An Introduction to Bluetooth low energy

Robin Heydon, Senior Director, Technology
Qualcomm Technologies International, Ltd.

Agenda

What is Bluetooth low energy
What are the important features?

How does it work?
What are the next steps?

What is it good for today?
What else can you do with it?

What is low energy?

It is a new technology

blank sheet of paper design

optimized for ultra low power

different to *Bluetooth* classic

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

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

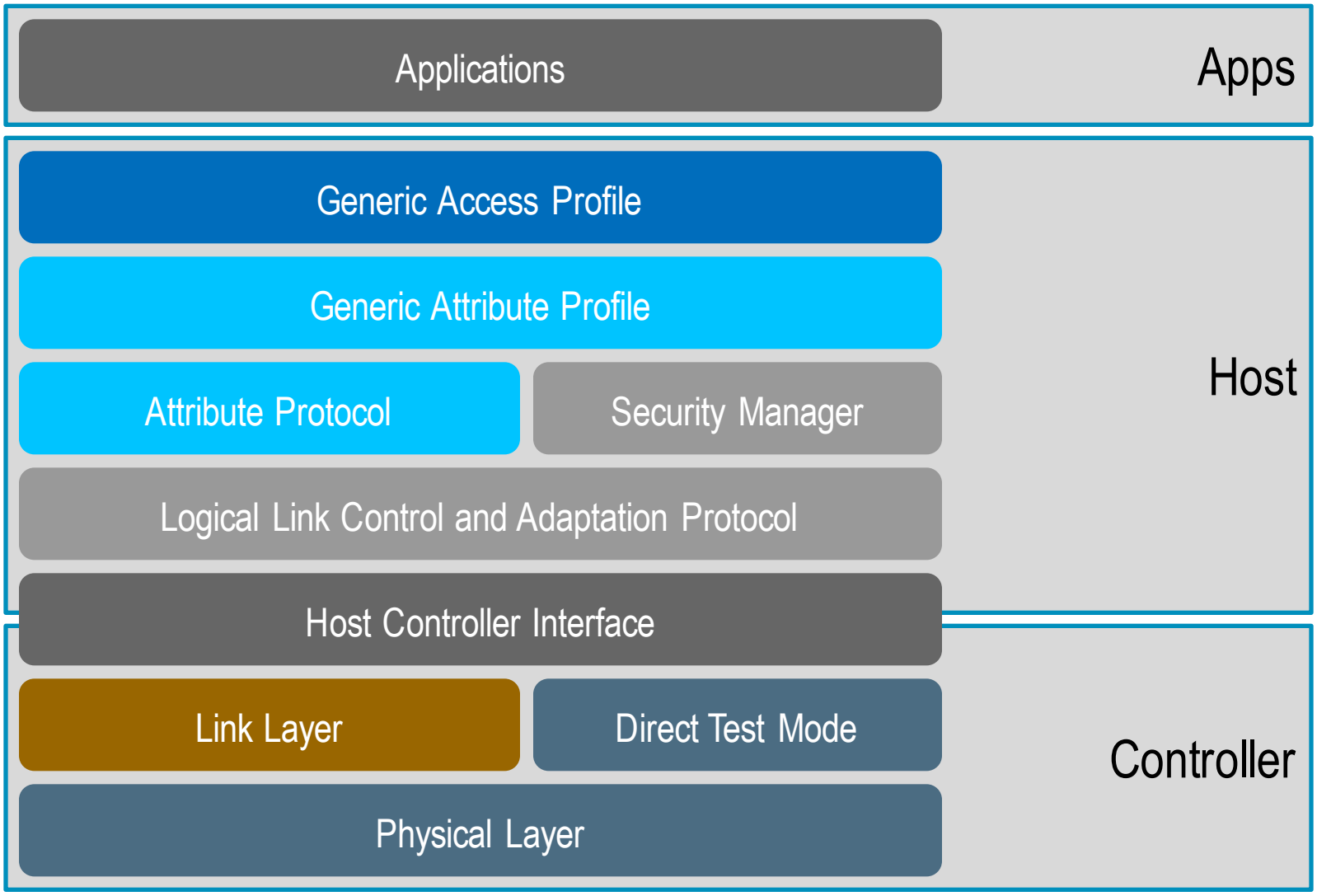
Clients can use the state exposed on servers

- read it – get current temperature
- write it – increase set point temperature for room

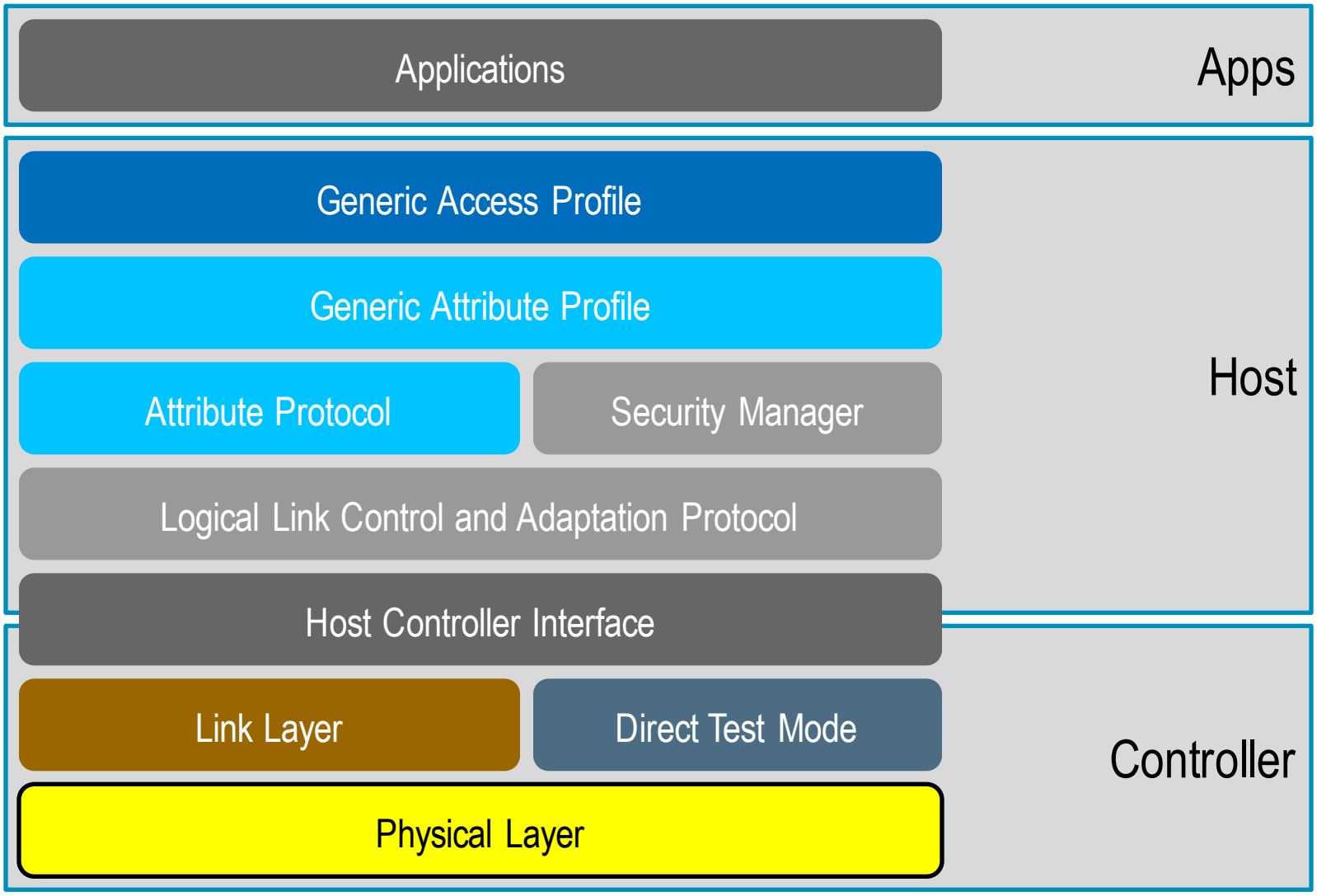
Servers can tell clients when state updates

- notify it – temperature up to set point

Stack Architecture



Physical Layer



Physical Layer

Uses 2.4 GHz ISM Band

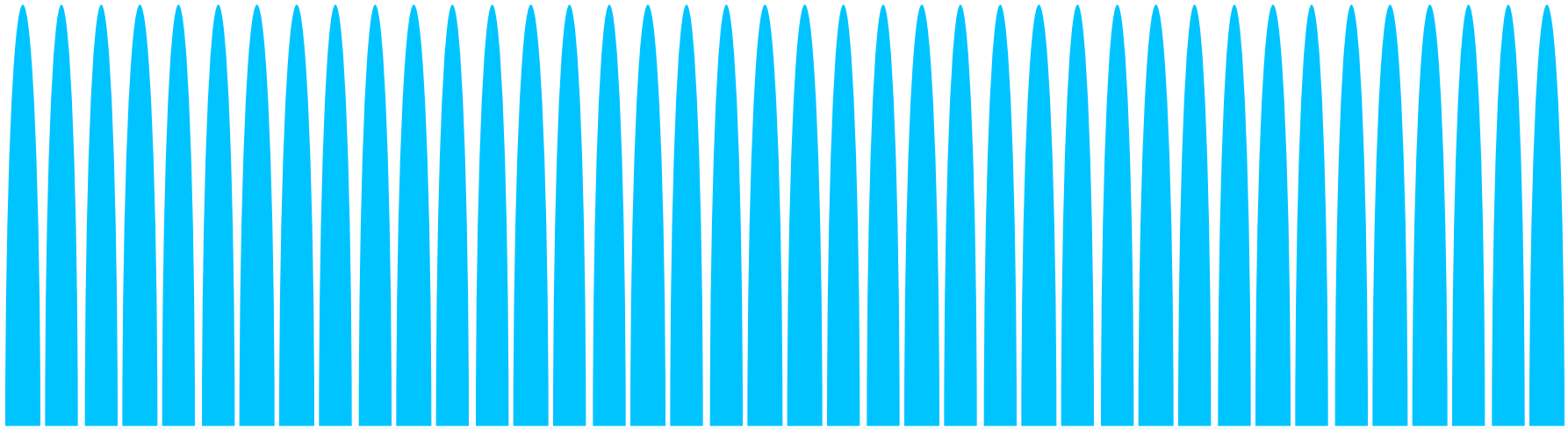
- Industrial Scientific Medical band
- License Free – with certain rules
- 2400 MHz to 2483.5 MHz

Used by many other standards

- IEEE 802.11, IEEE 802.15
- and many proprietary radios

40 Physical Channels

$$f = 2402 + 2 \cdot k \text{ MHz}$$



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

Modulation

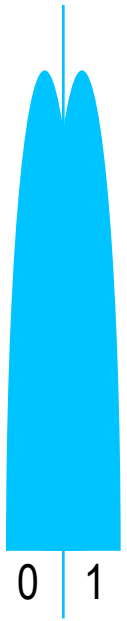
GFSK Modulation

- 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



PHY Summary

2.4 GHz GFSK

- Modulation Index = ~0.5

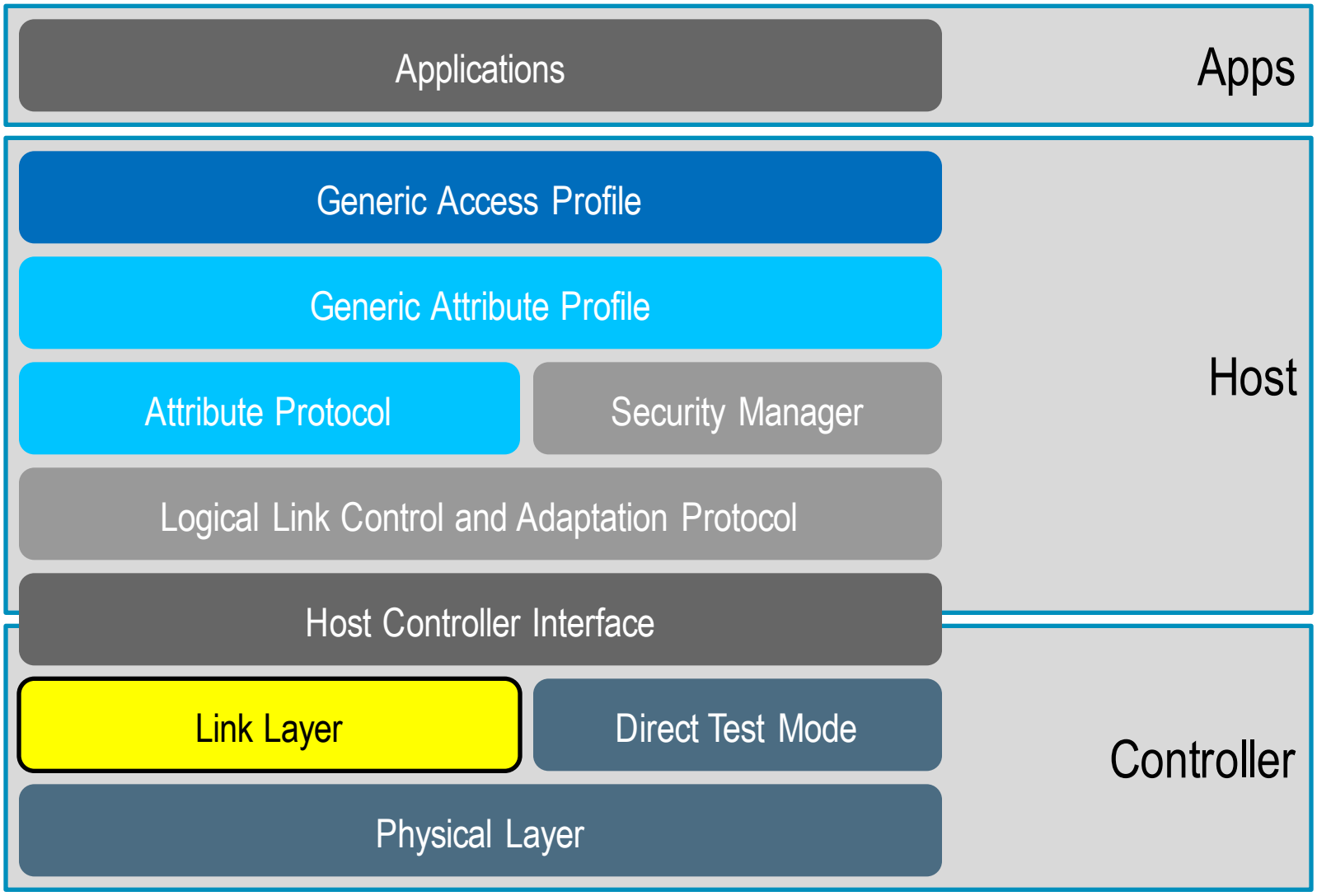
40 channels

- 2 MHz channel spacing
- 2402 MHz to 2480 MHz

Range

- 50m to further than 150 m

Layer Layer


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

Link Layer (LL)

Link Layer State Machine

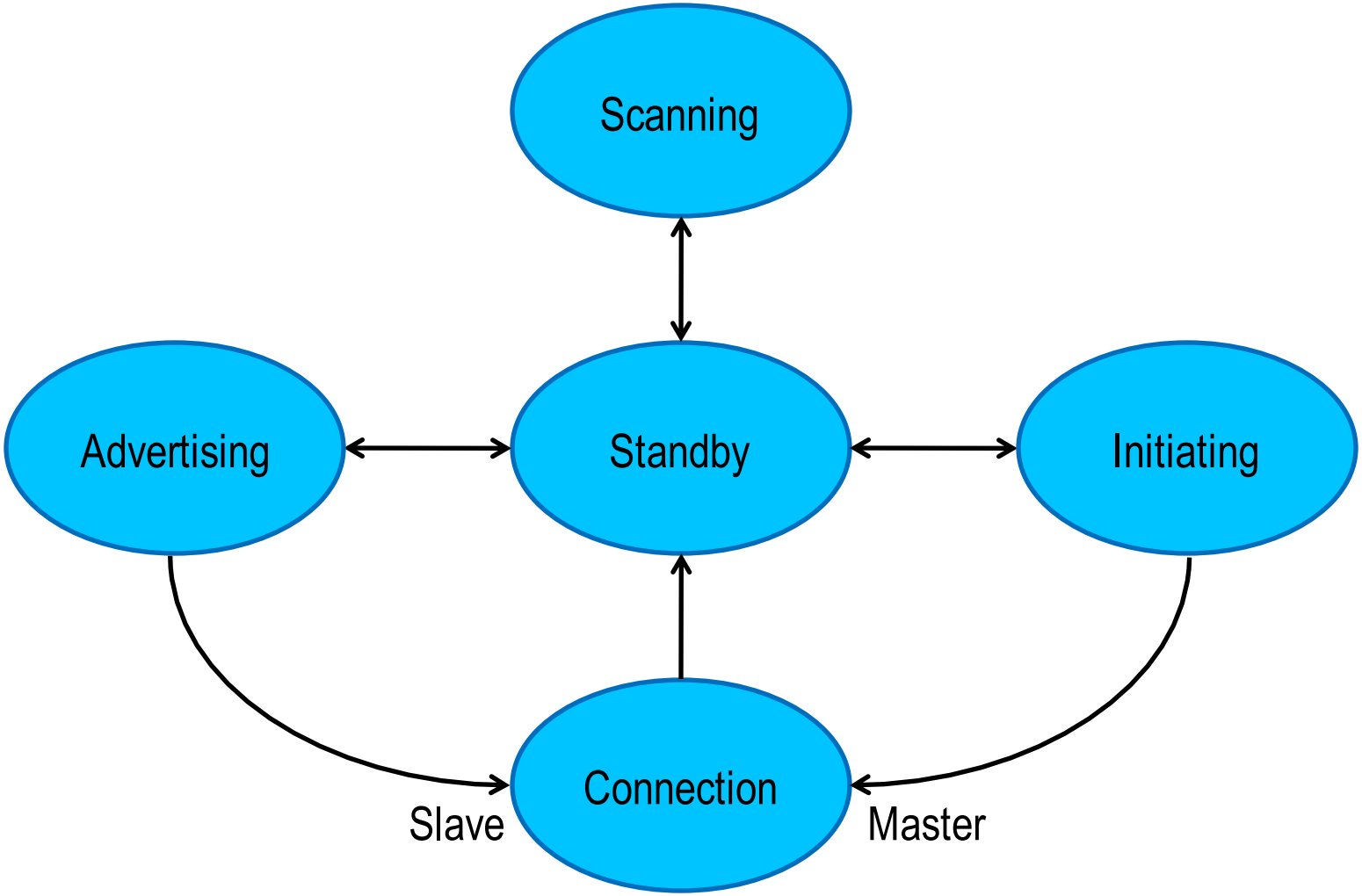
- can have multiple state machines active in device

Link Layer Channels

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

Link Layer Control Procedures

Link Layer State Machine

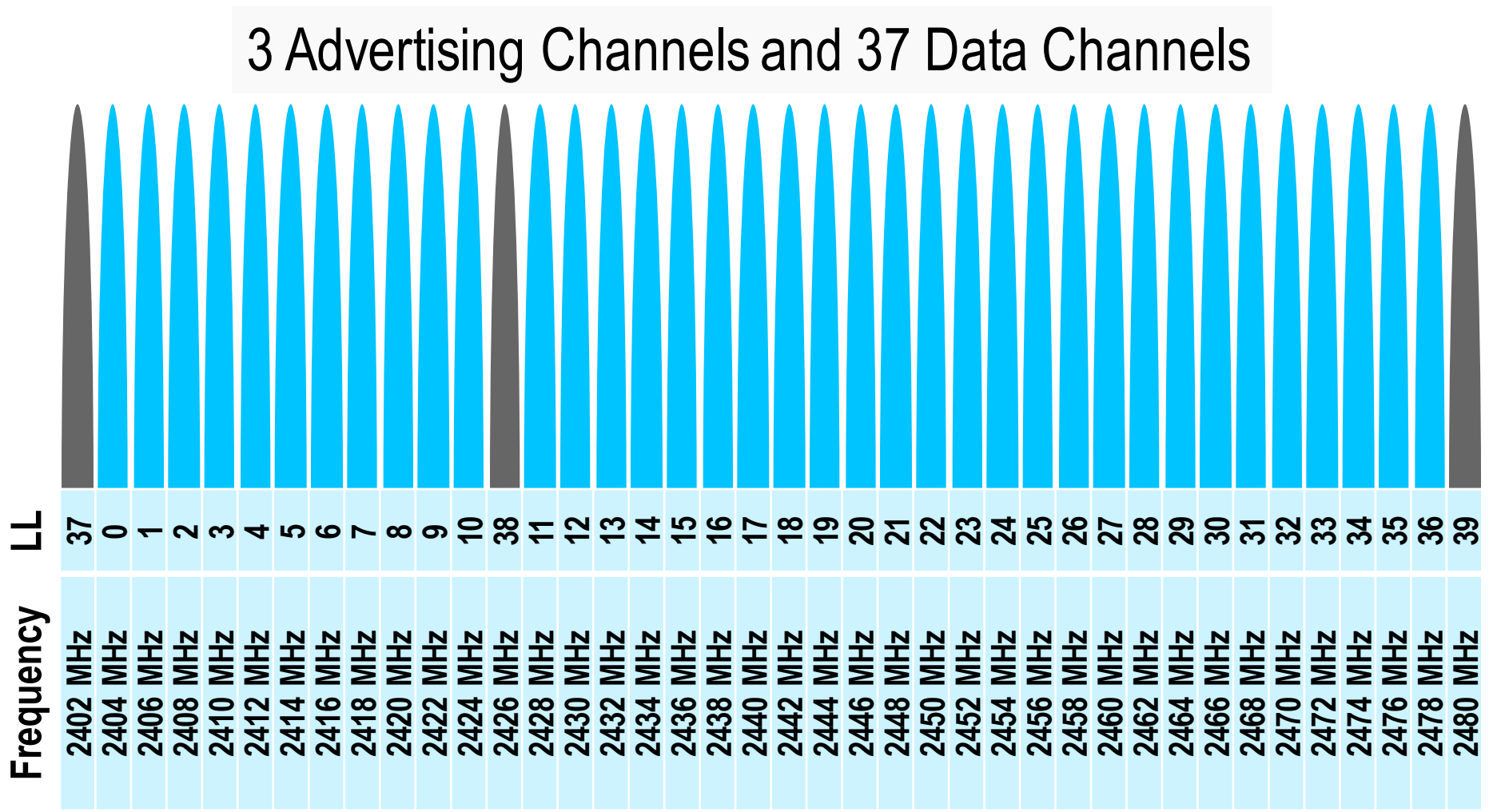


Two Types of Channels

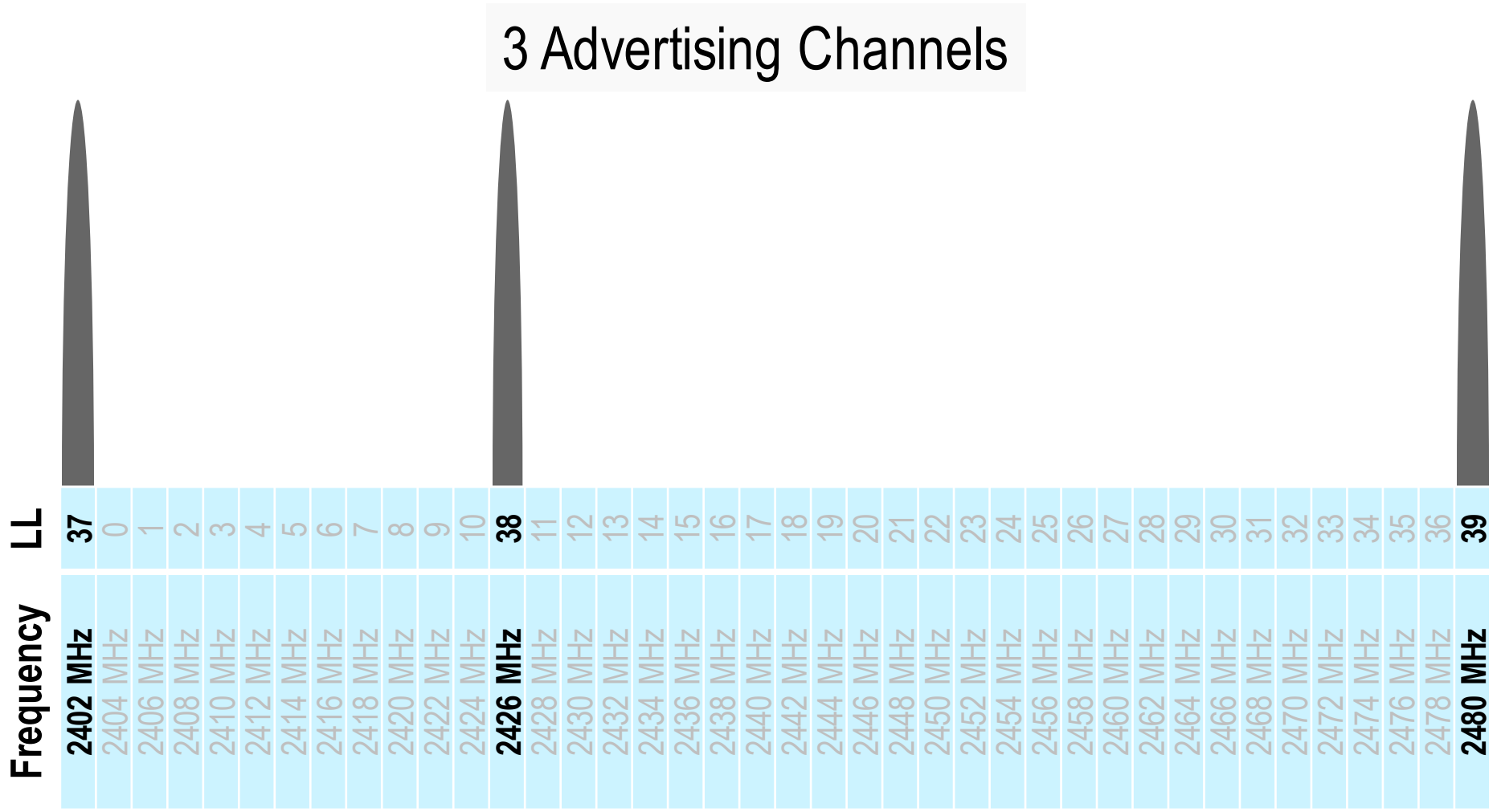
Advertising Channels
- Advertising Channel Packets
- Used for Discoverability / Connectability
- Used for Broadcasting / Observing

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

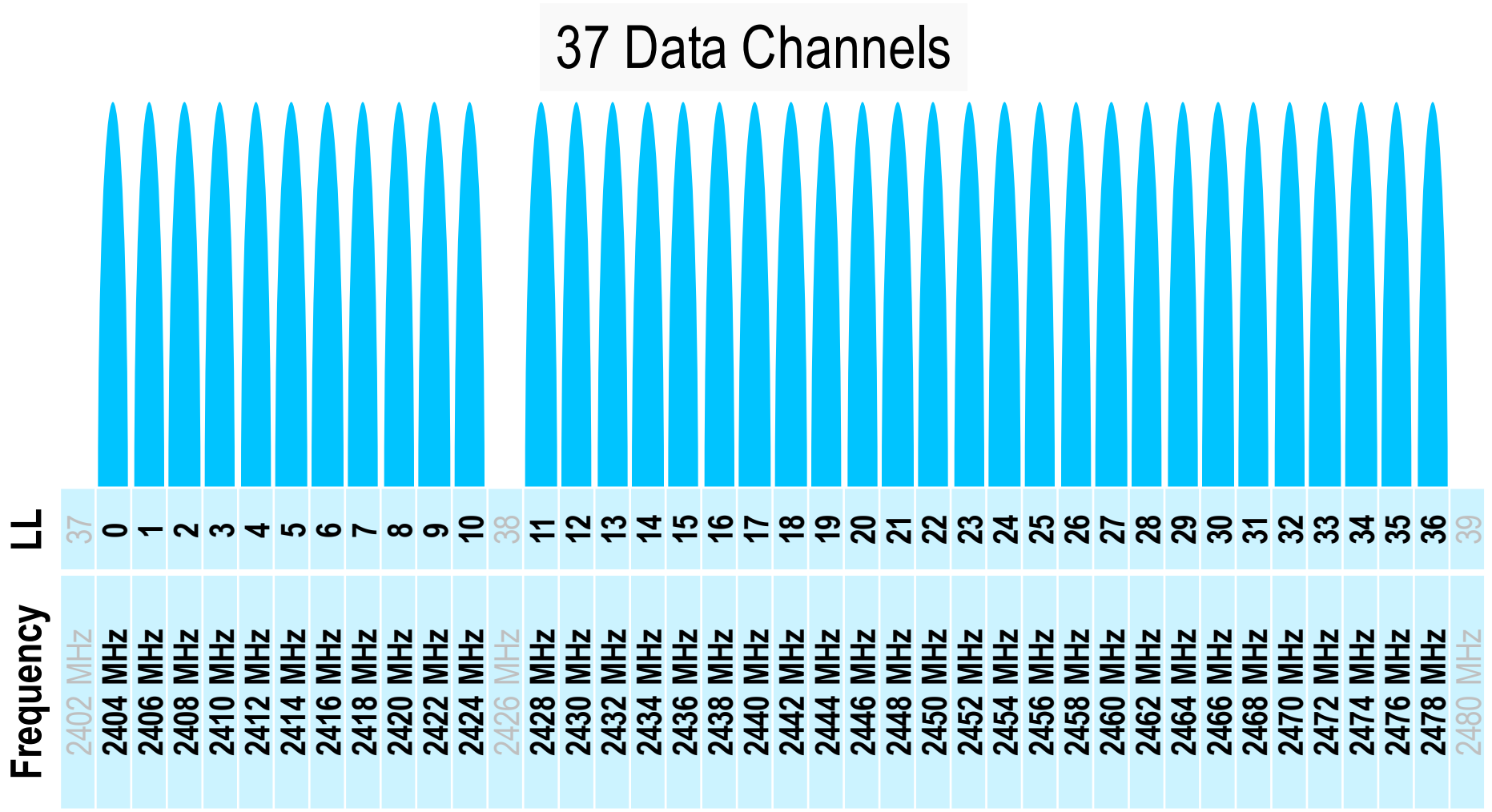
Link Layer Channels



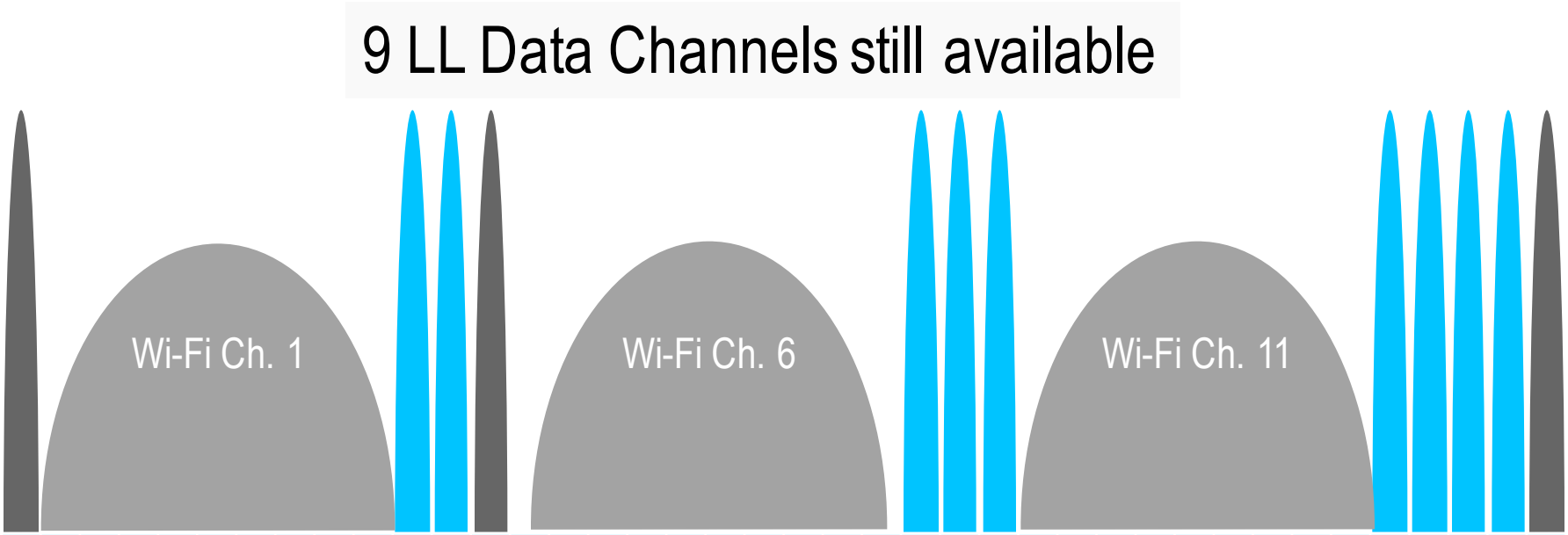
Link Layer Channels



Link Layer Channels

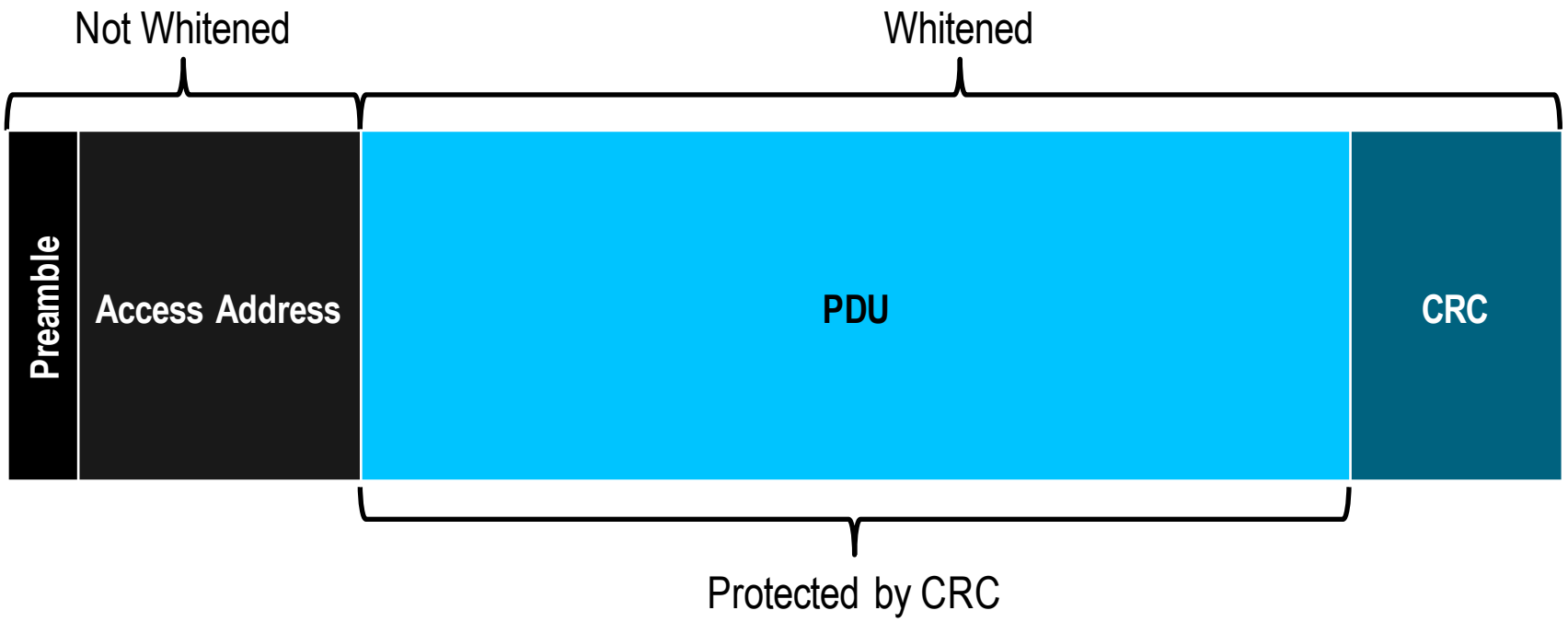


Link Layer Channels

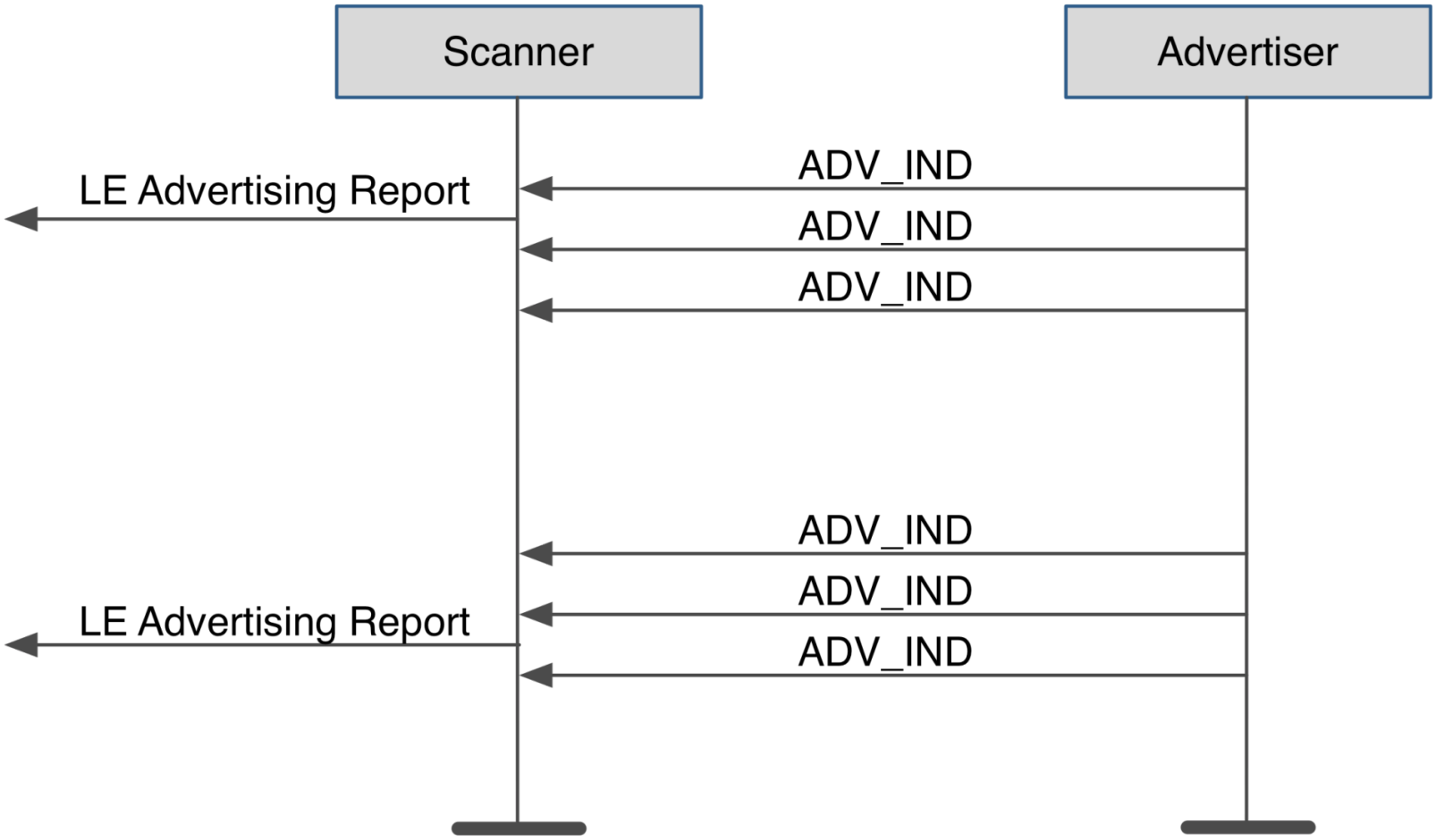


LL	Frequency
37	2402 MHz
0	2404 MHz
1	2406 MHz
2	2408 MHz
3	2410 MHz
4	2412 MHz
5	2414 MHz
6	2416 MHz
7	2418 MHz
8	2420 MHz
9	2422 MHz
10	2424 MHz
38	2426 MHz
11	2428 MHz
12	2430 MHz
13	2432 MHz
14	2434 MHz
15	2436 MHz
16	2438 MHz
17	2440 MHz
18	2442 MHz
19	2444 MHz
20	2446 MHz
21	2448 MHz
22	2450 MHz
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

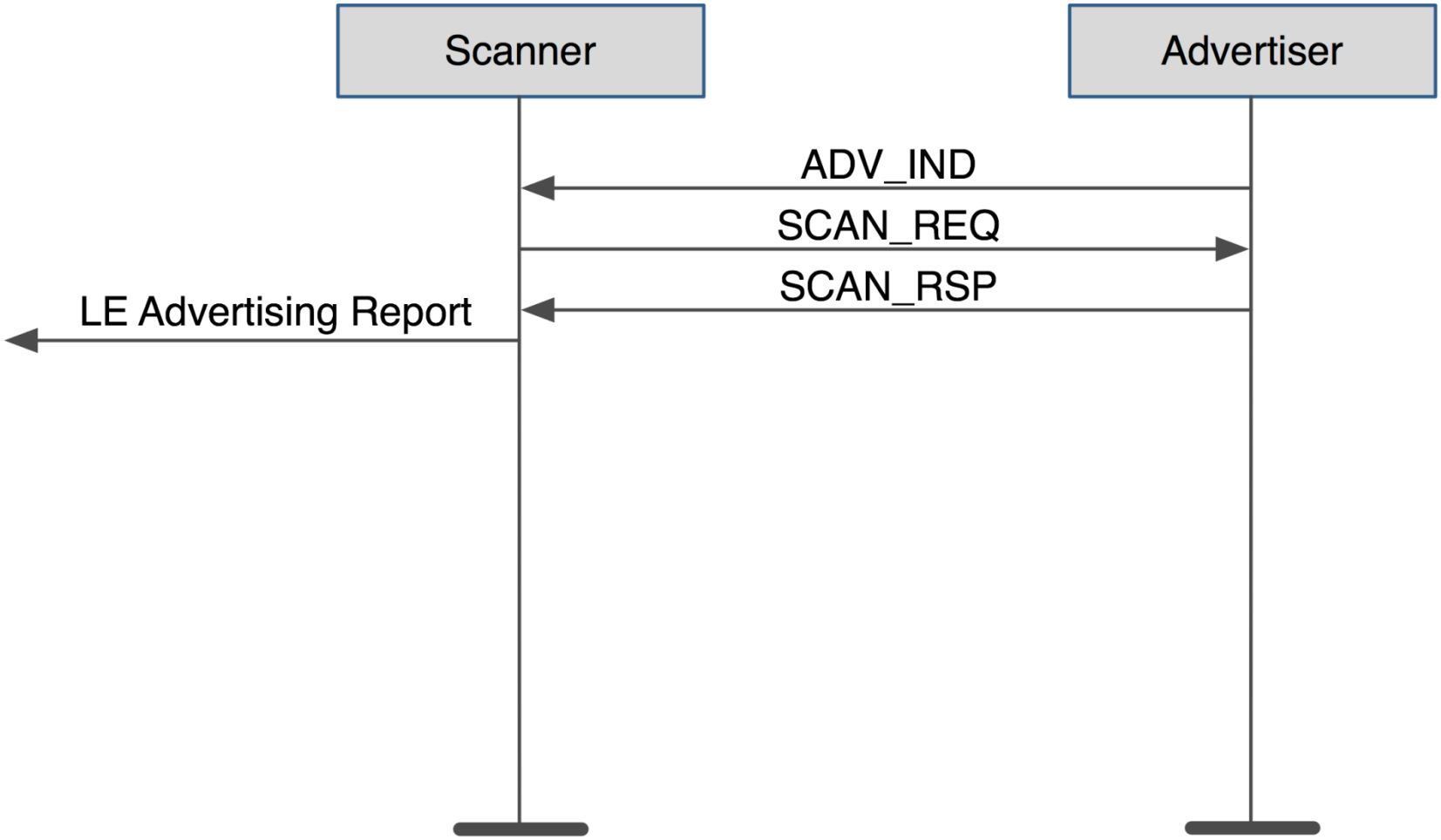
One Packet Format



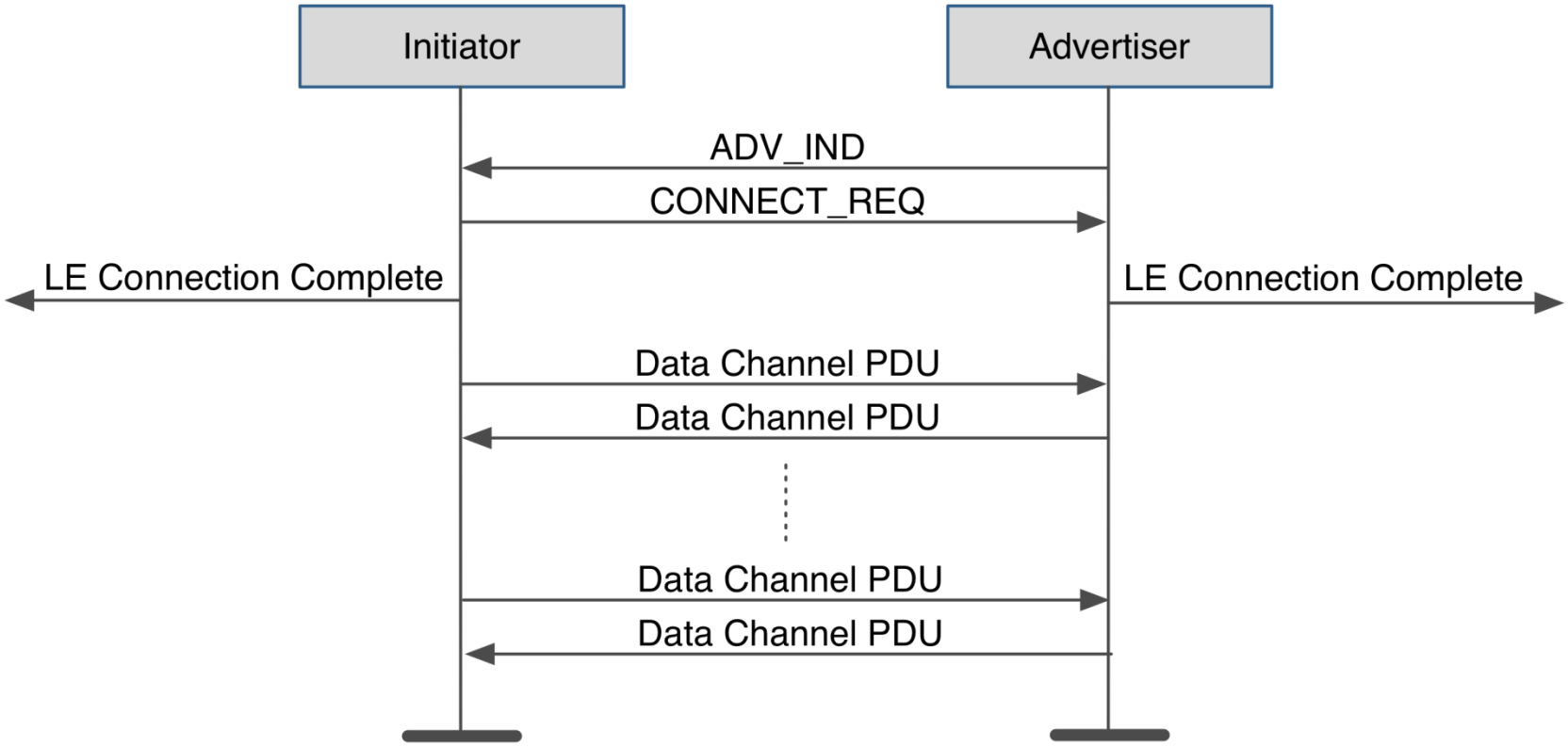
Passive Scanning



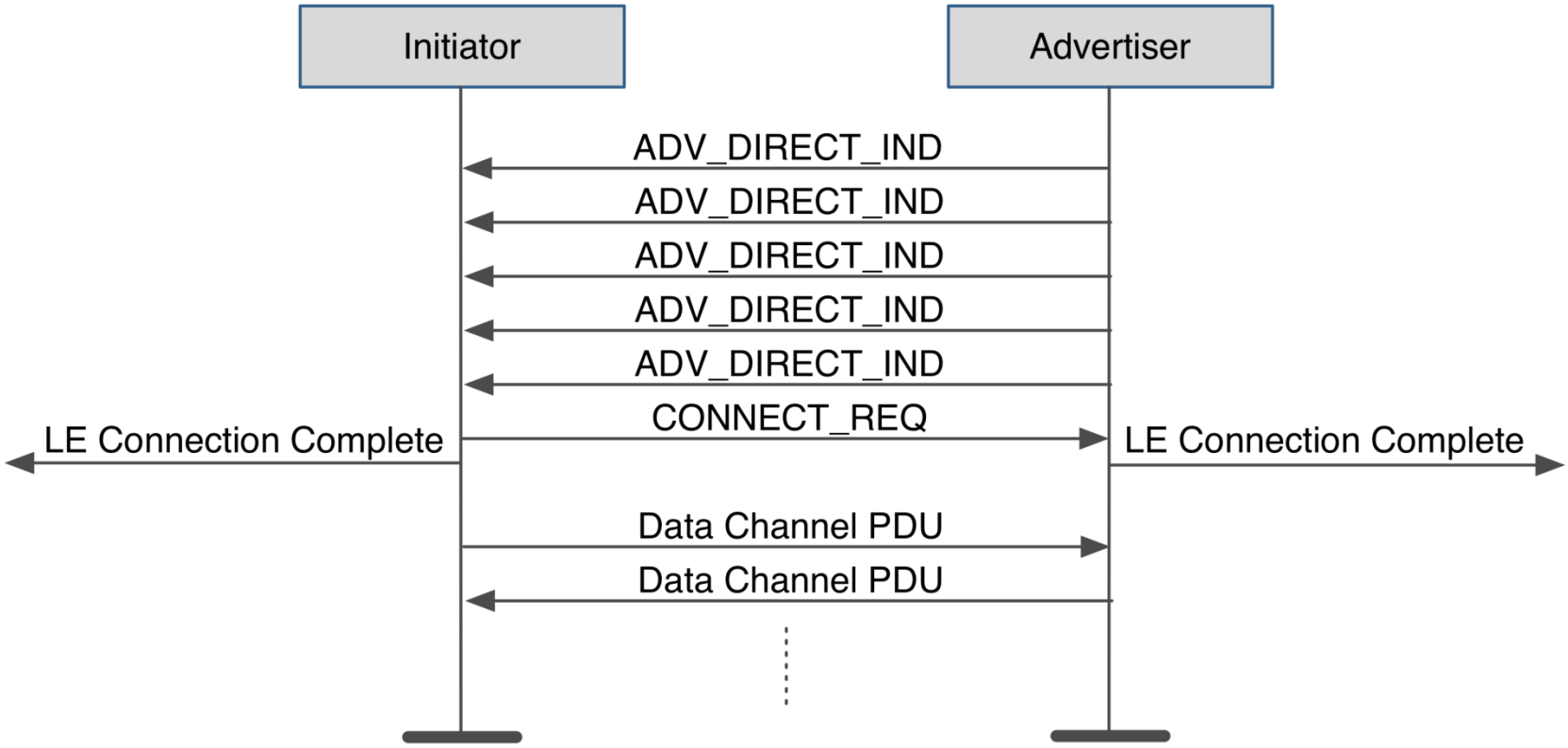
Active Scanning



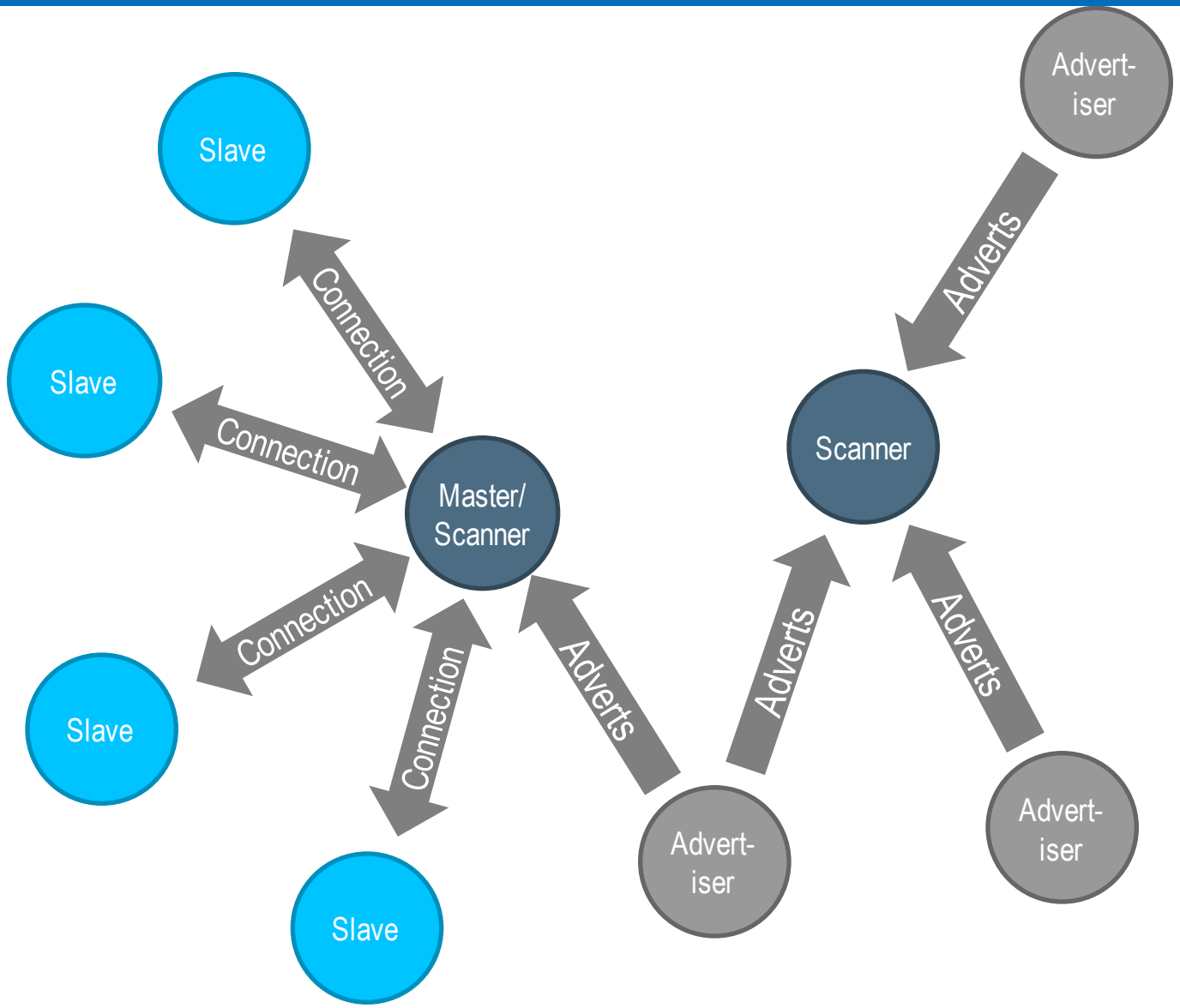
Initiating Connections



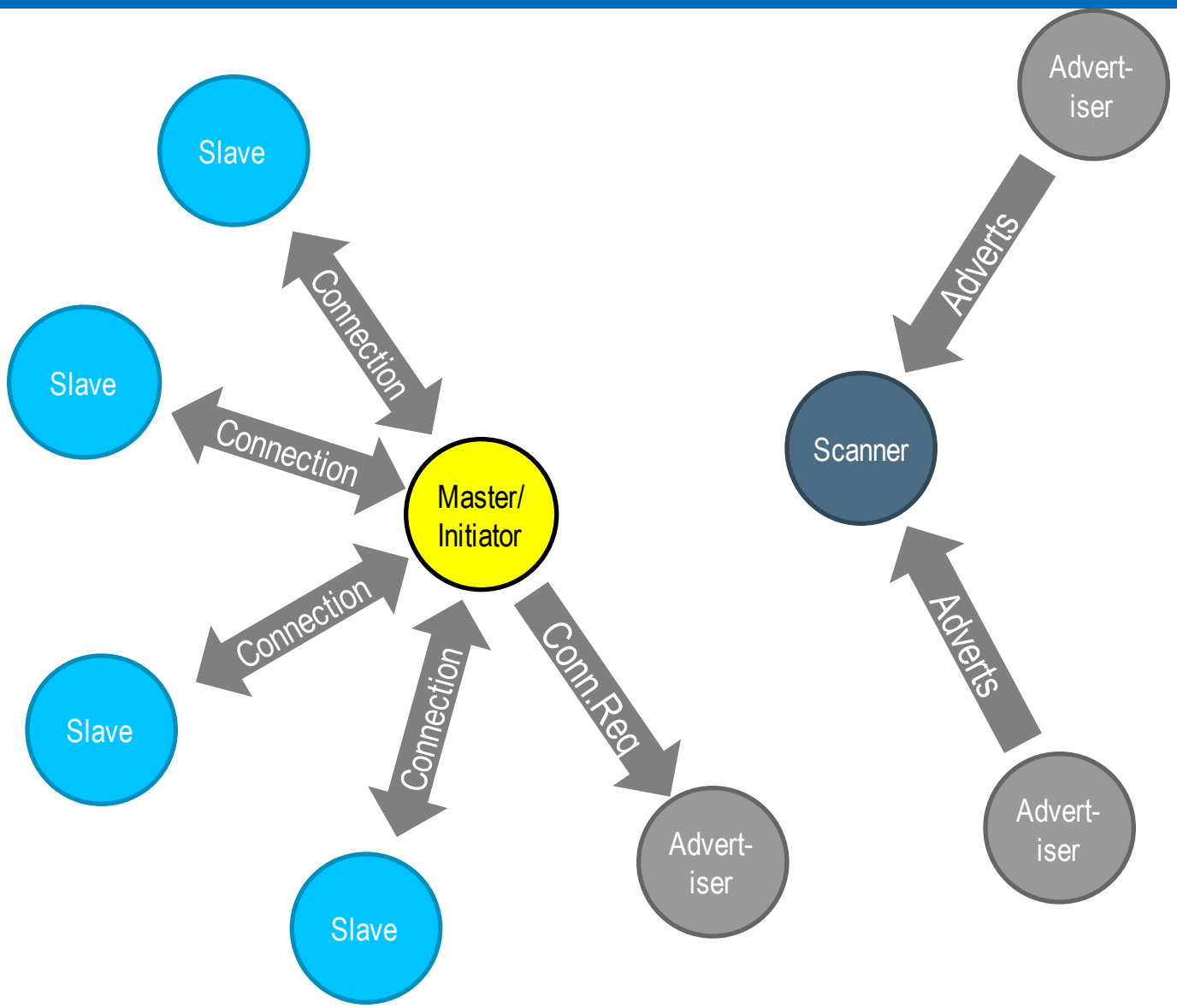
Directed Connections



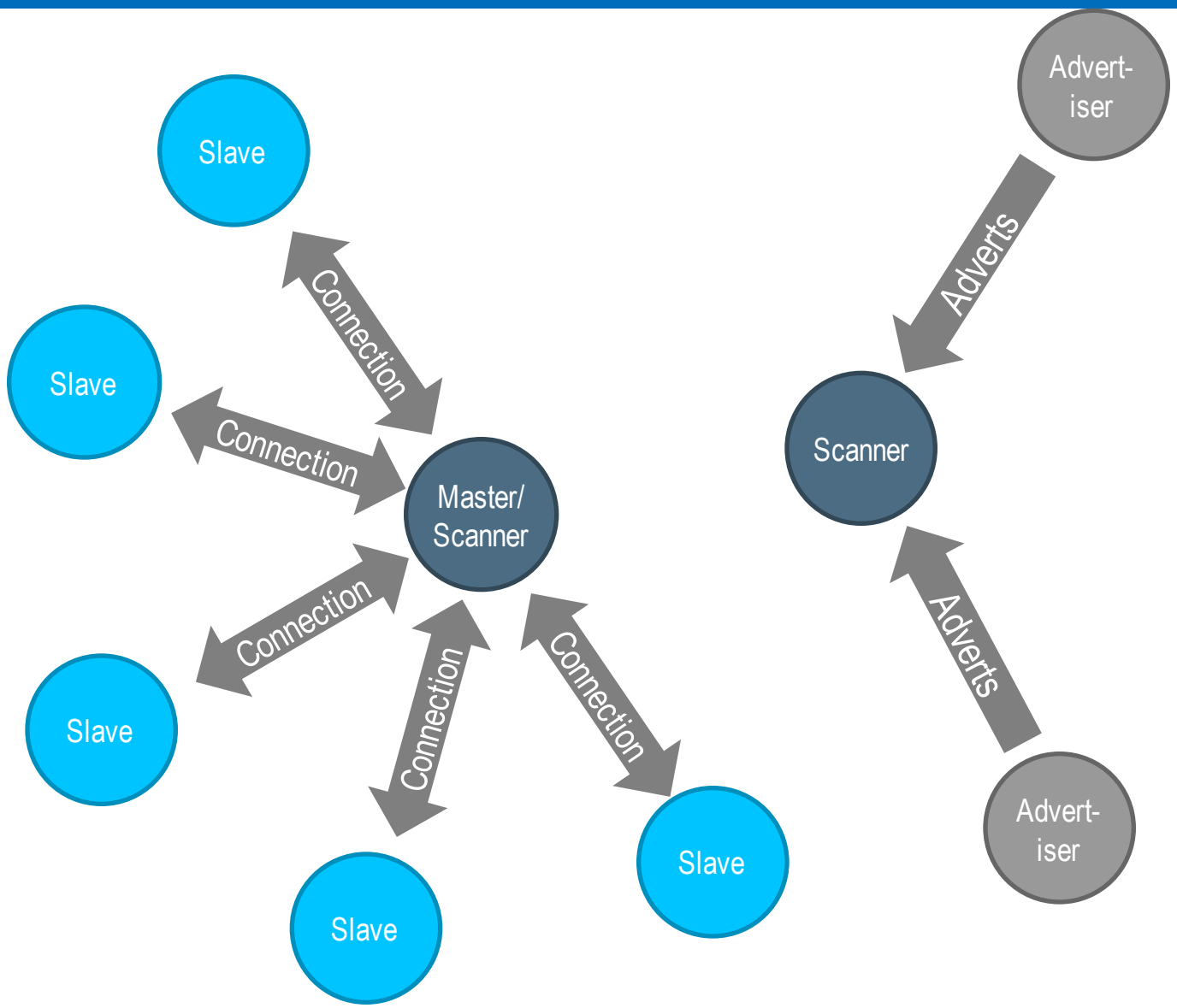
Topology


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

Topology


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

Topology


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

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

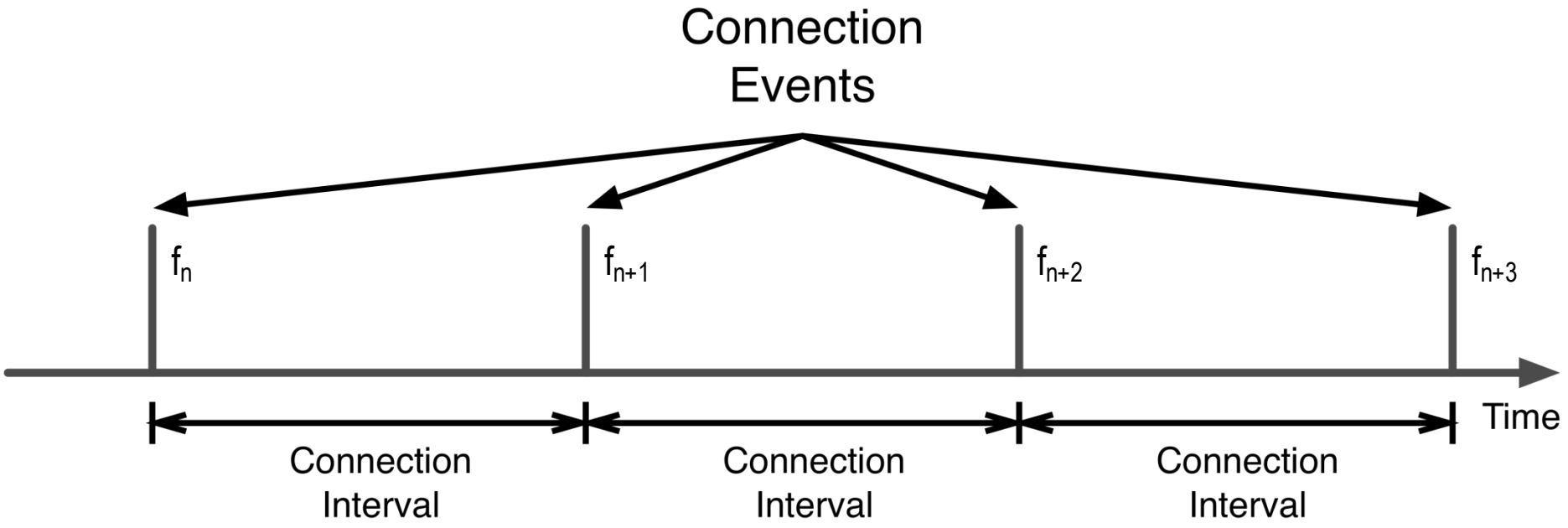
Used to send application data
- reliably, robustly

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

Connection Events

Each connection event uses a different channel

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



Latency

Master Latency

how often the master will transmit to slave

Slave Latency

how often the slave will listen to master

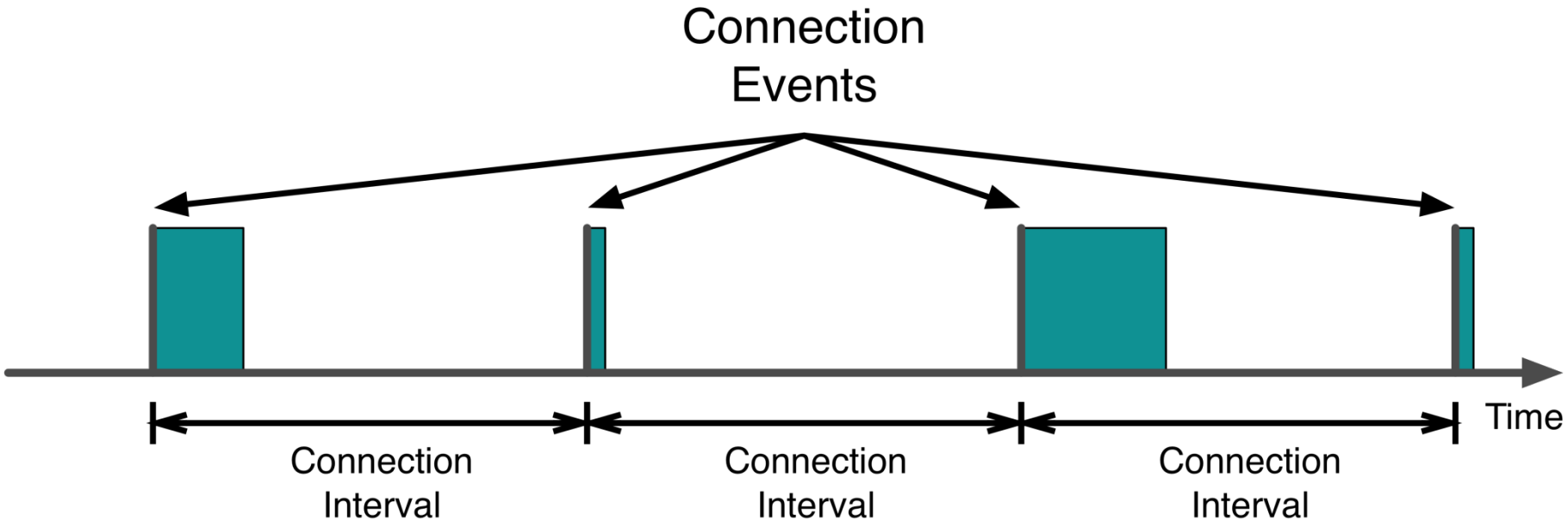
The two latencies don't have to be the same

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

Slave Latency = Connection Interval * Slave Latency

Connection Events

More Data bit automatically extends connection events



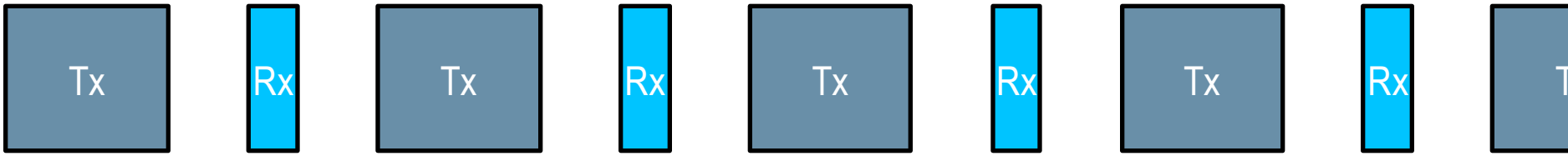
Maximum Data Rate

Asymmetric Tx/Rx Packet Sequence

328 + 150 + 80 + 150 = 708 µs

Transmitting 27 octets of application data

~305 kbps



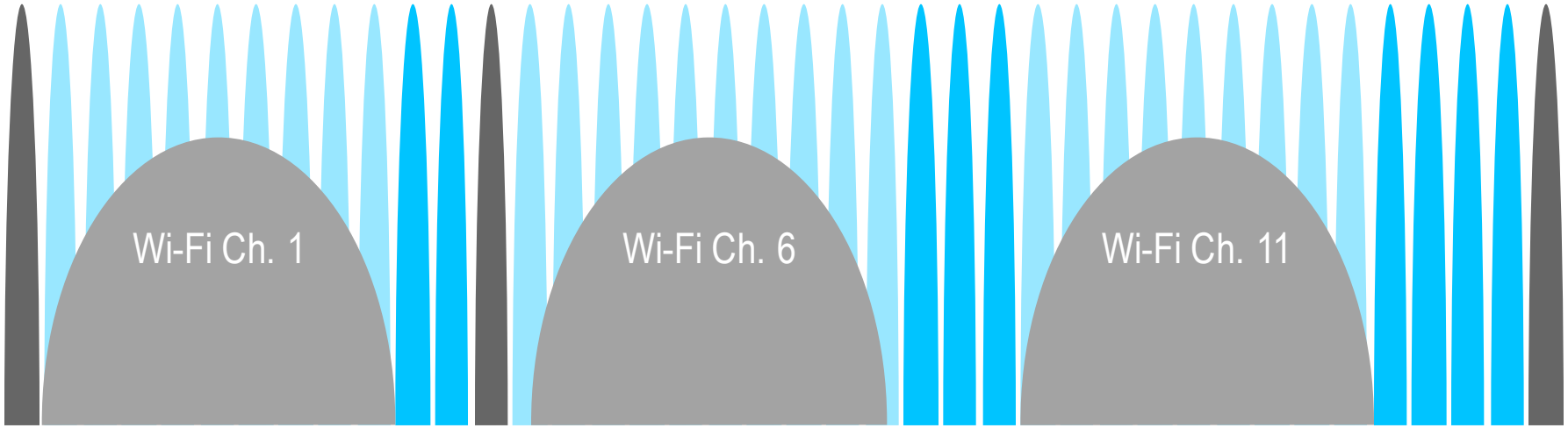
Adaptive Frequency Hopping

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



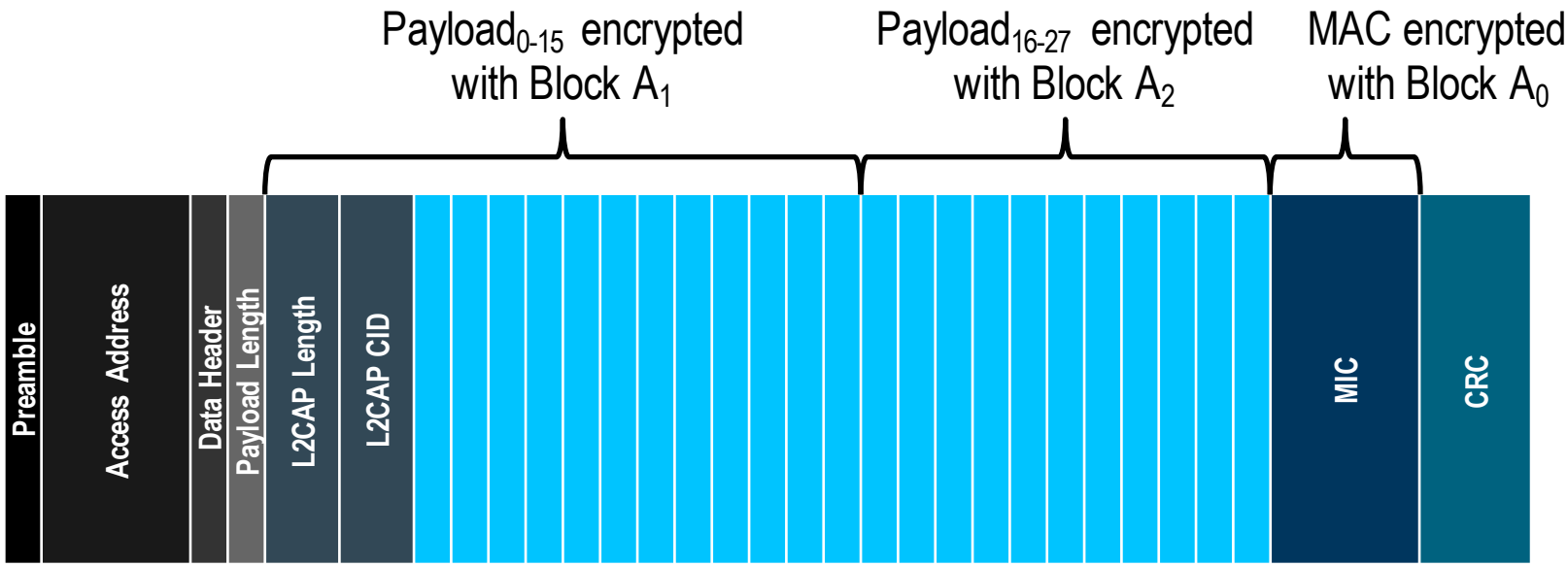
Link Layer Control Procedures

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

Link Layer Encryption

Uses AES 128 encryption block
and CCM as defined by RFC 3610



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

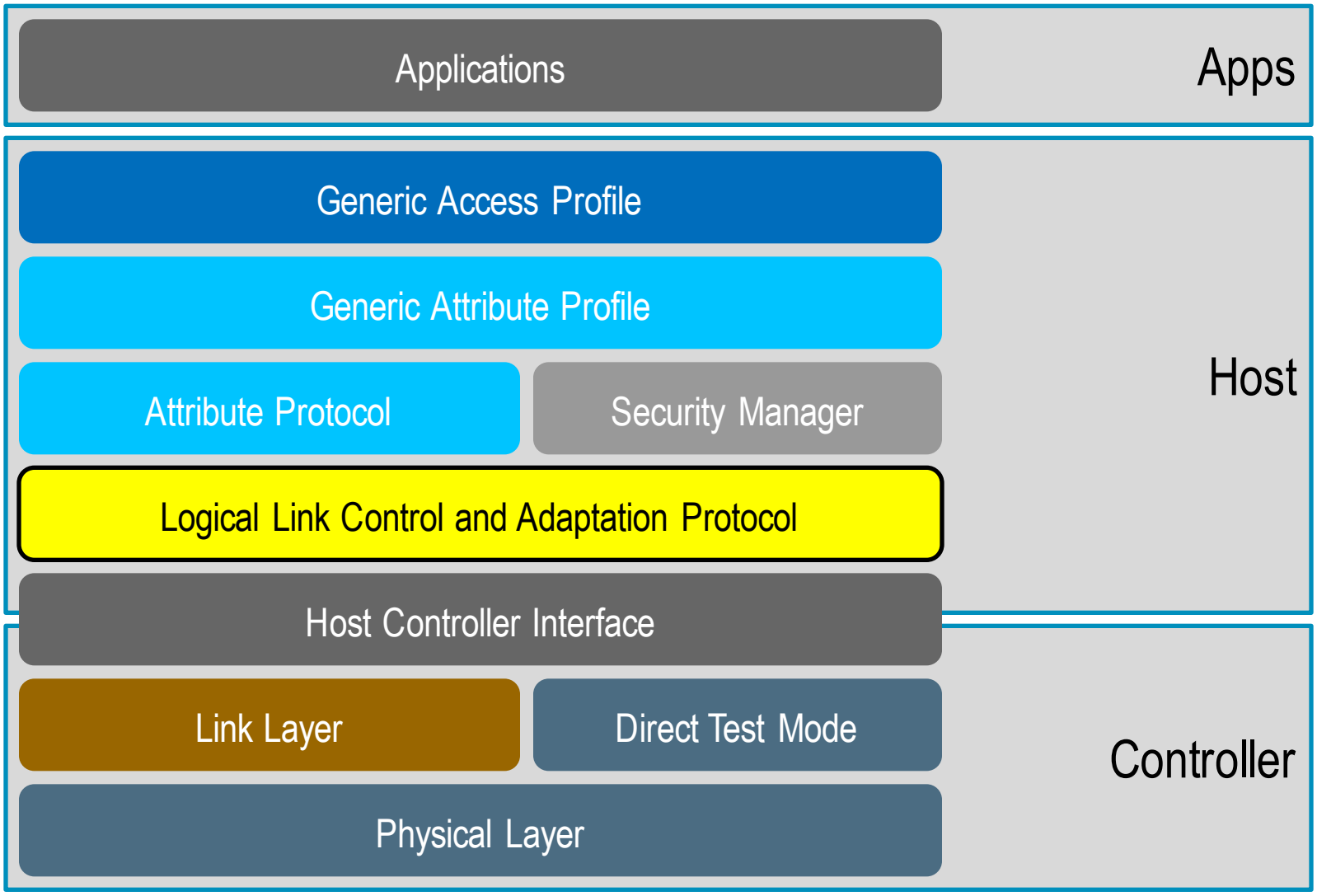
Low Complexity

- 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

L2CAP


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

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

Fixed L2CAP Channels

CIDs from 0x0001 to 0x003F are fixed channels
0x0040 to 0xFFFF are dynamically allocated

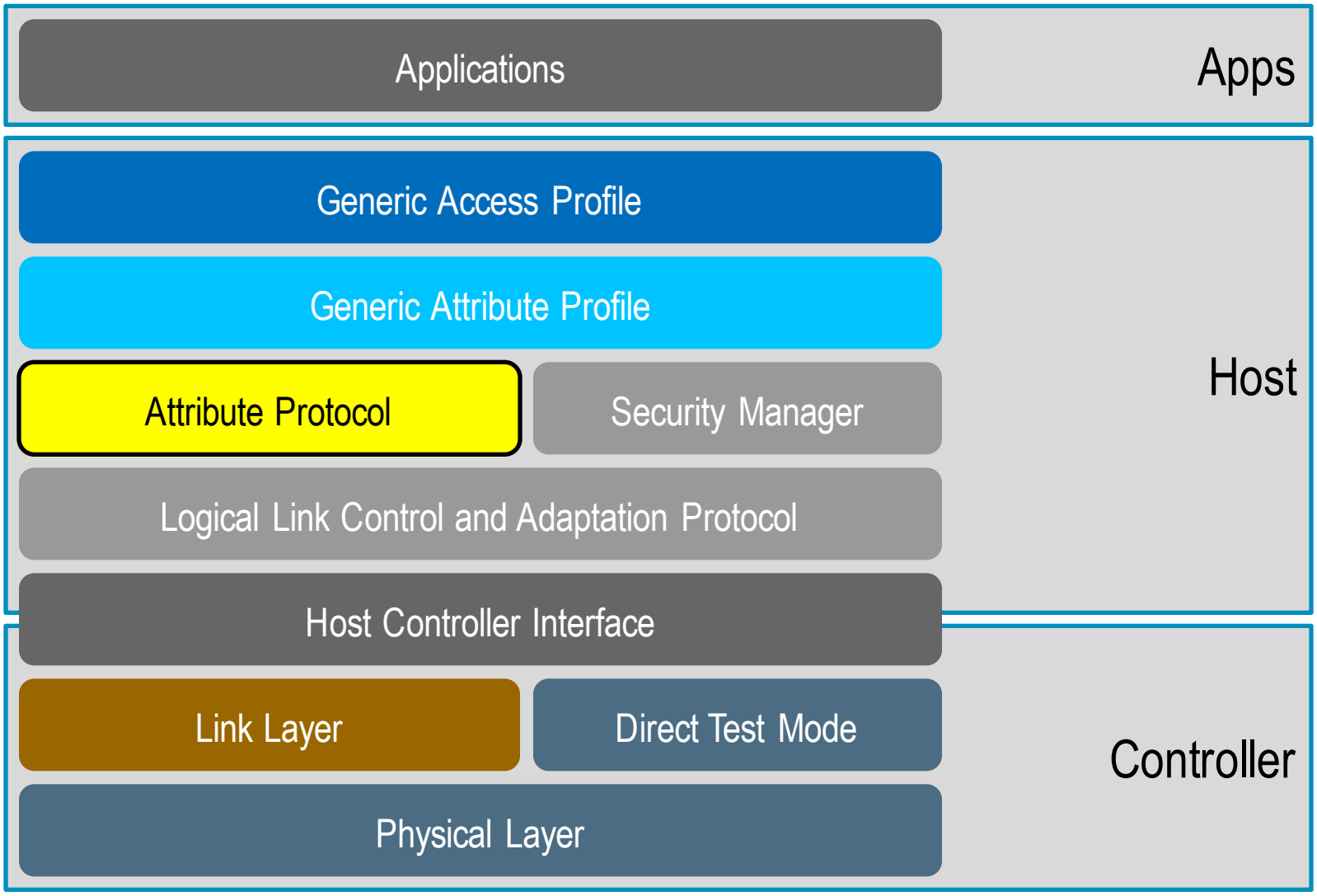
CID	Description	Notes
0x0000	Null identifier	Not used (ever)
0x0001	L2CAP Signaling Channel	Used over BR/EDR
0x0002	Connectionless Channel	Used over BR/EDR
0x0003	AMP Manager Protocol	Used over BR/EDR
0x0004	**Attribute Protocol**	**Used over LE only**
0x0005	**LE L2CAP Signaling Channel**	**Used over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

Connection Oriented Channels

New feature in v4.1

- allows a connection oriented channels
- proposed to be used for “Object Transfer”

Attribute Protocol



Attribute Protocol (ATT)

Client Server Architecture

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

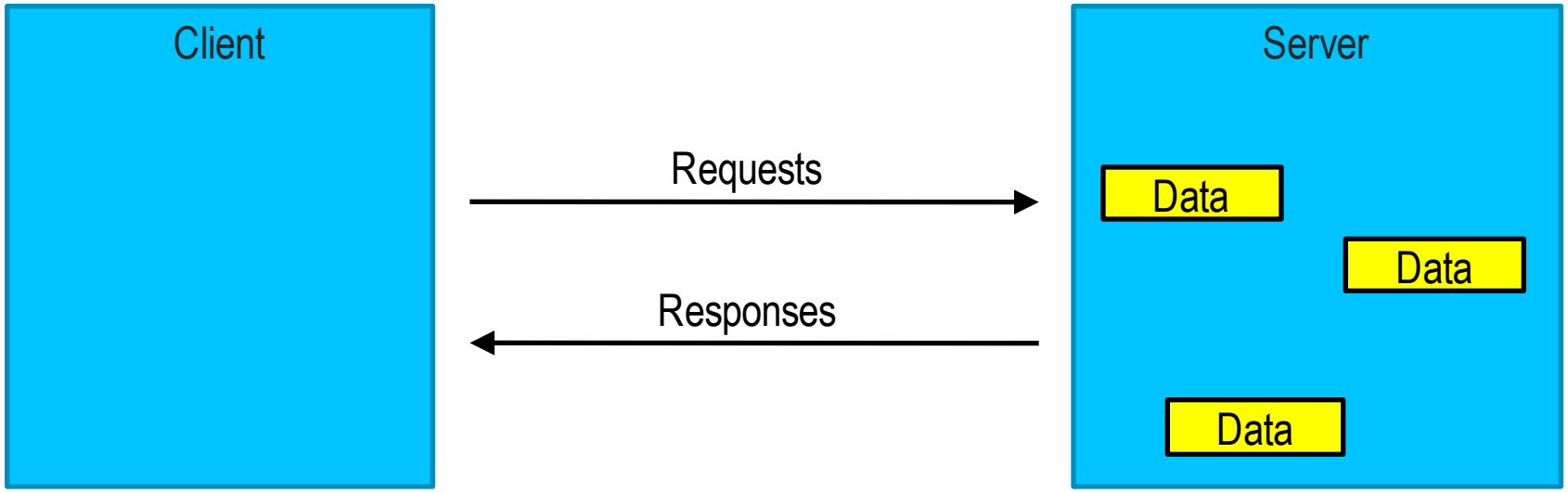
Protocol Methods

- 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
0x54656d7065726174757265205365e736f72
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	0x54656d70657261747572652053656e736f72
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»	0x54656d7065726174757265205365e736f72
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

Attributes 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

0000**xxxx**-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

0000**1234**-0000-1000-8000-00805F9B34FB

= 16 bit UUID 0x**1234**

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

0000**xxxx**-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

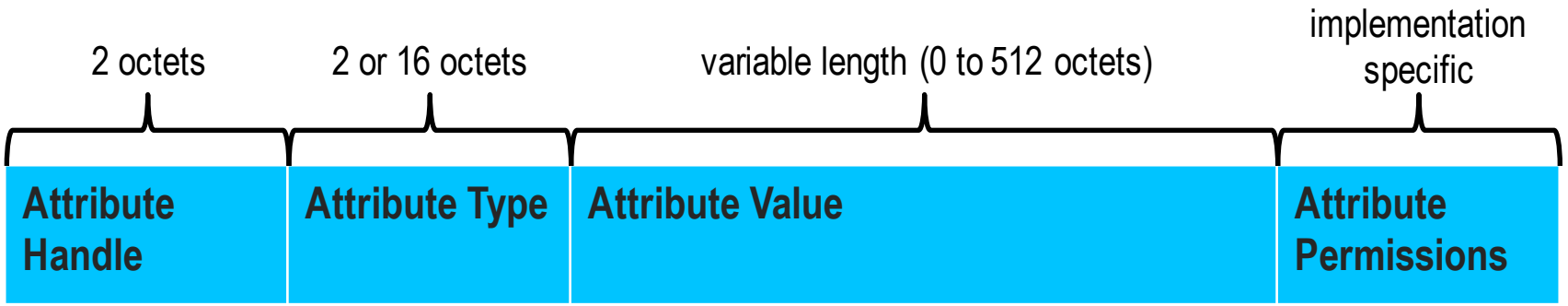
Attribute Handles are public information
Attribute Types are public information
Attribute Values can be protected

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



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»
- «Include»
- «Characteristic»
  - «Data»
  - «Data»

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 Request / 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

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

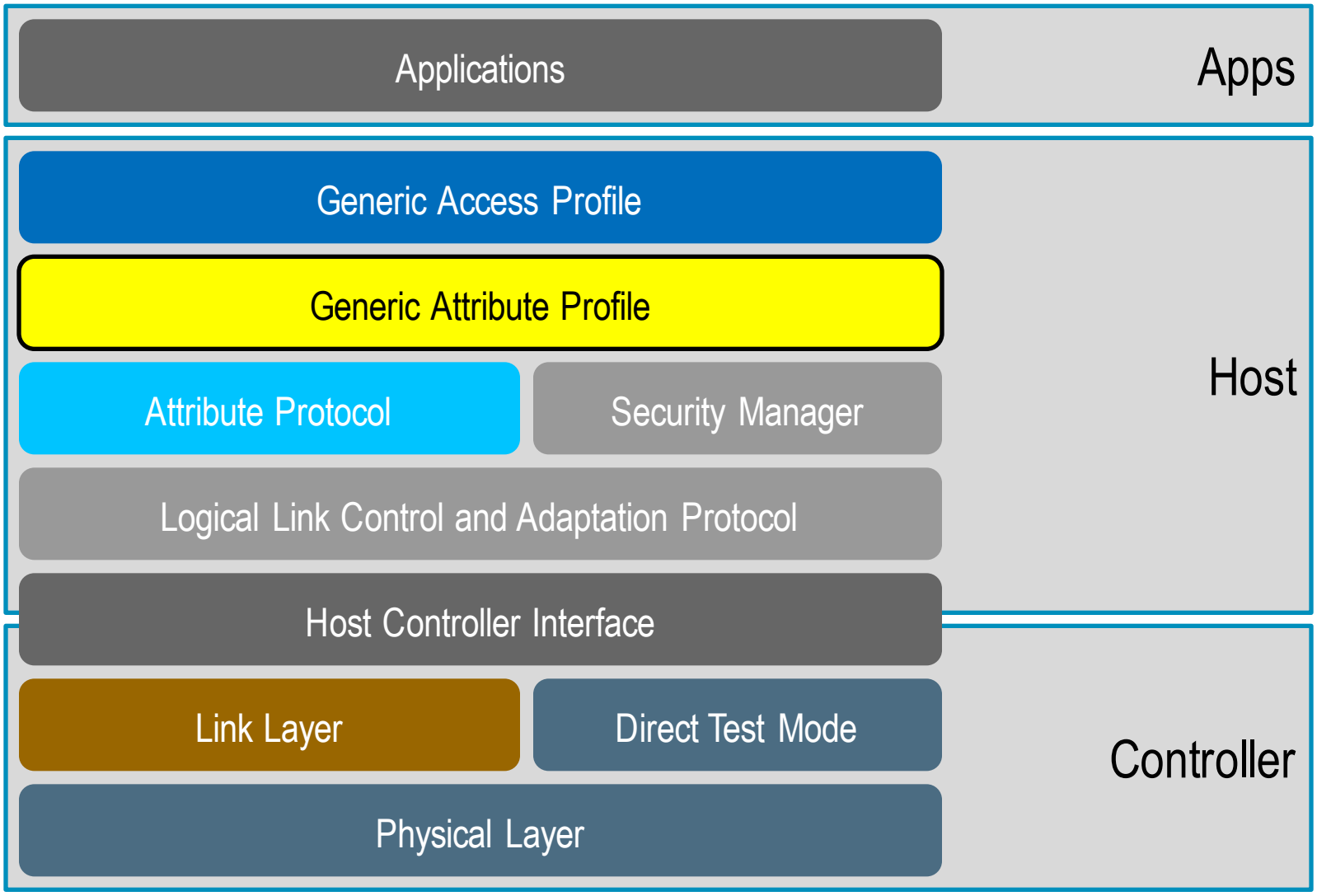
Attribute Protocol Summary

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 Protocol (GATT)



Generic Attribute Protocol (GATT)

Defines concepts of:

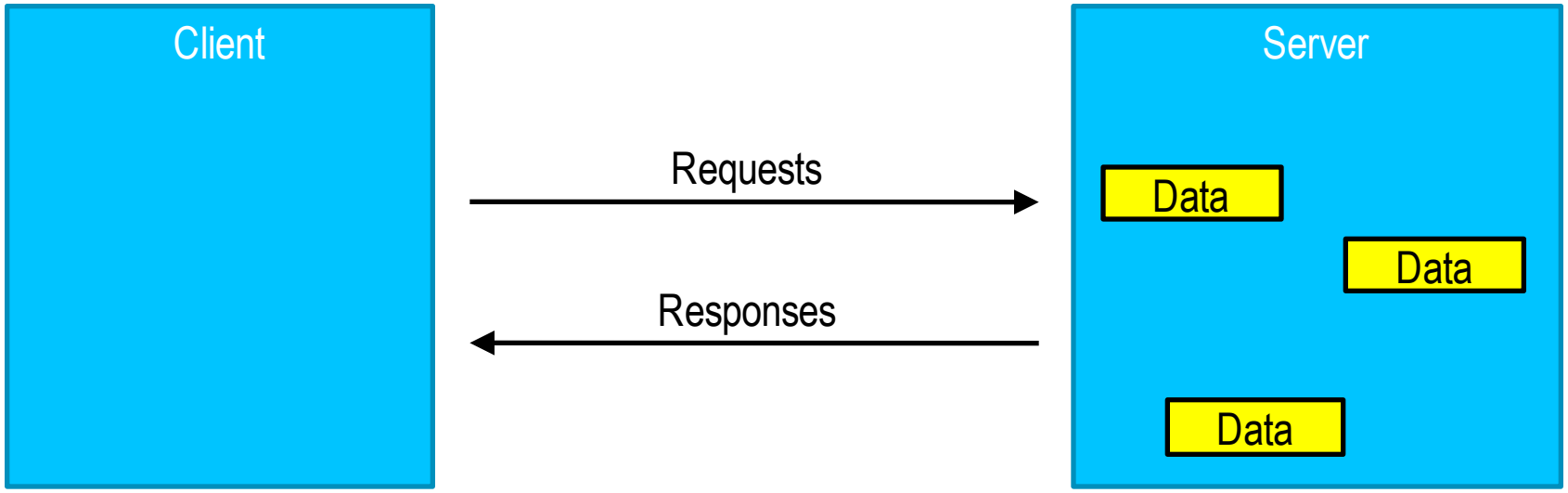
- Service Group
- Characteristic Group
- Declarations
- Descriptors

Does not define rules for their use

- this is separate but essential to understand

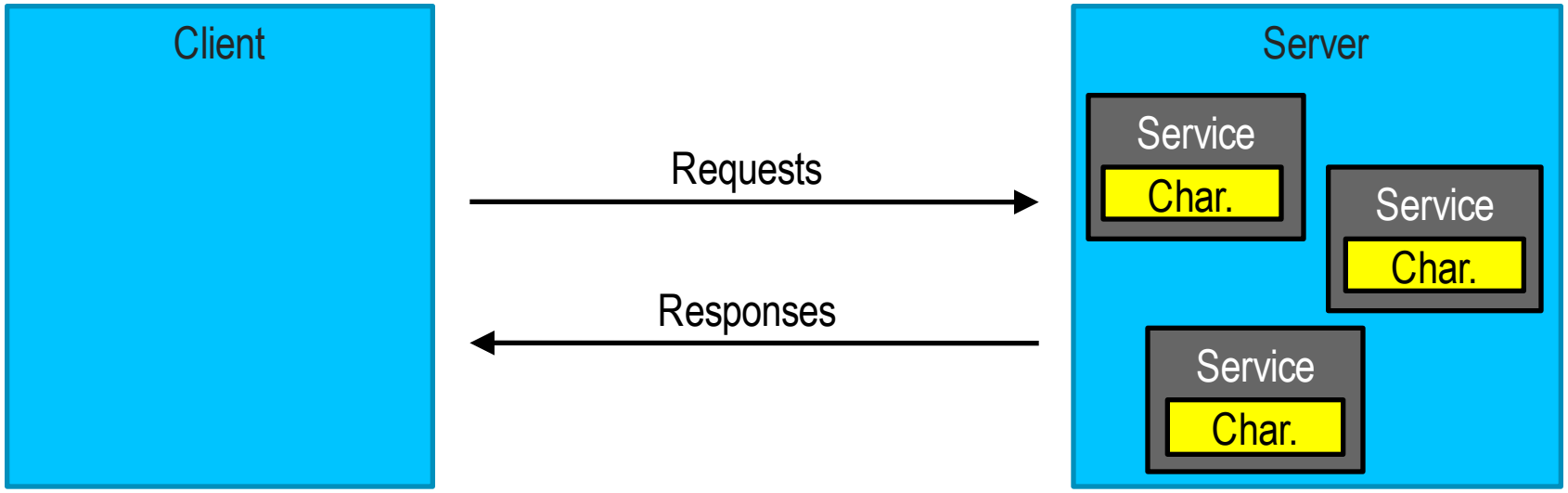
CLIENT SERVER ARCHITECTURE

Same client server architecture as Attribute Protocol



Client Server Architecture

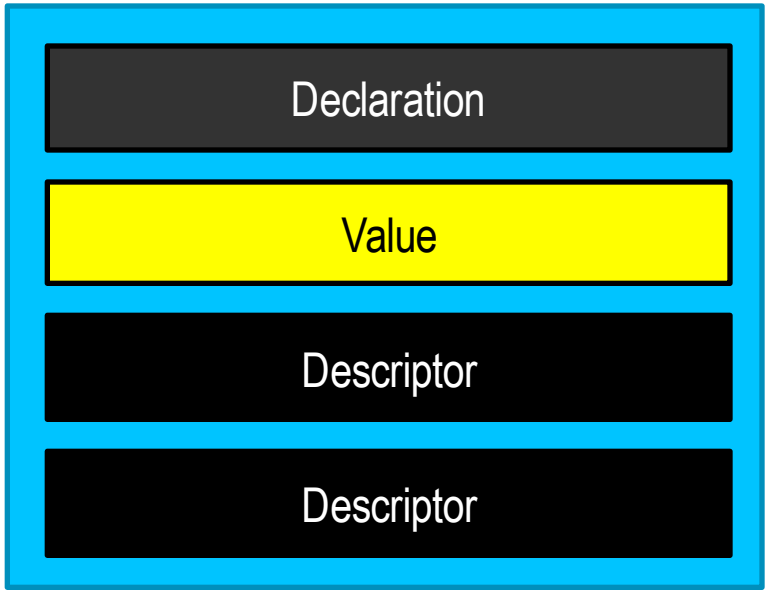
Same client server architecture as Attribute Protocol except that data is encapsulated in “Services” and data is exposed in “Characteristic”



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

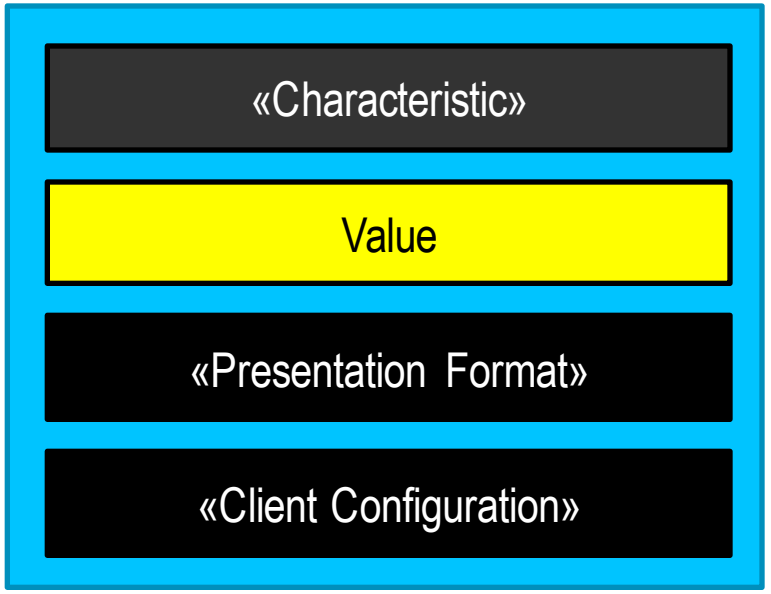
Descriptors

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.

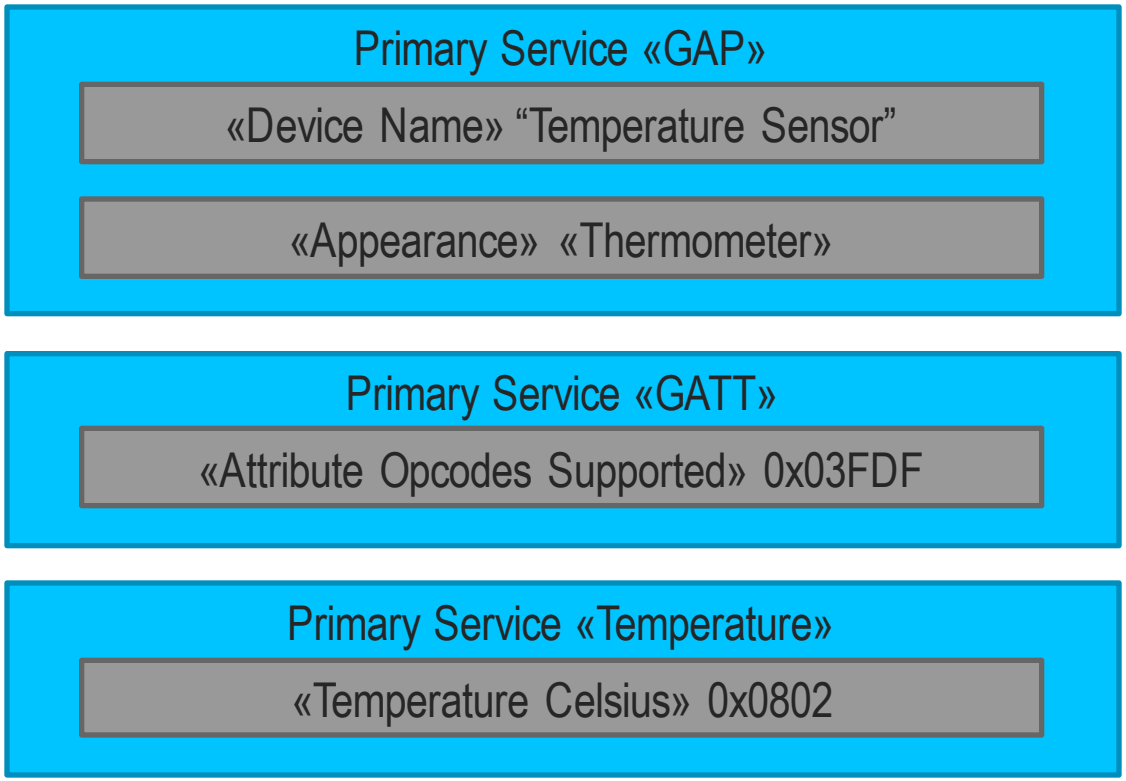
Secondary Service

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.

Attribute 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*

GATT Provides Structure



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*

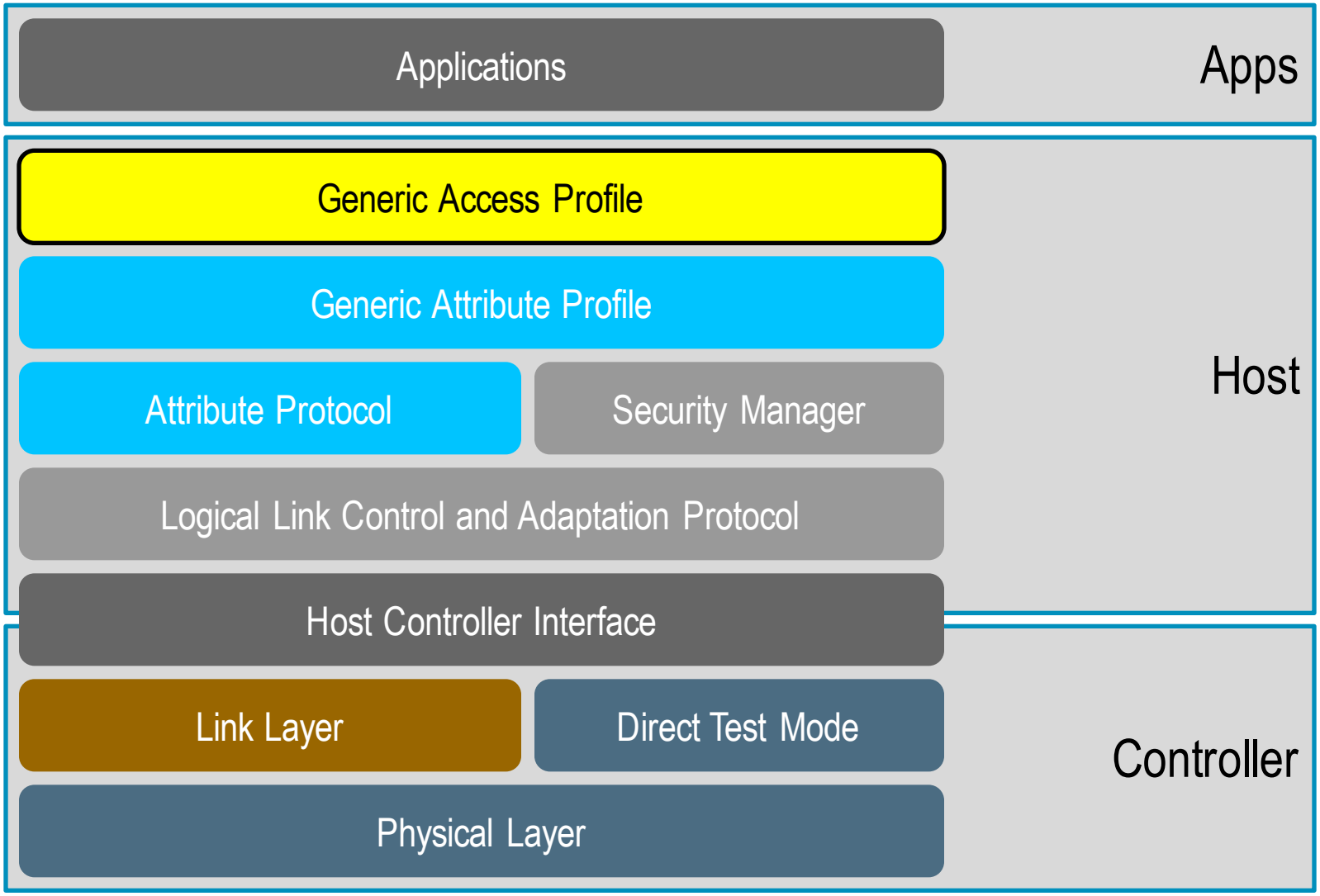
Grouping Gives Structure

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*

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

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

GENERIC ACCESS PROFILE



Generic Access Profile (GAP)

Profile Roles

- Broadcaster, Observer
- Peripheral, Central

Defines standard ways for devices to connect

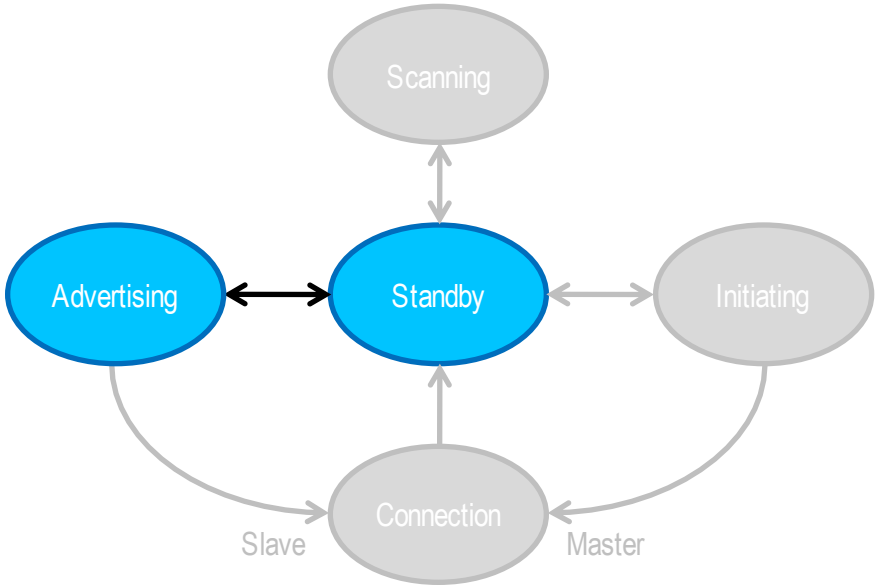
- Discoverable, Connectable, Bonding

Privacy

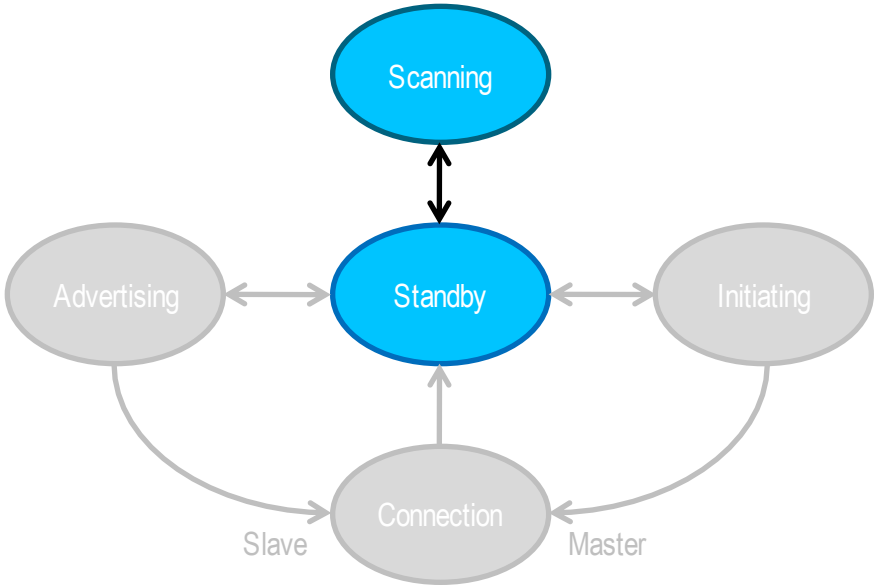
- Resolvable Private Addresses

Link Layer State Machines for Broadcaster and Observer

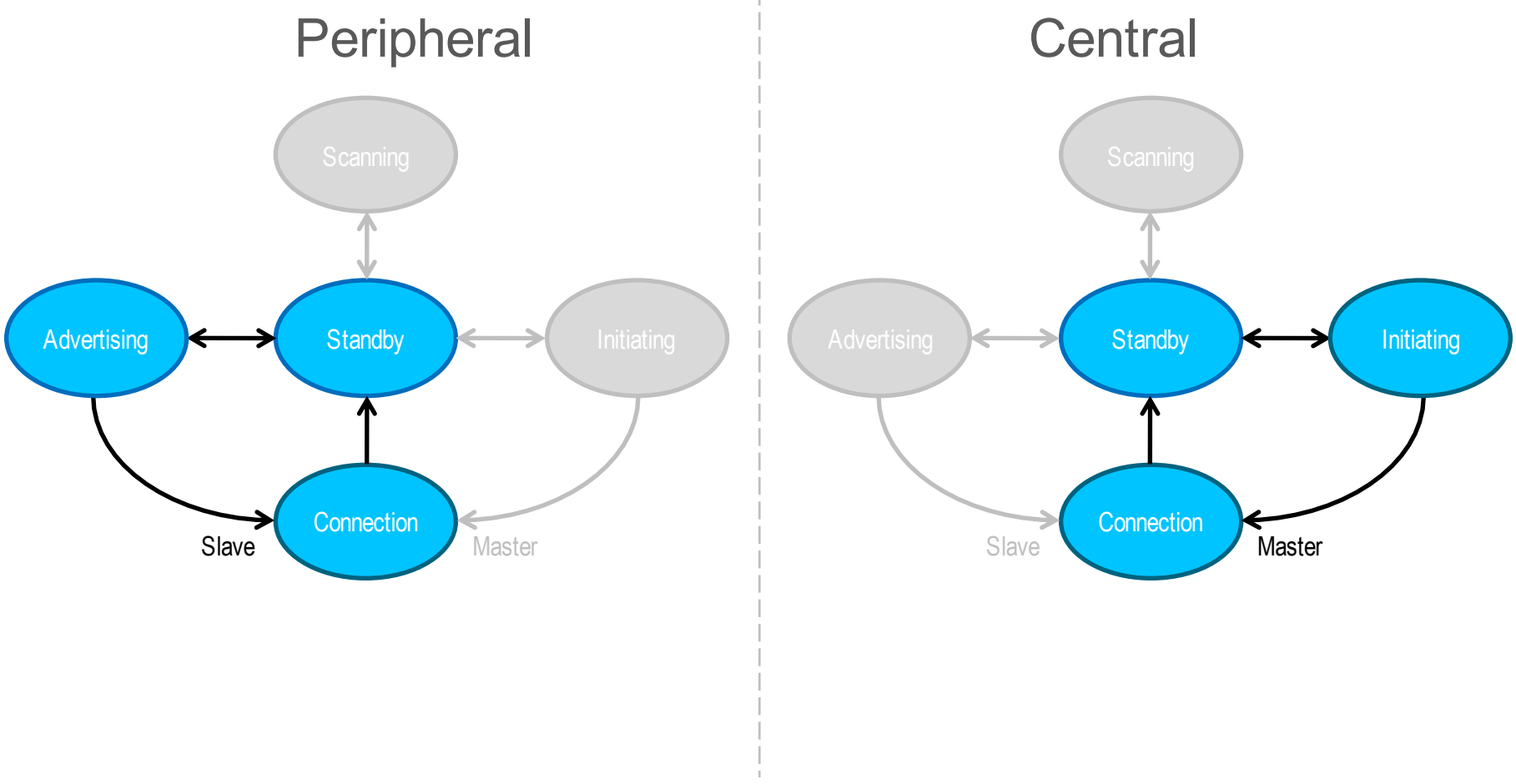
Broadcaster



Observer



Link Layer State Machines for Peripheral and 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

Tag Value	Description
0x01	Flags
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)

What can Discover 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

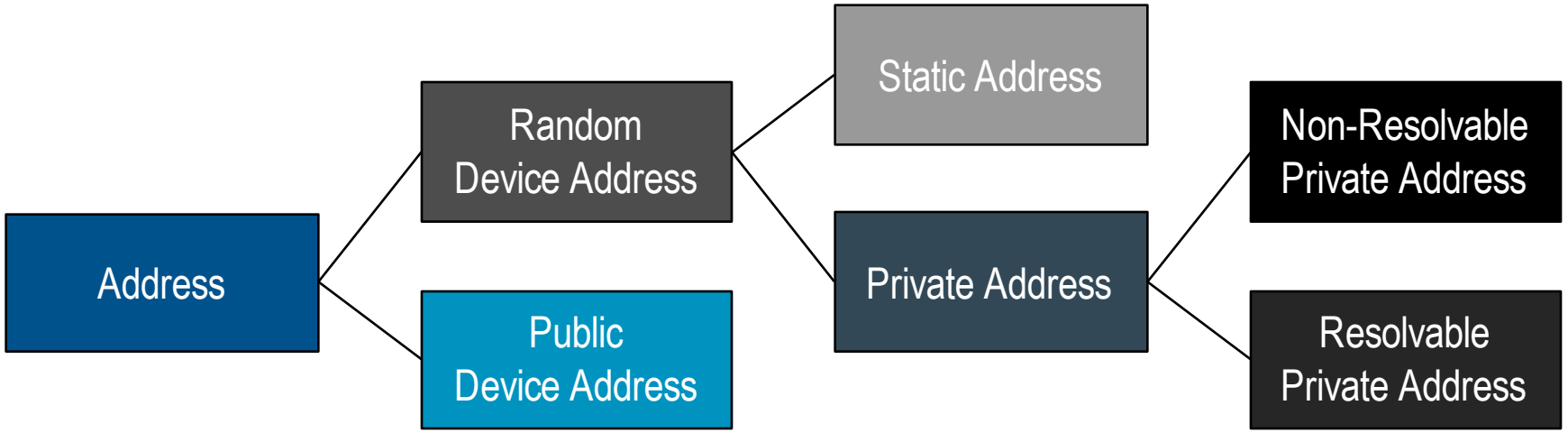
Security Manager defines how to “pair” devices
- authenticate and then encrypt a link

Bonding is the storing of
- Security and Identity Information

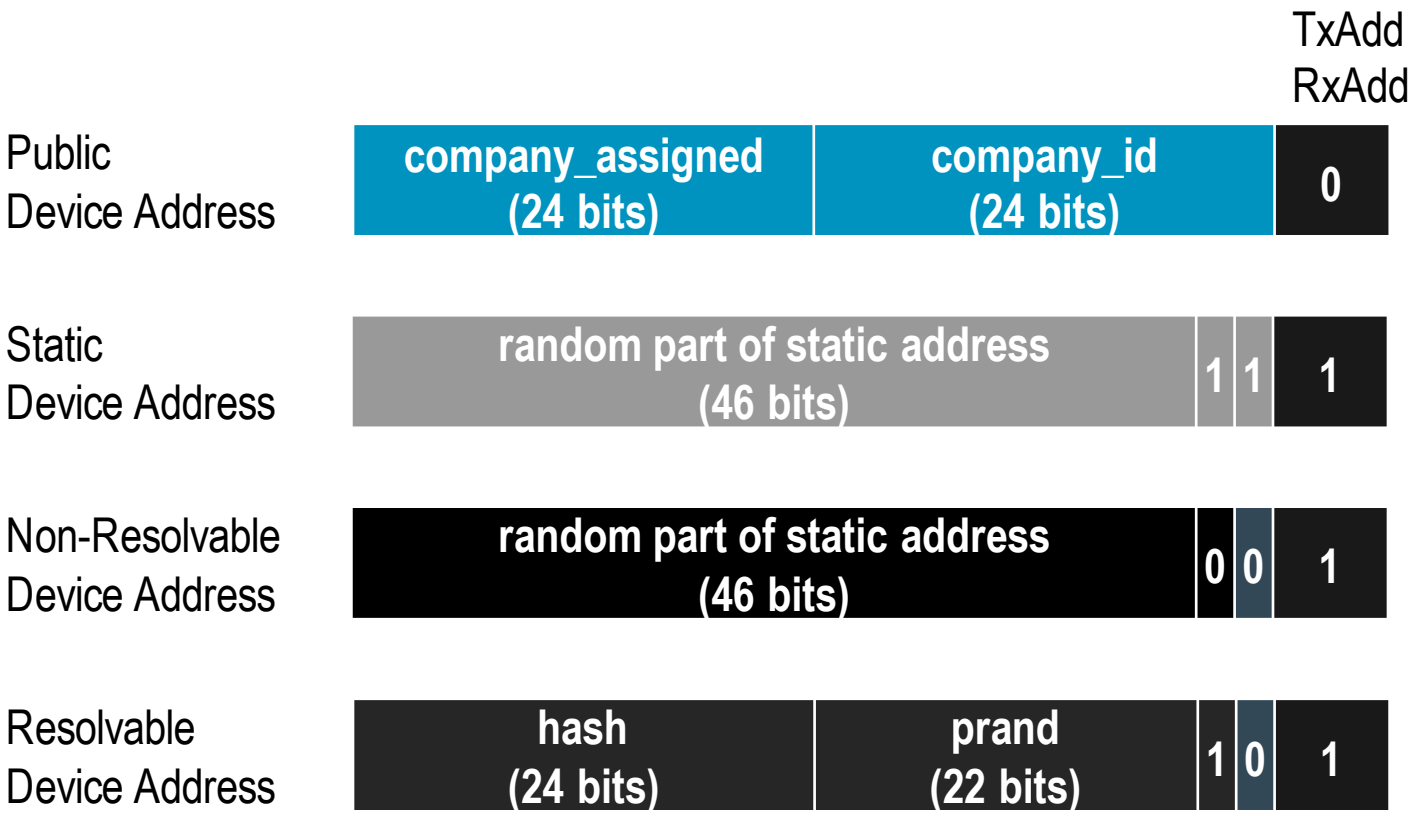
Once bonded
- a device can reconnect using the stored information
- GATT will store service change information for device

Private Addresses

A type of Random Device Address
sub-categorized into either:
Non-Resolvable Private Address
Resolvable Private Address



ADDRESS TYPES



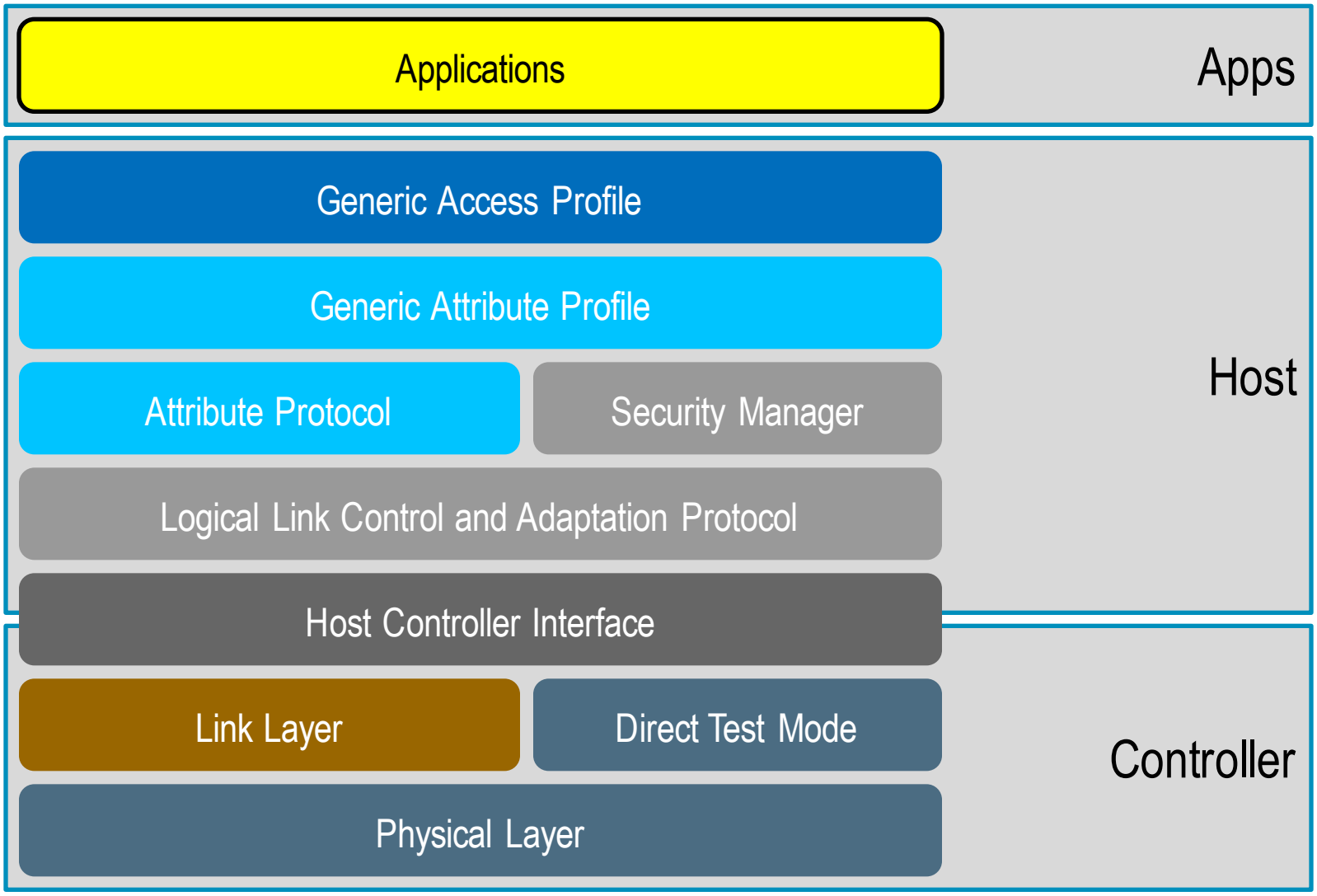
Resolvable Addresses

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

Applications

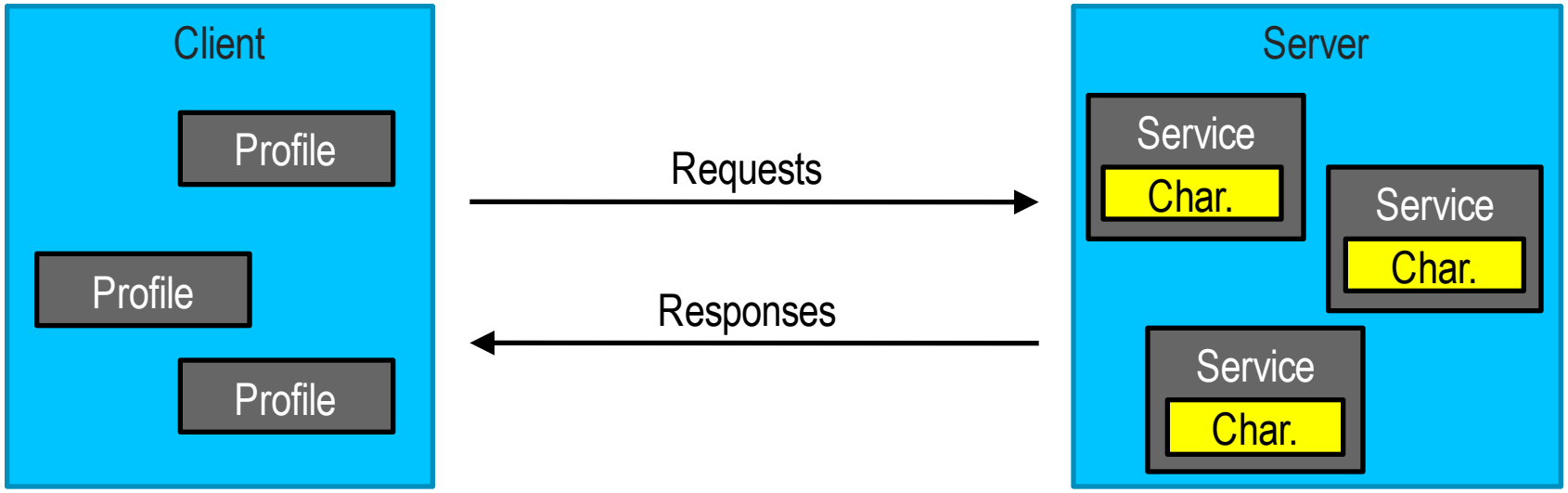


Applications

Client Server Architecture

Service – exposes behavior that have characteristics

Profile– define how to use services on a peer



Two Approaches

Use Standard Services
- defined by Bluetooth SIG
- includes lots of useful things
  - battery
  - heart rate
  - temperature

- can even request new ones
  - “New Work Proposal”

Great interoperability
- vast “app” ecosystem

Define Your Own Service
- using 128-bit UUIDs
- you can do what you want
  - motor controls
  - custom sensors
  - calibration

- can even buy 16-bit UUIDs
  - allow donation later

Closed ecosystems
- few supporting apps

Futures…

Faster Data Rates
- 2.5x to 4.5x faster

Low Power Optimizations
- privacy in Controller

Longer Range
- 500m to 1km ranges

Audio
- hearing aids

SUMMARY

New Technology
- Blank sheet of paper
- Optimized for ultra low powe

Asymmetric Architecture
- Broadcaster / Observer
- Peripheral / Central
- Client / Server
- Service / Use Case

SUMMARY

Leading Edge Technology
- AES-128 encryption, w/ CBC-MAC
- Simple Pairing
- Object Oriented Model
- XML based Testing

Robust
- 24-bit CRCs and 32-bit MICs
- Full AFH at Connection Setup
- Prepare / Execute Writes

SUMMARY

It does everything you need

- 30 – 50 meter range
- highly robust
- consultant free
- very fast connections
- signed transactions
- simple star topology
- very low cost

SUMMARY

Bluetooth low energy

the only open wireless standard

connecting everything else

sports sensors, fitness equipment, automotive keyless go, car climate control, cellular phones, games consoles, medical devices, proximity, smart energy, assisted living, phone accessories, wellness, animal tagging, intelligent transport, machine to machine, shoes, watches, bracelets, rings, toys, remote controls, location based services, broadcast services, find my car keys, children tracking, smart appliances, automatic maintenance of household goods, and your next great idea…

THANK YOU!