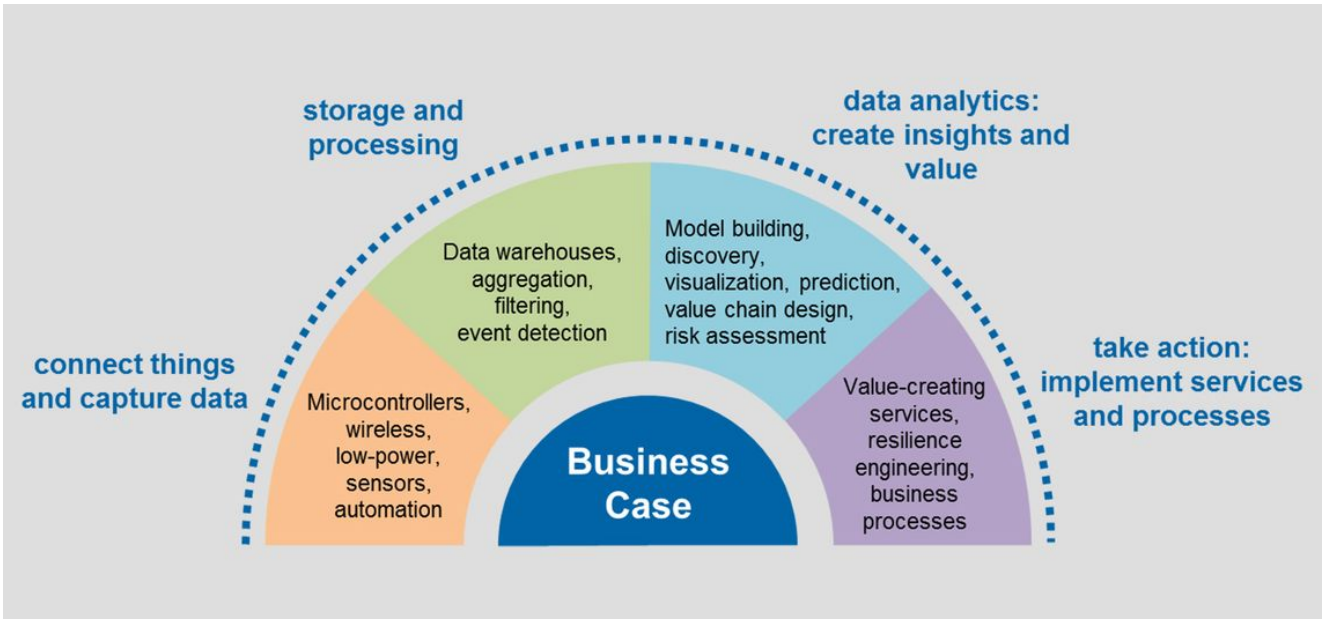
IOT2

Introduction

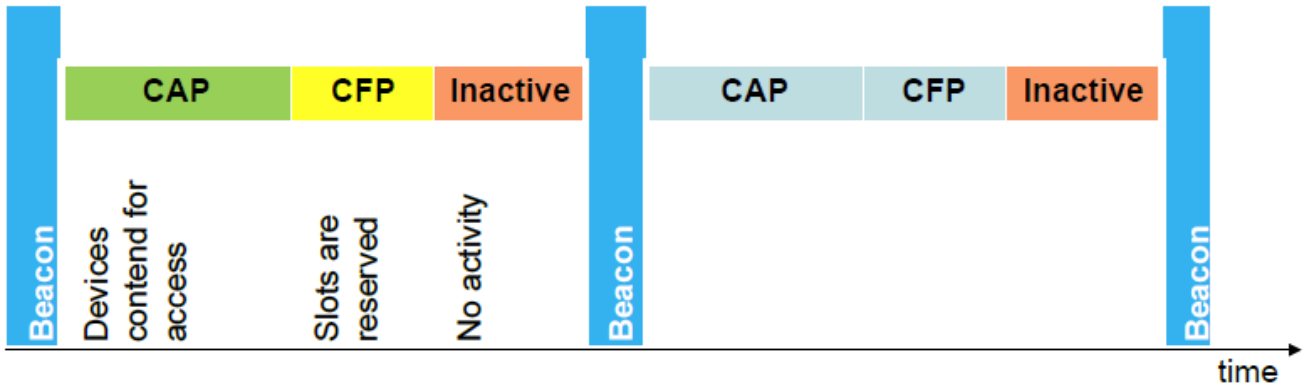
- IoT = more things than people connected to the internet; elements: data source/sink, communication, computing/storage, objects combining functions, frontend parts, comm infra, backend parts
- challenges: resources constrained (ROM, CPU, power, RAM, UI, developer accessibility), mobility/wireless, gateways, scalability, addressing, communication (range, regulation/certification, interference / medium access, social issues, power source)
- wired is better for bigger objects, when size/weight, mobility is non an issue, high data rate is needed



802.15.4

- reasons for 802.15.4: low complexity, low data rate (250 Kbps), low-cost, low-power, 1 meter precision, several bands; capabilities: star / p2p, 64-bit extended / 16-bit short address, optional guaranteed time slots, carrier sense multiple access w/ collision avoidance (CSMA-CA) or ALOHA, fully ack'ed proto, lower power, energy detection (ED), link quality indication (LQI)
- wireless comm has an inherent security problem: with the right kind of radio, you can listen in
- wireless problems: distance, physical obstacles, attenuation; increasing power can help, but disturbs other, regulations, more energy needed; interference from others
- PAN/LAN/NAN/WAN
- robustness & reliability is hampered; use CRC but that's an extra few bits which is expensive; using ACK frames also costs energy, time, bandwidth (no matter whether separate or attached to data frame)

- increase reliability by: working on different channels; use channel hopping (-> increases complexity); changing frequencies; reducing data rate; increasing power / use appropriate power (-> regulations); use repeaters; use mesh network
- CSMA/CA: listen before transmitting (= CS) or wait if channel is busy (= CA); CD (detecting collisions) is not possible for wireless; problems: hidden/too-far-away node (ACK helps)
- antenna diversity can help mitigate multipath effects
- 802.15.4 net model: extended 64-bit address, PAN address is 16 bits; 2 device types: FFD (full-function; more resources/expensive/energy, comm w/ RFD,FFD) or RFD (reduced-function; can only be end device, less energy than FFD, can only comm w/ FFD); different functions: coordinator (FFD-only), router, end device
- net topology: min 2 components: PAN coordinator (FFD) establishes net, others join and comm, each has unique extended address, short address received from coordinator, net-local comm can use both addresses; topos: star (everything through coordinator; e.g. personal health care), peer-to-peer (FFD-FFD comm possible, still w/ coordinator; e.g. building automation, agriculture; can be used for mesh nets), cluster trees
- PHY layer = RF transceiver, control mechanism of radio; ED, LQI; MAC = provides access to physical channels; beacon mgmt, channel access, GTS mgmt, frame validation, ACK, (dis-)assoc
- frames are basic unit of transport mechanism; 4 types: beacon, data, ack, MAC
- access mechanisms: non-beacon / unslotted (beacon still required for net discovery; positive ACK); beacon-enabled / slotted (coordinators emit regular beacons; used for synchronous nets / dedicated bandwidth / low latency; e.g. PC peripherals; low power consumption mode for coordinator)



- up to 7 GTSs (guaranteed time slots)
- data transfer models: at least 1 coord in net; data transfer transactions can be dev-coord (slotted: dev listens for beacons, syncs to it, then sends when appropriate, optional ACK; non-slotted: devs doesn't listen for beacon, optional ACK), coord-dev (slotted: beacon transmits intent to send, dev replies MAC to receive to req data, coord acks + data, ACK back to coord; non-slotted: dev has to req data), dev-dev (star only via coord, p2p all three types)
- MAC frame

<table>
<tr><td>Octets: 2</td><td>1</td><td>0/2</td><td>0/2/8</td><td>0/2</td><td>0/2/8</td><td>0/5/6/10/14</td><td>variable</td><td>2</td></tr>
<tr><td rowspan="2">Frame Control</td><td rowspan="2">Sequence Number</td><td>Destination PAN Identifier</td><td>Destination Address</td><td>Source PAN Identifier</td><td>Source Address</td><td rowspan="2">Auxiliary Security Header</td><td rowspan="2">Frame Payload</td><td rowspan="2">FCS</td></tr>
<tr><td colspan="4">Addressing fields</td></tr>
<tr><td colspan="7">MHR</td><td>MAC Payload</td><td>MFR</td></tr>
</table>

- frame control; type 000 beacon, 001 data, 010 ack, 011 command; sec 0/1, frame pending 0/1, ACK (request) 0/1, PAN ID compression (0 = PAN ID present, 1 = only dest PAN present if both addresses are used), dest addr. mode (read as; bit11,bit10; 00 = PAN ID + addr fields not present, 10 = addr field is short addr (16 bit), 11 = addr field is long addr (64 bit)), src add mode is same as dest addr mode

Bits: 0–2	3	4	5	6	7–9	10–11	12–13	14–15
Frame Type	Security Enabled	Frame Pending	AR	PAN ID Compression	Reserved	Dest. Addressing Mode	Frame Version	Source Addressing Mode

- beacon frame

Beacon frame format

Frame type =000 Beacon

Octets: 2	1	4/10	0/5/6/10/14	2	variable	variable	variable	2
Frame Control	Sequence Number	Addressing fields	Auxiliary Security Header	Superframe Specification	GTS fields	Pending address fields	Beacon Payload	FCS
MHR				MAC Payload				MFR

Octets: 1	variable
Pending Address Specification	Address List

0/2	0/2/8	0/2	0/2/8
Destination PAN Identifier	Destination Address	Source PAN Identifier	Source Address
Addressing fields			

Octets: 1	0/1	variable
GTS Specification	GTS Directions	GTS List

Bits: 0–3	4–7	8–11	12	13	14	15
Beacon Order	Superframe Order	Final CAP Slot	Battery Life Extension (BLE)	Reserved	PAN Coordinator	Association Permit

- data frame

Frame type =001 Data

Octets: 2	1	variable	0/5/6/10/14	variable	2
Frame Control	Sequence Number	Addressing fields	Auxiliary Security Header	Data Payload	FCS
MHR				MAC Payload	MFR

- ACK frame

Frame type =010 Ack

Octets: 2	1	2
Frame Control	Sequence Number	FCS
MHR		MFR

- command frame

Frame type =011 MAC Command

Octets: 2	1	variable	0/5/6/10/14	1	variable	2
Frame Control	Sequence Number	Addressing fields	Auxiliary Security Header	Command Frame Identifier	Command Payload	FCS
MHR				MAC Payload		MFR

Command Frame ID	Command name	FFD support	RFD support	Coord	PAN Coord	Dev
0x01	Association request Unassociated Src requests association to Dest	Tx Rx	Tx	Dest	Dest	Src
0x02	Association response Response of Src to Dest seeking association	Tx Rx	Rx	Src	Src	Dest
0x03	Disassociation notification Src wishes to disassociate from Dest	Tx Rx	Tx Rx	Src Dest	Src Dest	Dest Src
0x04	Data request Src requests data from Dest	Tx Rx	Tx	Dest	Dest	Src
0x05	PAN ID conflict notification Src informs Dest of PAN Identifier conflict	Tx Rx	Tx		Dest	Src
0x06	Orphan notification Src indicates that it has lost synchronisation	Tx Rx	Tx	Dest		Src
0x07	Beacon request Src tries to locate coordinators (Dest) within range	Tx Rx		Dest	Dest	Src
0x08	Coordinator realignment Realign information sent out by Src to Dest in PAN	Tx Rx	Rx	Src	Src	Dest
0x09	GTS request. Associated Src requests allocation/deallocation of GTS from Dest			Dest	Dest	Src
0x0A-0xFF	Reserved					

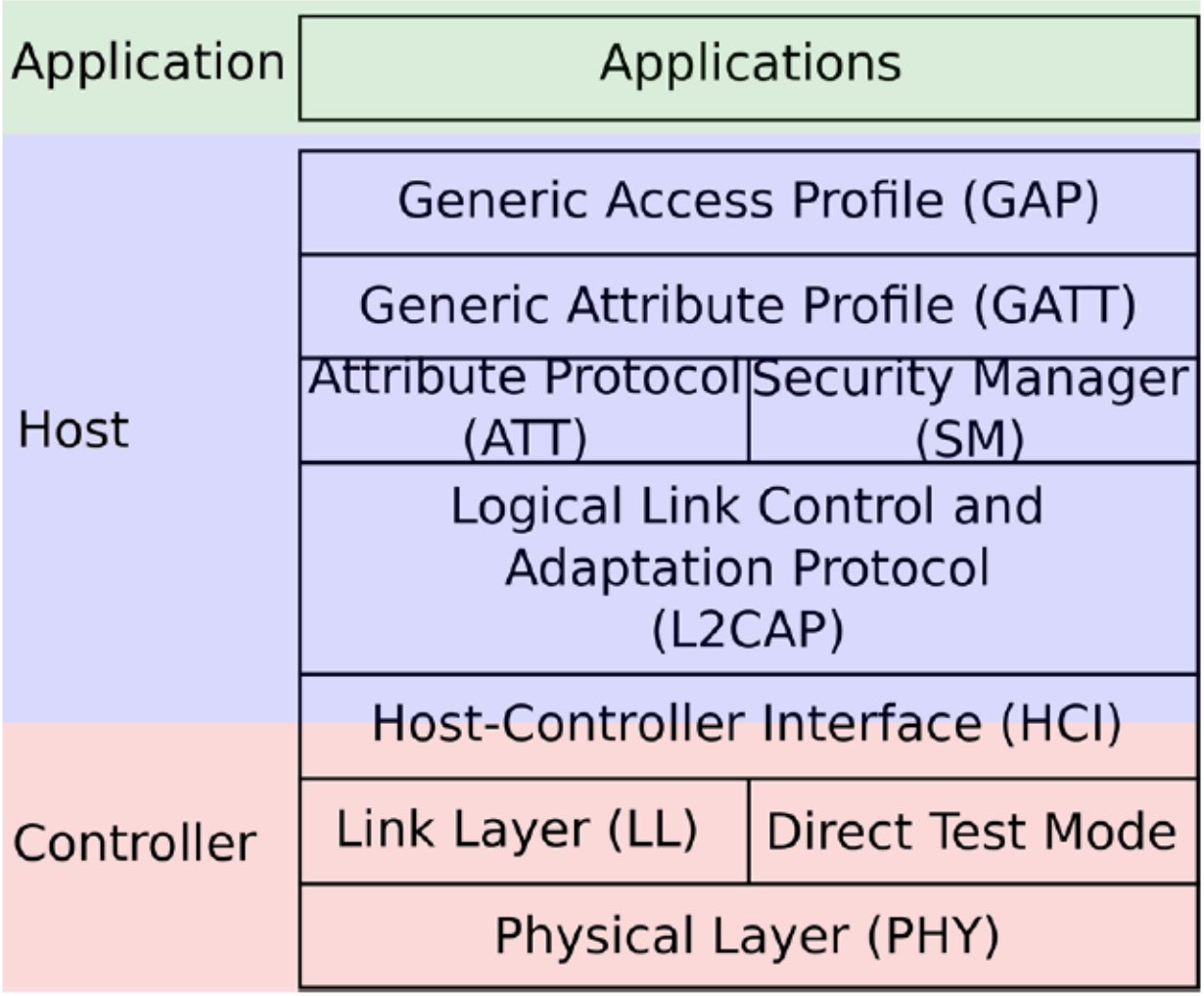
6LoWPAN

- often, WPAN are non-IP but use specific protocols (low power, small size, mobility, low-cost, targeted); WPAN nodes cannot always cope w/ large TCP/IP packets (processing, energy reqs); instead using translators for IP
- IPv6 is large, no NAT, stateless, larger addresses, IPSec etc.
- 802.15.4 IPv6 MTU is 1280 octets, frame of 127 octets is too small, header is too long (IPv6: 40 octets, UDP: 8 octets)
- 6LoWPAN uses unslotted CSMA/CA, fragments + reassembles IPv6 packets, compresses IPv6, UDP headers, mesh routing support, low processing and storage costs;
- 127 bytes: 23 mac header | 21 security header (AES-CCM-128; 9 for AES-CCM-32 or 13 for AES-CCM-64) | 40 IP6 header | ... | 2 FCS; 8 UDP header + 33 data OR 20 TCP header + 21 data
- payload / MAC layer max size is 102 (127 - 23 header - 2 FCS) octets (81 w/ security); for UDP: 54/33 octets; for TCP: 21 octets
- header compression: HC1/2, IPHC/NHC

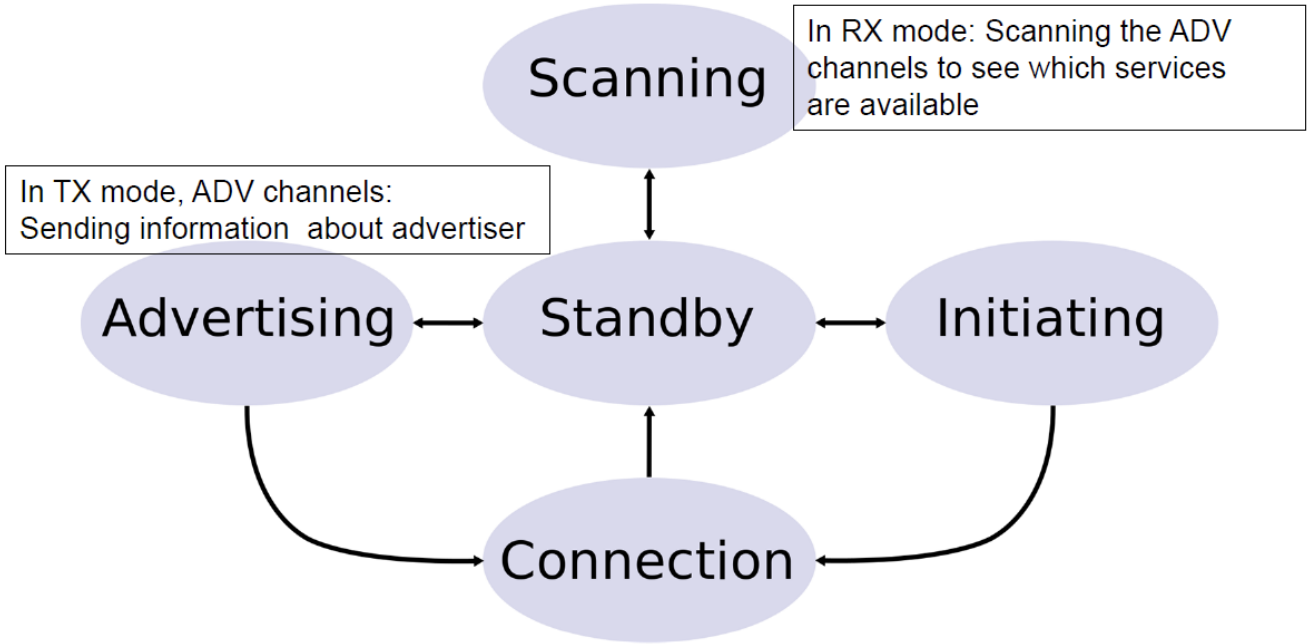
		Payload part in green parts			Comments/description in yellow parts	
IPv6 Dispatch	IPv6 Header	Payload			IPv6 datagram	
HC1 Dispatch	HC1 Header	Payload			LOWPAN_HC1 compressed IPv6 datagram	
Mesh Type	Mesh Header	HC1 Dispatch	HC1 Header	Payload	LOWPAN_HC1 compressed IPv6 datagram that requires mesh addressing	
Frag Type	Frag Header	HC1 Dispatch	HC1 Header	Payload	LOWPAN_HC1 compressed IPv6 datagram that requires fragmentation	
M Typ	M Hdr	F Typ	F Hdr	HC1 Dsp	HC1 Hdr	Payload
LOWPAN_HC1 compressed IPv6 datagram that requires both mesh addressing and fragmentation						
M Typ	M Hdr	B Dsp	B Hdr	HC1 Dsp	HC1 Hdr	Payload
LOWPAN_HC1 compressed IPv6 datagram that requires both mesh addressing and a broadcast header to support mesh broadcast/multicast						

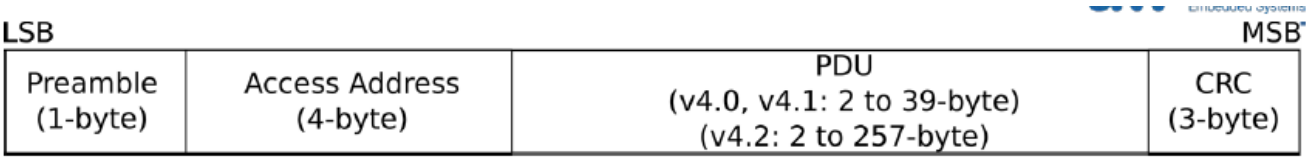
- IPv6 consists of 128 bits: unicast: 64 net prefix (48 routing, 16 subnet) + 64 interface ID OR link-local: 64 net prefix (10 prefix, 54 zeros) + 64 interface ID
- IPv6 has unicast, anycast, multicast; no broadcast
- IPv6 identifier ID is obtained from MAC header (dest/src nodes + PAN)
- fragmentation: send too long datagrams (up to 1280 octets), inefficient, possible retransmission

BLE / Bluetooth Smart



- BT has high penetration in devices; BT Smart = BTLE, long sleep, fast comm, quick & JIT wake-up; v5: higher data rate, longer range
- single mode: BT Smart; dual mode: support Smart and Classic
- PHY: 2.4 GHz, 2MHz spacing, GFSK modulation, output power between -20 and +10 DBm, sensitivity < -70 dbM, 10-100m
- LL (link layer): 3 advertising channels (can be used for data; ch 37-39), 37 data channels; FSM: scanning, standby, initiating/advertising, connection; master/slave principle (scan->init->conn/adv->conn); preamble + access address (conn ID) + PDF (protocol data unit) + CRC





- access address for ADV is always `0x8E89BED6`
- ADV types:
  - ADV_IND: Connectable undirected advertising. Advertises for all. Connections are accepted
  - ADV_DIRECT_IND: Connectable directed advertising. Not for all. Specific initiator (field initA, 6 bytes) can connect fast.
  - ADV_NONCONN_IND: Non connectable undirected advertising. Just broadcasts data(e.g. beacons). Connections are not possible
  - SCAN_REQ: Scanner makes a request to Advertiser. Requests information from Advertiser
  - SCAN_RSP: possible response of Advertiser to SCAN_REQ. Discloses more information about advertiser
  - CONNECT_REQ: Connection request (to request a connection)
  - ADV_SCAN_IND: Advertises a non connectable broadcaster that can provide additional information by SCAN_RSP
- LL summary:
  - 1 packet format. PDU and CRC whitened
  - 2 PDU types. PDU protected with 32-bit CRC; advertising channel PDU: 16-47 Bytes (0 -31-byte payload); data channel PDU: 10-41 Bytes or 10-265 Bytes (0 –27-byte / 0 -251-byte payload)
  - 7 Advertising Channel PDU: 4 Advertising packets, 2 Scanning packets, Connection request
  - device advs on 1..3 channels; uses frequency hopping
  - Very fast connection and timing very important; 7.5ms <= connInterval<= 4.0s (multiple of 1.25ms)
- Host-Controller Interface (HCI): uniform command method; packet types: command 0x01 (H>C), data 0x02 (H<>C), event 0x04 (C>H)
- Logical Link Control and Adaptation Protocol (L2CAP): mux between link layer and 3 channels (attr proto, link layer control signaling, sec mgmt)
- Attribute Protocol (ATT): client/server arch above BLE logical transport channel for GATT sever/client; each attr has attr type with UUID, handle (unique per server), value (fixed/var length), permissions (r/w, enc, authentication, authorization); PDU (requests, responses, commands, notifications, indications, confirmations); some type UUIDs are pre-allocated, can be shortened

- The full 128-bit value of a 16-bit or 32-bit UUID may be computed
  - 128_bit_value = 16_bit_value * 296 + Bluetooth_Base_UUID
  - 128_bit_value = 32_bit_value * 296 + Bluetooth_Base_UUID
  - 32_bit_value = 16_bit_value + zero-valued 32-bit UUID

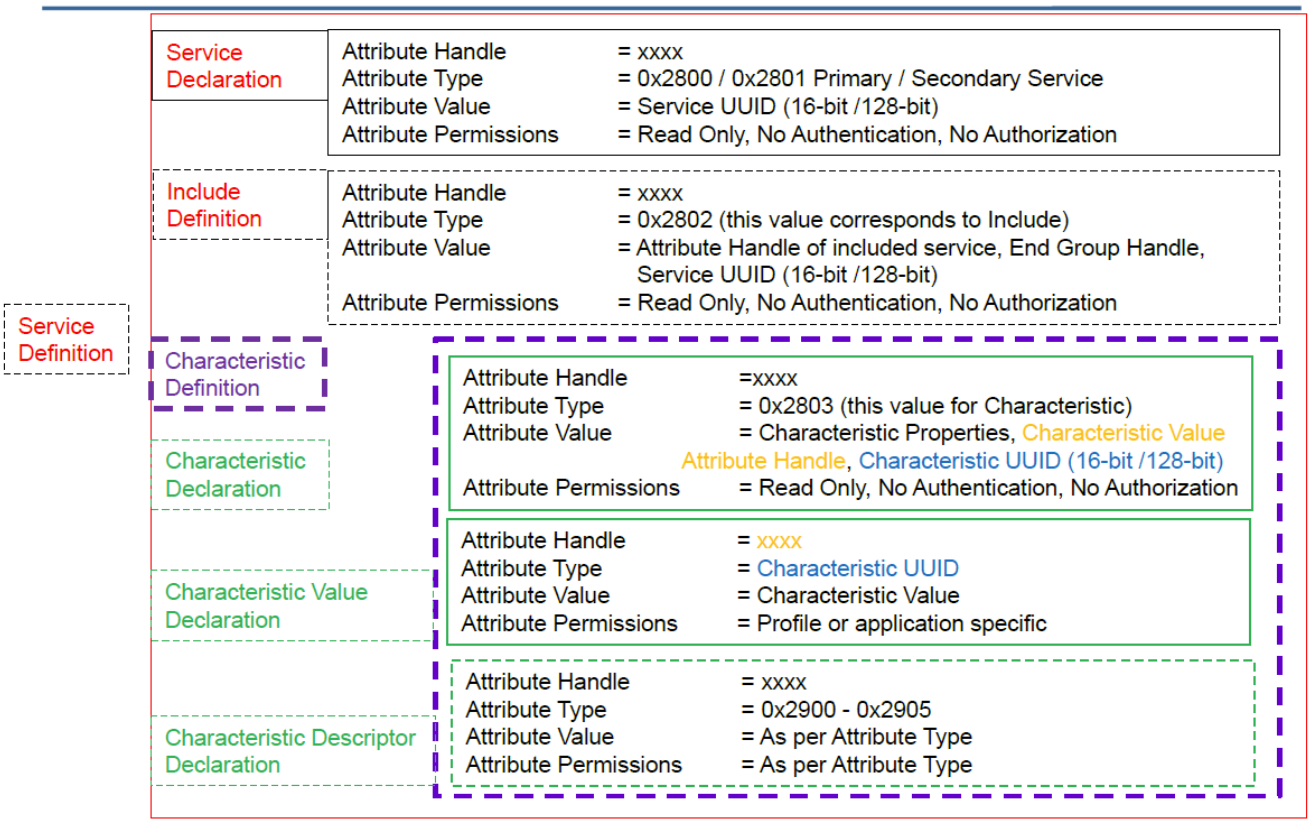
  That will mean:
  - 0000xxxx-0000-1000-8000-00805F9B34FB (16-bit_value)
  - xxxxxxxx-0000-1000-8000-00805F9B34FB (32-bit_value)
  - For the Heart rate Service, the 16-bit UUID is: 0x180D
    - ▶ The equivalent 128-bit UUID is:

    0000180D-0000-1000-8000-00805F9B34FB

- Generic Attribute Profile (GATT): client/server, generic service framework on ATT

GATT

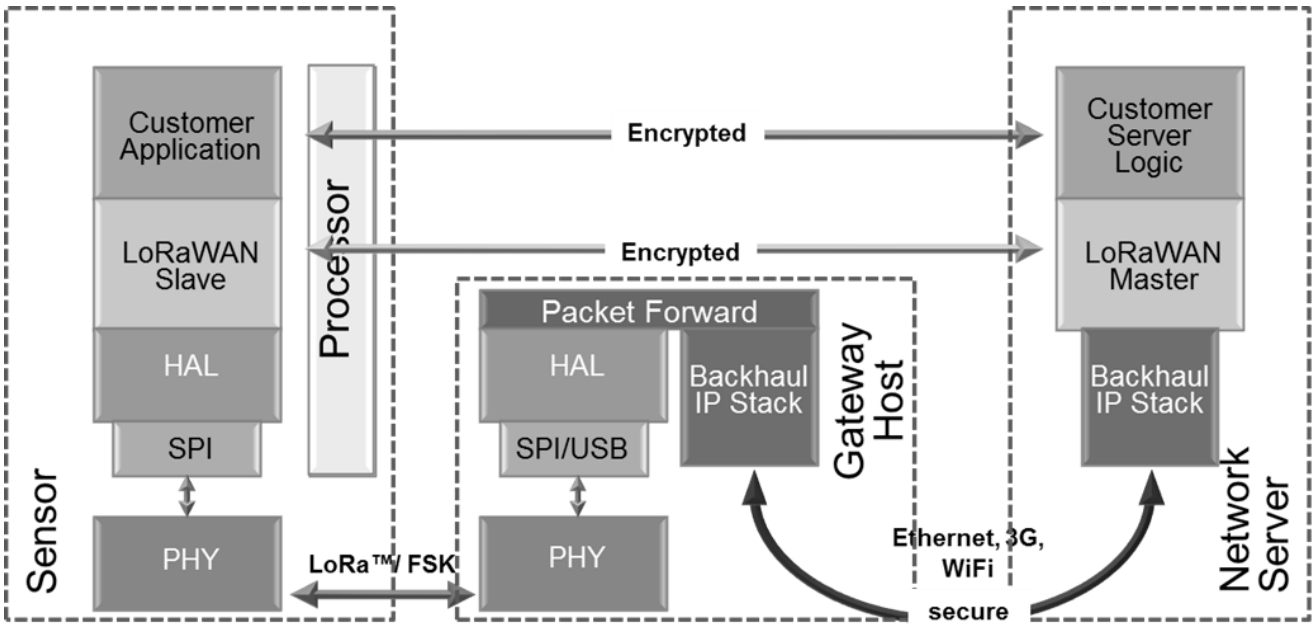
- data structure: grouping for three attribute types: primary svc, secondary svc, characteristic
- service is defined svc def; declaration, optional definitions, optional characteristic definitions (decl, prop, value, descriptors; how to access and display value)



- 11 features are defined in the GATT Profile: Server Configuration, Primary Service Discovery, Relationship Discovery, Characteristic Discovery, Characteristic Descriptor Discovery, Reading a Characteristic Value, Writing a Characteristic Value, Notification of a Characteristic Value, Indication of a Characteristic Value, Reading a Characteristic Descriptor, Writing a Characteristic Descriptor

LoRaWAN

- long-range (kilometers), low power (lower than cellular), small amount of data, low cost, small size
- uplink is reliable, downlink is optional and unreliable (i.e. may not be ACKed)
- LoRa is new, second sourcing of ICs is an issue
- main elements: end-point, gateway, master
- net arch: star-of-stars topo: gateways are transparent bridges, relay messages, connected to net server via IP; end-devices use single-hop wireless comm to 1..n gateways, normally bidirectional, limited amount of messages, support multicast (OTA upgrade, mass message)
- lower bitrate signals travel longer distances; longer listening time per bit helps reduce noise
- end-dev - gateway comm: spreading factors (SFx), 0.3..50 kbps (tradeoff range/energy/data rate), use ADR (adaptive data rate) algo
- sec: net-level key, e2e app-level key, dev-key
- end-dev classes: A (battery-powered sensors; energy efficient, downlink only after Tx, 2 downlink windows, other downlink comm is queued), B (battery-powered actuators; slotted comm w/ beacon, bi-directional comm, scheduled Rx slots), C (mains-powered actuators; no downlink latency, listen continuously, Rx always open except when Tx)



LoRaWAN Limits

- https://arxiv.org/pdf/1607.08011.pdf

Energy

- CPU activity can be controlled by firmware; different activities (and peripherals) draw different amounts of current which impacts consumption
  - active: several milliA, idle (CPU inactive, clocks active): hundreds microA, power down: some microA or tens of nanoA
- $P = UI, E = PT = UT, E = \sum_i P_i \cdot T_i$
- energy consumption can be estimated/calculated; use datasheets ("typical" values)
- lower voltage results in less energy and slower
- internal RC oscillators: fast, not much energy, not precise; flash memory is programmable, limited, needs more energy than RAM and ROM
- current depends on temperature (rises exponentially after some point)
- e.g. CR2032 w/ 240 mAh @ 2.5 V; req: $500\mu J$ every 2 seconds. available energy is $(240 \cdot 1000 \cdot 3600 \cdot 2.5)\mu J$, energy needed per second $500/2 = 250\mu J$, time (in hours) is $(240 \cdot 1000 \cdot 3600 \cdot 2.5)/(250)/(24 \cdot 3600) = 100d$

- e.g. 9 ms @ 3 mA, 1 ms @ 10 mA, 9990 ms @ 1 uA using a CR2032 w/ 240 mAh @ 2.5 V: $(240 \cdot 1000 \cdot 3600 \cdot 2.5)/(0.1 \cdot 2.5(9 \cdot 3 + 1 \cdot 10 + 9.99 \cdot 1))/(24 \cdot 3600 \cdot 365) = 5.8\text{years}$; discharge current $0.274\mu A$ results in 5.53 years @ 1.8 V or 4.4 years @ 2.4 V
- or a bit simpler: $240\text{mAh}/(0.9\text{ms} \cdot 3\text{mA} + 0.1\text{ms} \cdot 10\text{mA} + 0.999\text{ms} \cdot 1\mu\text{A})$ and to not have to deal with lots of 10^{-3}, multiply the numerator by 10^3
- generalized: $(\text{mAh of battery} \cdot 1000 \cdot 3600 \cdot \text{voltage of battery})$ $/(\text{voltage of battery} \cdot \text{energy consumption in one second using} \sum_i P_i \cdot T_i)/(24 \cdot 3600) = \text{lifetime in days}$
- simplified: $(\text{mAh of battery} \cdot 1000)/(\text{energy consumption in one second in milli-seconds and -amperes using} \sum_i P_i \cdot T_i) = \text{lifetime in hours}$
- reality: battery leakage, self discharge current; important: lifetime shouldn't be longer than what manufacturer guarantees
- energy harvesting (EH) solves issues: mobility issue w/ mains power cable, battery leakage (possibly faster if warm/hot), battery energy drops over time, batteries can be empty, battery replacement can be expensive, battery explosion danger, ecological issue of batteries
- EH is not always available (e.g. solar, Seebeck temperature gradient) and is expensive at investment time
- EH sources: solar: outdoor/indoor differences, e.g. indoor light spectrum; LED/photodiode: need more light, work outdoors, takes more time, low-cost; piezo: generate energy from vibrations at proper frequency; electro-dynamic: uses movement and electro-magnetic field (e.g. shake flashlight); Seebeck: temperature difference, small voltage, high current, energy reduces as temperatures converge (use in e.g. heating system controls); RF sources: radar/radio/WiFi AP/smartphone, weak, unreliable (use in e.g. difficult-to-reach sensors, rotating systems)
- to get optimal EH, use maximum power point tracking: the load is constantly adapted to match the internal resistance of the cell that changes with the illumination
- power mgmt needed to manage energy efficiently; booster: step-up voltage; Buck converters: opposite; supervisor: monitor parameters and take action of value out-of-range

RFID

- short range, suitable for confidential information, very low cost (if passive), low energy (EH possible), lots of experience, decreasing costs
- Radio Frequency IDentification; read/write tags without contact; tag is low-cost, stores data; reader is expensive, reads out + write information from/to tag; shifted complexity to keep tag small and cheap
- reader generates field that also powers the tag
- classifications: 120-150 KHz low-freq tags, can cope with liquids, low data rate, e.g. animals; 13.56 MHz high-freq, fast transfer, e.g. E-Pass, access control, books; 860-960 ultra-high-freq, long distances, e.g. logistics
- classifications: passive tags (powered by reader, cheapest), active tags (tags has own transmitter + source), BAP (battey assisted passive; has power source but no transmitter), read-only, read/write
- RFID and active wireless combined for low power and fast reacting system: LF is active when system sleeps (needs little energy), the LF signal generates a wake-up in case of event, device wakes up and transmits information using UHF active radio
- RFID sensor networks e.g. temperature sensors (are goods still fresh during/after transport?)
- NFC: 13.56 MHz (HF), short-range, bidirectional, 26-848 kbps, p2p / r/w / card emulation, possible to be passive, fast data exchange (both between active->powered and devices/passive tags), good for "secret"/secure data, useful to set up complex wireless system in a secure + user friendly way

Wireless Security

omitted

Labs

8 bits = 1 byte = 2 hex digits

TI Sensortag

- events are used for sensors as they are not synchronous

Power measurements

- $I_{OUT} = V_{OUT}/Gain$
- battery lifetime calculations: $\text{capacity of battery in mAh}/\text{current draw in mAh} = \text{lifetime in hours}$

Sending and sniffing 802.15.4

- header is reversed bit order i.e. MSB is on the left within a section/group, e.g. frame type data is `001` where b2=0,b1=0,b0=1
- e.g. `buf[3] = IEEE802154_PANID & 0x00ff;  buf[4] = IEEE802154_PANID >> 8;`

Energy consumption of 802.15.4

- save energy by turning radio off when not in use
- sending larger packets uses more energy
- printf requires energy, too

LoWPAN

-

Bluetooth ADV

-

URI, encrypted data ADV

- send temperature in 0..2 order ("normal")
- us correct lengths for ADV data, payload length etc.

LoRaWAN I

- use `union` to store data as float and `unsigned char[]`

LoRaWAN II

-

Stuff

- ALOHA: If you have data to send, send the data; If, while you are transmitting data, you receive any data from another station, there has been a message collision. All transmitting stations will need to try resending "later".