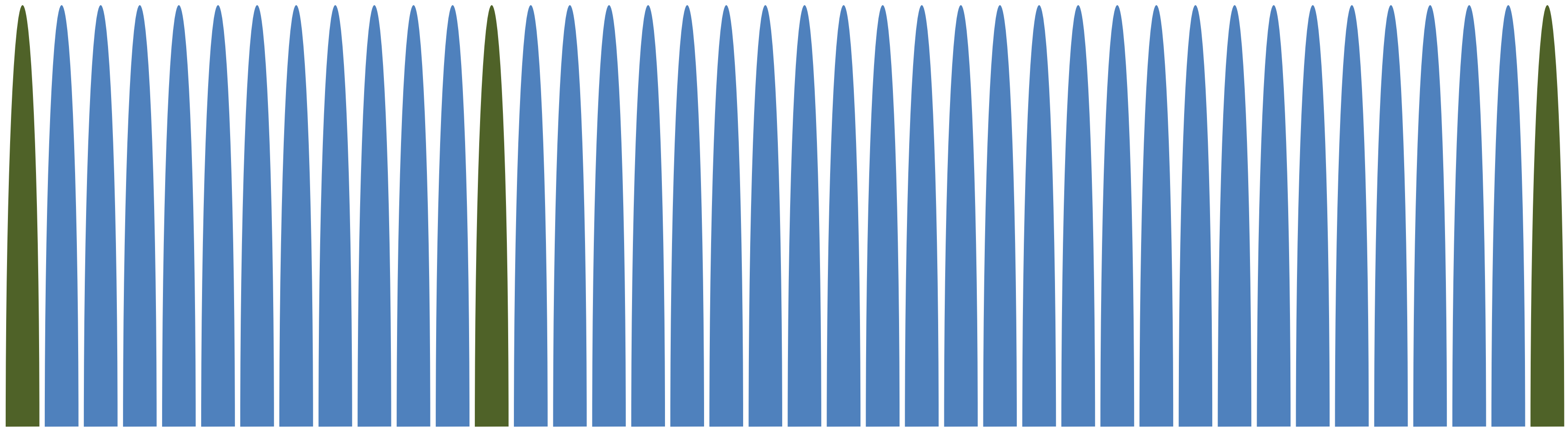
ADAPTIVE FREQUENCY HOPPING

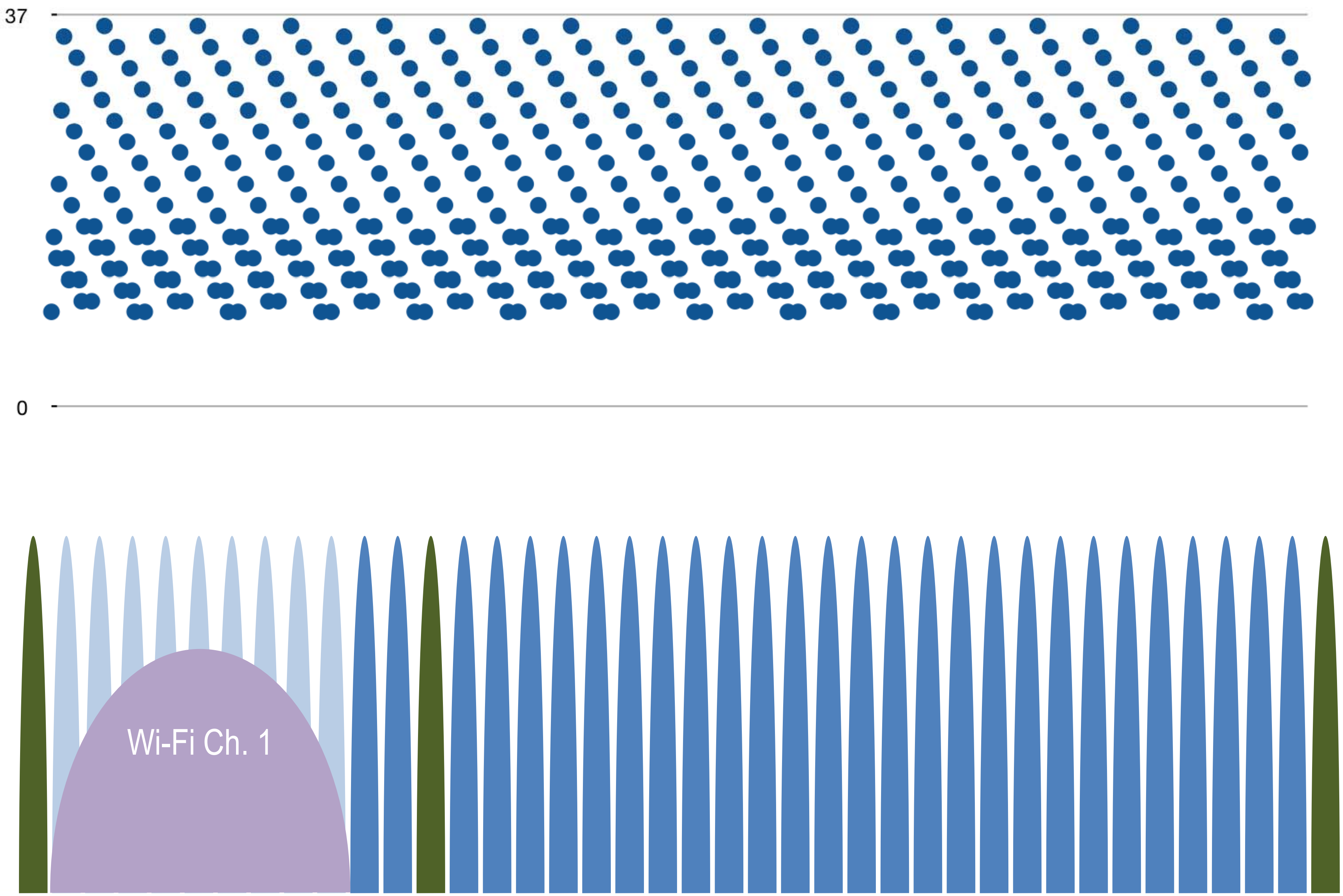
fn = 10, hop = 7, used = [9, 10, 21, 22, 23, 33, 34, 35, 36]

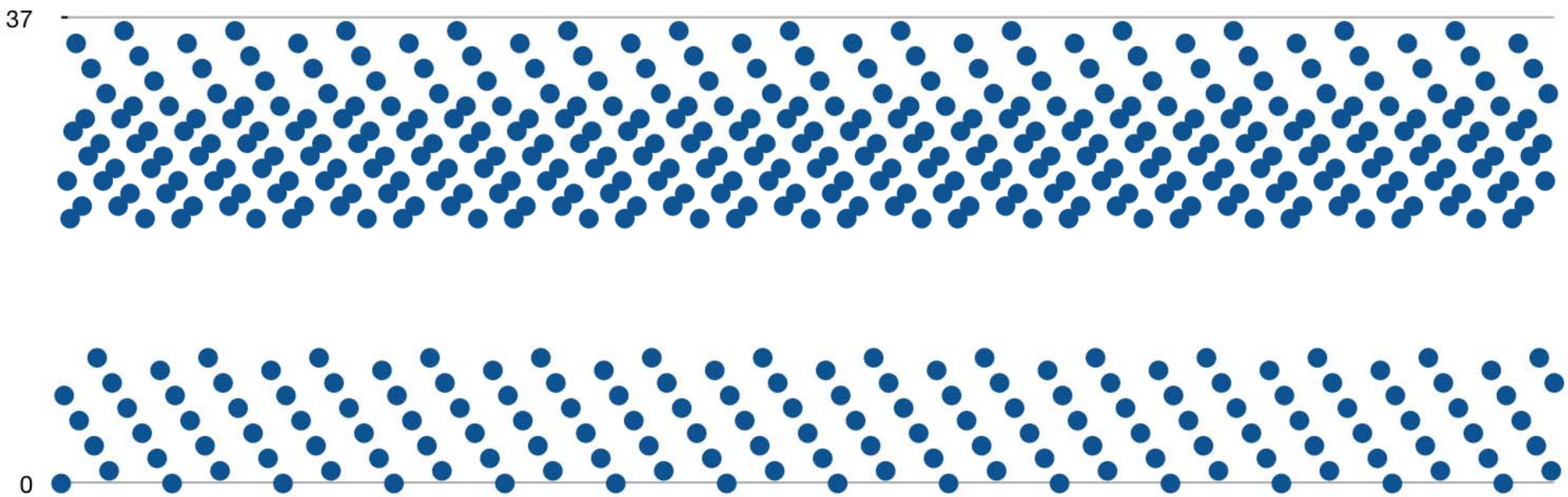
“used”; $f_{n+1} = f_n + 7 \bmod 37 = 17 \bmod 37 \rightarrow 17$

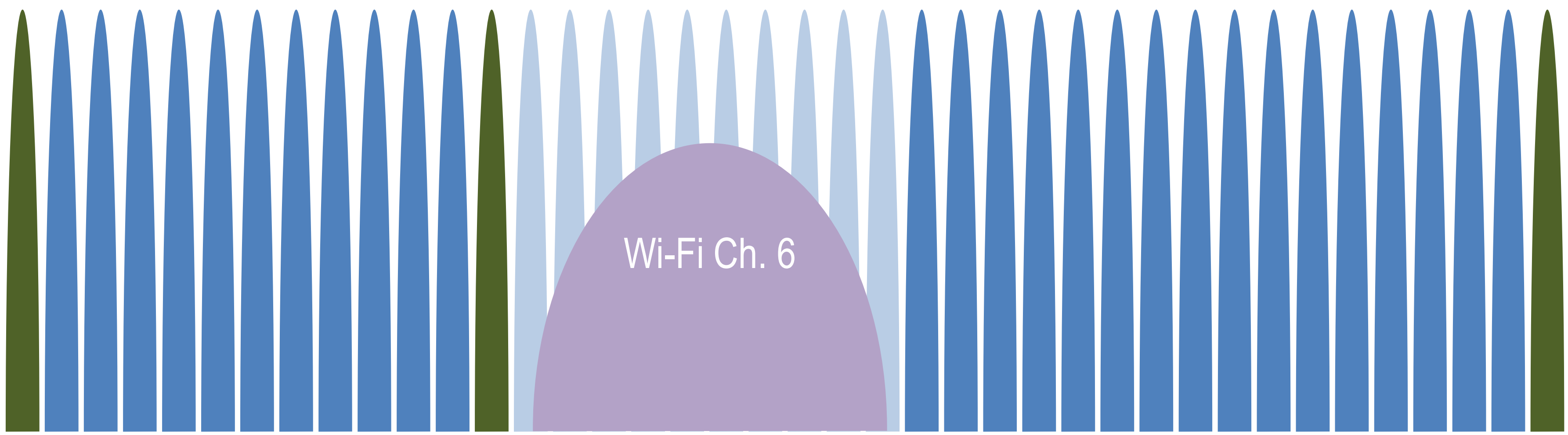


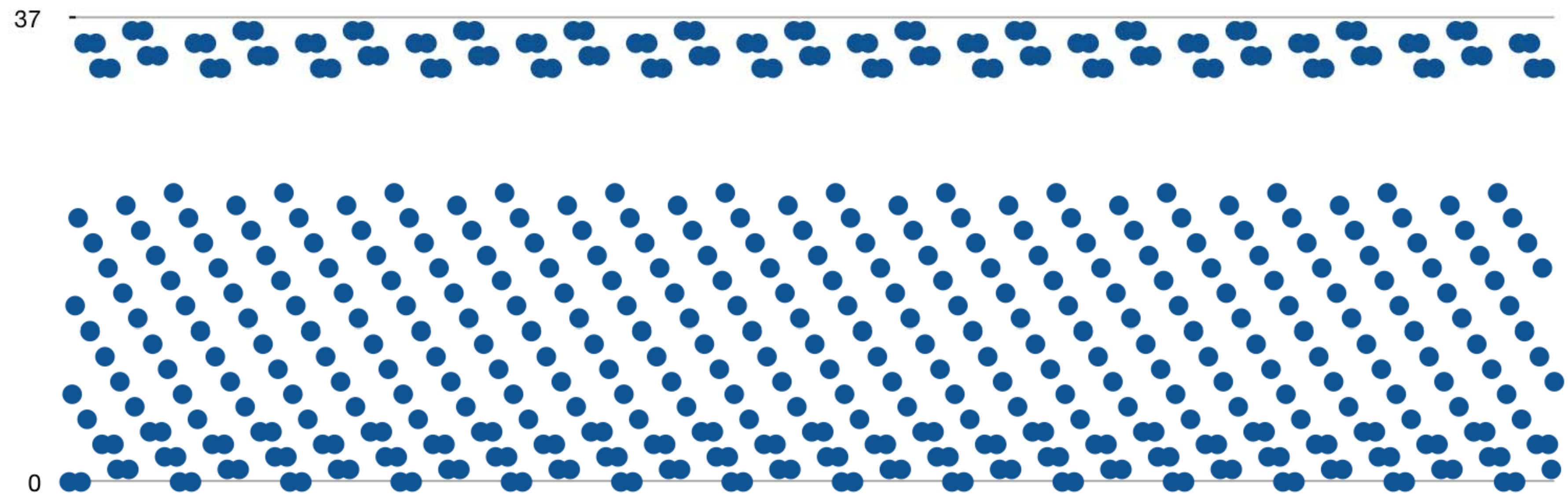

37
0

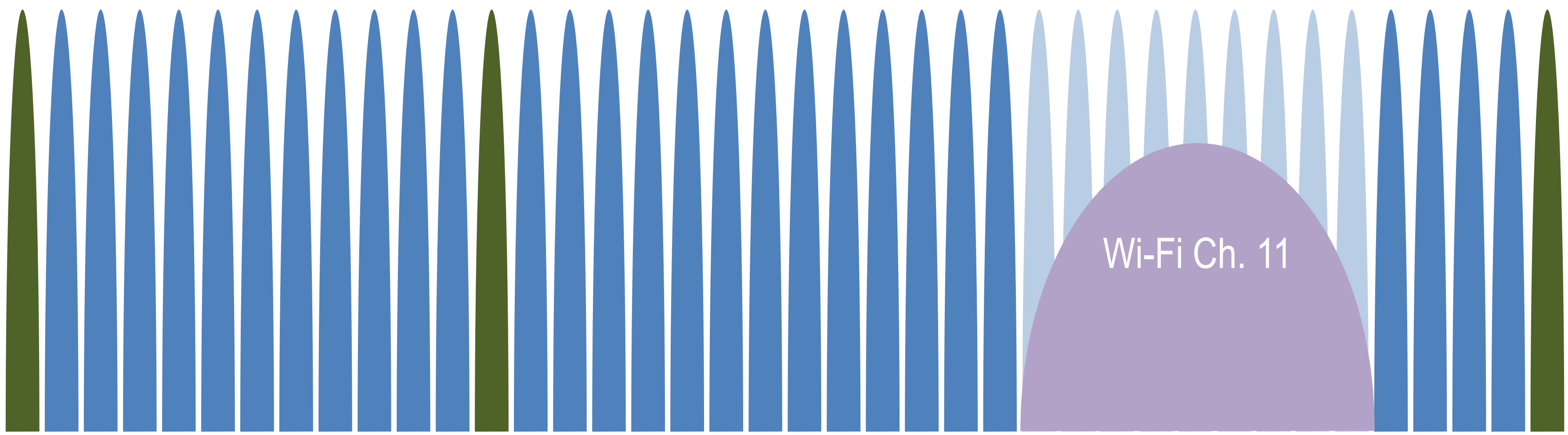


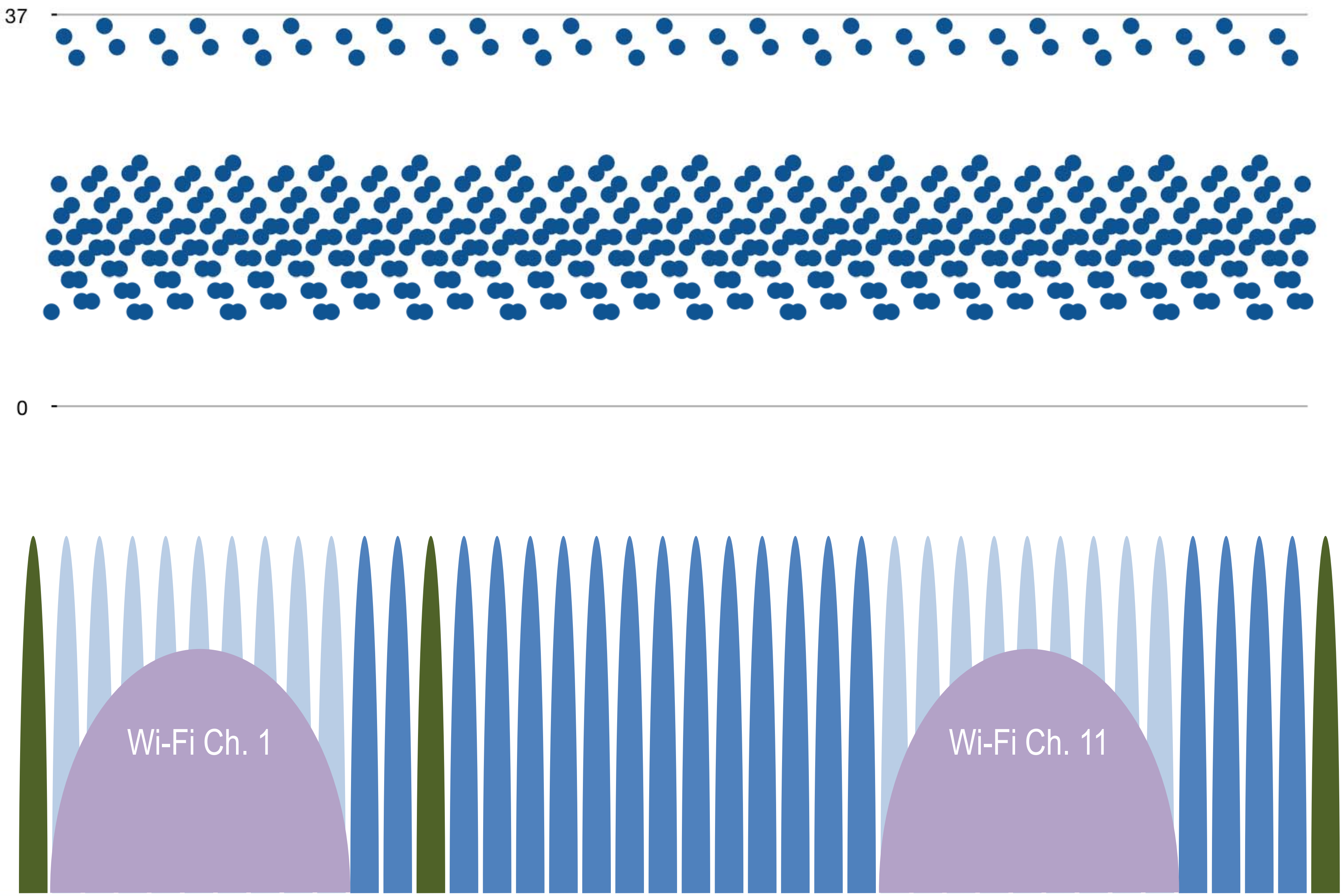

37
0
Wi-Fi Ch. 1

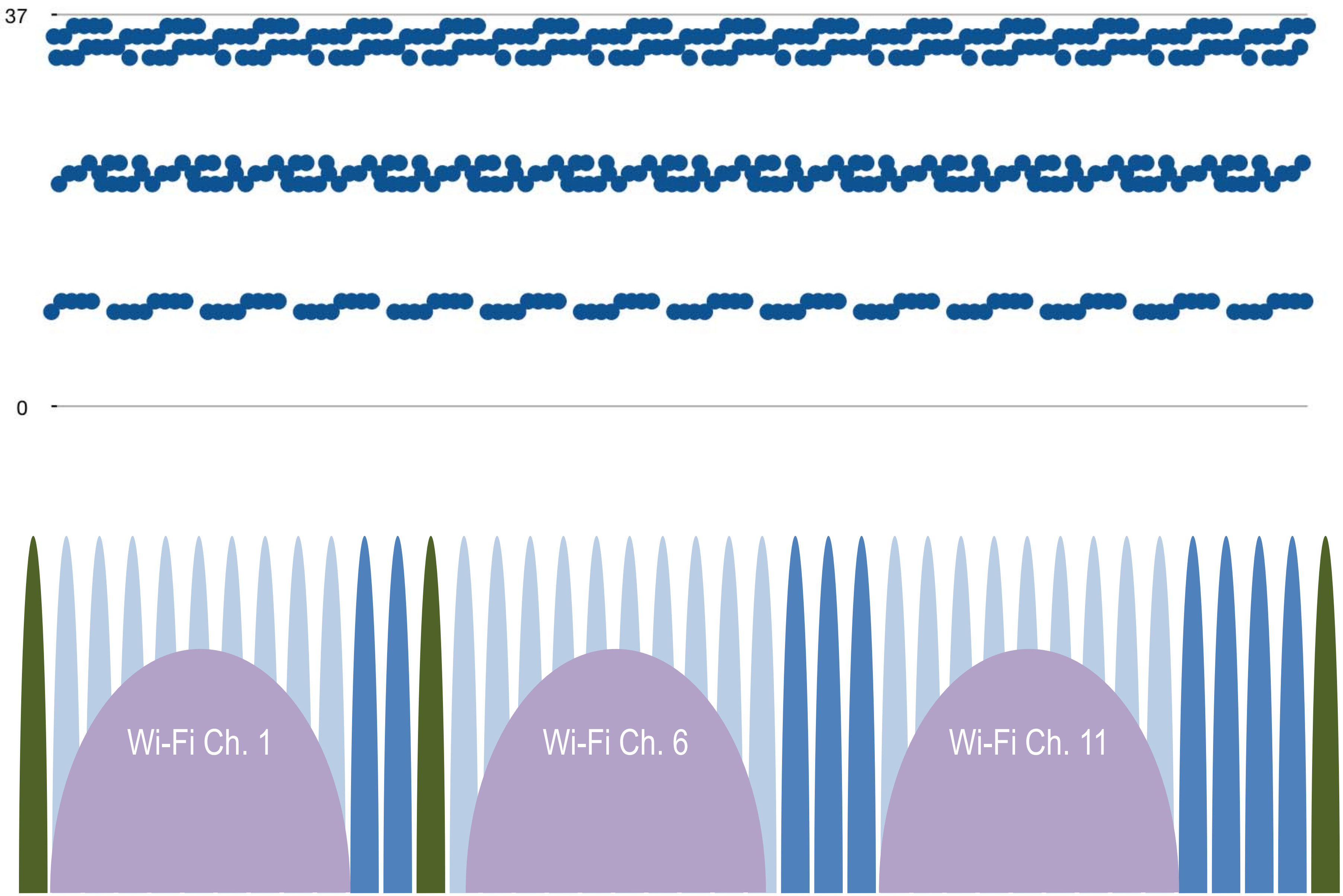

37
0


Wi-Fi Ch. 6


37
0


Wi-Fi Ch. 11


37
0
Wi-Fi Ch. 1
Wi-Fi Ch. 11


37
0
Wi-Fi Ch. 1
Wi-Fi Ch. 6
Wi-Fi Ch. 11

LINK LAYER CONTROL PROCEDURES

LLID = 11

Most procedures can only be initiated from Master

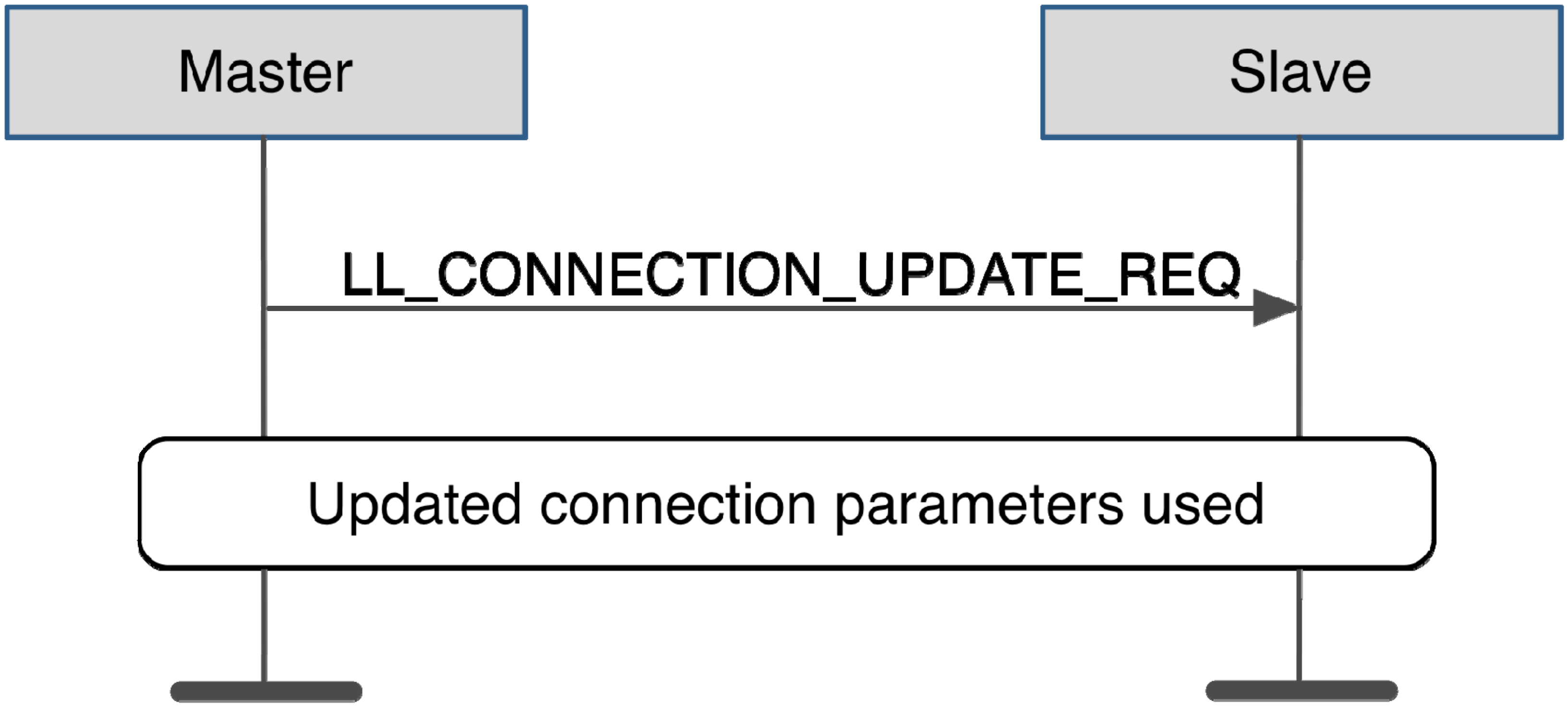
- except Termination Procedure
- except Version Exchange

LINK LAYER CONTROL PROCEDURES

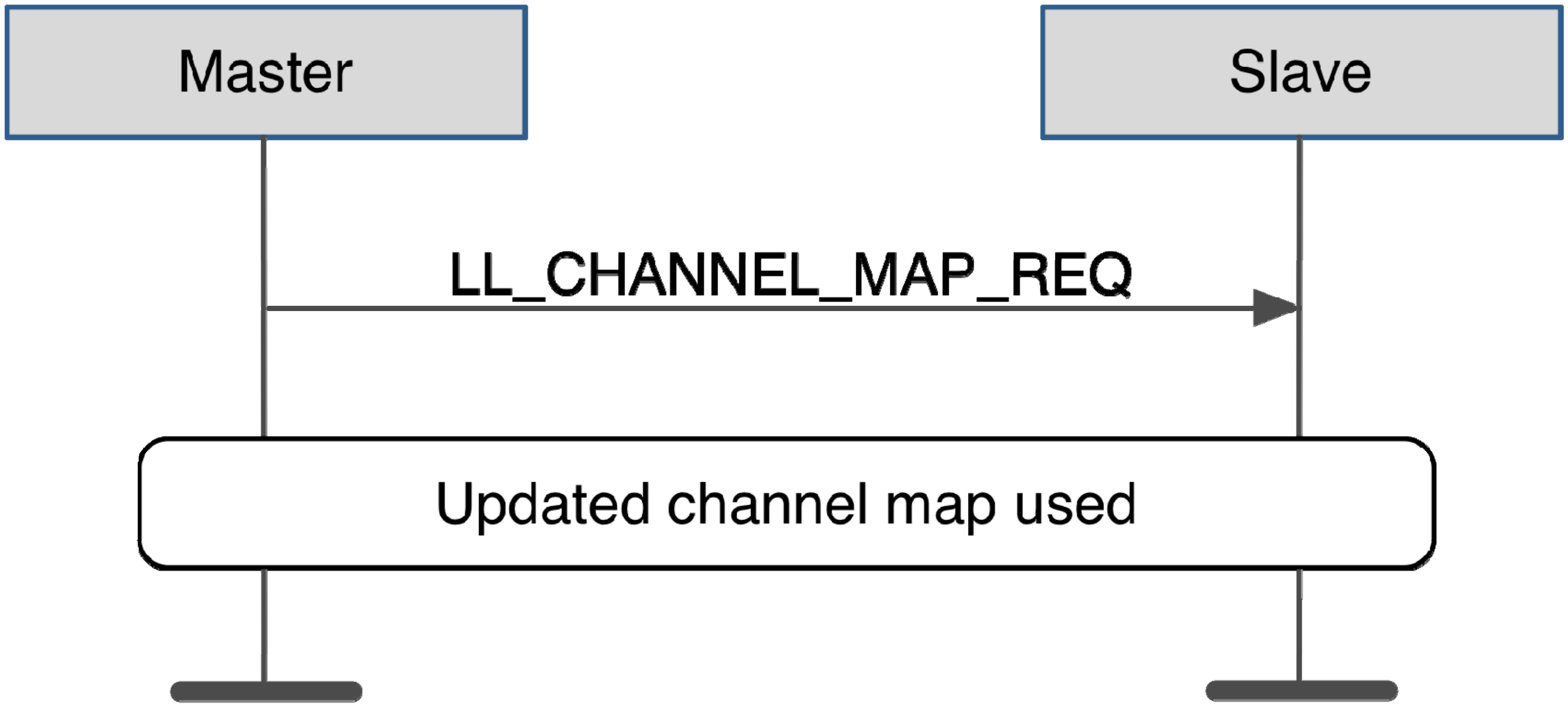
Name	Description
Connection Update Procedure	Update the connection intervals
Channel Map Update Procedure	Update the adaptive frequency hopping map
Encryption Start Procedure	Start encryption using a Long Term Key
Encryption Pause Procedure	Pause encryption, to change Long Term Key
Feature Exchange Procedure	Exchange the current supported feature set
Version Exchange Procedure	Exchange the current version information
Termination Procedure	Voluntary terminate the connection

Opcode	Control PDU Name	Description
0x00	LL_CONNECTION_UPDATE_REQ	Update Connection Intervals
0X01	LL_CHANNEL_MAP_REQ	Update Channel Maps
0X02	LL_TERMINATE_IND	Disconnect the connection
0X03	LL_ENC_REQ	Encryption Request
0X04	LL_ENC_REQ	Encryption Response
0x05	LL_START_ENC_REQ	3-way Handshake for Starting Encryption
0x06	LL_START_ENC_RSP	3-way Handshake for Starting Encryption
0x07	LL_UNKNOWN_RSP	Control PDU Unknown
0x08	LL_FEATURE_REQ	Master sends Features to Slave
0x09	LL_FEATURE_RSP	Slave sends Features to Master
0x0A	LL_PAUSE_ENC_REQ	Pause Encryption to Refresh Keys
0x0B	LL_PAUSE_ENC_RSP	Pause Encryption to Refresh Keys
0x0C	LL_VERSION_IND	Version Exchange
0x0D	LL_REJECT_IND	Reject Control PDU

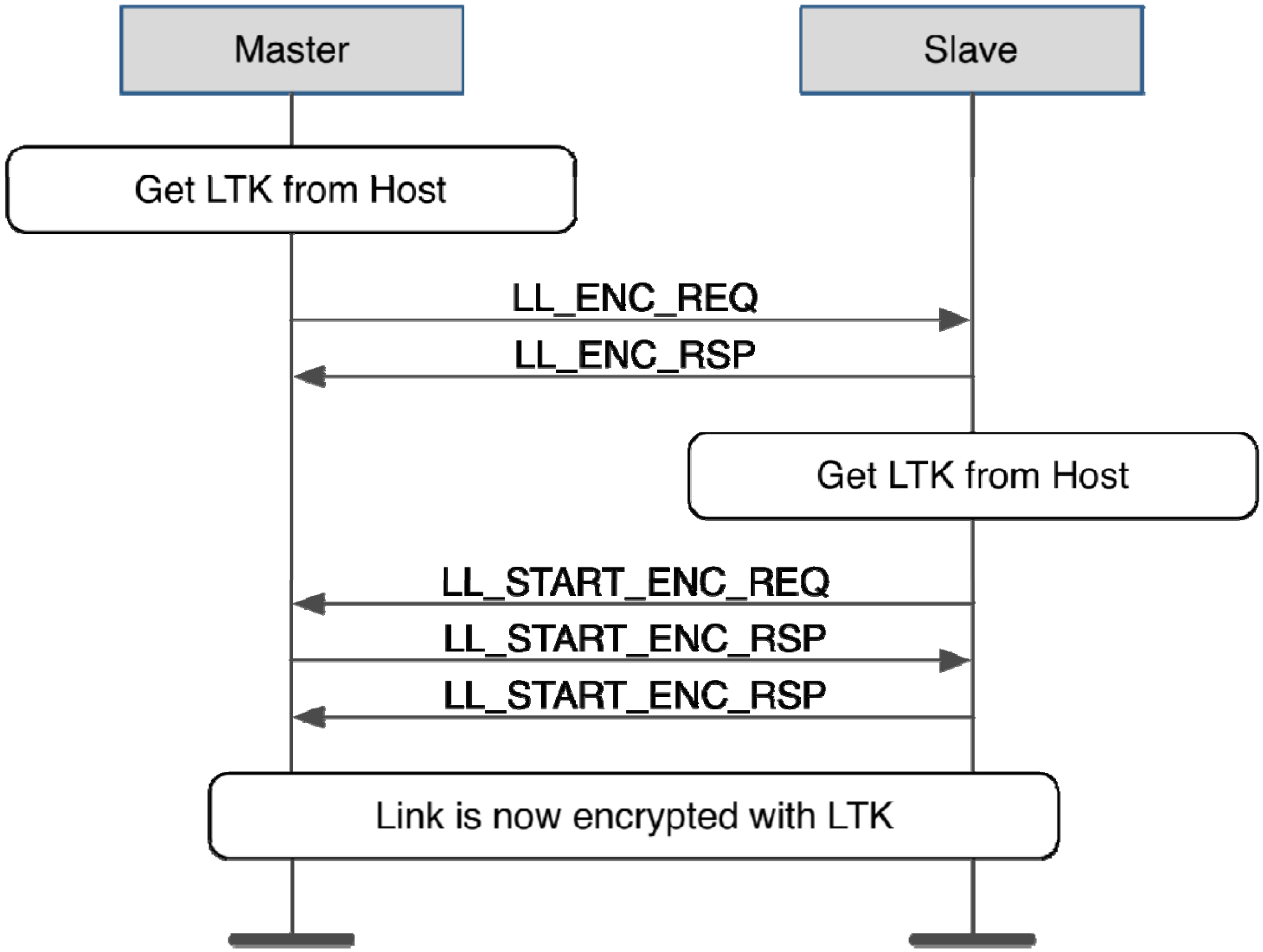
CONNECTION UPDATE



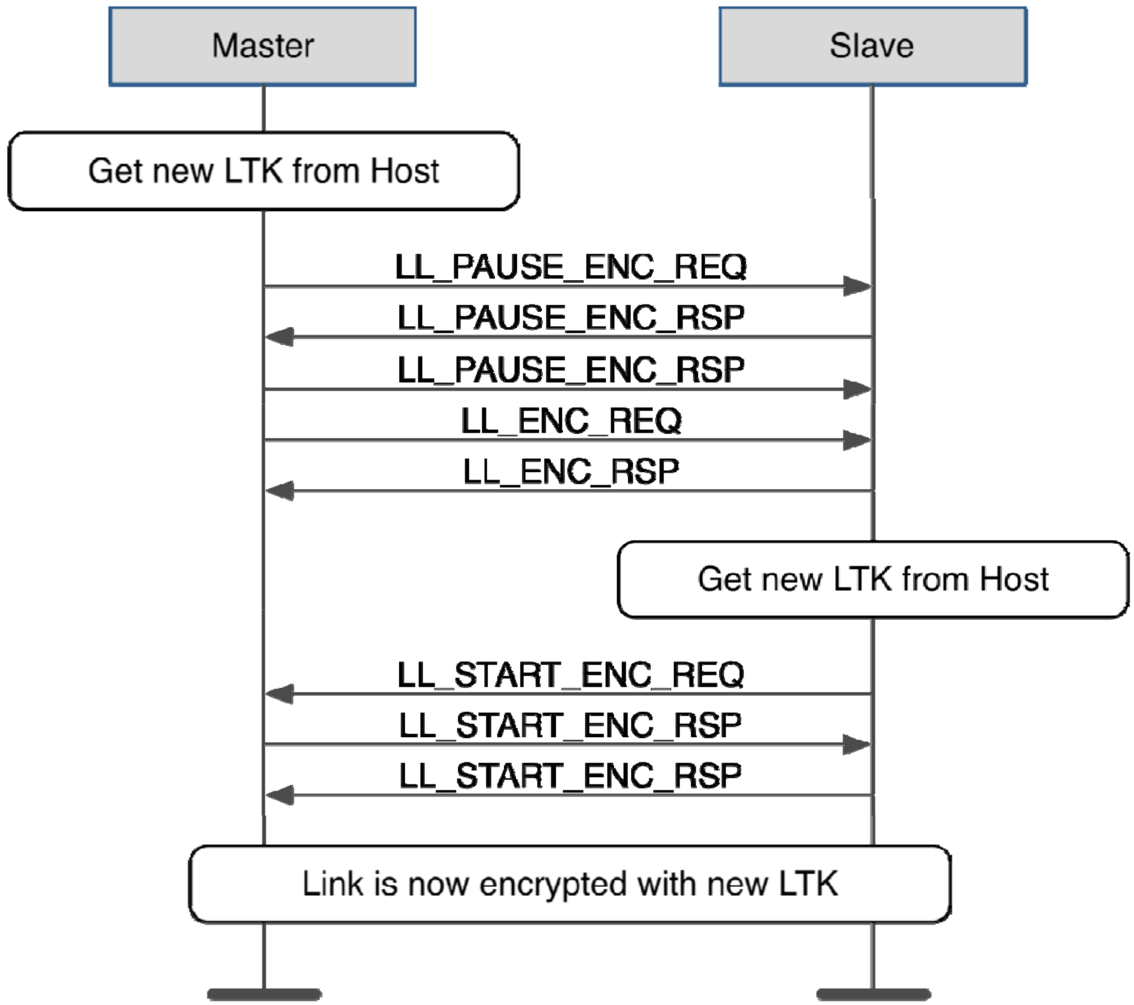
CHANNEL MAP UPDATE



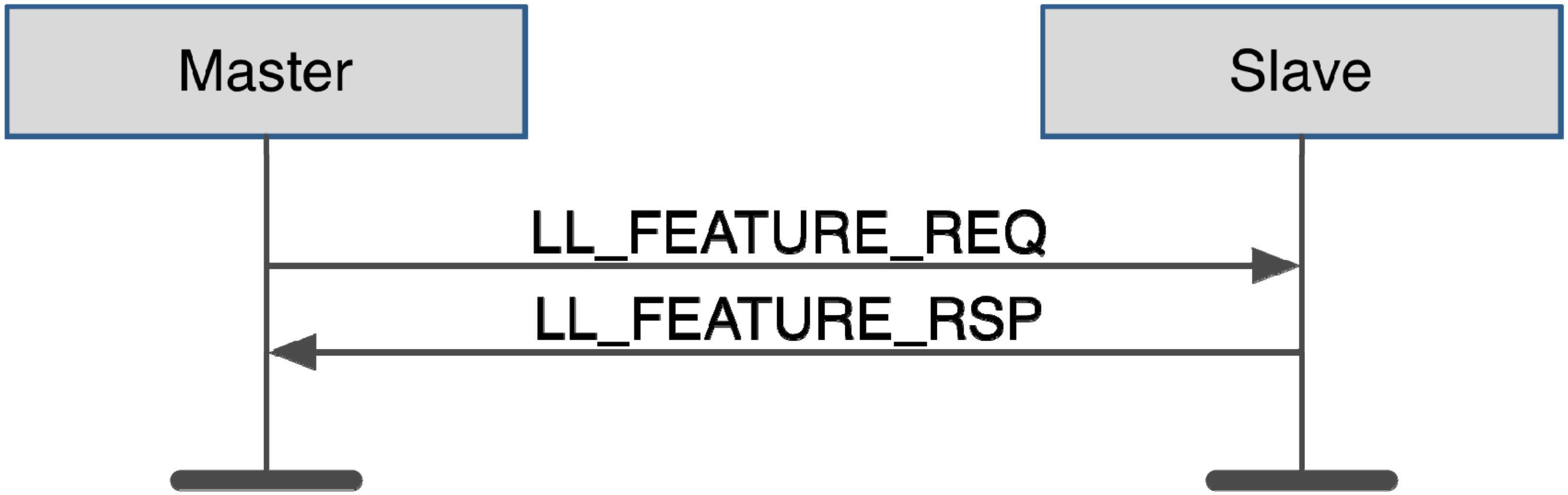
START ENCRYPTION



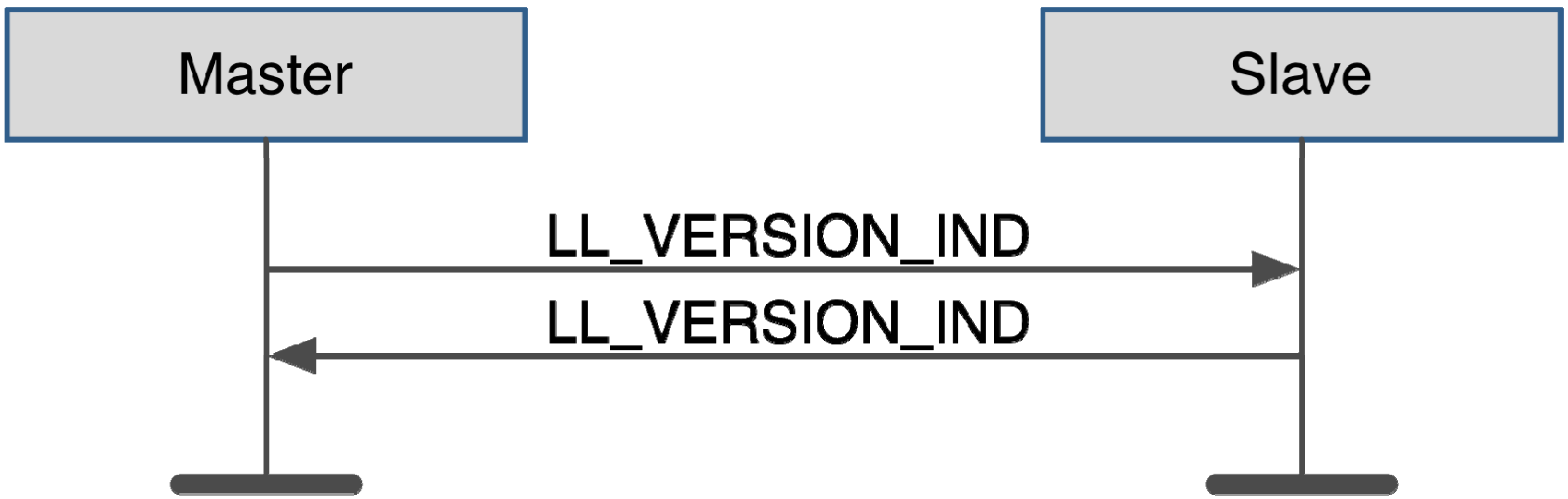
RESTART ENCRYPTION



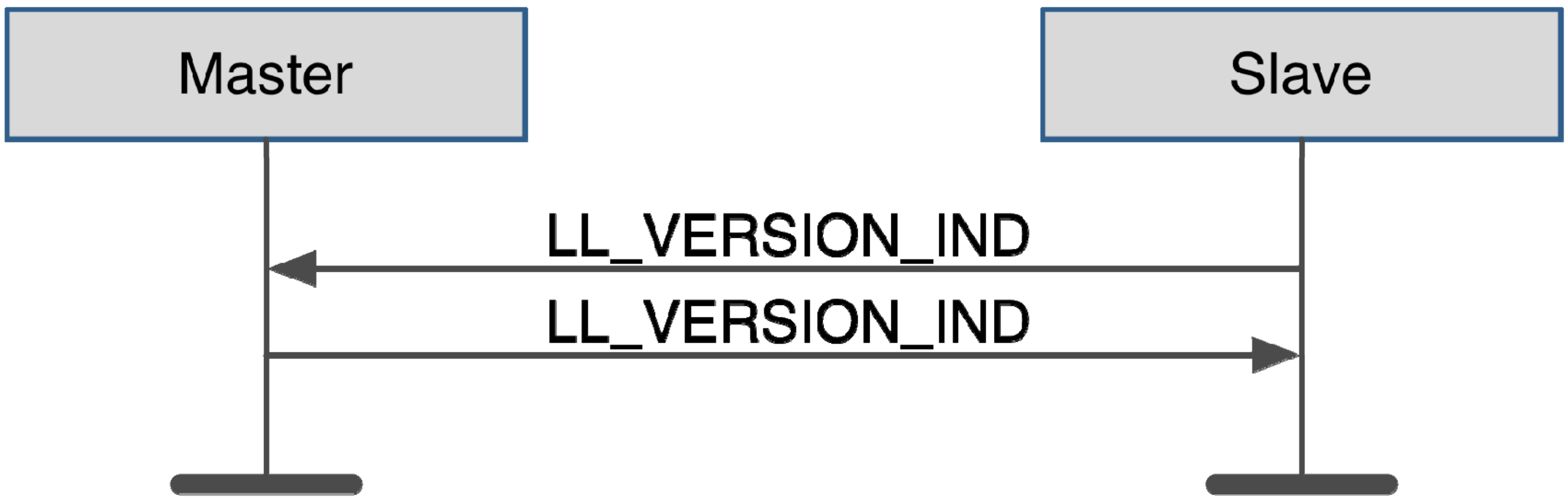
FEATURE EXCHANGE



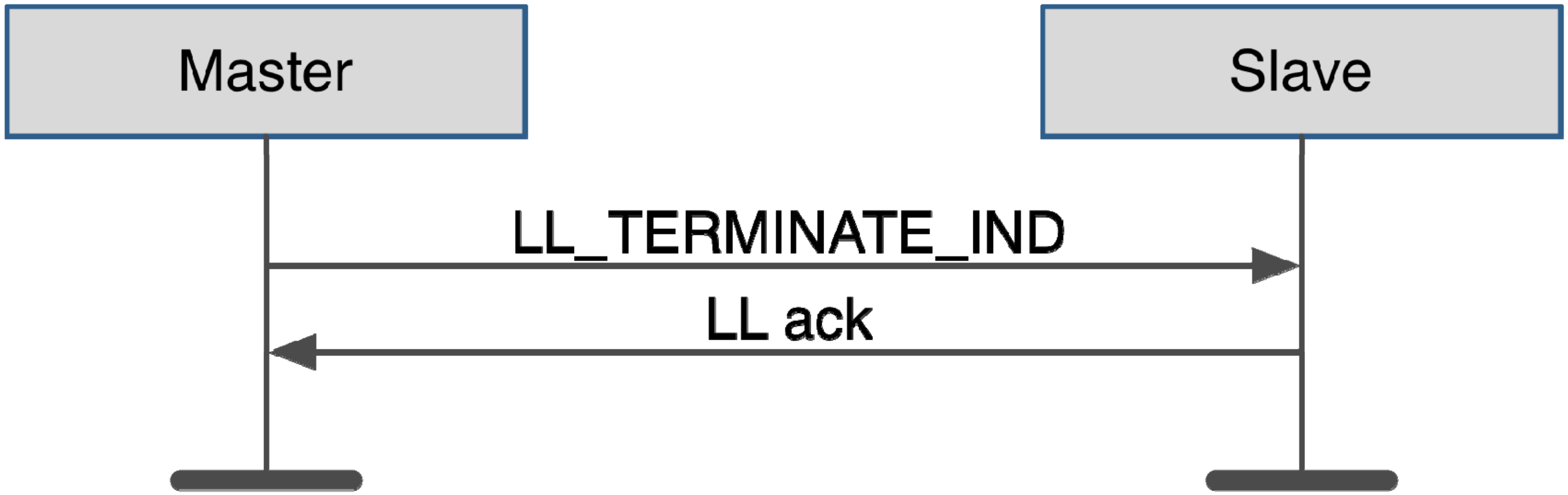
VERSION EXCHANGE



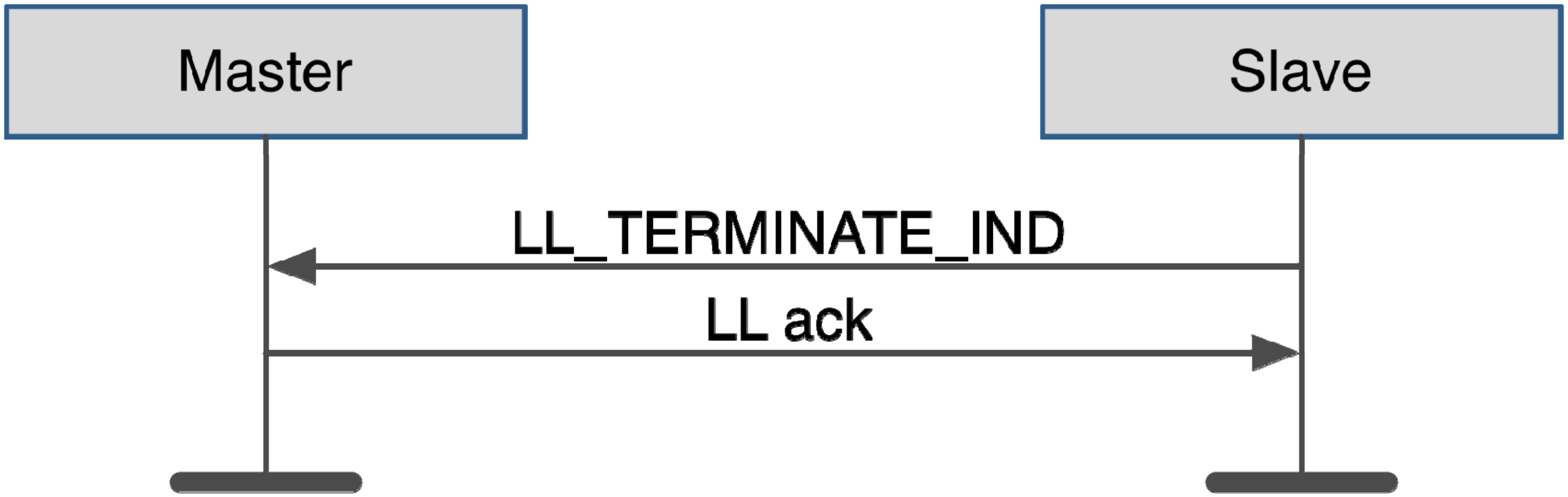
VERSION EXCHANGE



TERMINATE



TERMINATE



LINK LAYER ENCRYPTION

Uses AES 128 encryption block

Counter Mode Cipher Block Chaining Message Authentication Code

LINK LAYER ENCRYPTION

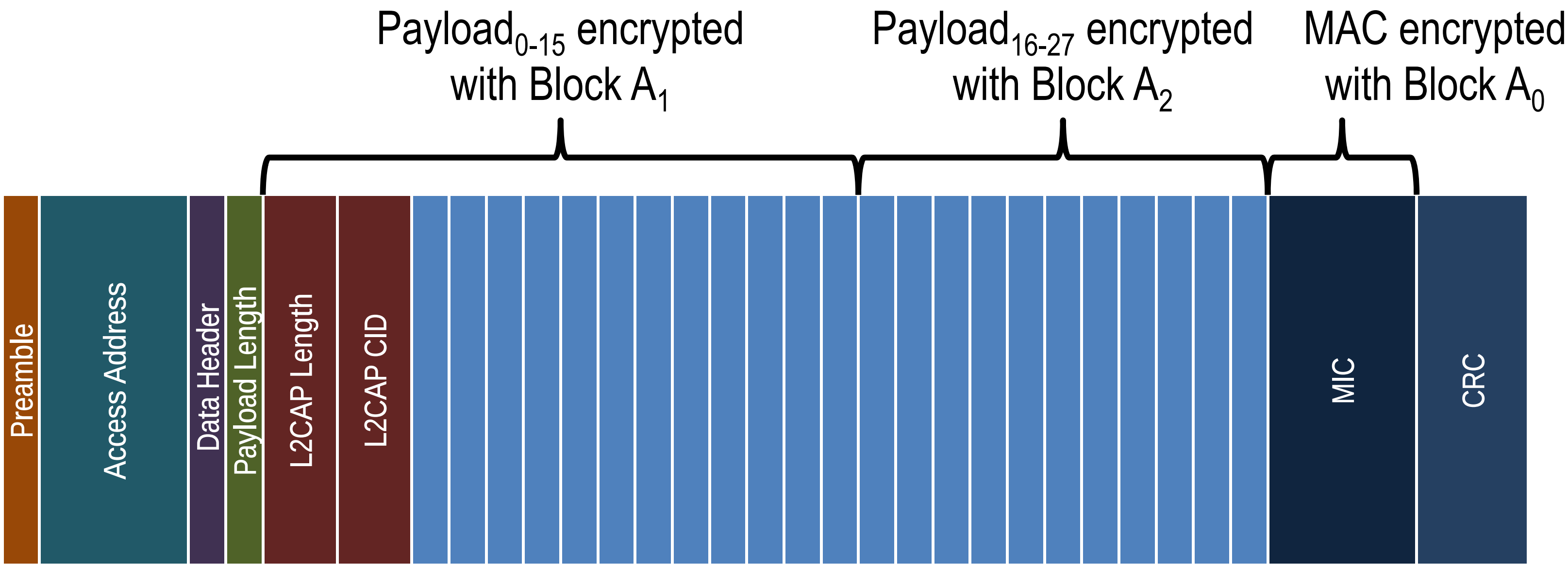
Uses AES 128 encryption block

Counter Mode CBC MAC

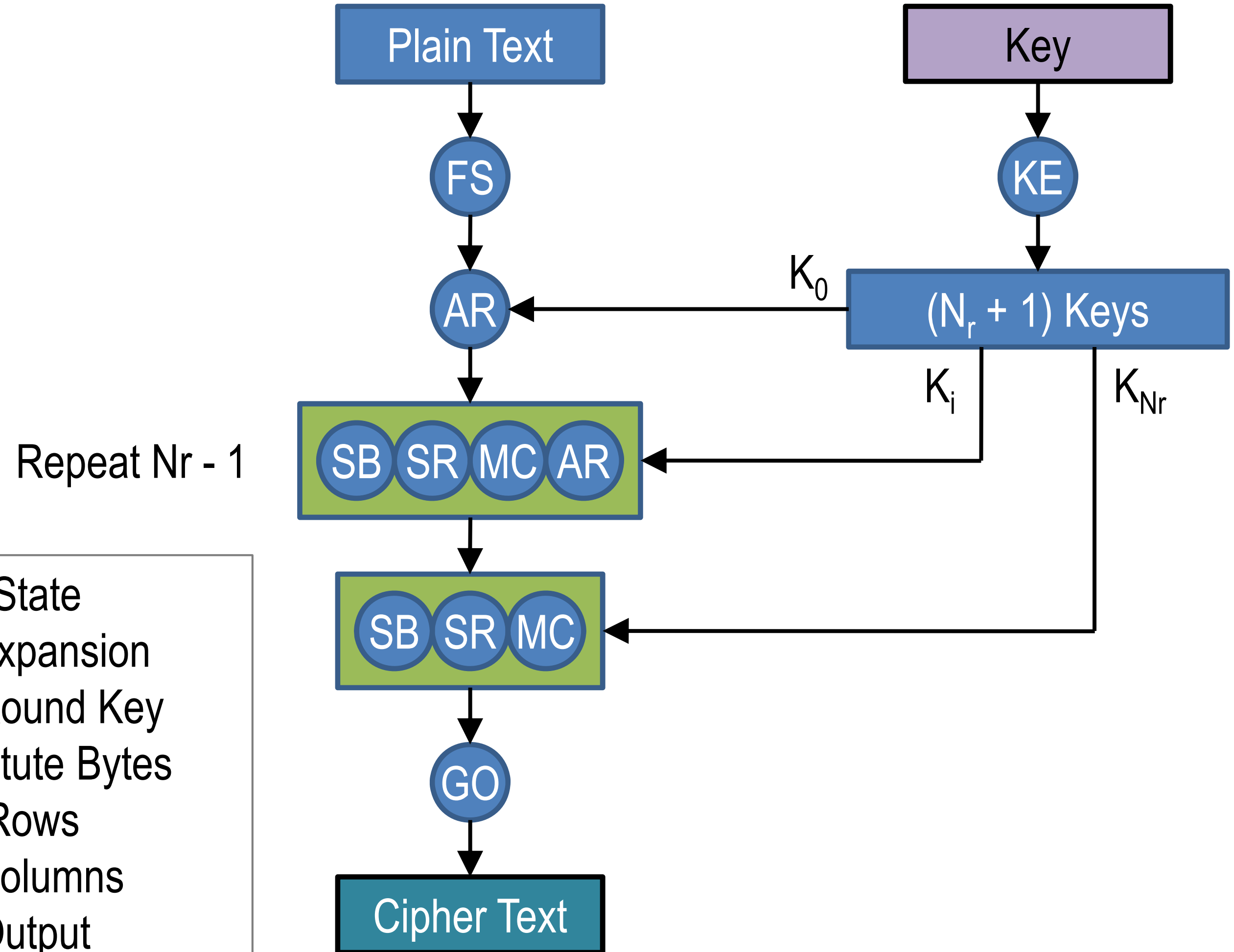
LINK LAYER ENCRYPTION

Uses AES 128 encryption block

and CCM as defined by RFC 3610

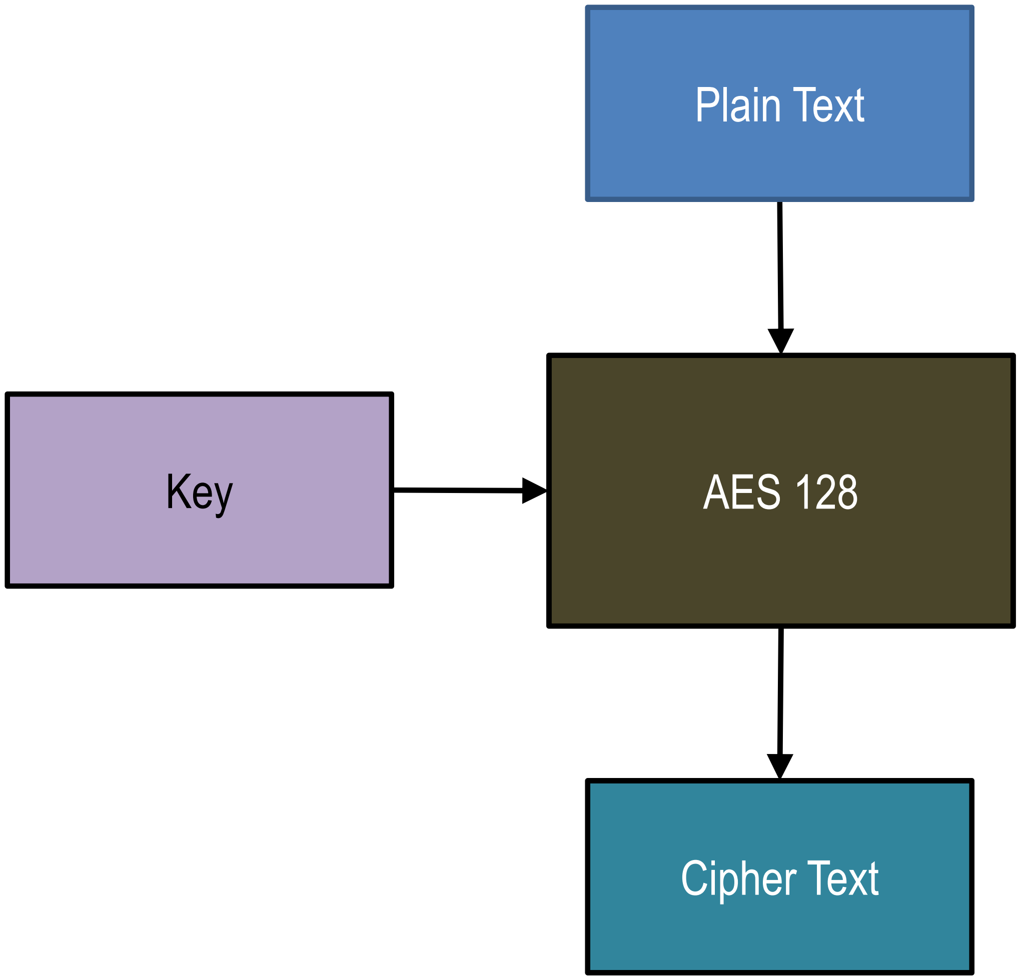


AES ENCRYPTION

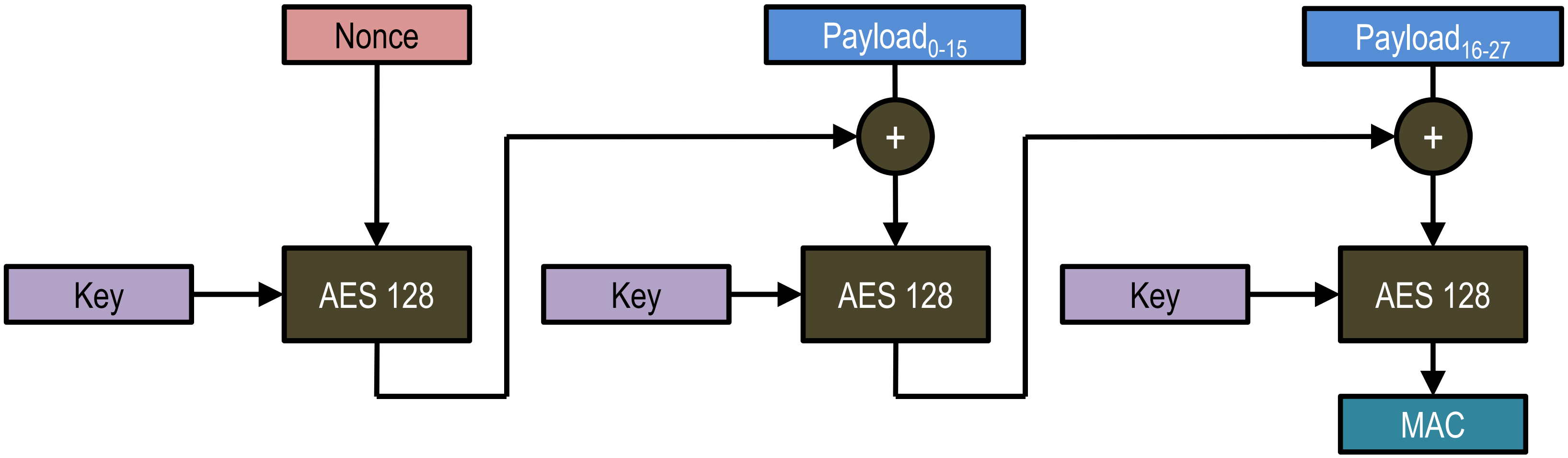


FS = Form State
KE = Key Expansion
AR = Add Round Key
SB = Substitute Bytes
SR = Shift Rows
MC = Mix Columns
GO = Get Output

AES BLOCK

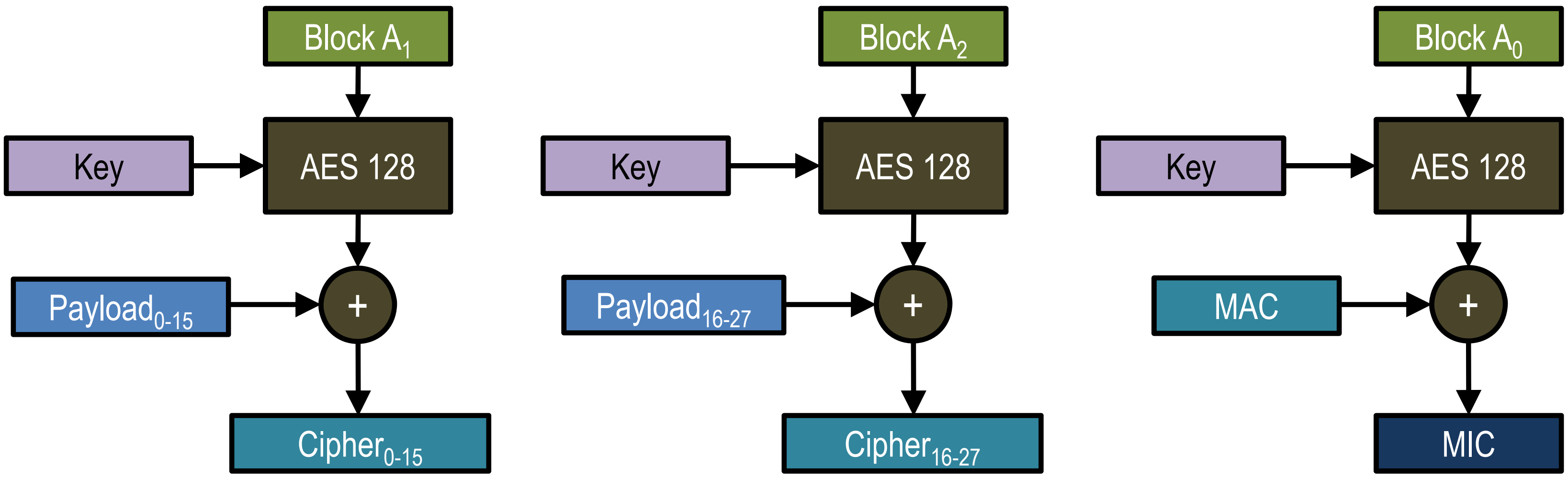


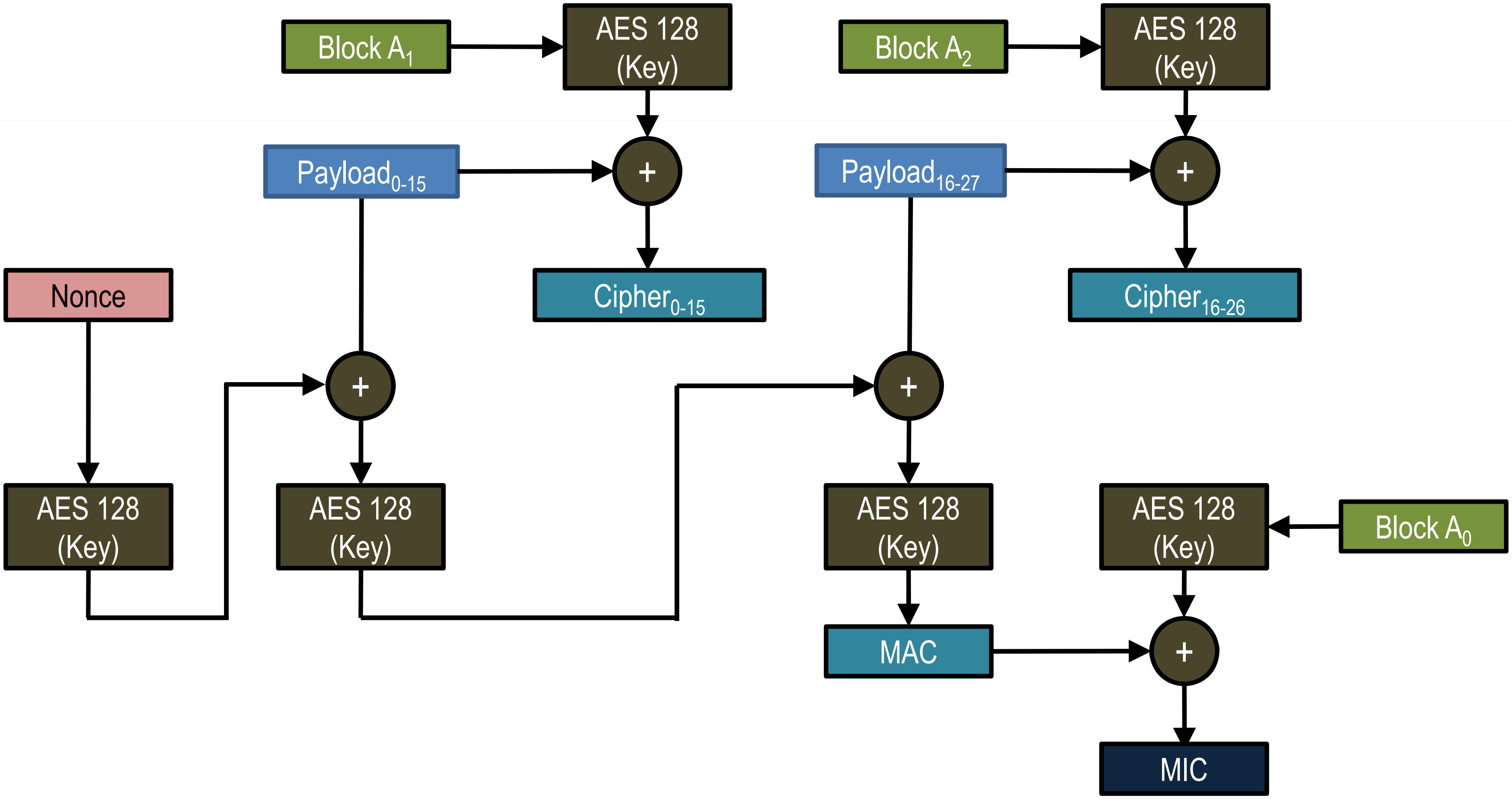
CIPHER BLOCK CHAINING MESSAGE AUTHENTICATION CODE



Nonce

AES CCM ENCRYPTION




Block A1
AES 128 (Key)
Block A2
AES 128 (Key)
Payload0-15
+
Payload16-27
+
Nonce
Cipher0-15
Cipher16-26
+
+
AES 128 (Key)
AES 128 (Key)
AES 128 (Key)
AES 128 (Key)
Block A0
MAC
+
MIC


P
AA
H
L
Cipher0-15
Cipher16-26
MIC
CRC

LIMITS

Maximum 2^{39} packets per LTK per direction

- Each packet can contain up to 27 octets data
- Max 13.5 Terabytes of data per connection
- ~12 years at maximum data rate

Then you have to change the encryption key

- using Restart Encryption Procedure

LINK LAYER SUMMARY

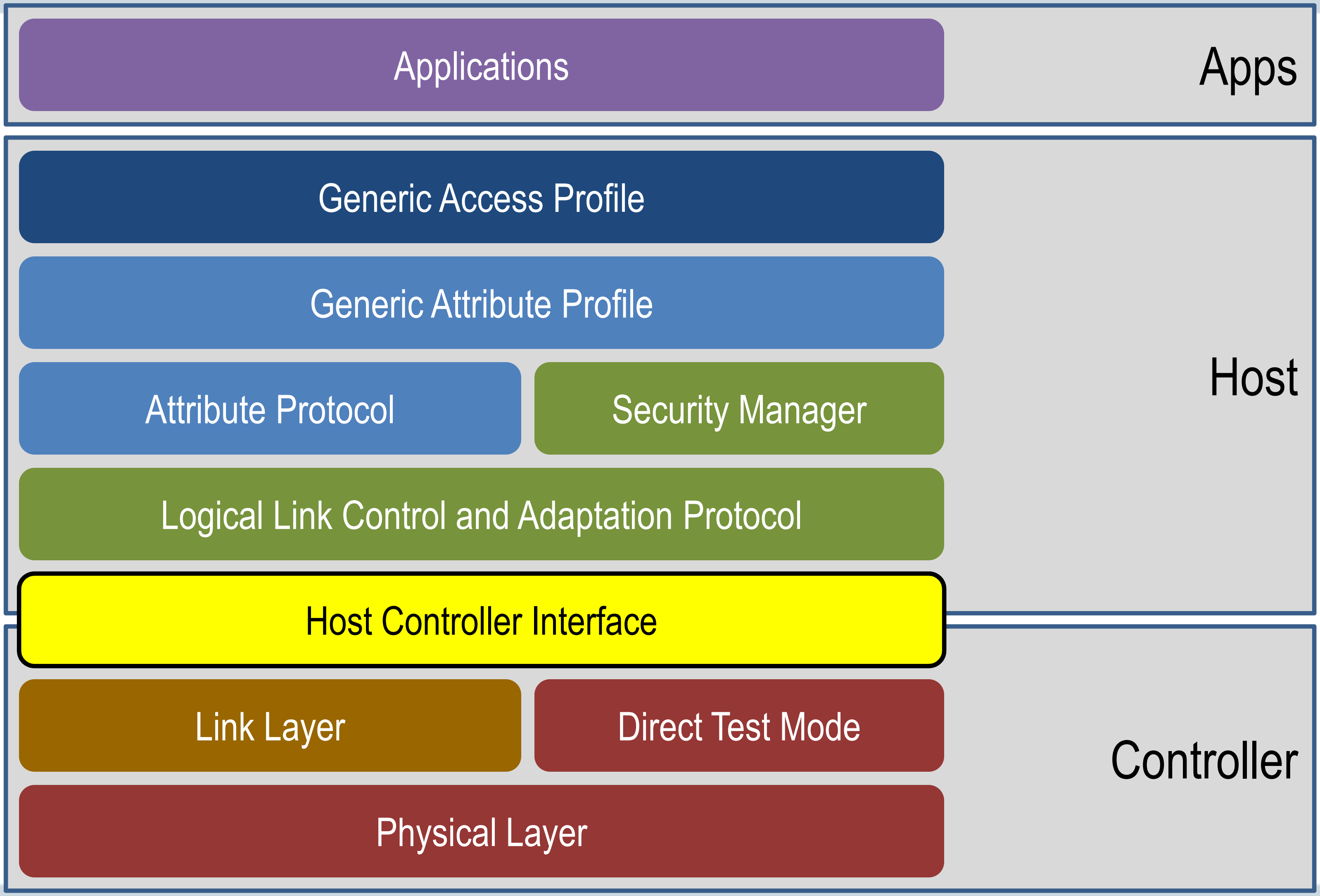
Low Complexity

- 1 packet format
- 2 PDU types – depending on Advertising / Data Channel
- 7 Advertising PDU Types
- 7 Link Layer Control Procedures

Useful Features

- Adaptive Frequency Hopping
- Low Power Acknowledgement
- Very Fast Connections

HOST CONTROLLER INTERFACE



HOST CONTROLLER INTERFACE (HCI)

Transport Layer

- UART
- USB
- SD
- 3-wire UART

Functional Layer

- Commands / Events / Data

COMMANDS / EVENTS / COMMANDS

Four HCI Packet Types

- HCI Command Packet
- HCI ACL Data Packet
- HCI Synchronous Data Packet (not used in LE)
- HCI Event Packet

Transports describe how to send these HCI Packet Types

UART HCI TRANSPORT LAYER

Each packet type assigned a HCI packet indicator

Command = 0x01, Data = 0x02, Event = 0x04

Send HCI packet indicator, and then HCI packet

0x01	HCI Command Packet
0x02	HCI ACL Data Packet
0x04	HCI Event Packet

UART HCI TRANSPORT LAYER

RS232 Settings

- Baud rate : manufacturer specific
- Data bits : 8
- Parity : none
- Stop bits : 1
- Flow Control : RTS / CTS
- Flow-off response time : manufacturer specific
- Configured as null-modem

USB HCI TRANSPORT LAYER

Defines one endpoint for ACL Data Packets

- Endpoint (out) 0x02 / (in) 0x82
- suggested max packet size is 32 or 64

Uses Control Endpoint for Commands

Uses Interrupt Endpoint for Events

- Endpoint (in) 0x81
- 1 ms interval

SECURE DIGITAL HCI TRANSPORT LAYER

Uses SDIO to transport HCI packets

- references SDIO Card Type-A Specification

Uses same codes as UART for SDIO Type-A service ID

- 0x01 = Command, 0x02 = Data, 0x04 = Event

3-WIRE UART HCI TRANSPORT LAYER

Places all HCI Packets on top of a SLIP layer

- RFC 1055

Adds framing to detect bit errors on UART

- allows use of long UART wires on product
- ideal for use when lots of interference in UART cables

Can support automatic UART Baud rate detection scheme

- has support for low power, software flow control

HCI FUNCTIONAL LAYER

Reuses existing HCI commands / events / data packets

  except where LE is different

All LE specific HCI commands have LE in name

All LE specific HCI events have LE in name

HCI BUFFERS

Dual mode controllers can either:

Expose one set of HCI buffers

Shared between BR/EDR and LE

OR

Expose two sets of HCI buffers

One set for BR/EDR and one set for LE

RANDOM DEVICE ADDRESSES

LE Rand

asks controller to generate a random number

very good random number using non-linear algorithms

LE Encrypt

asks controller to encrypt some plain text with key

LE Set Random Address

sets the random address used by controller

WHITE LISTS

LE Read Write List Size

gets the size of the controllers white list

LE Clear White List

removes all devices from white list

LE Add Device To White List

adds a single device to the white list

LE Remove Device From White List

removes a single device from the white list

ADVERTISING

LE Set Advertising Parameters

sets timing, advertising event type, address to use, channel map, filter policy

LE Read Advertising Channel Tx Power

allows filling in the Advertising Data for Tx Power

LE Set Advertising Data - defines the data to be broadcast

LE Set Scan Response Data - defines the data contained in scan responses

LE Set Advertising Enable - turn on and off advertising

SCANNING

LE Set Scan Parameters

sets timing, filter policy, address type, scan type

LE Set Scan Enable

turn on/off scanning

LE Advertising Report

we found something interesting

INITIATING

LE Create Connection

set timing, filter policy, target address, address type, connect to white list, connection interval, supervision timeout, expected length of communication events

LE Create Connection Cancel

stop trying to create a connection

LE Connection Complete

a connection was created (or cancelled, or timed out)

CONNECTION MANAGEMENT

LE Connection Update

- update connection event timing parameters
- connection interval, slave latency, supervision timeout, expected length of communication events

LE Connection Update Complete

- connect update has succeeded

AFH CHANNEL MAP MANAGEMENT

LE Set Host Channel Classification

- which LE data channels are “good / bad”
- controller still determines if channels are “used / unused”

LE Read Channel Map

- read the current LE channel map for a connection

INFORMATIONAL EXCHANGES

LE Read Local Supported Features

reads what LE features a controller supports

LE Read Remote Used Features

requests what a remote device supports

LE Read Remote Used Features Complete

report on what the remote device supports

STARTING ENCRYPTION

LE Start Encryption

request that encryption is started on a connection

requires random numbers & LTK

LE Long Term Key Request

Controller needs an LTK from Host to start encryption

LE Long Term Key Request Reply

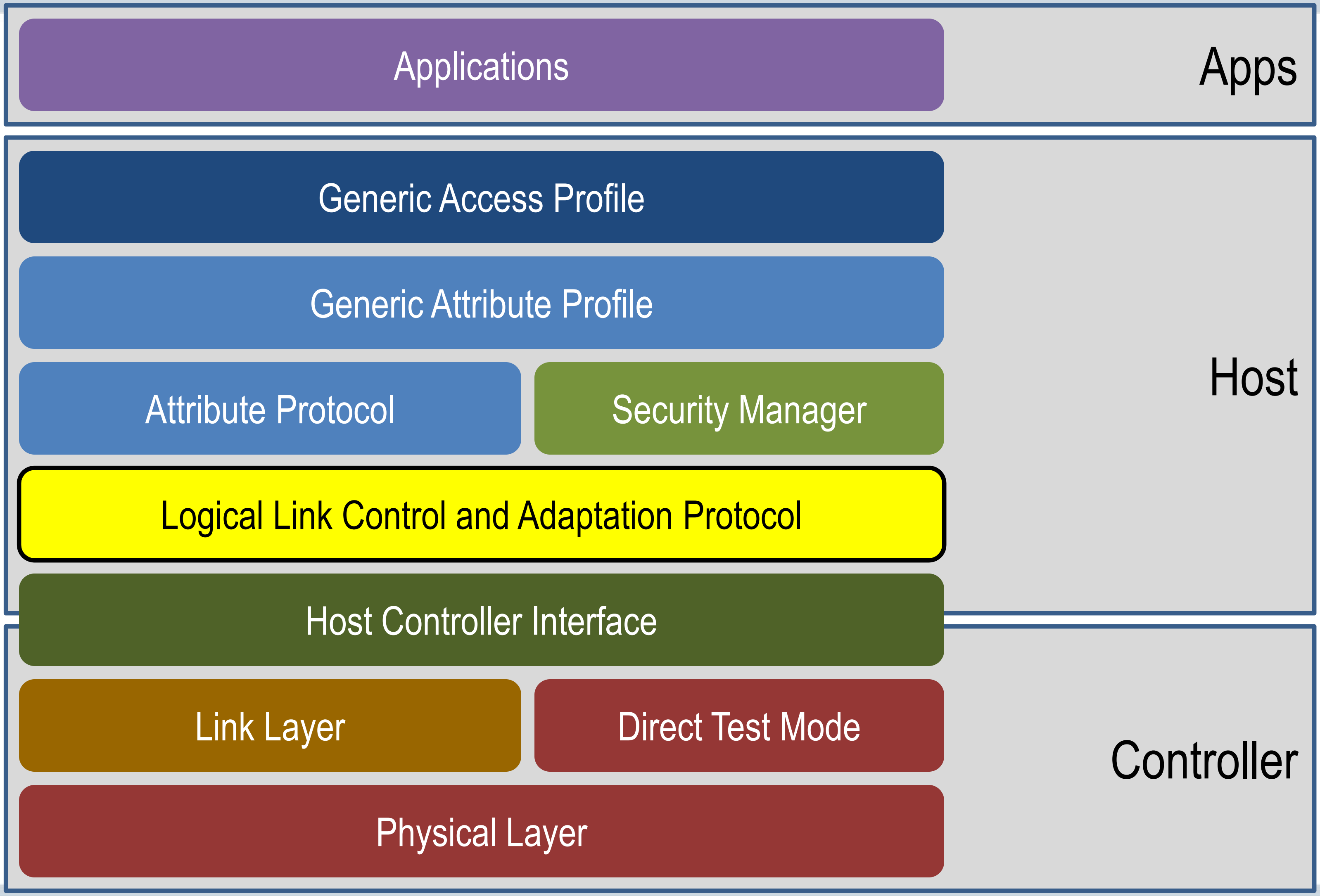
Host gives LTK to Controller for encyption

LUNCH

return at 13:30

Bluetooth SPECIAL INTEREST GROUP

L2CAP


Applications
Apps
Generic Access Profile
Generic Attribute Profile
Attribute Protocol
Security Manager
Host
Logical Link Control and Adaptation Protocol
Host Controller Interface
Link Layer
Direct Test Mode
Controller
Physical Layer

L2CAP

Logical Link Control and Adaptation Protocol

- protocol multiplexer
- segmentation and reassembly

Provides logical channels

- multiplexed over one or more logical links

L2CAP PACKETS

All application data is sent using L2CAP packets

- Length is the length of the L2CAP Information Payload
- CID is the destination logical channel

CIDs can be either

- fixed channels
- connection oriented channels (not used in LE)

Length	CID	Information Payload

FIXED L2CAP CHANNELS

CIDs from 0x0001 to 0x003F are fixed channels

0x0040 to 0xFFFF are dynamically allocated

CID	Description	Notes
0x0000	Null identifier	Not used (ever)
0x0001	L2CAP Signaling Channel	Used over BR/EDR
0x0002	Connectionless Channel	Used over BR/EDR
0x0003	AMP Manager Protocol	Used over BR/EDR
0x0004	Attribute Protocol	Used over LE only
0x0005	LE L2CAP Signaling Channel	Used over LE only
0x0006	Security Manager Protocol	Used over LE only

LE L2CAP SIGNALING PROTOCOL

Identical to L2CAP Signaling Protocol

except only one command per packet

supported commands limited

- Command reject
- Connection Parameter Update request
- Connection Parameter Update response

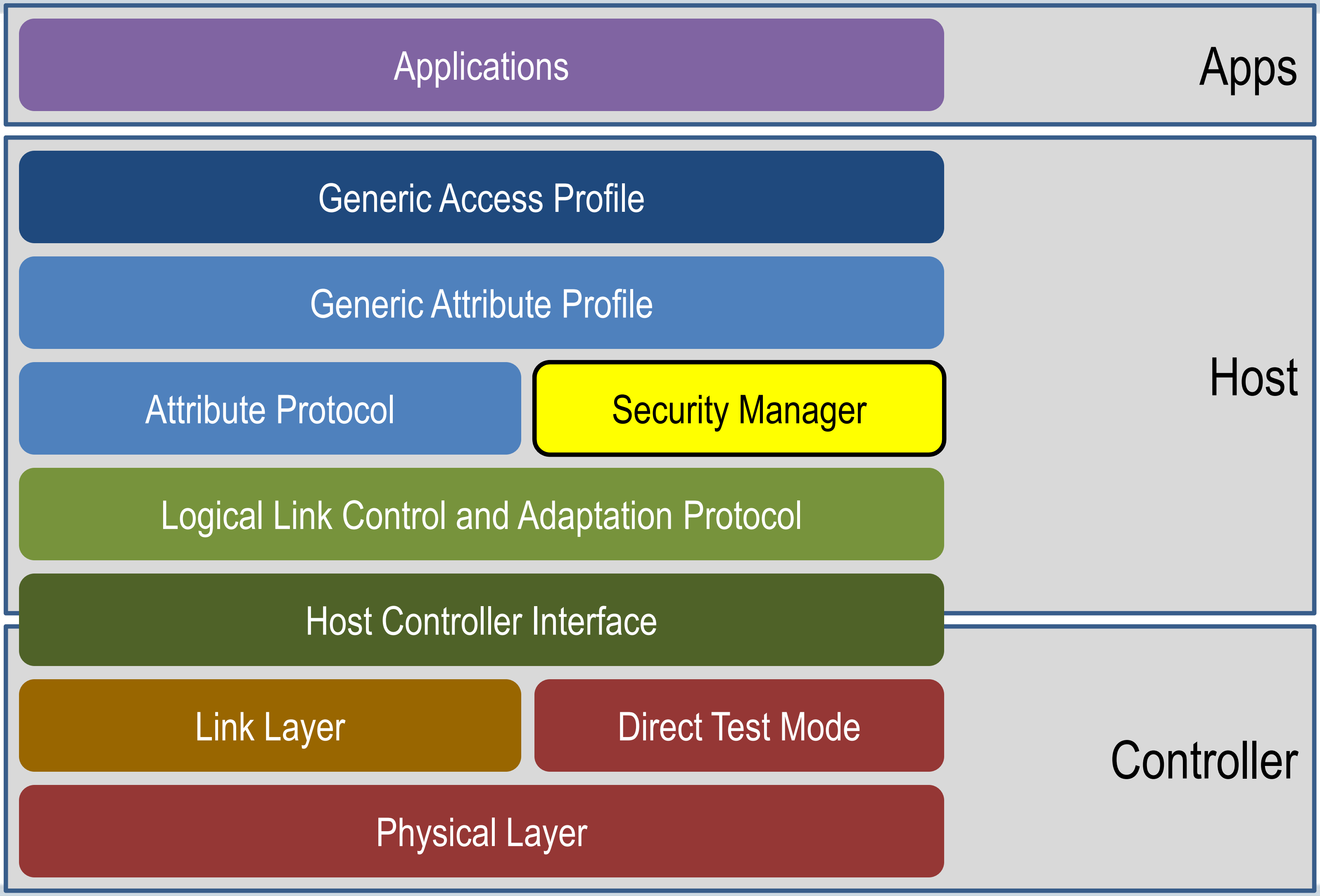
L2CAP SUMMARY

Three fixed channels for LE

- Attribute Protocol
- LE L2CAP Signaling Protocol
- Security Manager Protocol

Retains existing L2CAP packet structure

SECURITY MANAGER



SECURITY MANAGER (SM)

Security Manager Protocol

Pairing & Key Distribution

- I trust this device, and here is a key to prove it

Security Toolbox

- generating hashes
- confirmation values
- generate short term keys during pairing

SECURITY MANAGER PROTOCOL

Uses L2CAP fixed channel 0x0006

- MTU = 23 octets
- Best Effort, Basic Mode, Infinite Flush Timeout

Code	Data

BASIC CONCEPTS

Use distributing key model
- Slave generates and distributes key information to master
- Master can use this key information when reconnecting

Pairing
- authentication based on their capabilities / security requirements
- side effect is encrypted link / key distribution

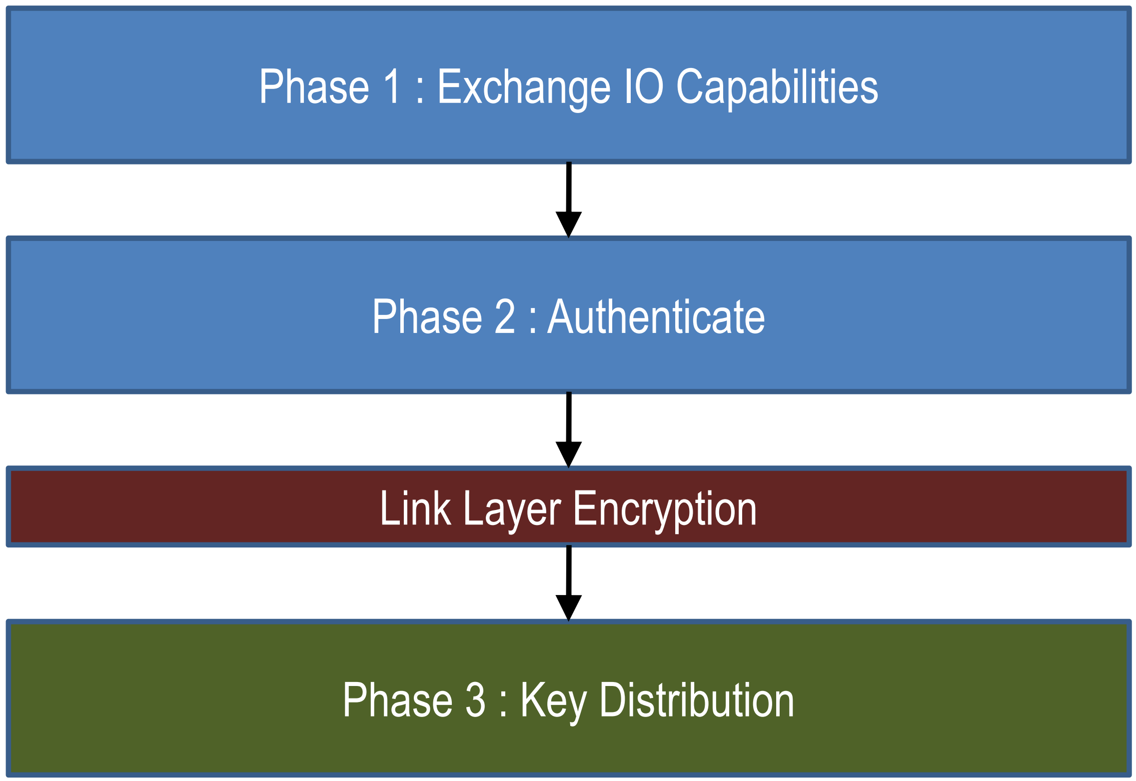
Signing Data
- Signing allows authentication of sender without encryption

Bonding
- GAP concept – device save keys for bonded devices

SM PROTOCOL CODES

Code	Name	Description
0x01	Pairing Request	starts the pairing procedure
0x02	Pairing Response	completes pairing exchange
0x03	Pairing Confirm	sends confirm value used in pairing
0x04	Pairing Random	sends random value used in pairing
0x05	Pairing Failed	oh no, its failed – including reason
0x06	Encryption Information	distributed Long Term Key
0x07	Master Identification	information used when reconnecting to a master
0x08	Identity Information	distributed Identity Resolving Key
0x09	Identify Address Information	address information used for reconnecting
0x0A	Signing Information	distributed Signature Key
0x0B	Security Request	slave wants security – master always initiates

PAIRING MODEL



IO CAPABILITIES

	No Input	Yes / No	Keyboard
No Output	No Input No Output	No Input No Output	Keyboard Only
Numeric Output	Display Only	Display Yes No	Keyboard Display

IO CAPABILITIES TO ALGORITHM

	Display Only	Display Yes No	Keyboard Only	No Input No Output	Keyboard Display
Display Only	Just Works Unauthenticated	Just Works Unauthenticated	Passkey Entry Authenticated	Just Works Unauthenticated	Passkey Entry Authenticated
Display Yes No	Just Works Unauthenticated	Just Works Unauthenticated	Passkey Entry Authenticated	Just Works Unauthenticated	Passkey Entry Authenticated
Keyboard Only	Passkey Entry Authenticated	Passkey Entry Authenticated	Passkey Entry Authenticated	Just Works Unauthenticated	Passkey Entry Authenticated
No Input No Output	Just Works Unauthenticated	Just Works Unauthenticated	Just Works Unauthenticated	Just Works Unauthenticated	Just Works Unauthenticated
Keyboard Display	Passkey Entry Authenticated	Passkey Entry Authenticated	Passkey Entry Authenticated	Just Works Unauthenticated	Passkey Entry Authenticated

OTHER PAIRING REQUIREMENTS

OOB Data

- has any authentication data been exchanged over OOB

Authentication Requirements

- No Bonding / Bonding
- Man In The Middle protection required

Key Distribution

- which keys does this device want from peer

ALGORITHMS

Just Works

TK = 0

Passkey Entry

TK = passkey (6 digit number, 000000 to 999999)

Out Of Band

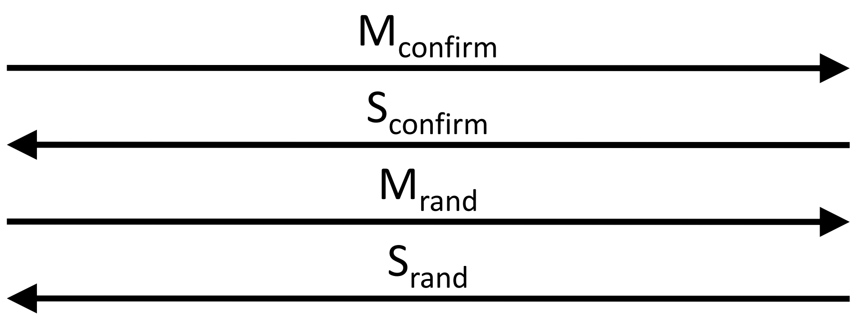
TK = from out of band

AUTHENTICATION

$$M_{confirm} = f_{c1}\ (TK, M_{rand}, Slave_{ADDR})$$

$$S_{confirm} = f_{c1}\ (TK, S_{rand}, Master_{ADDR})$$

$$f_{c1} = AES_{TK}\ (padding\ ||\ ADDR\ ||\ random)$$



SHORT TERM KEY

Need to generate a Short Term Key

after authentication

$$STK = f_{s1}\ (TK, S_{rand}, M_{rand})$$

$$f_{s1} = AES_{TK}\ (S_{rand} \mid\mid M_{rand})$$

KEY DISTRIBUTION

Many keys can be distributed in both directions

- Long Term Key
- Identity Resolving Key
- Signature Resolving Key
- Encrypted Diversifier / Random Number

LONG TERM KEY

128 bit random number given by slave to master

If a device can be both master and slave
  then a master can also give key to slave

Can be “derived” from EDIV / Rand
  LTK = f (EDIV, Rand)

e.g. NIST Special Publication 800-108 could be used

IDENTITY RESOLVING KEY

Two types of address

- Public Address
- Random Address

GAP defines sub-categories of Random Address

- Static Address
- Non-Resolvable Private Address
- Resolvable Private Address ← derived from IRK

RESOLVABLE PRIVATE ADDRESSES

Resolvable addresses need to use a known secret

- Identify Resolving Key (IRK)

$$hash = AES_{IRK}\ (random)$$

hash (24 bits)	random (24 bits)	1

When receive an address, can search all known IRKs with random, and check if hash matches

SIGNED DATA

Allows authentication without encryption

Uses SignCounter

- incremented on each new message sent between devices

Message Authentication Code

- calculated over plain text and SignCounter
- uses CMAC as defined by RFC 4493

plain text	SignCounter (32 bits)	MAC (64 bits)

SUMMARY

Pairing / Authentication

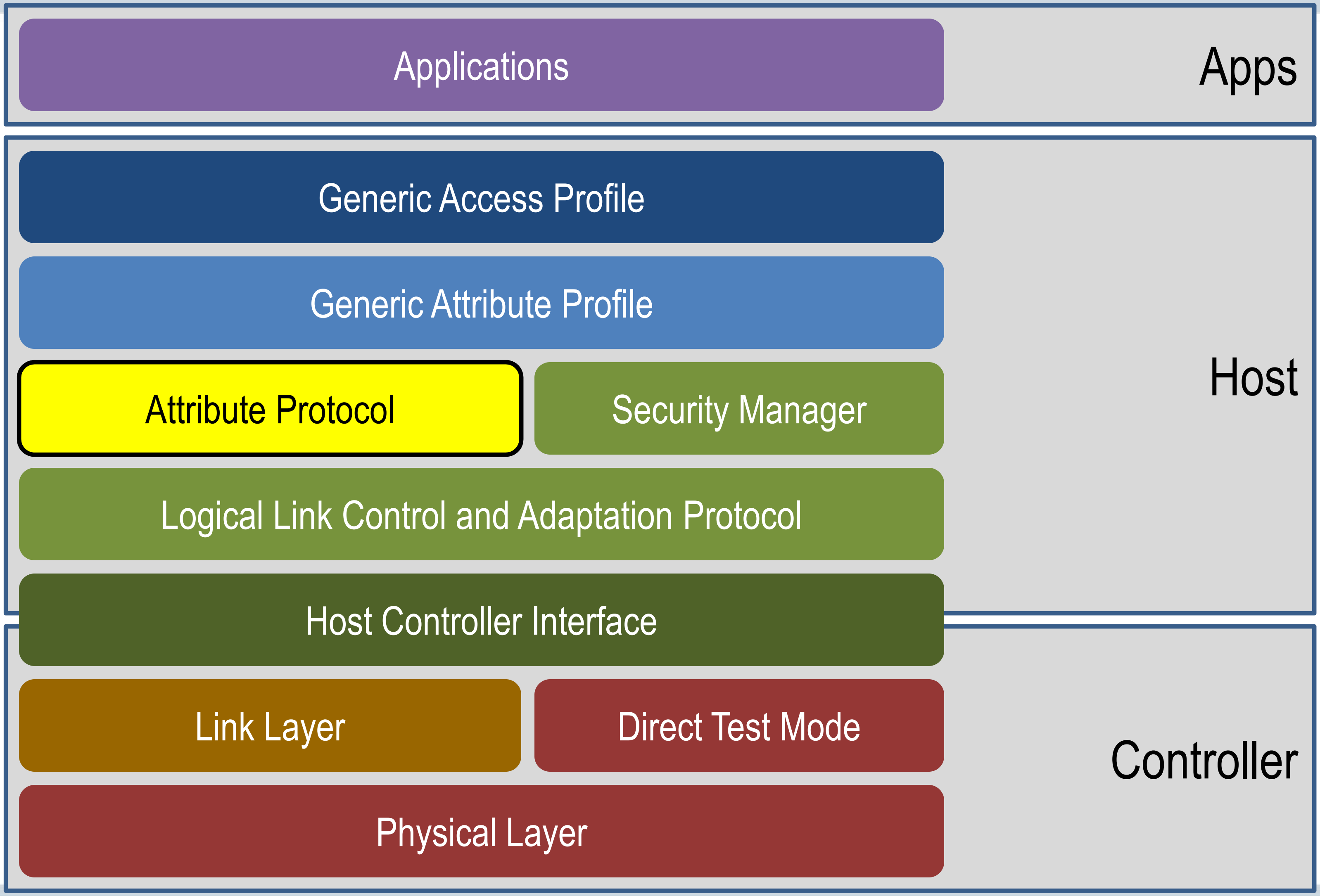
- uses same IO Capabilities as SSP from v2.1
- allows upgrade to public key cryptography
- uses Key Distribution

Defines cryptographic algorithms for privacy

- Resolvable Private Addresses

Allows authentication using Signed Data

ATTRIBUTE PROTOCOL



ATTRIBUTE PROTOCOL (ATT)

Client Server Architecture

- servers have data
- clients request data to/from servers

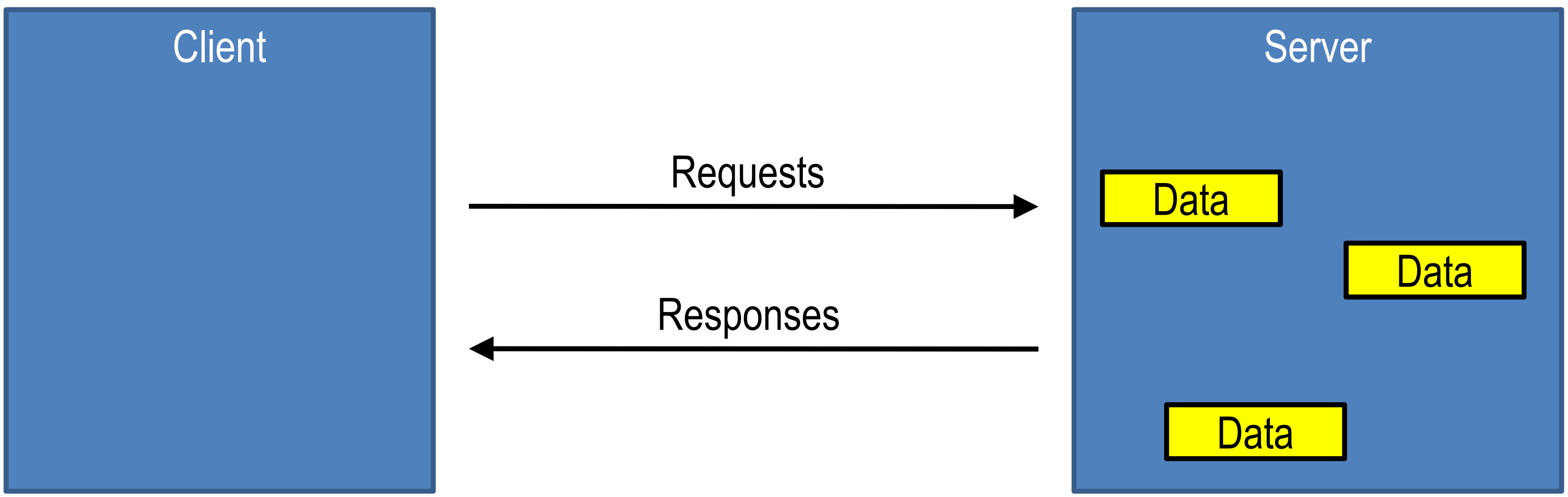
Protocol Methods

- request, response, command,
- notification, indication, confirmation

CLIENT SERVER ARCHITECTURE

Servers have data, Clients want to use this data

Servers expose Data using Attributes



SERVERS EXPOSE DATA USING ATTRIBUTES

Attributes have values

- array of octets
- 0 to 512 octets in length
- can be fixed or variable length

Value
0x54656d706572617475726520536656e736f72
0x04
0x0802

ATTRIBUTES ARE ADDRESSABLE

Each attribute has a "handle"

used to address an individual attribute by a client

Clients use handles to address attributes

Read (0x0022) => 0x04 ; Read (0x0098) => 0x0802

Handle	Value
0x0009	0x54656d7065726174757265205365e736f72
0x0022	0x04
0x0098	0x0802

ATTRIBUTES ARE TYPED

Attributes have a type

- type is a «UUID», determines what the value means

Types are defined by “Characteristic Specifications”

- or Generic Access Profile or Generic Attribute Profile

Handle	Type	Value
0x0009	«Device Name»	0x54656d706572617475726520536e736f72
0x0022	«Battery State»	0x04
0x0098	«Temperature»	0x0802

ATTRIBUTES ARE TYPED

«Device Name»

- defined by GAP
- formatted as UTF-8

0x54656d70657261747572652053656e736f72 =
"Temperature Sensor"

Handle	Type	Value
0x0009	«Device Name»	"Temperature Sensor"
0x0022	«Battery State»	0x04
0x0098	«Temperature»	0x0802

ATTRIBUTES ARE TYPED

«Battery State»

- defined by “Battery State Characteristic” specification
- enumerated value

0x04 = Discharging

Handle	Type	Value
0x0009	«Device Name»	“Temperature Sensor”
0x0022	«Battery State»	Discharging
0x0098	«Temperature»	0x0802

ATTRIBUTES ARE TYPED

«Temperature»

defined by “Temperature Characteristic” specification

Signed 16 bit Integer in 0.01 ºC

0x0802 = 2050 * 0.01 ºC = 20.5 ºC

Handle	Type	Value
0x0009	«Device Name»	“Temperature Sensor”
0x0022	«Battery State»	Discharging
0x0098	«Temperature»	20.5 ºC

ATTRIBUTE TYPE

Type is a «UUID»

UUIDs are 128 bits in length

Bluetooth defines a Bluetooth Base UUID

allowing a 16 bit «UUID» to be defined

00000000-0000-1000-8000-00805F9B34FB

Same Bluetooth Base UUID as SDP

ATTRIBUTE TYPE

Type is a «UUID»

UUIDs are 128 bits in length

Bluetooth defines a Bluetooth Base UUID

allowing a 16 bit «UUID» to be defined

0000xxxx-0000-1000-8000-00805F9B34FB

ATTRIBUTE TYPE

Type is a «UUID»

UUIDs are 128 bits in length

Bluetooth defines a Bluetooth Base UUID

allowing a 16 bit «UUID» to be defined

00001234-0000-1000-8000-00805F9B34FB

= 16 bit UUID 0x1234

ATTRIBUTE HANDLE

Handle is a 16 bit value

0x0000 is reserved – shall never be used

0x0001 to 0xFFFF can be assigned to any attributes

Handles are “sequential”

- 0x0005 is “before” 0x0006
- 0x0104 is “after” 0x00F8

ATTRIBUTE TYPE

Type is a «UUID»

UUIDs are 128 bits in length

Bluetooth defines a Bluetooth Base UUID

allowing a 16 bit «UUID» to be defined

0000xxxx-0000-1000-8000-00805F9B34FB

ATTRIBUTE PERMISSIONS

Attributes values may be:

- readable / not readable
- writeable / not writeable
- readable & writeable / not readable & not writeable

Attribute values may require:

- authentication to read / write
- authorization to read / write
- encryption / pairing with sufficient strength to read / write

ATTRIBUTE PERMISSIONS

Permissions not “discoverable” over Attribute Protocol

- determined by implication

If request to read an attribute value that cannot be read

- Error Response «Read Not Permitted»

If request to write an attribute value that requires authentication

- Error Response «Insufficient Authentication»
- Client must create authenticated connection and then retry
- There is no “pending” state

ATTRIBUTE PERMISSIONS

Attribute Handles are public information

Attribute Types are public information

Attribute Values can be protected

It is up to the server to not reveal any values that it considers are protected to a client it does not “trust” enough

- Server responds by saying what is wrong – not with value
  - Insufficient Authentication / Authorization / Key Size
  - Read / Write Not Permitted

LOGICAL ATTRIBUTE REPRESENTATION

2 octets	2 or 16 octets	variable length (0 to 512 octets)	implementation specific
Attribute Handle	Attribute Type	Attribute Value	Attribute Permissions

PROTOCOL METHODS

Protocol PDU Type	Sent by	Description
Request	Client	Client requests something from server – always causes a response
Response	Server	Server sends response to a request from a client
Command	Client	Client commands something to server – no response
Notification	Server	Server notifies client of new value – no confirmation
Indication	Server	Server indicates to client new value – always causes a confirmation
Confirmation	Client	Confirmation to an indication

PROTOCOL IS STATELESS

After transaction complete

- no state is stored in protocol

A transaction is:

- Request -> Response
- Command
- Notification
- Indication -> Confirmation

SEQUENTIAL PROTOCOL

Client can only send one request at a time

- request completes after response received in client

Server can only send one indication at a time

- indication completes after confirmation received in server

Commands and Notifications are no response / confirmation

- can be sent at any time
- could be dropped if buffer overflows – consider unreliable

ATOMIC OPERATIONS

Each request / command is an atomic operation

- cannot be affected by another client at the same time

If link is disconnected halfway through a transaction

- value of attribute(s) undefined
- there is no “rollback” or “transactional processing”

ATTRIBUTE GROUPING

Generic Attribute Profile defines a concept of Grouping

- Grouping is done by Attribute Type

```
«Grouping Type»
    «Another Grouping Type»
        «Data»
        «Data»
    «Another Grouping Type»
        «Data»
        «Data»
«Grouping Type»
    «Data»
    «Another Grouping Type»
        «Data»
        «Data»
```

ATTRIBUTE GROUPING

Generic Attribute Profile defines a concept of Grouping

- Grouping is done by Attribute Type

«Secondary Service»
    «Characteristic»
        «Data»
        «Data»
    «Characteristic»
        «Data»
        «Data»
«Primary Service»
    «Data»
    «Characteristic»
        «Data»
        «Data»

MTU SIZES

Over BR/EDR

- Attribute protocol uses a dynamic channel (fixed PSM)
- MTU negotiated by L2CAP

Over LE

- Attribute protocol uses a fixed channel
- MTU “exchanged” by ATT
  - ATT_MTU = min (Server_Rx_MTU, Client_Rx_MTU)
  - ATT_MTU is symmetrical (same on client / server)

ATTRIBUTE PDU FORMAT

Attribute Opcode

- bit 6-0 : Method
- bit 7 : Authentication Signature Flag = 0

Can append Signed Data to some methods

ATTRIBUTE PDU FORMAT

Attribute Opcode

bit 6-0 : Method

bit 7 : Authentication Signature Flag = 1

Can append Signed Data to some methods

Name	Description
Error Response	Something was wrong with a request
Exchange MTU Request / Response	Exchange new ATT_MTU
Find Information Request / Response	Find information about attributes
Find By Type Value Request / Response	Find specific attributes
Read By Group Type Request / Resposne	Find specific group attributes and ranges
Read By Type Request / Response	Read attribute values of a given type
Read Read / Response	Read an attribute value
Read Blob Request / Response	Read part of a long attribute value
Read Multiple Request / Response	Read multiple attribute values
Write Command	Write this – no response
Write Request / Response	Write an attribute value
Prepare Write Request / Response	Prepare to write a value (long)
Execute Write Request / Response	Execute these prepared values
Handle Value Notification	Notify attribute value – no confirmation
Handle Value Indication / Confirmation	This attribute now has this value

MTU EXCHANGE

```
Exchange MTU (Client Rx MTU) => (Server Rx MTU)
```

Only used on LE (not on BR/EDR)

ATT_MTU = min (Client Rx MTU, Server Rx MTU)

Can only be sent once during a connection

only be initiated by client

optional to initiate this – if you are happy with 23, don't

server has no control over if this is done

FINDING INFORMATION

```
Find Information (Starting Handle, Ending Handle) =>
  (format, [Handle, Type]*)
```

Handles / Types are public information

- Find Information used to find this information

Responds with a list of attribute handles and their types

- can only send all 16 bit or all 128 bit UUIDs in response
- if mix of 16/128 UUIDs, multiple requests must be used
  - each with different format

FINDING INFORMATION

```
Find By Type Value (Starting Handle, Ending Handle,
  Attribute Type, Attribute Value) =>
  ([HandlePair]*)
```

Returns the handle of all found attribute

with type and value matching exactly

AND

handle of the last attribute in the attribute group

READING ATTRIBUTES

```
Read By Group Type (Starting Handle, Ending Handle,
  UUID) => (Length, [Handle:EndGroupHandle:Value]*)
```

Reads the value of each attribute of a given type in a range

- responds with handle:end group handle:value tuples
- allows reading group semantics

If multiple values have the size length, then many can be returned in a single response

READING ATTRIBUTES

```
Read By Type (Starting Handle, Ending Handle, UUID)
  => (Length, [Handle:Value]*)
```

Reads the value of each attribute of a given type in a range

responds with handle:value pairs

If multiple values have the size length, then many can be returned in a single response

READING ATTRIBUTES

```
Read (Handle) => (Value)
```

Simple?

Only works up to ATT_MTU – 1 octets

READING LONG ATTRIBUTES

```
Read Blob (Handle, Offset) => (Part Value)
```

Can be used to read very long attributes

If ATT_MTU = 23 (default on LE)

- a 512 octet value would require 24 transactions to read

If ATT_MTU = 48 (default on BR/EDR)

- a 512 octet value would require 11 transactions to read

READ MULTIPLE

```
Read Multiple ([Handle]*) => ([Value]*)
```

Used to read multiple values at the same time

Can only be used if len (value) < ATT_MTU – 1

Can only be used if size of each value is known
except last value

WRITING ATTRIBUTES

```
Write Command (handle, value) =>
```

I want this attribute set to this value NOW

- with no response

Used for “sending commands” to a “state machine”

- can also be signed if need authentication with cost of encryption setup / confidentiality requirements

WRITING ATTRIBUTES

```
Write (handle, value) => ()
```

Simple?

Includes a Response to allow:

- flow control of transactions – no buffer overflows
- confirm that the value has been written

COMPLEX WRITES

```
Prepare Write (handle, offset, value) => (handle,
  offset, value)
Execute Write (exec/cancel) => ()
```

Prepared writes are queued in server until they are executed

- queue size can be discovered
- response includes value again (double checking)
- offset parameter allows writes to long attributes

execution is a single atomic transaction

SERVER INITIATED METHODS

=> Handle Value Notification (handle, value)

Server notifies client of an attribute with a new value

Can be sent at any time, no flow control

- inherently unreliable – can be dropped by buffer overflow

Typically used to notify client of state updates

- turned on / off by clients using Generic Attribute Profile

SERVER INITIATED METHODS

```
Handle Value Indication(handle, value) => Handle
  Value Confirmation ()
```

Server indicates to client an attribute with a new value

Client must confirm receipt before server can send again

Used when flow control is important

or when confirmation of state change is required

ERROR RESPONSE

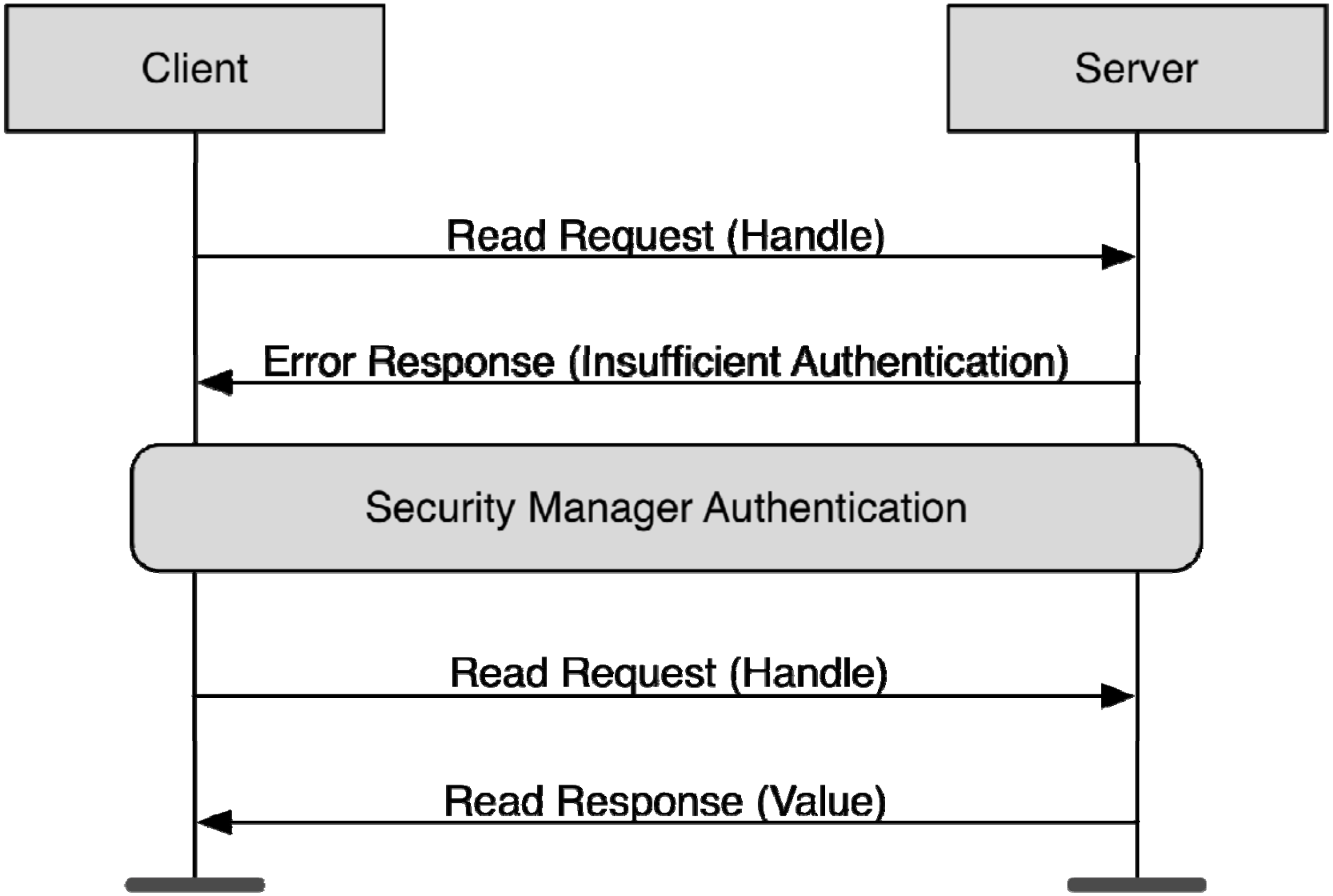
```
(any) Request (*) => Error Response (Opcode, Handle,
  Error Code)
```

Any request can cause an error
- response must always be sent
- error response includes information about error

Opcode from the request – what request caused this error
Handle from the request – what handle caused this error
Error Code – reason why this error is raised

Name	Description
Invalid Handle	for example handle = 0x0000
Read Not Permitted	not readable attribute : permissions
Write Not Permitted	not writeable attribute : permissions
Invalid PDU	PDU was invalid – wrong size?
Insufficient Authentication	needs authentication : permissions
Request Not Supported	server doesn't support request
Invalid Offset	offset beyond end of attribute
Insufficient Authorization	need authorization : permissions
Prepare Queue Full	server has run out of prepare queue space
Attribute Not Found	no attributes in attribute range found
Attribute Not Long	should use Read requests
Insufficient Encryption Key Size	needs encryption key size : permissions
Invalid Attribute Value Length	value written was invalid size
Unlikely Error	something went wrong – oops
Insufficient Encryption	needs encryption : permissions
Application Error	application didn't like what you requested

SECURITY CONSIDERATIONS



ATTRIBUTE PROTOCOL SUMMARY

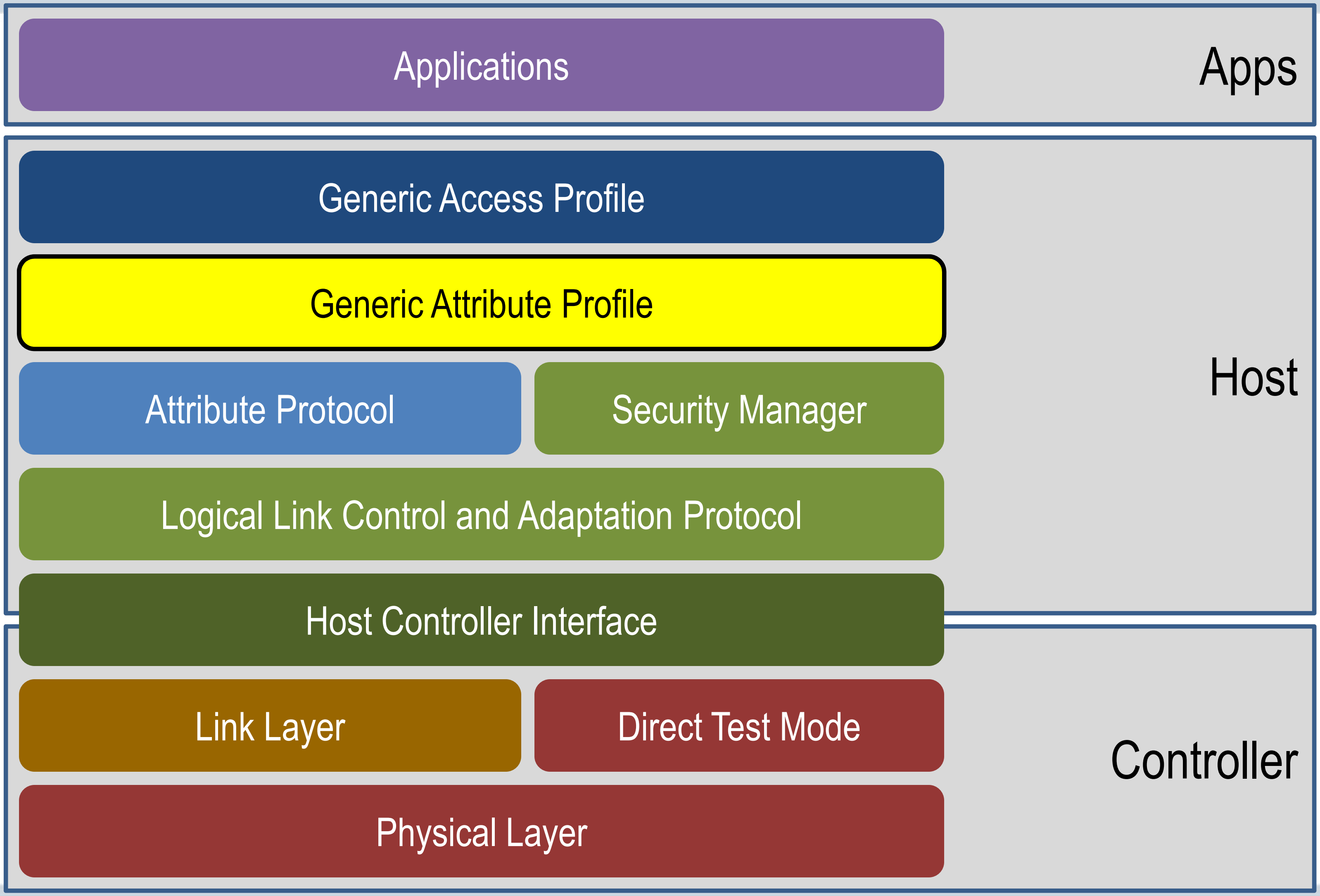
Exposes Data using Typed, Addressable Attributes

- Handle
- Type
- Value

Methods for finding, reading, writing attributes by client

Methods for sending notifications / indications by server

GENERIC ATTRIBUTE PROFILE



GENERIC ATTRIBUTE PROFILE (GATT)

Defines concepts of:

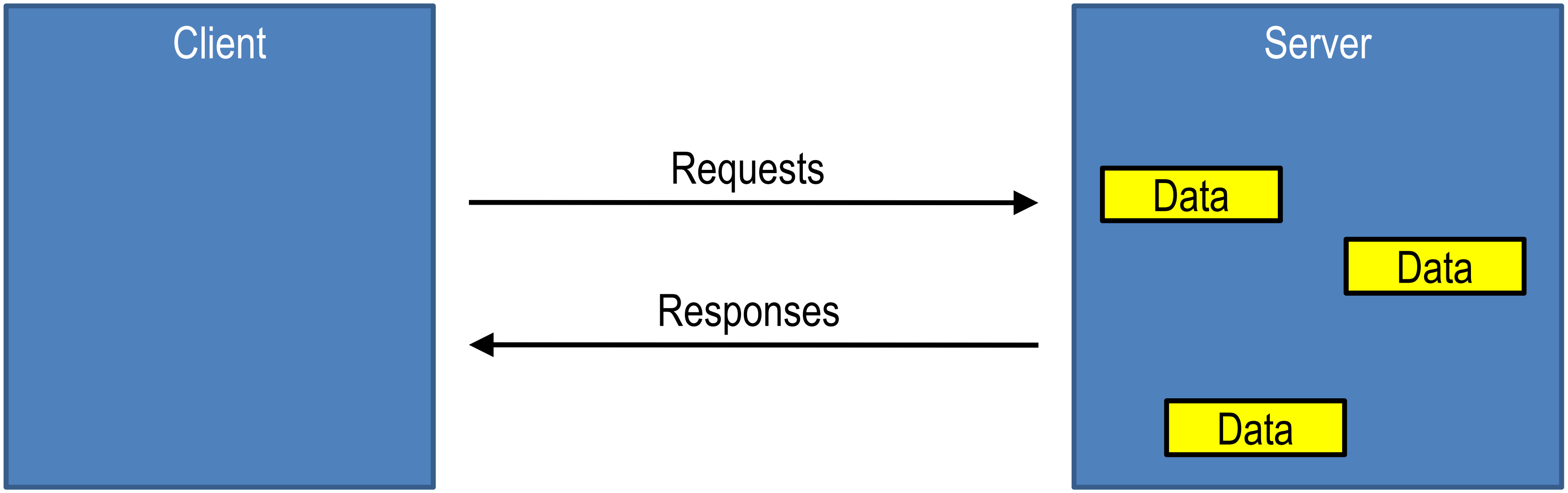
- Service Group
- Characteristic Group
- Declarations
- Descriptors

Does not define rules for their use

- this is separate but essential to understand

CLIENT SERVER ARCHITECTURE

Same client server architecture as Attribute Protocol

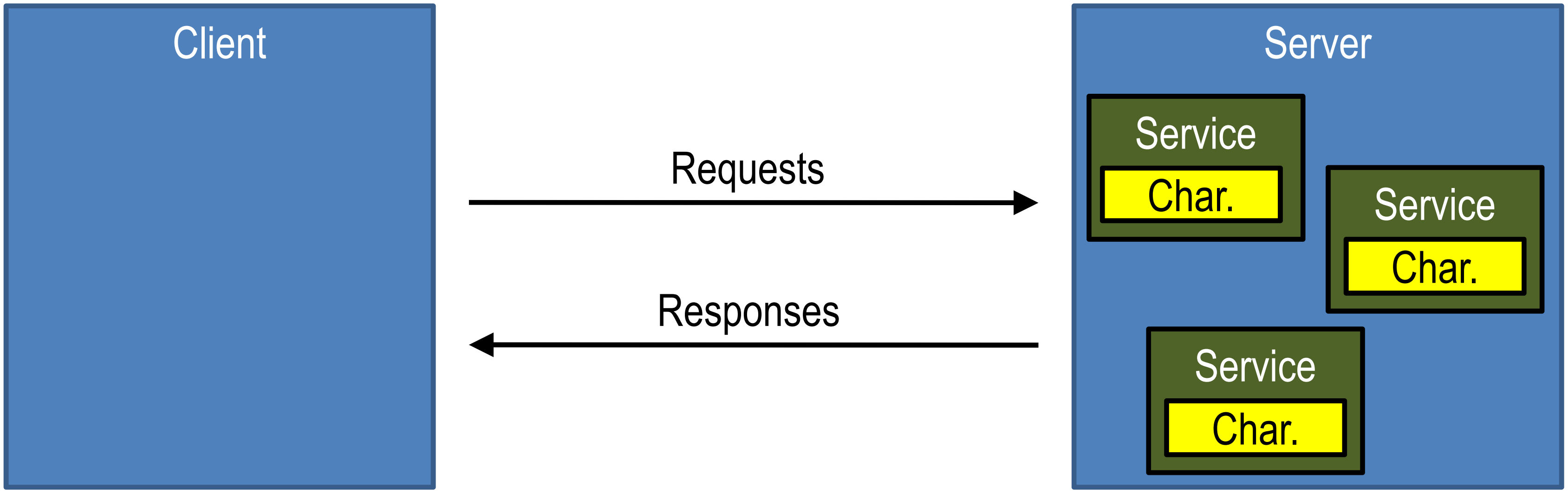


CLIENT SERVER ARCHITECTURE

Same client server architecture as Attribute Protocol

except that data is encapsulated in “Services”

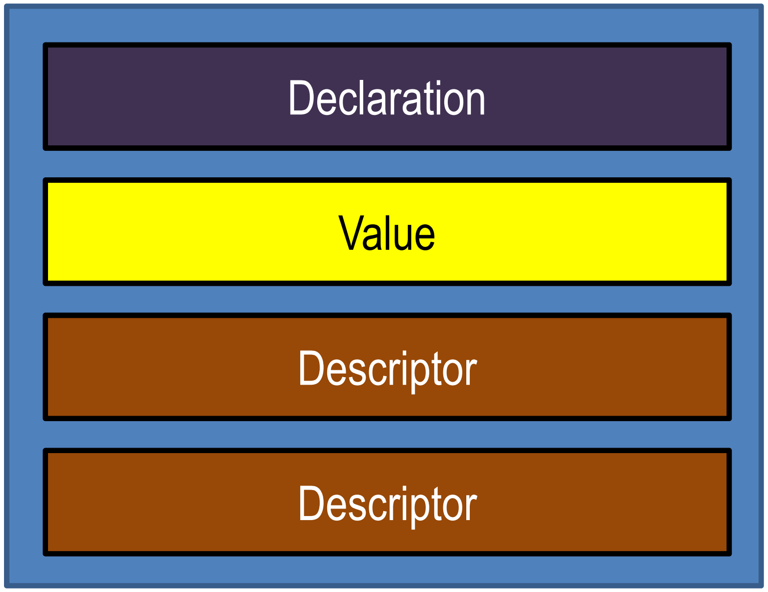
and data is exposed in “Characteristic”



WHAT IS A CHARACTERISTIC?

It's a value, with a known type, and a known format
- defined in a "Characteristic Specification"

Characteristic Declaration
- Characteristic Value
- Characteristic Descriptors



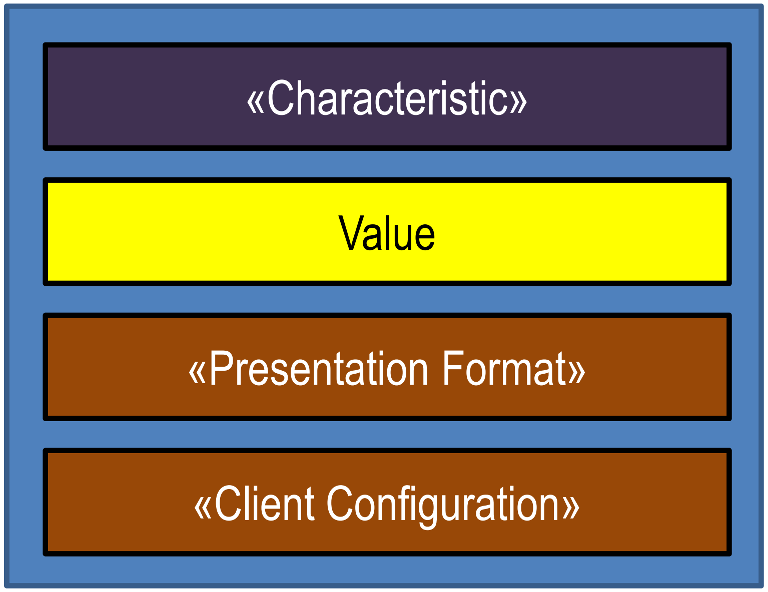
WHAT IS A CHARACTERISTIC?

Characteristics are grouped by «Characteristic»

- Value attribute is always immediately after «Characteristic»
- followed by descriptors

Descriptors

- additional information
- any number
- any order
- can be vendor specific



WHAT IS A SERVICE?

A service is:

- defined in a “Service Specification”
- collection of characteristics
- references to other services

Service Declaration

- Includes
- Characteristics

TWO TYPES OF SERVICE:

Primary Service

- A primary service is a service that exposes primary usable functionality of this device. A primary service can be included by another service.

Secondary Service

- A secondary service is a service that is subservient to another secondary service or primary service. A secondary service is only relevant in the context of another service.

GENERIC ATTRIBUTE PROFILE

Attribute Protocol defines a server with a set of attributes

- addressable with a handle
- typed using a UUID
- with data in an attribute value
- with some permissions

2 octets	2 or 16 octets	variable length (0 to 512 octets)	implementation specific
Attribute Handle	Attribute Type	Attribute Value	Attribute Permissions

ATTRIBUTES ARE FLAT

Handle	Type	Value	Permissions
0x0001	«Primary Service»	«GAP»	R
0x0002	«Characteristic»	{r, 0x0003, «Device Name»}	R
0x0003	«Device Name»	“Temperature Sensor”	R
0x0004	«Characteristic»	{r, 0x0006, «Appearance»}	R
0x0006	«Appearance»	«Thermometer»	R
0x000F	«Primary Service»	«GATT»	R
0x0010	«Characteristic»	{r, 0x0012, «Attribute Opcodes Supported»}	R
0x0012	«Attribute Opcodes Supported»	0x00003FDF	R
0x0020	«Primary Service»	«Temperature»	R
0x0021	«Characteristic»	{r, 0x0022, «Temperature Celsius»}	R
0x0022	«Temperature Celsius»	0x0802	R*

GENERIC ATTRIBUTE PROFILE IMPOSES STRUCTURE

Primary Service «GAP»

- «Device Name» “Temperature Sensor”
- «Appearance» «Thermometer»

Primary Service «GATT»

- «Attribute Opcodes Supported» 0x03FDF

Primary Service «Temperature»

- «Temperature Celsius» 0x0802

GROUPING GIVES STRUCTURE

Handle	Type	Value	Permissions
0x0001	«Primary Service»	«GAP»	R
0x0002	«Characteristic»	{r, 0x0003, «Device Name»}	R
0x0003	«Device Name»	“Temperature Sensor”	R
0x0004	«Characteristic»	{r, 0x0006, «Appearance»}	R
0x0006	«Appearance»	«Thermometer»	R
0x000F	«Primary Service»	«GATT»	R
0x0010	«Characteristic»	{r, 0x0012, «Attribute Opcodes Supported»}	R
0x0012	«Attribute Opcodes Supported»	0x00003FDF	R
0x0020	«Primary Service»	«Temperature»	R
0x0021	«Characteristic»	{r, 0x0022, «Temperature Celsius»}	R
0x0022	«Temperature Celsius»	0x0802	R*

GROUPING GIVES STRUCTURE

Handle	Type	Value	Permissions
0x0001	«Primary Service»	«GAP»	R
0x0002	«Characteristic»	{r, 0x0003, «Device Name»}	R
0x0003	«Device Name»	“Temperature Sensor”	R
0x0004	«Characteristic»	{r, 0x0006, «Appearance»}	R
0x0006	«Appearance»	«Thermometer»	R
0x000F	«Primary Service»	«GATT»	R
0x0010	«Characteristic»	{r, 0x0012, «Attribute Opcodes Supported»}	R
0x0012	«Attribute Opcodes Supported»	0x00003FDF	R
0x0020	«Primary Service»	«Temperature»	R
0x0021	«Characteristic»	{r, 0x0022, «Temperature Celsius»}	R
0x0022	«Temperature Celsius»	0x0802	R*

GROUPING GIVES STRUCTURE

Handle	Type	Value	Permissions
0x0001	«Primary Service»	«GAP»	R
0x0002	«Characteristic»	{r, 0x0003, «Device Name»}	R
0x0003	«Device Name»	"Temperature Sensor"	R
0x0004	«Characteristic»	{r, 0x0006, «Appearance»}	R
0x0006	«Appearance»	«Thermometer»	R
0x000F	«Primary Service»	«GATT»	R
0x0010	«Characteristic»	{r, 0x0012, «Attribute Opcodes Supported»}	R
0x0012	«Attribute Opcodes Supported»	0x00003FDF	R
0x0020	«Primary Service»	«Temperature»	R
0x0021	«Characteristic»	{r, 0x0022, «Temperature Celsius»}	R
0x0022	«Temperature Celsius»	0x0802	R*

«INCLUDE» DECLARATION

Allows services to reference other services

Value is reference to another Service:

- Service Handle
- End Group Handle
- Service UUID

Type	Value	Permissions
«Include»	Service Handle End Group Handle Service UUID	Read only, no authentication, no authorization

«PRIMARY SERVICE» DECLARATION

Groups attributes for a primary service

- followed by «Include»
- then followed by «Characteristic»

Attribute value is UUID for Service

- e.g. «GAP», «GATT», «Temperature», «Battery», etc…

Type	Value	Permissions
«Primary Service»	UUID for Service	Read only, no authentication, no authorization

«SECONDARY SERVICE» DECLARATION

Groups attributes for a secondary service

followed by «Include»

then followed by «Characteristic»

Attribute value is UUID for Service

e.g. «GAP», «GATT», «Temperature», «Battery», etc…

Type	Value	Permissions
«Secondary Service»	UUID for Service	Read only, no authentication, no authorization

«CHARACTERISTICS» DECLARATION

Groups attributes for a characteristic within a service
followed by Characteristic Value attribute, descriptors

Attribute Value is:
properties for characteristic value (b,r,c,w,n,i,a,e)
handle of characteristic value
type of characteristic

Type	Value	Permissions
«Characteristic»	Properties, Value Handle, Characteristic UUID	Read only, no authentication, no authorization

«CHARACTERISTICS» PROPERTIES

Properties for Characteristic Value

- Broadcast
- Read
- Command
- Write
- Notify
- Indicate
- Signed Command
- Extended Properties

CONTROL-POINT CHARACTERISTICS

Characteristics that are only:

Command, Write, Notify or Indicate

are called “Control-Point Characteristics”

Cannot read Control-Point Characteristics

they expose no state

can represent “instantaneous” state

«CHARACTERISTIC» HANDLE AND TYPE

Handle for Characteristic Value not incremental to Declaration

Type is repeated for optimized searches

- Read By Type «Device Name»
- Read By Type «Characteristic»

Handle	Type	Value	Permissions
0x0001	«Primary Service»	«GAP»	R
0x0002	«Characteristic»	{r, 0x0003, «Device Name»}	R
0x0003	«Device Name»	“Temperature Sensor”	R
0x0004	«Characteristic»	{r, 0x0006, «Appearance»}	R
0x0006	«Appearance»	«Thermometer»	R

CHARACTERISTIC DESCRIPTORS

«Characteristic Extended Properties»

Additional properties (that didn't fit in «Characteristic»)

Bit: Reliable Write

can "prepare / execute write"

Bit: Writable Descriptors

can write descriptors (when allowed)

CHARACTERISTIC DESCRIPTORS

«Characteristic User Description»

UTF-8 string

Description of this characteristic

typically written by user to label a characteristic

may require authentication / authorization to write

max of one User Description per Characteristic

CHARACTERISTIC DESCRIPTORS

«Client Characteristic Configuration»

- Notification / Indication configuration

Allows each client to turn on / off notifications / indications

- does not define when these are sent
- just that they are sent

Different value for each client that is “bonded” with device

- may require authentication / authorization to write

CHARACTERISTIC DESCRIPTORS

«Server Characteristic Configuration»

- Broadcast configuration

Allows any client to turn on / off broadcast

- does not define when these are sent
- just that they are sent

One value for all clients

- may require authentication / authorization to write

CHARACTERISTIC DESCRIPTORS

«Characteristic Presentation Format»

- defines how the characteristic value if formatted
- used to "present" value to user via a "Generic Client"

Fields include:

- Format (boolean, uint8, sint16, float32, utf8s, etc…)
- Exponent (multiply value by $10^{Exponent}$)
- Unit (SI Units / Derived Units)
- Name Space (Who allocated Description: Bluetooth, Continua)
- Description (Above, Below, Armpit, Ear, Inside, Outside, etc…)

CHARACTERISTIC DESCRIPTORS

«Characteristic Aggregate»

list of characteristic presentation formats

for each part of the aggregated value

«Longitude», «Latitude», «Elevation»

«3D Position»

Aggregate Longitude / Latitude / Elevation

GATT PROCEDURES

Procedure	Sub-Procedures
Server Configuration	Exchange MTU
Primary Service Discovery	Discovery All Primary Service Discover Primary Service by Service UUID
Relationship Discovery	Find Included Services
Characteristic Discovery	Discover All Characteristics of a Service Discover Characteristics by UUID
Characteristic Descriptor Discovery	Discover All Characteristic Descriptors
Characteristic Value Read	Read Characteristic Value Read Using Characteristic UUID Read Long Characteristic Values Read Multiple Characteristic Values

GATT PROCEDURES

Procedure	Sub-Procedures
Characteristic Value Write	Write Without Response Write Without Response With Authentication Write Characteristic Value Write Long Characteristic Values Reliable Writes
Characteristic Value Notifications	Notifications
Characteristic Value Indications	Indications
Characteristic Descriptors	Read Characteristic Descriptors Read Long Characteristic Descriptors Write Characteristic Descriptors Write Long Characteristic Desc

GATT CHARACTERISTICS

GATT defines it own «GATT» Service, and characteristics

- «Service Changed»
- «Attribute Opcodes Supported»

«SERVICE CHANGED» CHARACTERISTIC

Control-Point Characteristic

- notified when client uses attributes on a server
- after server has change attribute handles
- or after server has added or removed services

Client must perform "Service Discovery" procedure

- re-establish services / characteristics

GATT CHARACTERISTICS

«Attribute Opcodes Supported»

What attribute protocol opcodes are supported

just a bit mask

GATT SUMMARY

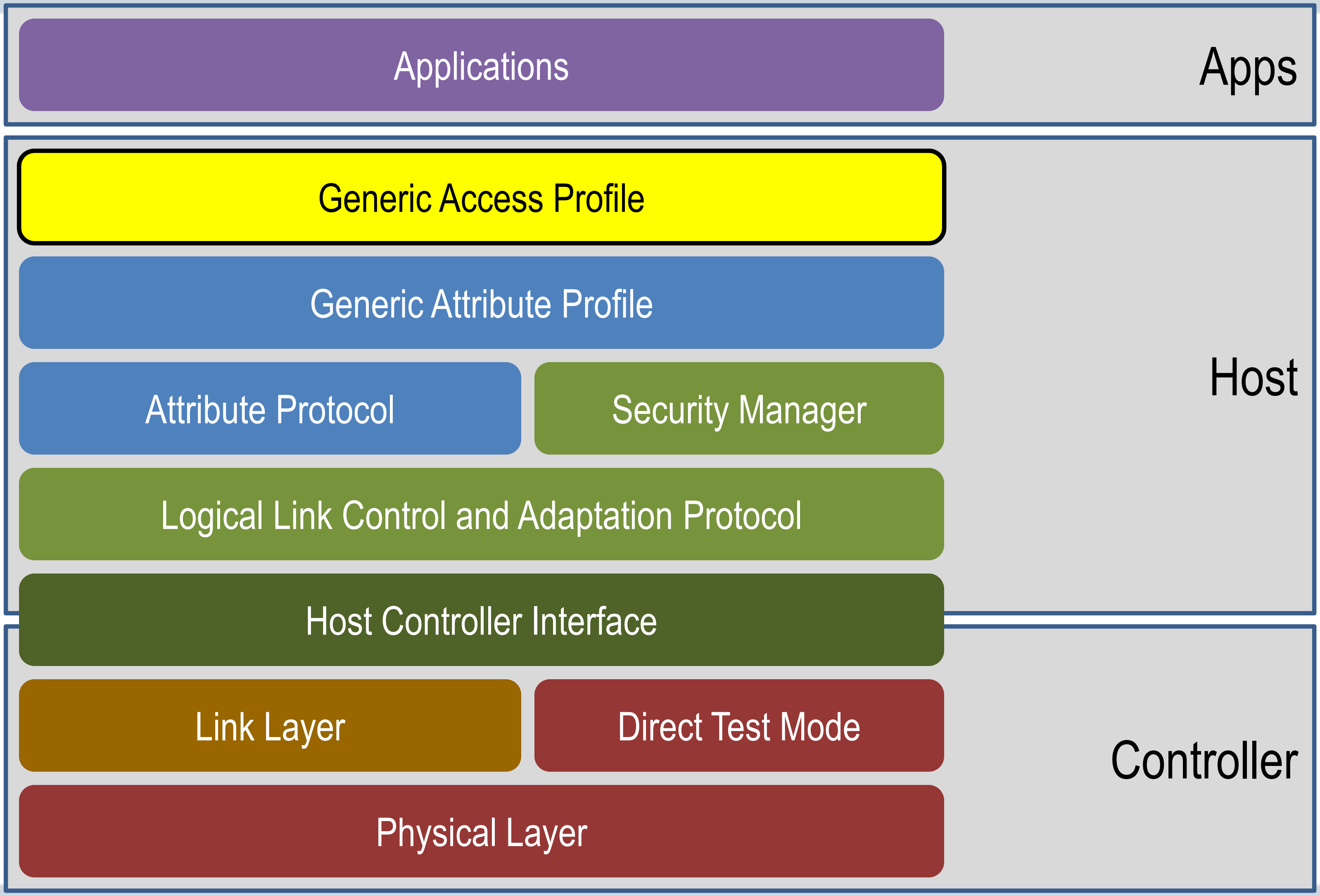
Defines grouping of attributes

- Services
  - Include
- Characteristics
  - Descriptors

Grouping Attributes are called Descriptors

- «Primary Service», «Secondary Service», «Characteristic»

GENERIC ACCESS PROFILE



GENERIC ACCESS PROFILE (GAP)

Profile Roles

- Broadcaster, Observer
- Peripheral, Central

Defines standard ways for devices to connect

- Discoverable, Connectable, Bonding

Privacy

- Resolvable Private Addresses

PROFILE ROLES

Broadcaster

- sends advertising events
- including characteristics
- including service data

Doesn’t need Receiver

- if it does have Receiver,
- can be discoverable

Observer

- receives advertising events
- listens for characteristics
- listens for service data

Doesn’t need Transmitter

- if it does have Transmitter,
- can discover devices

PROFILE ROLES

Peripheral

- Has Transmitter & Receiver
- Always slave
- Connectable advertising

Must support

- All LL Control Procedures
- Encryption optional

Central

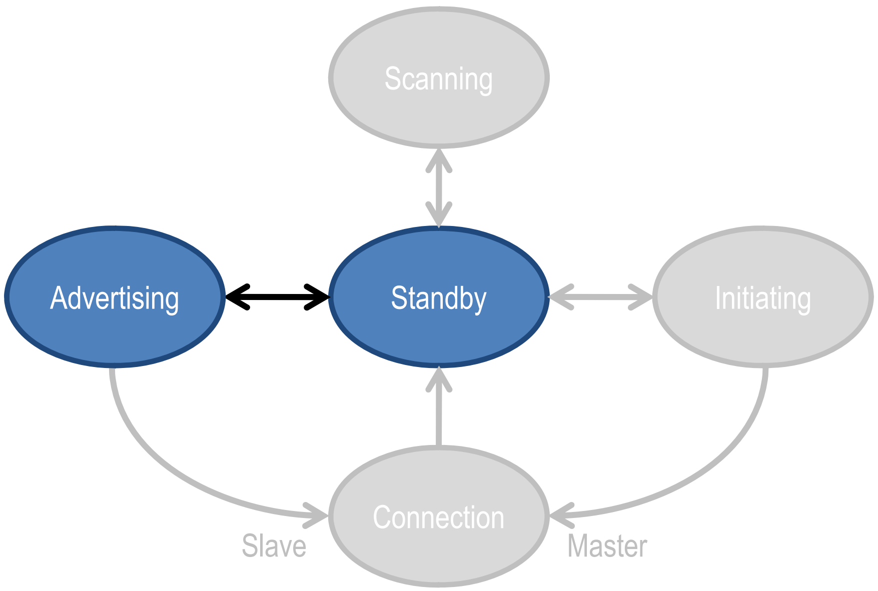
- Has Transmitter & Receiver
- Always master
- Never advertisers

Must support

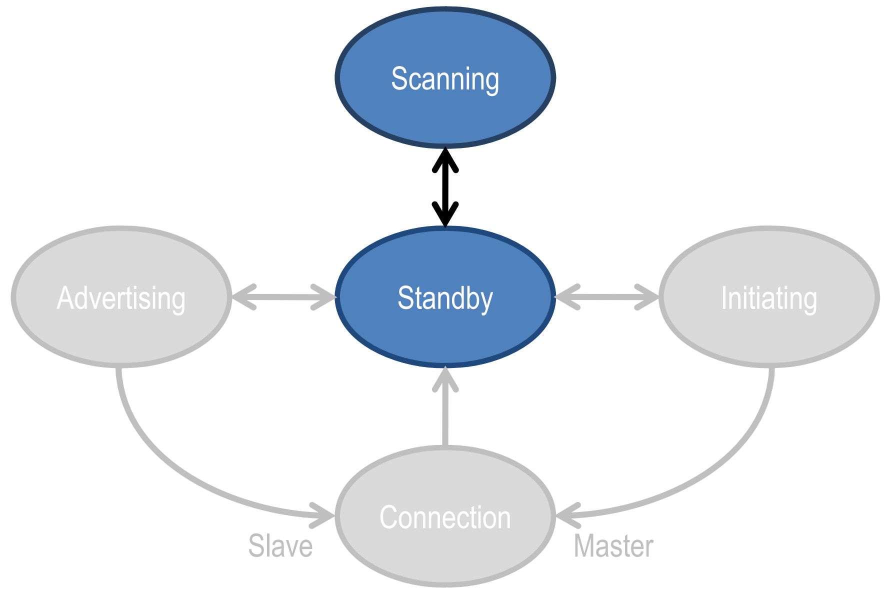
- active or passive scanning
- All LL Control Procedures
- Encryption optional

LINK LAYER STATE MACHINES FOR BROADCASTER AND OBSERVER

Broadcaster

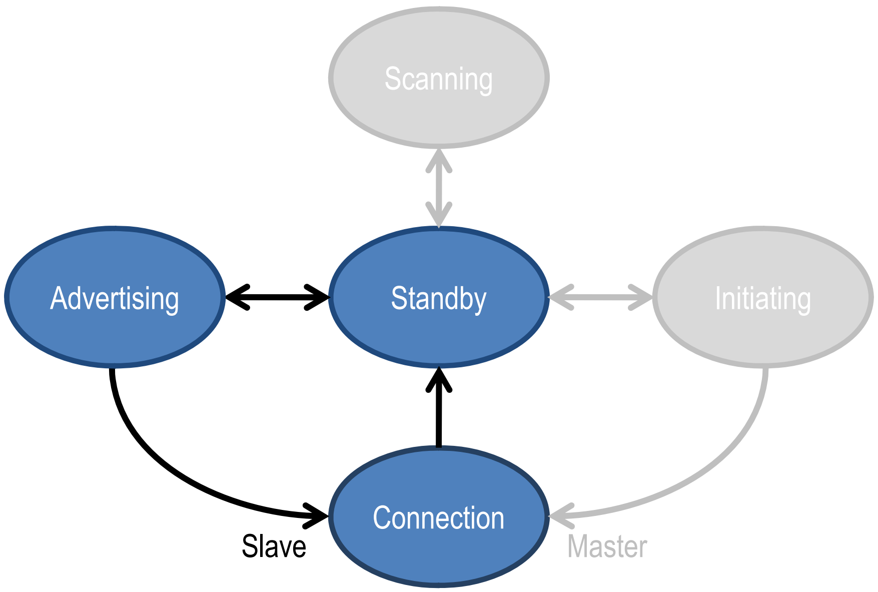


Observer

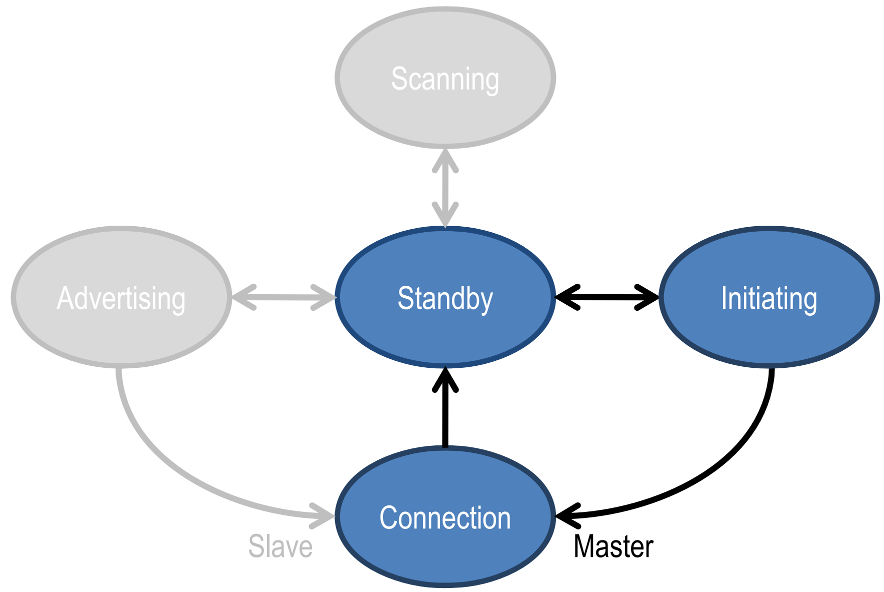


LINK LAYER STATE MACHINES FOR PERIPHERAL AND CENTRAL

Peripheral



Central



ADVERTISING DATA

Can be sent when broadcaster / discoverable peripheral

Many Advertising Data (AD) Types defined:

- Flags
- Service UUIDs
- Local Name
- TX Power Level
- Slave Connection Interval Range
- Signed Data
- Service Solicitation
- Service Data
- Manufacturer Specific Data

FLAGS AD TYPE

Tag Value	Bit	Description
0x01	0	LE Limited Discoverable Mode
	1	LE General Discoverable Mode
	2	BR/EDR Not Supported
	3	Simultaneous LE and BR/EDR to Same Device Capable (Controller)
	4	Simultaneous LE and BR/EDR to Same Device Capable (Host)
	5-7	Reserved

Tag Value	Description
0x02	Non-complete list of 16-bit Service UUIDs
0x03	Complete list of 16-bit Service UUIDs
0x06	Non-complete list of 128-bit Service UUIDs
0x07	Complete list of 128-bit Service UUIDs
0x08	Non-complete shortened local name
0x09	Complete local name
0x0A	Tx Power Level (-127 dBm to +127 dBm)
0x12	Slave Connection Interval Range (min, max)
0x14	Service Solicitation for 16 bit Service UUIDs
0x15	Service Solicitation for 128 bit Service UUIDs
0x16	Service Data (16 bit Service UUID, service data)
0xFF	Manufacturer Specific Data (Company Identifier Code, data)

INTERESTING AD TYPES

Service UUIDs

- can determine which services a device supports without connecting to it

Tx Power Level

- combined with RSSI of advertising packet
- allows sorting of devices by distance in UI
  - sort by distance = Tx Power Level - RSSI

SERVICE SOLICITATION DATA

“I’m desperate – I want a simple remote control to talk to me”

Allows a Peripheral to “advertise” what services it use

e.g.

- Central wants to expose «Simple Remote Service»
  - but is not advertising
- Peripheral wants to use this service
  - Central connects to peripherals that solicit its services

BROADCASTER & OBSERVER MODES AND PROCEDURES

Broadcast Mode

- Flags AD Type
  - LE General Discoverable Mode = 0
  - LE Limited Discoverable Mode = 0
- Uses ADV_NONCONN_IND or ADV_DISCOVER_IND

Observation Procedure

- Listens for all advertising packets
- Can active or passive scan

DISCOVERABLE MODES – PERIPHERAL

Non-Discoverable Mode

- not connectable – default mode
- can advertise
- Flags AD Type
  - LE Limited Discoverable Mode flag = 0
  - LE General Discoverable Mode flag = 0

DISCOVERABLE MODES – PERIPHERAL

Limited Discoverable Mode

- used after user interaction (power on / button press)
- Flags AD Type
  - LE Limited Discoverable Mode flag = 1
  - LE General Discoverable Mode flag = 0

can only be limited discoverable for 30.72 seconds

- ensures that limited discoverable devices have recently been touched by user

DISCOVERABLE MODES – PERIPHERAL

General Discoverable Mode

- used at any time
- Flags AD Type
  - LE Limited Discoverable Mode flag = 0
  - LE General Discoverable Mode flag = 1

typically used with low duty cycle advertising

- should include: TX Power Level AD, Local Name AD, Service UUIDs AD, Slave Connection Interval Range AD

DISCOVERY PROCEDURES

Limited Discovery Procedures

finds devices where LE Limited Discoverable Mode flag = 1

finds only "Limited Discoverable" devices

General Discovery Procedures

finds devices where LE Limited Discoverable Mode flag = 1

or LE General Discoverable Mode flag = 1

finds all "Discoverable" devices

WHAT CAN DISCOVER TO WHAT?

	Non-Discoverable	Limited Discoverable	General Discoverable
Limited Discovery	No	Yes	No
General Discovery	No	Yes	Yes

CONNECTABLE MODES – PERIPHERAL

Non-Connectable Mode

not connectable – default mode

Directed Connectable Mode

connect to specific device – using ADV_DIRECT_IND

Undirected Connectable Mode

connect to any device – using ADV_IND

CONNECTION ESTABLISHMENT PROCEDURES – CENTRAL

Auto Connection Establishment Procedures
- automatically connect to a set of devices – uses white lists

General Connection Establishment Procedure
- connect to any device – supports private connections

Selective Connection Establishment Procedure
- connect to set of devices – separate configuration per device

Direct Connection Establishment Procedure
- connect to “that” device – any private / unknown device possible

WHAT CAN CONNECT TO WHAT?

	Non-Connectable	Directed Connectable	Undirected Connectable
Auto Connection	No	Yes if in list	Yes if in list
General Connection	No	Yes if in list	Yes if in list
Selective Connection	No	Yes if in list	Yes if in list
Direct Connection	No	Yes	Yes

BONDING

Security Manager defines how to “pair” devices

- authenticate and then encrypt a link

Bonding is the storing of

- Security and Identity Information

Once bonded

- a device can reconnect using the stored information
- GATT will store service change information for device

BONDING IS SIMPLE

Non-Bondable Mode

default mode – does not allow bonding

Bondable Mode

allows bonding

Bonding Procedure

only when both devices are in bondable mode

ONLY BOND IF BOTH ARE BONDABLE

	Non-Bondable	Bondable
Non-Bondable	No	No
Bondable	No	Yes

LE SECURITY MODES

GAP defines two security modes:

- LE Security Mode 1 (Link Layer Encryption)
  - Level 1: No Security (no authentication, no encryption)
  - Level 2: Unauthenticated pairing with encryption
  - Level 3: Authenticated pairing with encryption
- LE Security Mode 2 (Signed Data)
  - Authenticated Pairing with Data Signing

PRIVACY

Two characteristics on Peripheral

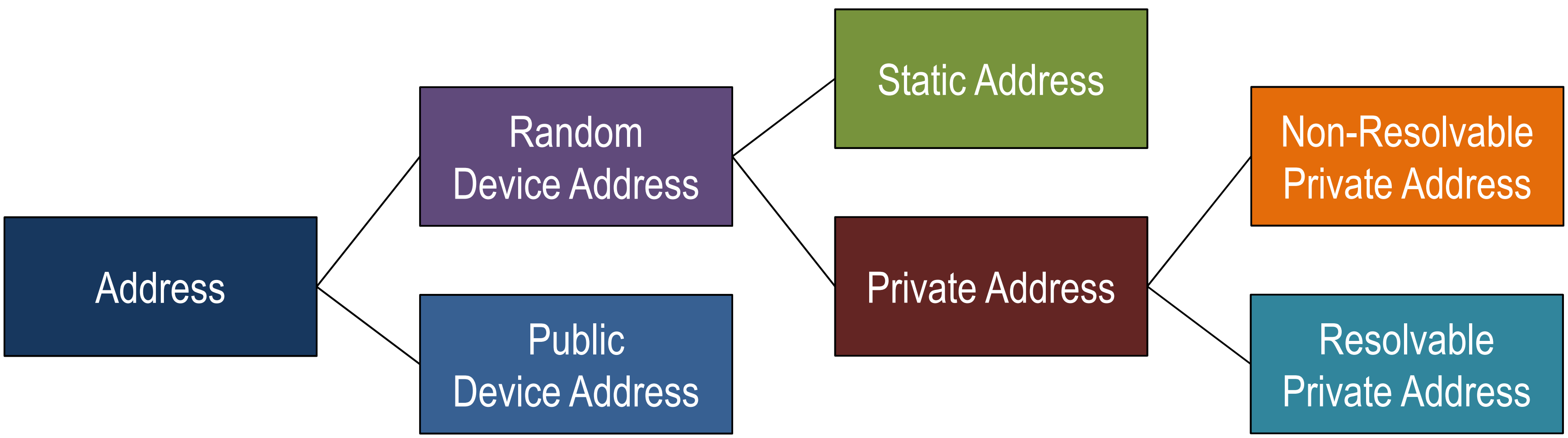
- Peripheral Privacy Flag Characteristic
- Reconnection Address Characteristic

Peripheral Privacy Flag Characteristic	Reconnection Address Characteristic	Privacy Support in Peripheral
Disabled	Not Exists	No Privacy
Enabled	Not Exists	Privacy in Undirected Connectable Mode only
Disabled	Exists	Not Allowed
Enabled	Exists	Privacy in Directed Connectable Mode and in Undirected Connectable Mode

PRIVATE ADDRESSES

A type of Random Device Address

- sub-categorized into either:
  - Non-Resolvable Private Address
  - Resolvable Private Address



ADDRESS TYPES

					TxAdd RxAdd
Public Device Address	company_assigned (24 bits)	company_id (24 bits)			0
Static Device Address	random part of static address (46 bits)		1	1	1
Non-Resolvable Device Address	random part of static address (46 bits)		0	0	1
Resolvable Device Address	hash (24 bits)	prand (22 bits)	1	0	1

ADDRESS TYPES

					TxAdd RxAdd
Resolvable Device Address	hash (24 bits)	prand (22 bits)	1	0	1

hash = func (IRK, prand)

GAP CHARACTERISTICS

Characteristic	Description
«Device Name»	the local name of the device
«Appearance»	enumeration of what the device “looks like”
«Peripheral Privacy Flag»	does this peripheral support privacy – is it enabled
«Reconnection Address»	address to use when reconnecting to a private device
«Peripheral Preferred Connection Parameters»	what this peripheral would prefer the central connect with

GAP SUMMARY

Profile Roles

- Broadcaster, Observer, Peripheral, Central

Defines modes and procedures for

- Discovery, Connections, Bonding

Privacy

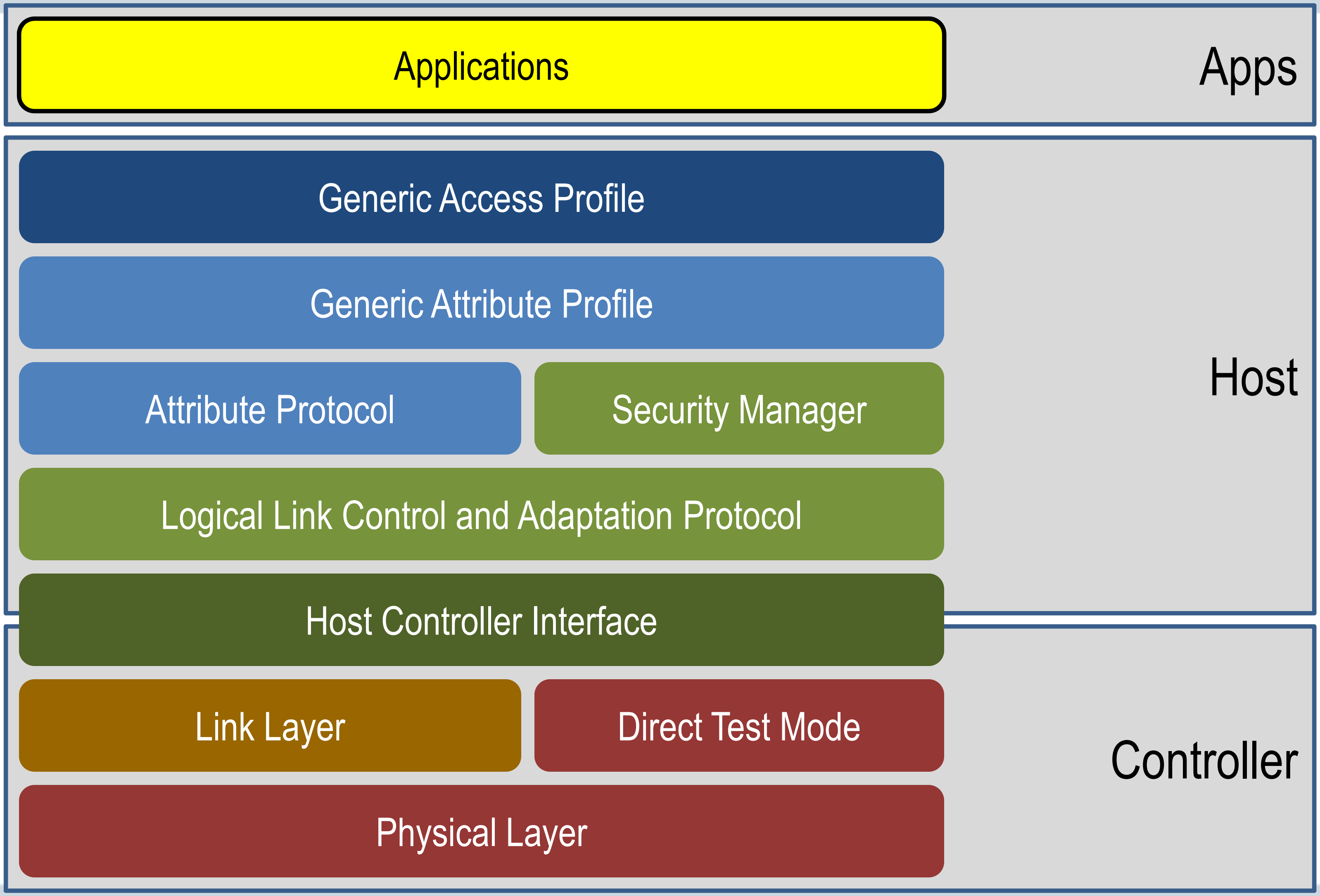
- Non-Resolvable and Resolvable Private Addresses

AFTERNOON BREAK

return at 15:45

Bluetooth®
SPECIAL INTEREST GROUP

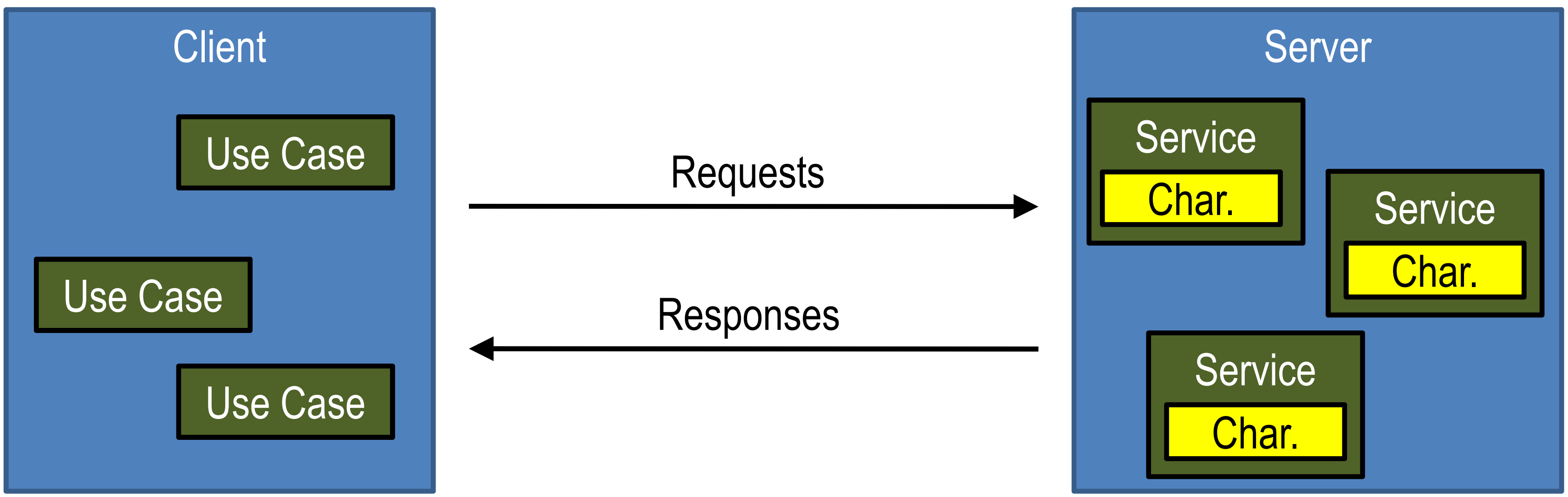
APPLICATIONS



APPLICATIONS

Client Server Architecture

- Services – exposes behavior that have characteristics
- Use Cases– define how to use services on a peer



Service != Profile

Use Case != Profile

Use Case != Service

WHY SPLIT PROFILES?

There is not a one to one link between services and use cases

«Immediate Alert» Service

Could be used by

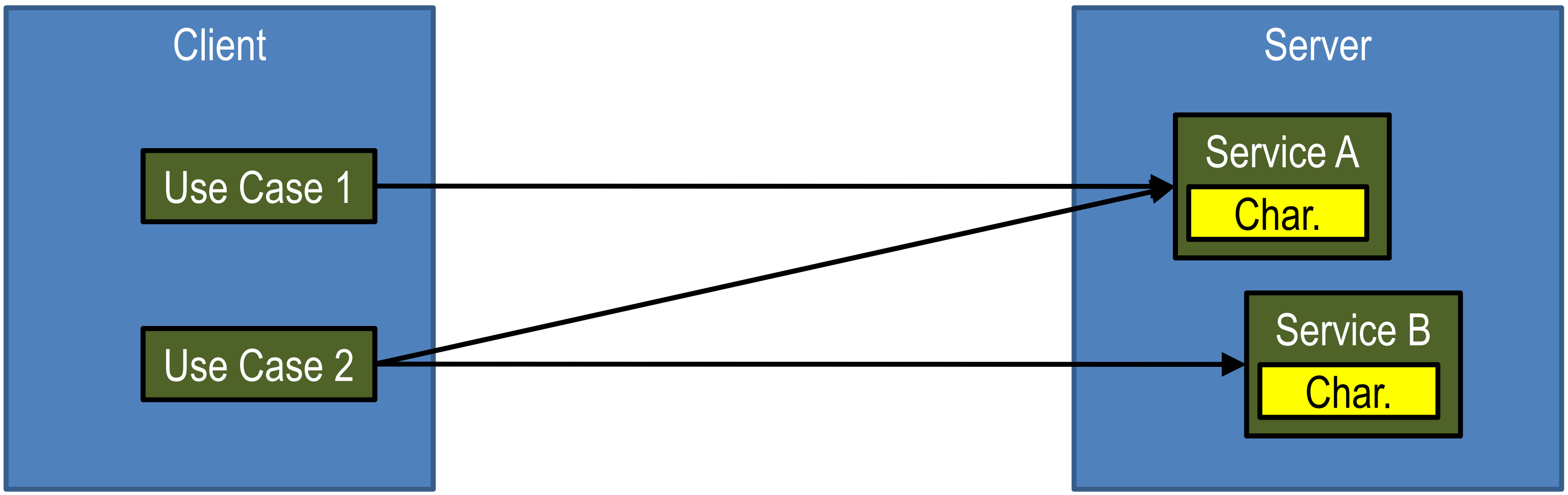
Proximity Use Case

Device Selection Use Case

APPLICATIONS

Use Case 1 uses Service «A»

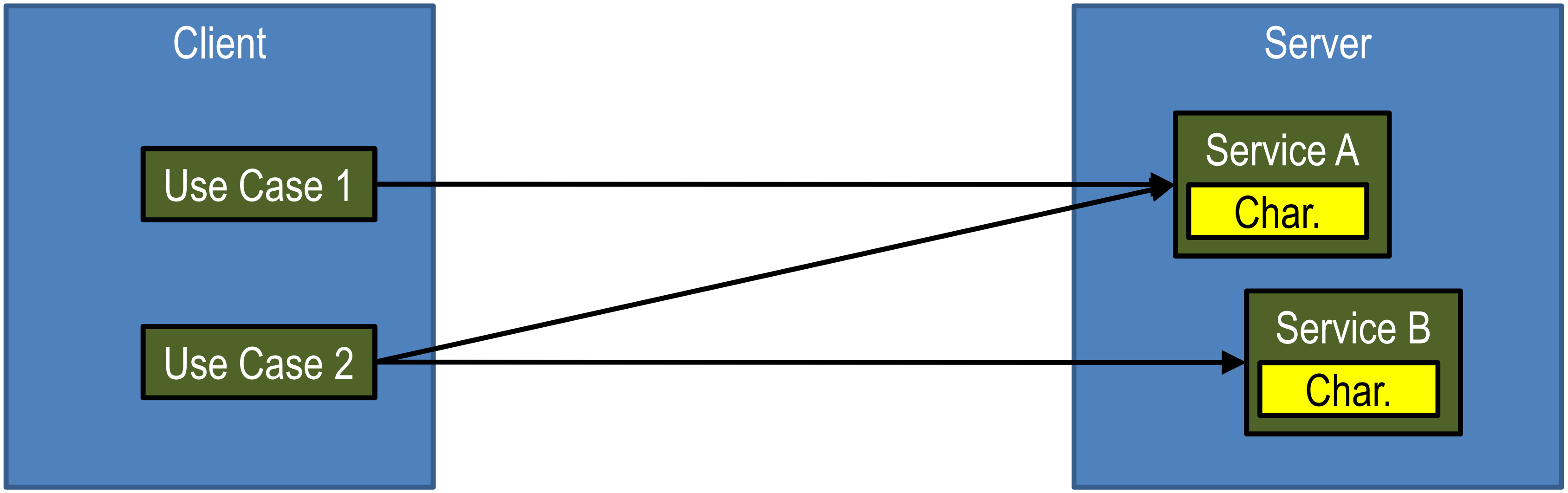
Use Case 2 uses Service «A» and Service «B»



APPLICATIONS

Device Selection uses «Immediate Alert»

Proximity uses «Immediate Alert» and «Transmit Power»



WHY ATOMIZE SERVICES

Services expose an atomic bit of behavior in a device

If you have an X, you implement Service «X»

- If you have a temperature sensor
  - you implement «Temperature» Service

- If you have the ability to monitor the battery status
  - you implement «Battery Status» Service

SERVICE != USE CASE

Services expose what the device does
not how a peer device uses it

Use Cases determine what Service are required

Proximity uses:
«Transmit Power», «Immediate Alert», «Link Loss Alert»

Device Selection uses:
«Immediate Alert»

CLIENTS AND SERVICES

Clients implement Use Cases, Servers implement Services

A device can be a client and a server at the same time

Phone may implement:

- «Network Availability» Service (as a Server)
- «Immediate Alert» Service (as a Server)
- Proximity Use Case (as a Client)
- Sports Sensor Use Case (as a Client)

SCHEMAS CAN USE MANY SERVICES

Sports Sensor Use Case

- defines how to talk to many sports sensor services
- can use one or more of these services:
  - «Heart Rate», «Peddle Power», «Cadence», «Pedometer», «Foot Fall», «Position», «Speed», «Elevation», «Inclination», etc…

WHAT IS AN APPLICATION?

An Application uses a set of Use Cases

- Use Cases use a set of Services on a peer device
  - Services expose Characteristics
  - Services define behavior exposed by Characteristics

HOW DO CREATE LE APPLICATIONS

Steps:

- What state is there?
- Where is this state?
- What data does the state expose?
- How does the data work?
- How is the state machine controlled?
- What is common and can be abstracted?
- What if the client disconnects?
- What is the final set of schemas?

WHAT STATE IS THERE?

Application	State
Thermometer	Current Temperature
Light	On / Off
Clock	Current Time
Radio Clock	Current Time Trigger Time Synchronization Time Synchronization Progress

WHERE IS THE STATE?

Application	State	Server is located in
Thermometer	Current Temperature	Sensor Device
Light	On / Off	Light Controller
Clock	Current Time	Clock
Radio Clock	Current Time Trigger Time Synchronization Time Synchronization Progress	Clock

WHAT DATA DOES THE STATE EXPOSE?

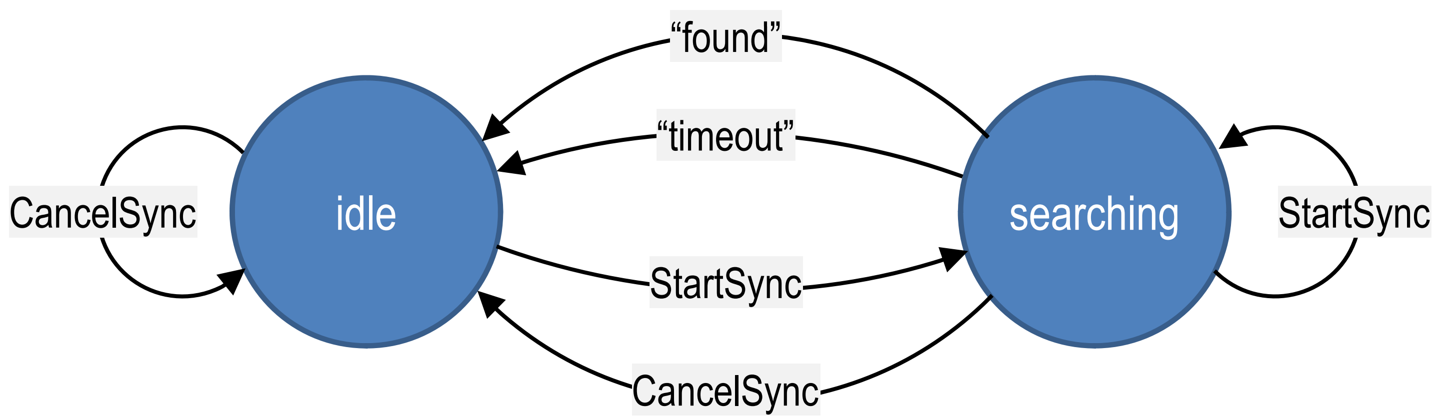
Application	Characteristic	Data Format
Thermometer	Current Temperature	sint16, 10^{-1}, Celsius
Light	On / Off	uint8, 10^{0}, Binary
Clock	Current Time	Localtime Characteristic
Radio Clock	Current Time Trigger Time Synchronization Time Synchronization Progress	Localtime Characteristic

HOW DOES THE DATA WORK?

Application	Service	Behavior
Thermometer	Temperature Service	Read current temperature
Light	Light Controller Service	Read current light state Write light on / off / toggle
Clock	Current Time	Read current time
Radio Clock	Time Synchronization	Read current time Start time synchronization Cancel time synchronization Notification of sync. progress

HOW IS THE STATE MACHINE CONTROLLED?

State	Signal	New State
idle	StartSync	searching
idle	CancelSync	idle
searching	StartSync	searching
searching	CancelSync	idle
searching	“found”	idle
searching	“timeout”	idle



HOW IS THE STATE MACHINE CONTROLLED?

Define “Control Point” characteristics

- enumerate all possible inputs
- define behavior when input received
- define even when input is received in “invalid” state

Also expose “State Machine” state characteristic

- allows a client to discover current state
- even after a reconnection
- allows lower power “disconnected” modes

WHAT IS COMMON AND CAN BE ABSTRACTED?

Time Synchronization Service requires

- Current Time, Time Sync. CP, Time Sync. State

Local Time Service requires

- Current Time

Abstract Current Time into own Service

- Time Synchronization references Current Time
- «Current Time» Service may be useful in other contexts

WHAT IF THE CLIENT DISCONNECTS?

No state or behavior in clients

- clients can disconnect at any time
- clients can cache information from server

Upon reconnection

- clients can re-establish current server state
- servers should keep state small – to speed this up

WHAT IF THE CLIENT DISCONNECTS?

Server can remember that client wants state updates

- reconnect to client when state changes
- then notify client with new state

Alternatively

- provide control points to allow control without state

- “Toggle Light” instead of read light state, write !light state

WHAT IS THE FINAL SET OF SCHEMAS?

An Application uses a set of Use Cases

- Use Cases use a set of Services on a peer device
  - Services expose Characteristics
  - Services define behavior exposed by Characteristics

Remember:

- Use Cases define how to use services for given application
- Services define behavior on characteristics
- Characteristics define interoperable data formats

EXAMPLE APPLICATIONS

Proximity

Device Selection

PROXIMITY APPLICATION

Description:

When is a device close / far away / very far away;

cause an alert

How will this work?

if a device disconnects, cause an alert

alert on link loss

if the device is too far away, cause an alert

alert on "path loss"

PROXIMITY APPLICATION

What state is there?

- ability to cause an alert – on server
- ability to measure Tx Power of peer device – on server
- ability to get RSSI of packets from peer device – on client

Server Characteristics:

- «Tx Power»
- «Alert Level»

PROXIMITY APPLICATION

How does the data work?

«Tx Power» Characteristic

dBm of transmitter when in a connection

Range: -20 dBm to +10 dBm

Format: sint8, 10^0, dBm

«Alert Level» Characteristic

enumeration of {none, mild, high}

Format: uint8, 10^0, enumeration

PROXIMITY APPLICATION

What behavior is exposed:

if a device disconnects, cause an alert

- alert on link loss
  - «Link Loss» Service

if the device is too far away, cause an alert

- alert on “path loss”
  - «Immediate Alert» Service
  - «Tx Power» Service

PROXIMITY APPLICATION

«Link Loss» Service

«Alert Level» Characteristic

Behavior: On link loss, cause alert as enumerated

«Immediate Alert» Service

«Alert Level» Characteristic

Behavior: when written, cause alert as enumerated

«Tx Power» Service

«Tx Power» Characteristic

Behavior: when read, reports current Tx Power for connection

PROXIMITY APPLICATION

Proximity Use Case

- Defines use of «Tx Power» Service with local RSSI
  - calculate “Path Loss”
  - write «Immediate Alert» «Alert Level» above threshold

Link Loss Use Case

- Defines use of «Link Loss» Service
  - write «Link Loss» «Alert Level» with alert level
  - could cause alert on client also
  - if write ‘none’, disabled this service
  - allows client to gracefully disconnect

PROXIMITY APPLICATION

Handle	Type	Value
0x0010	«Primary Service»	«Link Loss»
0x0011	«Characteristic»	{rw, 0x0012, «Alert Level»}
0x0012	«Alert Level»	0x00
0x0020	«Primary Service»	«Tx Power»
0x0021	«Characteristic»	{r, 0x0022, «Tx Power dBm»}
0x0022	«Tx Power dBm»	0x04
0x0030	«Primary Service»	«Immediate Alert»
0x0031	«Characteristic»	{w, 0x0032, «Alert Level»}
0x0032	«Alert Level»	

PROXIMITY APPLICATION

Handle	Type	Value
0x0010	«Primary Service»	«Link Loss»
0x0011	«Characteristic»	{rw, 0x0012, «Alert Level»}
0x0012	«Alert Level»	0x00
0x0020	«Primary Service»	«Tx Power»
0x0021	«Characteristic»	{r, 0x0022, «Tx Power dBm»}
0x0022	«Tx Power dBm»	0x04
0x0030	«Primary Service»	«Immediate Alert»
0x0031	«Characteristic»	{w, 0x0032, «Alert Level»}
0x0032	«Alert Level»	

DEVICE SELECTION APPLICATION

Description:

When is a device is selected on user interface

cause that device to alert user

State:

device selected on client

alert level on server

DEVICE SELECTION APPLICATION

What state is there?

ability to cause an alert – on server

Server Characteristics:

«Alert Level»

DEVICE SELECTION APPLICATION

How does the data work?

«Alert Level» Characteristic

- enumeration of {none, mild, high}
- Format: uint8, 10^0, enumeration

DEVICE SELECTION APPLICATION

What behavior is exposed:

when the device is selected, cause an alert

- alert on device selection on client
  - «Immediate Alert» Service

DEVICE SELECTION APPLICATION

«Immediate Alert» Service

- «Alert Level» Characteristic
- Behavior: when written, cause alert as enumerated

DEVICE SELECTION APPLICATION

Device Selection Use Case

- Defines use of «Immediate Alert» Service
  - write «Alert Level» when selected
  - clear «Alert Level» when unselected

DEVICE SELECTION APPLICATION

Handle	Type	Value
0x0030	«Primary Service»	«Immediate Alert»
0x0031	«Characteristic»	{w, 0x0032, «Alert Level»}
0x0032	«Alert Level»	

WHAT DOES THIS DEVICE SUPPORT?

Handle	Type	Value
0x0010	«Primary Service»	«Link Loss»
0x0011	«Characteristic»	{rw, 0x0012, «Alert Level»}
0x0012	«Alert Level»	0x00
0x0020	«Primary Service»	«Tx Power»
0x0021	«Characteristic»	{r, 0x0022, «Tx Power dBm»}
0x0022	«Tx Power dBm»	0x04
0x0030	«Primary Service»	«Immediate Alert»
0x0031	«Characteristic»	{w, 0x0032, «Alert Level»}
0x0032	«Alert Level»	

WHAT USE CASES DOES THIS SERVER SUPPORT?

Services	Use Cases
«Link Loss»	Link Loss
«Immediate Alert», «Tx Power»	Proximity
«Immediate Alert»	Device Selection

Behavior of «Tx Power» and «Immediate Alert» separated

- interaction defined in Use Cases
- no multi-profile issues

WHAT SERVICES DOES THIS CLIENT SUPPORT?

Use Cases	Services
Link Loss	«Link Loss»
Proximity	«Immediate Alert», «Tx Power»
Device Selection	«Immediate Alert»

Behavior of «Tx Power» and «Immediate Alert» separated

- interaction defined in Use Cases
- no multi-profile issues

SERVICES ARE ATOMIC

A service exposes behavior around one bit of a device

must not be effected by any other service in the device

If a bit of a device is effected

a service must be defined

that defined the behavior between referenced services

network effect

1 Service = 1 Use Case

A

network effect

2 Services = 3 Use Cases

A, B, AB

network effect

3 Services = 7 Use Cases

A, B, AB, C, AC, BC, ABC

network effect

4 Services = 15 Use Cases

A, B, AB, C, AC, BC, ABC, D, AD, BD, ABD, CD, ACD, BCD, ABCD

network effect

5 Services = 31 Use Cases

A, B, AB, C, AC, BC, ABC, D, AD, BD, ABD, CD, ACD, BCD, ABCD, E, AE, BE, ABE, CE, ACE, BCE, ABCE, DE, ADE, BDE, ABDE, CDE, ACDE, BCDE, ABCDE

network effect

6 Services = 63 Use Cases

A, B, AB, C, AC, BC, ABC, D, AD, BD, ABD, CD, ACD, BCD, ABCD, E, AE, BE, ABE, CE, ACE, BCE, ABCE, DE, ADE, BDE, ABDE, CDE, ACDE, BCDE, ABCDE, F, AF, BF, ABF, CF, ACF, BCF, ABCF, DF, ADF, BDF, ABDF, CDF, ACDF, BCDF, ABCDF, EF, AEF, BEF, ABEF, CEF, ACEF, BCEF, ABCEF, DEF, ADEF, BDEF, ABDEF, CDEF, ACDEF, BCDEF, ABCDEF

network effect

7 Services = 127 Use Cases

A, B, AB, C, AC, BC, ABC, D, AD, BD, ABD, CD, ACD, BCD, ABCD, E, AE, BE, ABE, CE, ACE, BCE, ABCE, DE, ADE, BDE, ABDE, CDE, ACDE, BCDE, ABCDE, F, AF, BF, ABF, CF, ACF, BCF, ABCF, DF, ADF, BDF, ABDF, CDF, ACDF, BCDF, ABCDF, EF, AEF, BEF, ABEF, CEF, ACEF, BCEF, ABCEF, DEF, ADEF, BDEF, ABDEF, CDEF, ACDEF, BCDEF, ABCDEF, G, AG, BG, ABG, CG, ACG, BCG, ABCG, DG, ADG, BDG, ABDG, CDG, ACDG, BCDG, ABCDG, EG, AEG, BEG, ABEG, CEG, ACEG, BCEG, ABCEG, DEG, ADEG, BDEG, ABDEG, CDEG, ACDEG, BCDEG, ABCDEG, FG, AFG, BFG, ABFG, CFG, ACFG, BCFG, ABCFG, DFG, ADFG, BDFG, ABDFG, CDFG, ACDFG, BCDFG, ABCDFG, EFG, AEFG, BEFG, ABEFG, CEFG, ACEFG, BCEFG, ABCEFG, DEFG, ADEFG, BDEFG, ABDEFG, CDEFG, ACDEFG, BCDEFG, ABCDEFG

network effect

8 Services = 255 Use Cases

A, B, AB, C, AC, BC, ABC, D, AD, BD, ABD, CD, ACD, BCD, ABCD, E, AE, BE, ABE, CE, ACE, BCE, ABCE, DE, ADE, BDE, ABDE, CDE, ACDE, BCDE, ABCDE, F, AF, BF, ABF, CF, ACF, BCF, ABCF, DF, ADF, BDF, ABDF, CDF, ACDF, BCDF, ABCDF, EF, AEF, BEF, ABEF, CEF, ACEF, BCEF, ABCEF, DEF, ADEF, BDEF, ABDEF, CDEF, ACDEF, BCDEF, ABCDEF, G, AG, BG, ABG, CG, ACG, BCG, ABCG, DG, ADG, BDG, ABDG, CDG, ACDG, BCDG, ABCDG, EG, AEG, BEG, ABEG, CEG, ACEG, BCEG, ABCEG, DEG, ADEG, BDEG, ABDEG, CDEG, ACDEG, BCDEG, ABCDEG, FG, AFG, BFG, ABFG, CFG, ACFG, BCFG, ABCFG, DFG, ADFG, BDFG, ABDFG, CDFG, ACDFG, BCDFG, ABCDFG, EFG, AEFG, BEFG, ABEFG, CEFG, ACEFG, BCEFG, ABCEFG, DEFG, ADEFG, BDEFG, ABDEFG, CDEFG, ACDEFG, BCDEFG, ABCDEFG, H, AH, BH, ABH, CH, ACH, BCH, ABCH, DH, ADH, BDH, ABDH, CDH, ACDH, BCDH, ABCDH, EH, AEH, BEH, ABEH, CEH, ACEH, BCEH, ABCEH, DEH, ADEH, BDEH, ABDEH, CDEH, ACDEH, BCDEH, ABCDEH, FH, AFH, BFH, ABFH, CFH, ACFH, BCFH, ABCFH, DFH, ADFH, BDFH, ABDFH, CDFH, ACDFH, BCDFH, ABCDFH, EFH, AEFH, BEFH, ABEFH, CEFH, ACEFH, BCEFH, ABCEFH, DEFH, ADEFH, BDEFH, ABDEFH, CDEFH, ACDEFH, BCDEFH, ABCDEFH, GH, AGH, BGH, ABGH, CGH, ACGH, BCGH, ABCGH, DGH, ADGH, BDGH, ABDGH, CDGH, ACDGH, BCDGH, ABCDGH, EGH, AEGH, BEGH, ABEGH, CEGH, ACEGH, BCEGH, ABCEGH, DEGH, ADEGH, BDEGH, ABDEGH, CDEGH, ACDEGH, BCDEGH, ABCDEGH, FGH, AFGH, BFGH, ABFGH, CFGH, ACFGH, BCFGH, ABCFGH, DFGH, ADFGH, BDFGH, ABDFGH, CDFGH, ACDFGH, BCDFGH, ABCDFGH, EFGH, AEFGH, BEFGH, ABEFGH, CEFGH, ACEFGH, BCEFGH, ABCEFGH, DEFGH, ADEFGH, BDEFGH, ABDEFGH, CDEFGH, ACDEFGH, BCDEFGH, ABCDEFGH

network effect

9 Services = 511 Use Cases

A, B, AB, C, AC, BC, ABC, D, AD, BD, ABD, CD, ACD, BCD, ABCD, E, AE, BE, ABE, CE, ACE, BCE, ABCE, DE, ADE, BDE, ABDE, CDE, ACDE, BCDE, ABCDE, F, AF, BF, ABF, CF, ACF, BCF, ABCF, DF, ADF, BDF, ABDF, CDF, ACDF, BCDF, ABCDF, EF, AEF, BEF, ABEF, CEF, ACEF, BCEF, ABCEF, DEF, ADEF, BDEF, ABDEF, CDEF, ACDEF, BCDEF, ABCDEF, G, AG, BG, ABG, CG, ACG, BCG, ABCG, DG, ADG, BDG, ABDG, CDG, ACDG, BCDG, ABCDG, EG, AEG, BEG, ABEG, CEG, ACEG, BCEG, ABCEG, DEG, ADEG, BDEG, ABDEG, CDEG, ACDEG, BCDEG, ABCDEG, FG, AFG, BFG, ABFG, CFG, ACFG, BCFG, ABCFG, DFG, ADFG, BDFG, ABDFG, CDFG, ACDFG, BCDFG, ABCDFG, EFG, AEFG, BEFG, ABEFG, CEFG, ACEFG, BCEFG, ABCEFG, DEFG, ADEFG, BDEFG, ABDEFG, CDEFG, ACDEFG, BCDEFG, ABCDEFG, H, AH, BH, ABH, CH, ACH, BCH, ABCH, DH, ADH, BDH, ABDH, CDH, ACDH, BCDH, ABCDH, EH, AEH, BEH, ABEH, CEH, ACEH, BCEH, ABCEH, DEH, ADEH, BDEH, ABDEH, CDEH, ACDEH, BCDEH, ABCDEH, FH, AFH, BFH, ABFH, CFH, ACFH, BCFH, ABCFH, DFH, ADFH, BDFH, ABDFH, CDFH, ACDFH, BCDFH, ABCDFH, EFH, AEFH, BEFH, ABEFH, CEFH, ACEFH, BCEFH, ABCEFH, DEFH, ADEFH, BDEFH, ABDEFH, CDEFH, ACDEFH, BCDEFH, ABCDEFH, GH, AGH, BGH, ABGH, CGH, ACGH, BCGH, ABCGH, DGH, ADGH, BDGH, ABDGH, CDGH, ACDGH, BCDGH, ABCDGH, EGH, AEGH, BEGH, ABEGH, CEGH, ACEGH, BCEGH, ABCEGH, DEGH, ADEGH, BDEGH, ABDEGH, CDEGH, ACDEGH, BCDEGH, ABCDEGH, FGH, AFGH, BFGH, ABFGH, CFGH, ACFGH, BCFGH, ABCFGH, DFGH, ADFGH, BDFGH, ABDFGH, CDFGH, ACDFGH, BCDFGH, ABCDFGH, EFGH, AEFGH, BEFGH, ABEFGH, CEFGH, ACEFGH, BCEFGH, ABCEFGH, DEFGH, ADEFGH, BDEFGH, ABDEFGH, CDEFGH, ACDEFGH, BCDEFGH, ABCDEFGH, …

… I, AI, BI, ABI, CI, ACI, BCI, ABCI, DI, ADI, BDI, ABDI, CDI, ACDI, BCDI, ABCDI, EI, AEI, BEI, ABEI, CEI, ACEI, BCEI, ABCEI, DEI, ADEI, BDEI, ABDEI, CDEI, ACDEI, BCDEI, ABCDEI, FI, AFI, BFI, ABFI, CFI, ACFI, BCFI, ABCFI, DFI, ADFI, BDFI, ABDFI, CDFI, ACDFI, BCDFI, ABCDFI, EFI, AEFI, BEFI, ABEFI, CEFI, ACEFI, BCEFI, ABCEFI, DEFI, ADEFI, BDEFI, ABDEFI, CDEFI, ACDEFI, BCDEFI, ABCDEFI, GI, AGI, BGI, ABGI, CGI, ACGI, BCGI, ABCGI, DGI, ADGI, BDGI, ABDGI, CDGI, ACDGI, BCDGI, ABCDGI, EGI, AEGI, BEGI, ABEGI, CEGI, ACEGI, BCEGI, ABCEGI, DEGI, ADEGI, BDEGI, ABDEGI, CDEGI, ACDEGI, BCDEGI, ABCDEGI, FGI, AFGI, BFGI, ABFGI, CFGI, ACFGI, BCFGI, ABCFGI, DFGI, ADFGI, BDFGI, ABDFGI, CDFGI, ACDFGI, BCDFGI, ABCDFGI, EFGI, AEFGI, BEFGI, ABEFGI, CEFGI, ACEFGI, BCEFGI, ABCEFGI, DEFGI, ADEFGI, BDEFGI, ABDEFGI, CDEFGI, ACDEFGI, BCDEFGI, ABCDEFGI, HI, AHI, BHI, ABHI, CHI, ACHI, BCHI, ABCHI, DHI, ADHI, BDHI, ABDHI, CDHI, ACDHI, BCDHI, ABCDHI, EHI, AEHI, BEHI, ABEHI, CEHI, ACEHI, BCEHI, ABCEHI, DEHI, ADEHI, BDEHI, ABDEHI, CDEHI, ACDEHI, BCDEHI, ABCDEHI, FHI, AFHI, BFHI, ABFHI, CFHI, ACFHI, BCFHI, ABCFHI, DFHI, ADFHI, BDFHI, ABDFHI, CDFHI, ACDFHI, BCDFHI, ABCDFHI, EFHI, AEFHI, BEFHI, ABEFHI, CEFHI, ACEFHI, BCEFHI, ABCEFHI, DEFHI, ADEFHI, BDEFHI, ABDEFHI, CDEFHI, ACDEFHI, BCDEFHI, ABCDEFHI, GHI, AGHI, BGHI, ABGHI, CGHI, ACGHI, BCGHI, ABCGHI, DGHI, ADGHI, BDGHI, ABDGHI, CDGHI, ACDGHI, BCDGHI, ABCDGHI, EGHI, AEGHI, BEGHI, ABEGHI, CEGHI, ACEGHI, BCEGHI, ABCEGHI, DEGHI, ADEGHI, BDEGHI, ABDEGHI, CDEGHI, ACDEGHI, BCDEGHI, ABCDEGHI, FGHI, AFGHI, BFGHI, ABFGHI, CFGHI, ACFGHI, BCFGHI, ABCFGHI, DFGHI, ADFGHI, BDFGHI, ABDFGHI, CDFGHI, ACDFGHI, BCDFGHI, ABCDFGHI, EFGHI, AEFGHI, BEFGHI, ABEFGHI, CEFGHI, ACEFGHI, BCEFGHI, ABCEFGHI, DEFGHI, ADEFGHI, BDEFGHI, ABDEFGHI, CDEFGHI, ACDEFGHI, BCDEFGHI, ABCDEFGHI

network effect

100 Services =
1,267,650,600,228,229,401,496,703,205,375
Possible Use Cases

network effect

200 Services = ~$1.6 * 10^{60}$ Possible Use Cases

just 200 Services ≈ novemdecillion Use Cases

SIMPLE REMOTE CONTROL SERVICES AND SCHEMAS

«Simple Remote Control» service on TV / DVD / Player

- «Immediate Target CP» characteristic
- When written to – command performed

Remote Control Device Use Case

- just a client to the «Simple Remote Control» service

PROXIMITY SERVICES

«Immediate Alert» service

- «Alert Level» characteristic
- immediately alert to given level

«Tx Power» service

- «dBm» characteristic
- readable transmit power level

«Link Loss Alert» service

- «Alert Level» characteristic
- alert upon link loss

PROXIMITY USE CASES

Link Loss Proximity Use Case

- use the «Link Loss Alert» service to alert upon link loss
- can also alert locally upon link loss

Range Proximity Use Case

- read «Tx Power» service's characteristic
- calculate path loss based on Tx Power and RSSI
- use «Immediate Alert» service to alert when outside range

AUTOMATION SERVICES

«Light Control» service

- «Power On Off» characteristic
- readable / writable; turns light on and off
- «Power CP» characteristic
- control point to turn light on and off, or toggle state

«Appliance Power» service

- «Power On Off» characteristic
- readable / writable; turns appliance on and off

AUTOMATION SCHEMAS

Lighting Control Use Case

uses «Light Control» service to turn lights on and off

can read «Light Control» to determine if lights still on or off

Appliance Control Use Case

uses «Appliance Control» service to turn on appliances

Timed Appliance Use Case

uses «Appliance Control» and «Current Time» services

allows appliances to turn on / off at set times

AUTOMATION SERVICES

«Utility Meter» service

- «Current Use» characteristic
  - how much are you using now
- «Current Cost» characteristic
  - how much is it costing per unit
- «Future Costs» characteristic
  - how much will it cost in 15, 30, 45, etc. minutes time
- «Historic Usage Data» characteristic
  - averaged usage data (per hour, day, week)

AUTOMATION SERVICES

- «Energy Broker» service
  - «Energy Request CP» characteristic
    - written by appliances that want to “book” energy
    - can include QoS requirements
  - «Energy Grant» characteristic
    - notified to appliances with energy grants
    - can be re-notified if grant is taken away
    - due to high priority demands / grid demands / cost

MEDICAL SERVICES

«Heart Rate» Service

- «Heart Rate» characteristic
  - what is the current heart rate
  - can be notified once a second
- «Heart Rate RR» characteristic
  - what is the current heart rate / RR value
  - can be notified once a second

MEDICAL SERVICES

«Weighing Scale» Service

- «Weight kg» characteristic
  - what was the last weight being measured
- «Weight lbs» characteristic
  - what was the last weight being measured

MEDICAL SERVICES

«MDS» Service

- characteristics for IEEE 11073-20601 compatibility
- can be referenced by other services

MEDICAL SCHEMA

Sports & Fitness Use Case

- can use any of the sport device services
- will read data, either when it wants
- or will setup notifications to get constant reports
- just one schema for all sports devices

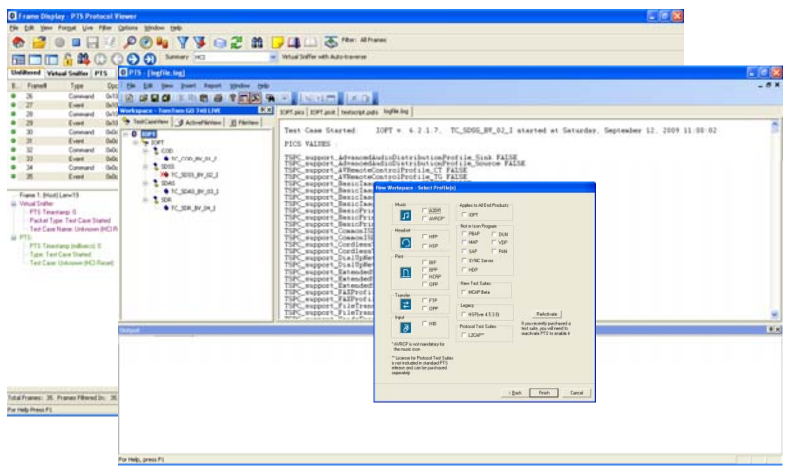
Continua Use Case

- uses «MDS» service to recreate 20601 state information
- optimized over the air – interoperable functionality after LE

PTS OVERVIEW

MISSION

Strengthen Bluetooth technology by creating a wireless testing tool with unmatched quality and test coverage. Lower cost of qualification thus making Bluetooth product development more accessible and efficient.



Bluetooth® SPECIAL INTEREST GROUP

RESPONSIBILITY

Support existing test suites

Add test support for new specifications

- Basic Rate
- Low Energy
- High Speed

Improve usefulness of PTS

- Improve usability
- Debugging functionality
- Test Reporting
- Automation

Test suite	Validated
A2DP	Yes
AVRCP	Yes
BPP	Yes
HCRP	Yes
DUN	Yes
HFP15	Yes
IOPT	Yes
PBAP	Yes
FTP	Yes
HID	Yes
BIP	Yes
OPP	Yes
SAP	Yes
PAN	Yes
HDP	Yes
HSP	Yes
MAP	Yes
SYNC	Yes
L2CAP	Yes
MCAP	No

VISION

Support Bluetooth testing for all profiles and protocols above HCI

Work toward a standardized PTS hardware that supports Bluetooth Basic Rate, Low Energy and High Speed

Release PTS test suites together with new specifications

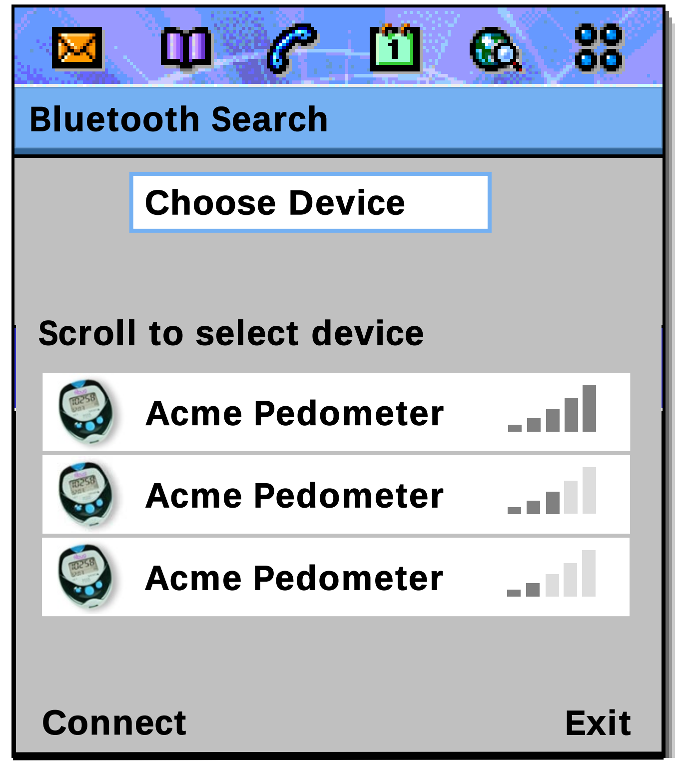
Provide tools required for members to automate their testing

NEW TOOLS MAKE PAIRING EASIER

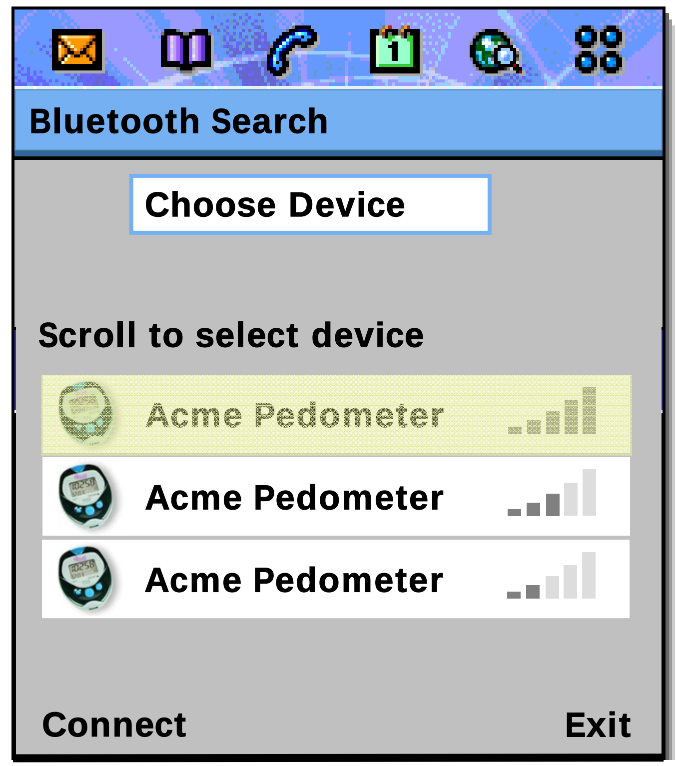
NEW TOOLS MAKE PAIRING EASIER

NEW TOOLS MAKE PAIRING EASIER

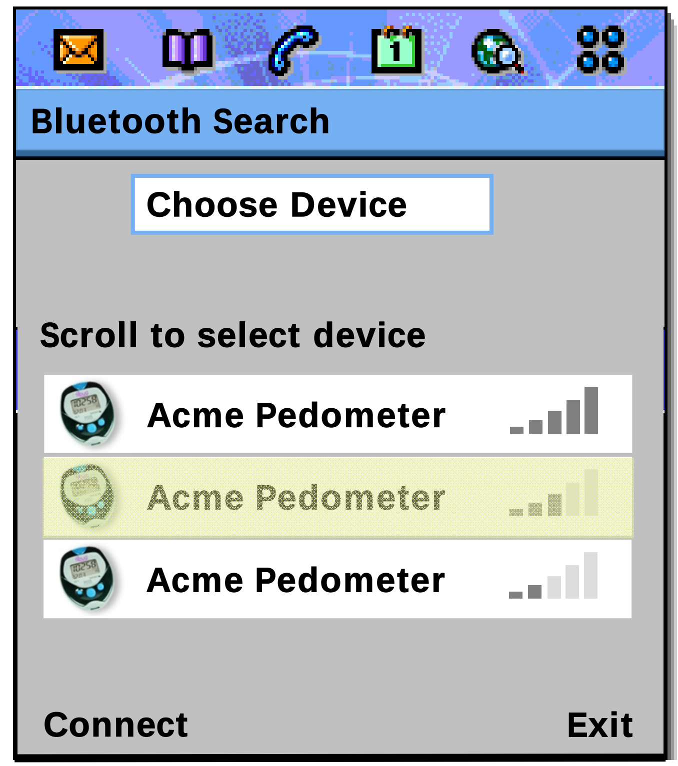
NEW TOOLS MAKE PAIRING EASIER



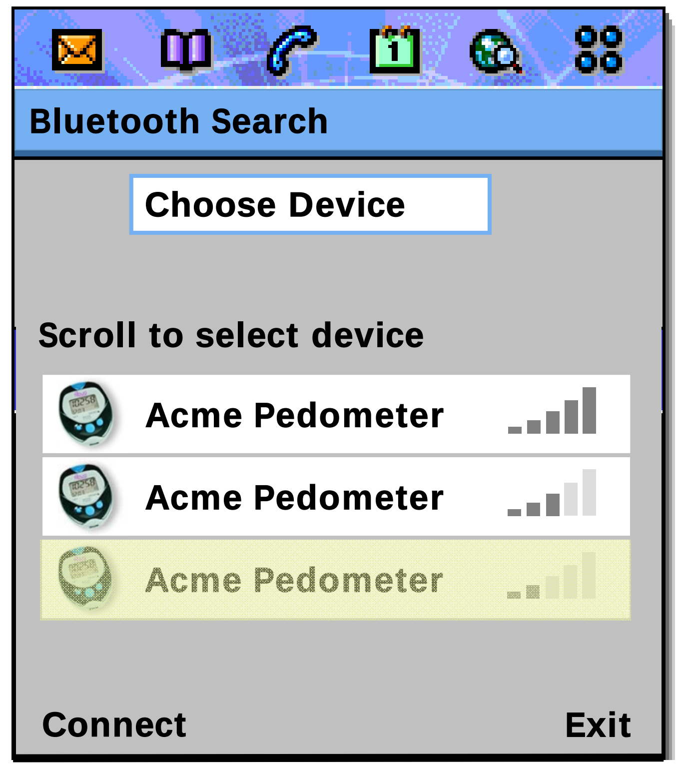
NEW TOOLS MAKE PAIRING EASIER



NEW TOOLS MAKE PAIRING EASIER



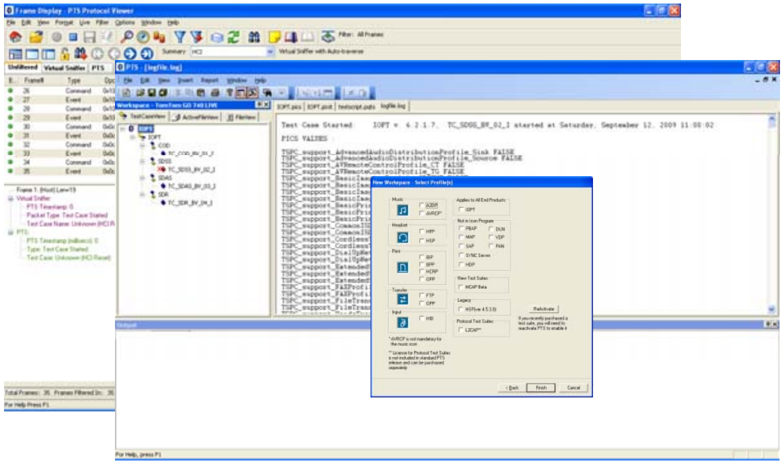
NEW TOOLS MAKE PAIRING EASIER



PTS OVERVIEW

MISSION

Strengthen Bluetooth technology by creating a wireless testing tool with unmatched quality and test coverage. Lower cost of qualification thus making Bluetooth product development more accessible and efficient.



RESPONSIBILITY

Support existing test suites

Add test support for new specifications

- Basic Rate
- Low Energy
- High Speed

Improve usefulness of PTS

- Improve usability
- Debugging functionality
- Test Reporting
- Automation

Test suite	Validated
A2DP	Yes
AVRCP	Yes
BPP	Yes
HCRP	Yes
DUN	Yes
HFP15	Yes
IOPT	Yes
PBAP	Yes
FTP	Yes
HID	Yes
BIP	Yes
OPP	Yes
SAP	Yes
PAN	Yes
HDP	Yes
HSP	Yes
MAP	Yes
SYNC	Yes
L2CAP	Yes
MCAP	No

VISION

Support Bluetooth testing for all profiles and protocols above HCI

Work toward a standardized PTS hardware that supports Bluetooth Basic Rate, Low Energy and High Speed

Release PTS test suites together with new specifications

Provide tools required for members to automate their testing

BLUETOOTH LOW ENERGY TESTING

Bluetooth SPECIAL INTEREST GROUP

GOALS

Dynamic product development

Interoperable products

Simple testing

Enable generic clients

Inexpensive solution

PRODUCTS

PRODUCTS

Implementations

HCI

LE chip
Below HCI
Few Implementations

TTCN test vectors

PRODUCTS

Implementations

PTS

LE stacks
L2CAP, SM
GAP, ATT , GATT
More implementations

HCI

LE chip
Below HCI
Few Implementations

TTCN test vectors

PRODUCTS

Implementations

Low Energy Tester

LE use cases

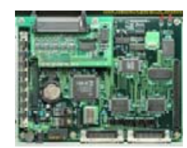
PTS

LE stacks
L2CAP, SM
GAP, ATT , GATT
More implementations

HCI

LE chip
Below HCI
Few Implementations

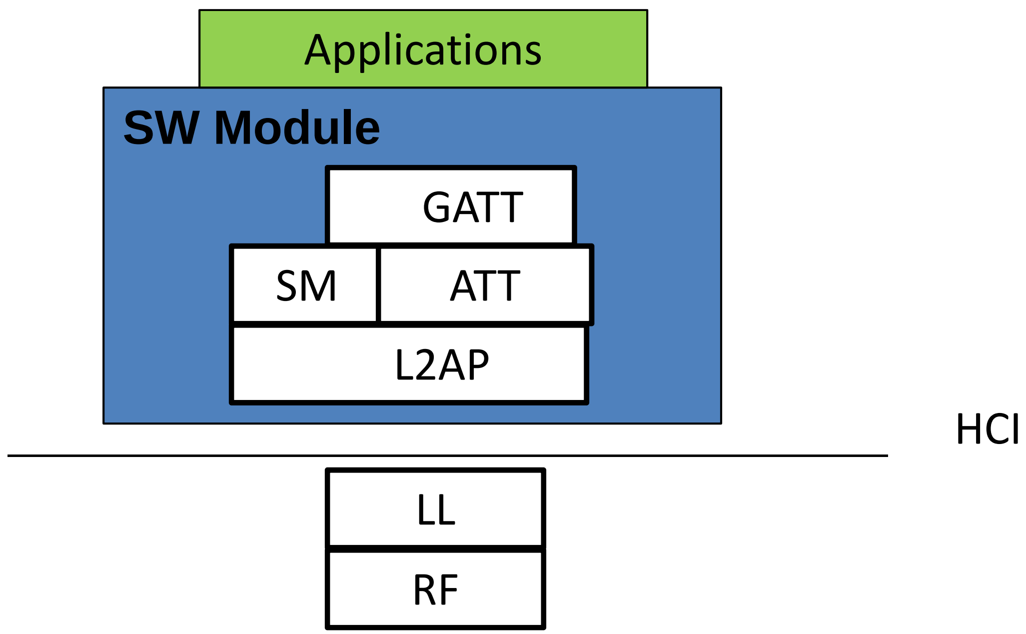
TTCN test vectors



LOW ENERGY DEVICES

GATT and below will be qualified as before

GATT based specifications will have a new model



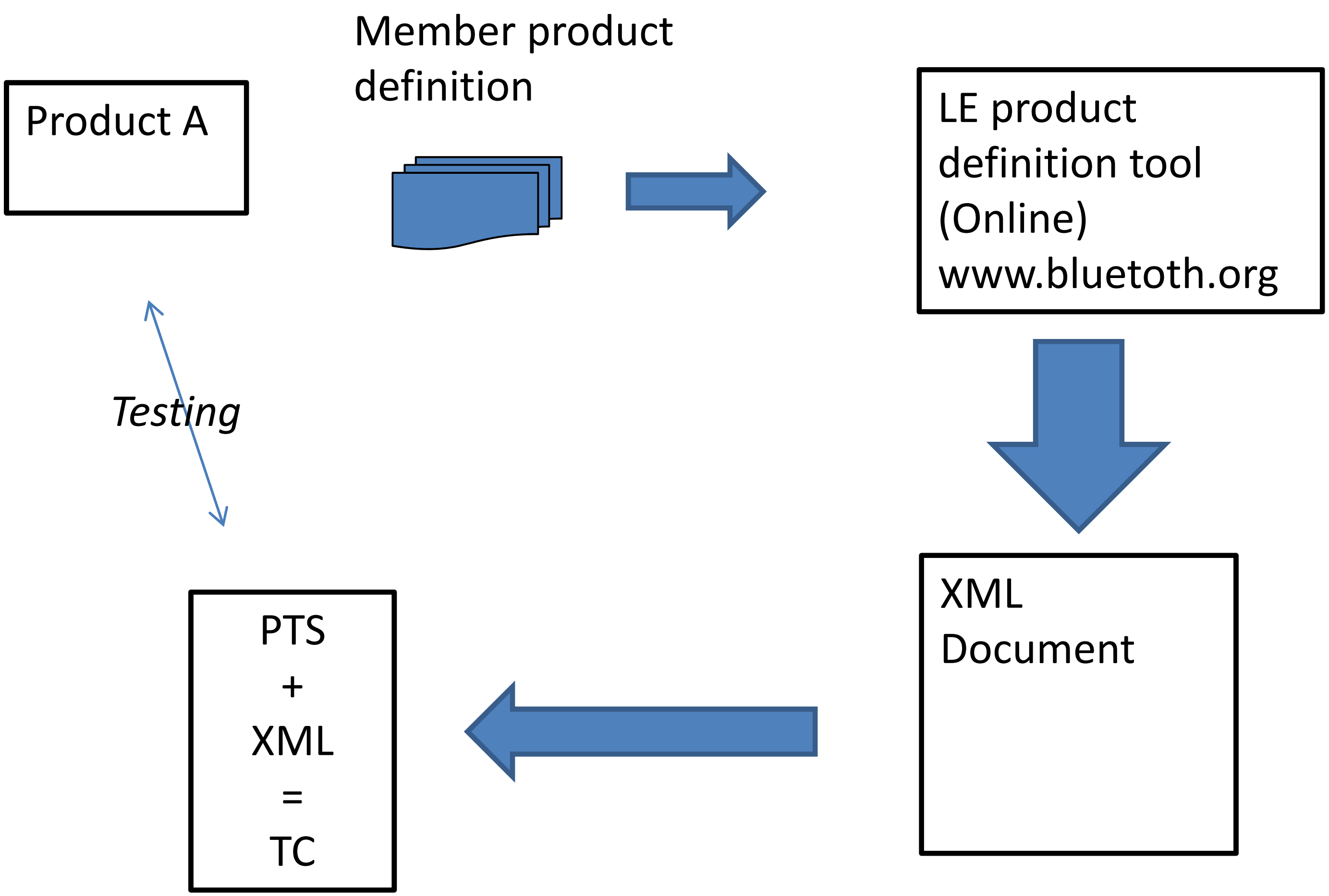
PTS TESTING

SM, GAP and GATT will be tested by PTS

GATT testing will include several configurations

A virtual device library can be used

XML CREATION TOOL AND PTS



ENABLING TECHNOLOGY

XML

- XSD (Schema Definition) created for GATT

Profile authoring tool

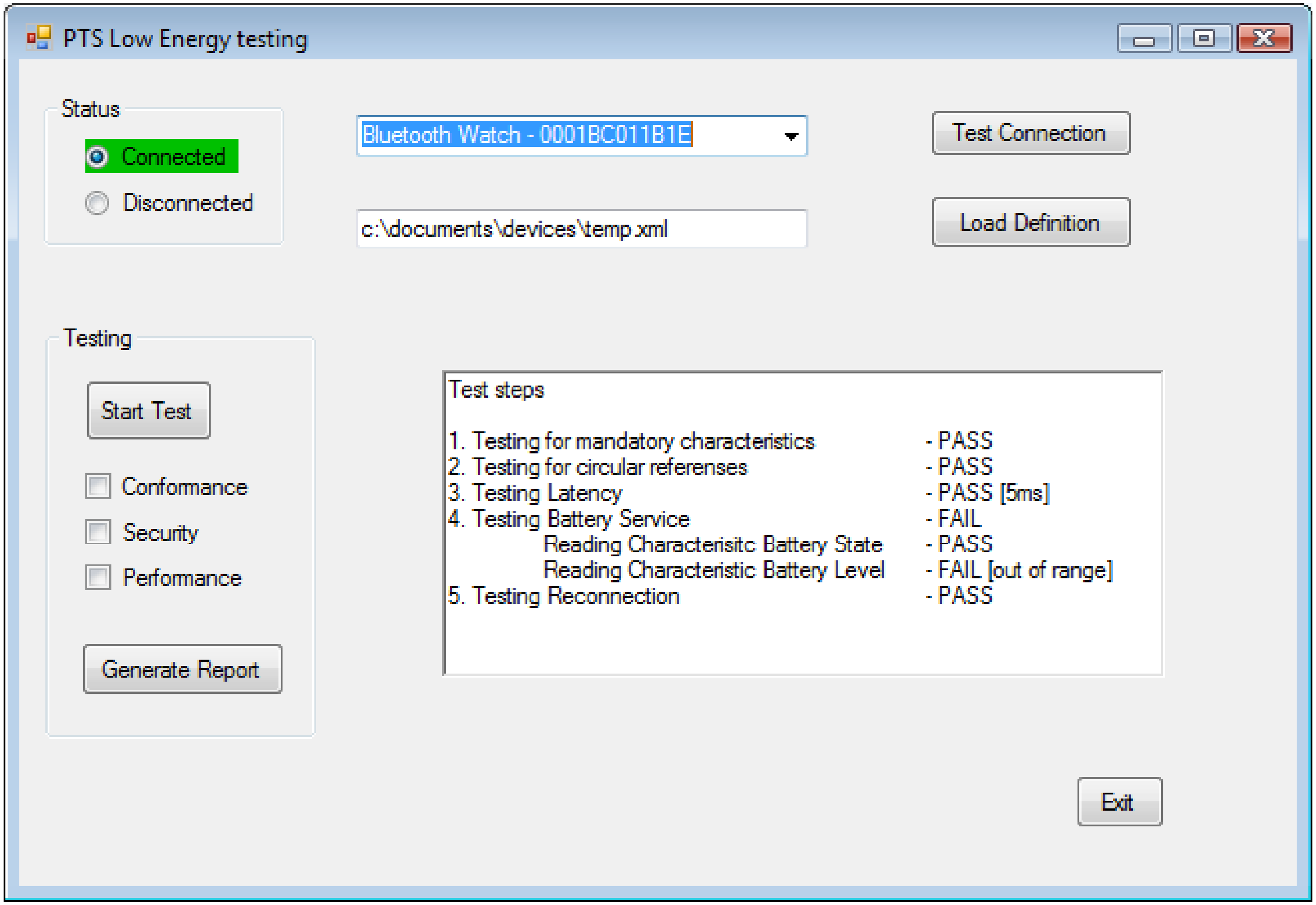
- Enable existing characteristics and services to be reused and configured
- Allow new characteristics and services to be defined and configured
- Export XML describing profile

Device definition authoring tool

- Enable profiles to be selected and configured for the device
- Exports xml schema for the device
- Exports sample device data structure for the device

Live attribute browser in PTS

LOW ENERGY PROFILE TESTING TOOLS



RECENT UPDATES LOW ENERGY

Low Energy Beta test suites

- L2CAP
- SM (not released)
- GAP (not released)
- GATT Testing approach

ACTIVITIES LOW ENERGY

L2CAP completed but need updates for spec changes

GAP test suite - May

SM test suite - May

GATT Testing (first version) - July

H2 – LE higher layer specs

PROBLEMS AND RISKS

Test Specifications is changing

Features not supported in stack and hardware

No products available for testing

PTS Low Energy hardware availability

PTS HARDWARE

Temporary USB based solution

- Ready this summer
- Limited samples available

A permanent USB dongle will come later

SUMMARY

THANK YOU!

Join Us at 17:30 at Fox Sports Grill for the Welcome Reception, co-sponsored by Frontline Test Equipment, Inc.