Answers to the pdf "Questions for the oral exam"

from the Mobile & Cyber-Physical Systems 's course

(Stefano Chessa, 2019-2020)

These notes are what I've understood from its slides.

There may be errors.

There may be incompleteness.

But hopefully they'll be helpful.

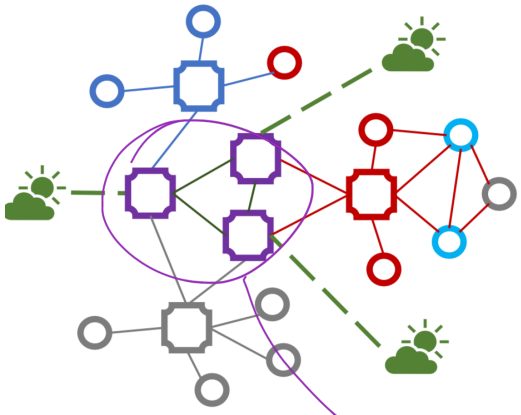
Giulio Purgatorio

To thank me, just

go star ⭐ → github.com/GPurgatorio/Master-Computer-Science-ICT !!

NOTES POST-ORALE: many APPLVLGATEWAYS per single-point of failure, exponential implosion in mapping, ...

1) interoperability



Different colors = different vendors

vendor lock-in = different standards

migrate to different Technologies is hard

introduce standards: 802.11, bluetooth, ZigBee, ...

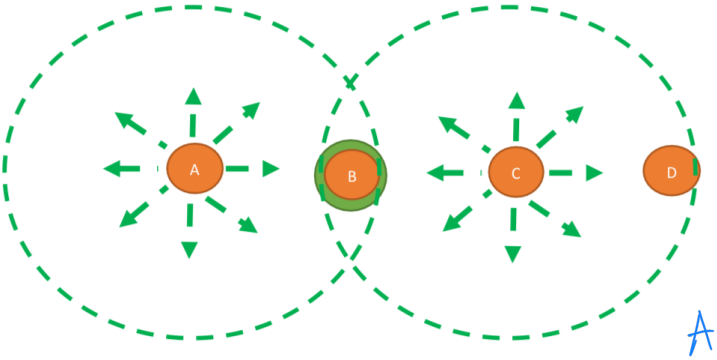
They need to communicate with each other

APPLICATION LEVEL GATEWAYS

map different protocols to allow communication between them

Extra: At the beginning it was an HW problem (IBM, Microsoft, ...), then abstract → SW so standards are LOCAL. Like in publish-subscribe there's MQTT, Celery, ...

2) hidden terminal



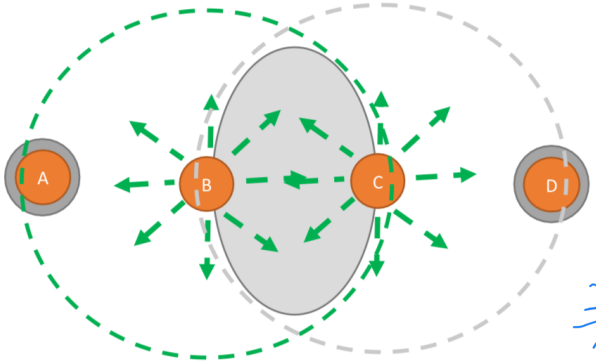
In Wireless Networks there's a challenge: mobile hosts have LIMITED RANGES and resources. Since a node can't hear all the others we introduce the Hidden Terminal (and exposed (3))

A wants to transmit to B

C wants to transmit to B

⇒ C is HIDDEN with respect to the transmission AB

3) exposed terminal



Similar to 4 but introducing the INTERFERENCE ZONE

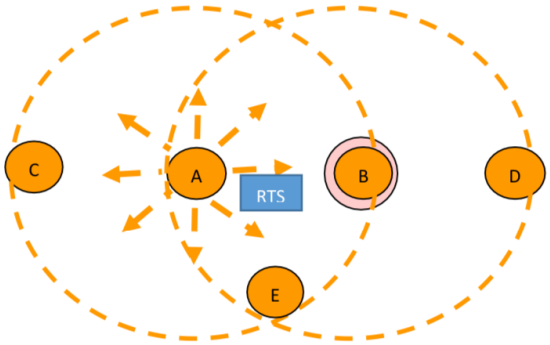
B wants to communicate with A

C wants to communicate with D

C senses the medium but B's Transmitting, so ...

⇒ C is EXPOSED w.r.t. the transmission BA

4) RTS/CTS



Context: can't use CSMA/CD Technique because it senses the problem at the sender and not at the receiver

⇒ MACA PROTOCOL

Send short frames (others will wait), then long ones

↑ Binary exponential Backoff

RTS — Request To Send — contains length of the frame that will be transmitted

CTS — Clear To Send

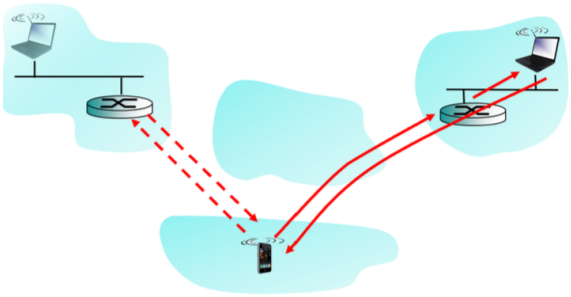
solves hidden/exposed Terminal!

Consider AB

- D is hidden (hears CTS but no RTS ≡ there's another transmission!)
- E same for D (hears both, so he'll wait too)
- C is exposed (hears RTS but no CTS ≡ he can transmit)

If both B ∧ C RTS (A), A sees the collision ⇒ won't generate CTS ⇒ timeout, retry

5) Mobile networks



DIRECT ROUTING

can always be used to reach mobile

Actors (from left to right)

- Home Network permanent home of mobile
- Home Agent entity that will perform mobility functions on behalf of (remote) mobile
- Correspondent
- Foreign Agent
- Visited Network another permanent address and a CARE OF ADDRESS

Correspondent request → Home Agent → reply → Foreign Address

Correspondent forward → Foreign Agent forward → Visited Network

it overcomes the triangle routing problem

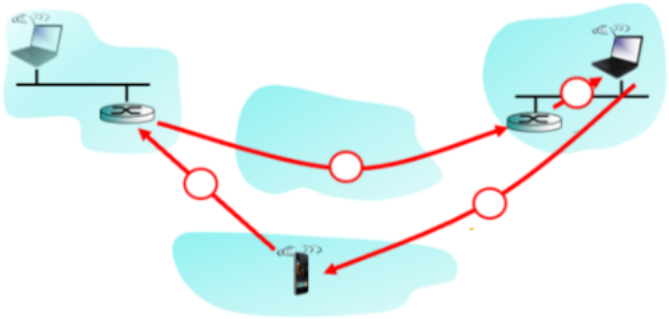
correspondent & mobile ∈ same network

↑ replies directly without FA

But it's non-transparent to correspondent because it must get the care-of-address from Home Agent when we consider MOBILITY

the first Foreign Agent will act as an ANCHOR to the new one

≡ CHAINING cow → FA → directly → new Fnetwork



INDIRECT ROUTING

correspondent → port → Home Net

Home Agent → forward → Foreign Agent → Visited Netw

Then replies are direct like in Fw Direct Routing

⇒ TRIANGLE ROUTING PROBLEM

communication follows the pattern

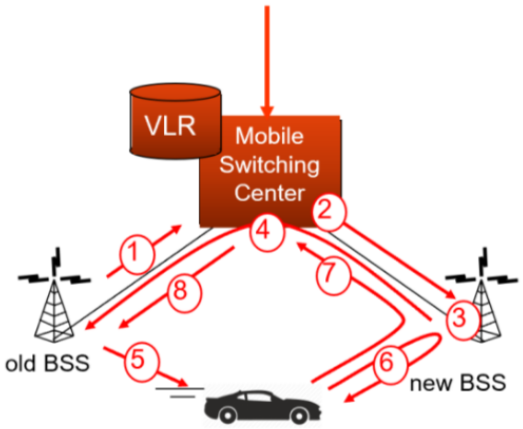
corresp → home net → mobile

inefficient when corresp & mobile are in the same net!

There are 2 addresses: ① Permanent used by correspondent

② Care-of used by Home Agent to send pkts to FA

7) Mobile networks



GSM

intro host is moving so it must change its base station

goal: change it without interrupting the communication

reason: stronger signal, load balancing

How? ① The OLD BASE STATION tells TO the MSC that he wants TO HANDOFF

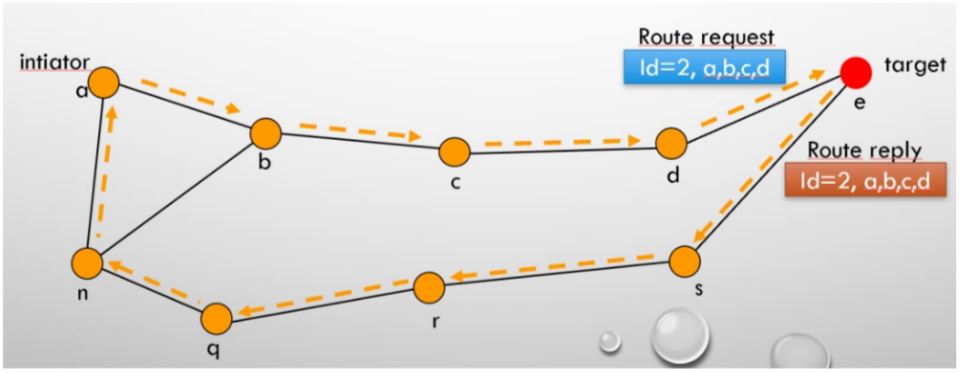
② MSC allocates resources for the new base station.
③ New BS allocates a radio channel for mobile's usage
④ NewBS signals TO MSC that he's ready
⑤ OldBS Tells mobile to perform handshake with newBS
⑥ Mobile does it: new canal is created
⑦ Mobile (through newBS) signals MSC that handoff is done, so that MSC can reroute through newBS
⑧ old BS 's resources are released

Handoff between MSC anchors here Too!

NOTE: handoff done by the anchor

Data will always be sent to Home MSC → Anchor → Anchor → ... until we find the MSC That can directly communicate with correspondant

8) DSR



AD HOC NETWORKS

Intro: wireless networks without infrastructure (no centralization, must support • load balancing
• low overhead
• async communication

Routing may be done in 3 ways:

① Proactive: maintain updated infos on all nodes

② Reactive: Find the route only when needed
DSR, AODV, DYMO

③ Hybrid

Routing Algos → GRAPHS
nodes (hosts)
arcs (communic between hosts)
Links are symmetric

Dynamic Source Routing

goals ① low overhead ② no centralization
③ Fast reactions to Topology changes

Assumptions ① nodes cooperate (forward packets to others)
② network has a small diameter (5-10 hops)
③ speed is moderate with respect to the transmission's range
a.k.a. A and B two close nodes. If B moves but is still in A's range, there's no Topology change
"changes aren't frequent & do not occur during route discovery"

How

Route Discovery Source wants to send a packet to Dest

- First search in its route cache
  if it's there, use that path ✓
- O/w start the protocol: send RREQ to all nodes in reach.
  RREQ contains • ID's receiver
  • ID's sender
  • ID's request
  • List of passed nodes
- A receiving node ≠ Dest
  → check if he's in the list already (or same ID req already received)
  → if not, it puts itself in it and broadcast
  if yes discard it
- Packet reached Dest

Before sending the reply, examine its route cache: if links are bidirectional and there's something, use that to reply

OTHERWISE (he knows no routes) he can reverse the root or start its own route discovery combining it with RREP

(contains the accumulated list)

- When RREP reaches Source, Source puts the route in its cache to reuse it if needed.

If there are packets we can't send at the moment due to the lack of a route → BUFFER (FIFO, timeouts) & route discovery

How Route Maintainance

when sending stuff, if there's a broken link, send to sender RERR. Sender will remove that route from its cache and all the routes that contain that link.
Error is reported to the upper layer to tell to retransmit
Try another valid route or route discovery again

IN THE PICTURE

NO BIDIRECTIONAL LINKS (see route reply)

E can receive many RREQs from many directions
→ choose the best one (latency, #hops, energy levels, ...)

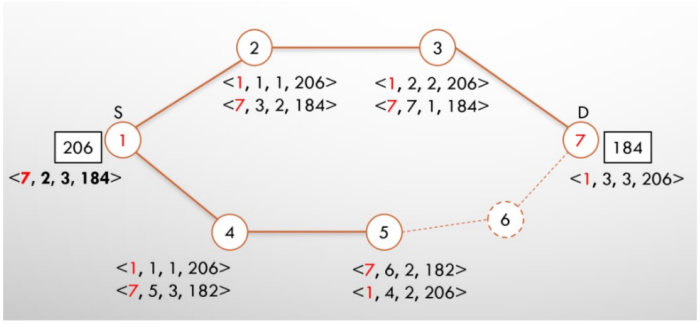
Improvements

① instead of simply caching the result of route discovery, cache the result of other RREQs that pass through
② if a node knows a valid path, return that
③ To avoid a storm of RREPs introduce a random timeout

$delay = L * (n - 1 + rand(0,1))$

L = constant
n = # hops of the Route

④ introduce TTLs



AODV

Ad hoc on Demand Distance Vector

same goals of DSR + uni/multi/broadcast

same assumptions + bidirectional links

Intermediary nodes store info only for source & destination

Again RREQ, RREP, RERR, but maintain 2 Routing Tables (uni & multicast)

AT MOST a route for each destination and for each destination we maintain only the next (and previous) hop.

Each entry has a LIFETIME

Each node has a sequence number (multicast group has one too)

How ROUTE DISCOVERY S → D, again check for RT's valid route (next hop!)

else route discovery:

→ broadcast RREQ
setting a timer ← contains:
- IP's sender
- S's sequence number
- IP's receiver
- last known sequence number
- broadcast ID
- Hop Count (=0)

↑ incremented when new RREQ [∧RREP]

→ receive RREQ:
check if already received
if not, PROCESS IT ≡ "sets up a reverse route entry for the source node in its routing table A,B,(C) → C:B → A. This way it's possible to RREP to the correct node"

→ if a node can't answer
INCREMENT HOPCOUNT and broadcast (retry if lost)

A valid answer must be an entry in RT with valid lifetime AND a sequence number ≥ last known sequence number of the receiver (TO PREVENT LOOPS)

AODV's flooding may be expensive (TTLs)

"FORWARD PATH ENTRY"

When receiving a RREP, create a path towards the destination

many may be received, forward the one with higher dest's seq number

S will use the first one and use next ones to adjust the RT

How ROUTE MAINTANANCE Send HELLO PACKETS To check neighbors

if they don't answer, remove them ← may trigger updates
↑ TTL=1

Hello Packets show broken links too: if they moved, just RouteDisc and you receive RERR ← send it back to source and everybody will know
send to all predecessors ↑ contains a list of many unavailable dests!

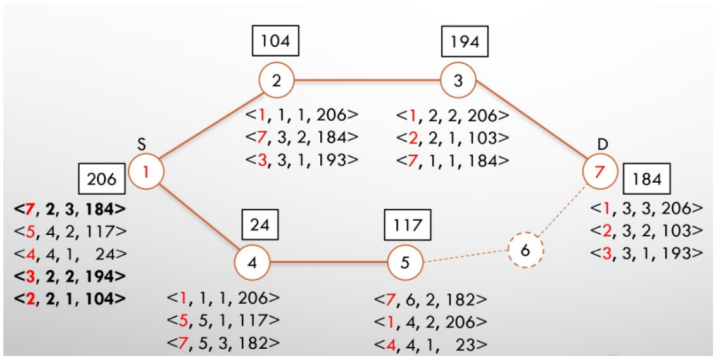
When source receives RERR, it will start a new route discovery.

ABOUT MULTICAST Every group has a leader and a bidir multicast tree.
Routing is made through the tree, starting from the first node receiving the msg
group's seq number is stored by the leader (node can join/leave anytime)
the MULTICAST ROUTE DISCOVERY starts when a node wants to join
or send data to the group → RREQ (with JOIN flag)
⇒ if a member receives RREQ it may answer, otherwise it's a relay node ↓ broadcast
if it's seqnum > the one in RREQ
When many RREPs reaches S ⇒ Leaders always answers!
choose the one with higher seqnum AND lower hop count
Then S sends in multicast MACT (multicast activation msg), activate the entry of the chosen entry
Leaving is a msg to the parent. NON-LEAF nodes will act as relay!
Routerant in Multicast is done asap, through Hello Msg.
If two parts can't reconnect (FURTHER NODE's RREQ) ⇒ PARTITION
elect a new leader
if it's a member → leader
else select one
Leader send periodically GroupHello ⇐
if heard by a leader = there's a path!
↳ lowest IP address sends

Compare AODV / DSR
- multicast support
- only next/prec
- only one route per dest
- both good but if traffic / mobility much
  - AODV control packets ↑
  - DSR save tons of multiple path

IMAGE FORMAT
<Dest, Next, Hops, Seq>

10) DYMO



Dynamic Manet On Demand Routing

goal combine advantages of DSR ∧ AODV
same assumptions of AODV

Route Disc ∧ Route Maintanance same thing but no HelloPackets
⇒ use timers

Takes idea from DSR, storing info not only of source and dest)
≡ RREP ∧ RREQ create entries in intermediary nodes ← each node appends itself to the route and updates its RT

An entry of the RT contains • destination address • seqNum • nextHop
- Route Broken = TRUE if nextHop is unreachable
- [optional] HopCount
- prefix (indicating if address is a network or host address)
- route forwarding flag (if it can be used to forward msg)

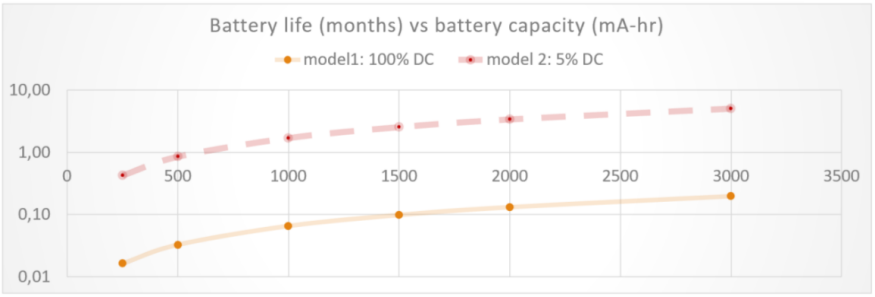
Timers are
ROUTE_AGE_MIN minimum time
ROUTE_SEC_NUM_AGE_MAX to discard
ROUTE_USED → ROUTE_USED_TIMEOUT when used
ROUTE_DELETE → ROUTE_DELETE_TIMEOUT when broken, then remove it

SeqNum increases as in AODV + intermediary node adds info to routing packet
↯ increment ∀ new RREQ ∧ RREP

when a node reboots he can't set its seqNum = 0 ⇒ LOOPS
→ store it or wait ROUTE_DELETE_TIMEOUT before participating
in this time it will only handle control msgs but can't forward packets
↑ RERR

FORMAT
<dest, next, metric, seq>
↑
hop count?

11) duty cycle



DUTY CYCLE

Idea save energy by reducing the period of activity of a sensor.

Alternate activity and inactivity period

Energy Cost

$$E_\mu = C_\mu^{full} * dc_\mu + C_\mu^{idle} * (1 - dc_\mu)$$

active cost * duty cycle + inactive cost * (1 - duty cycle)

Radio

$$E_\rho = C_\rho^{T} * dc_\rho^{T} + C_\rho^{R} * dc_\rho^{R} + C_\rho^{idle} * (1 - dc_\rho^{T} - dc_\rho^{R})$$

transmission cost * duty cycle Transmission + received radio signal cost * DC receiving radio + inactive cost * (1 - the two DCs)

$$Lifetime = \frac{B_0 - L}{E}$$

B_0 = initial charge L = lost battery at each cycle

E = total energy

Battery capacity at cycle n

$$B_n = B_0 * (1 - \varepsilon)^{n-1} + \frac{E((1-\varepsilon)^n - 1)}{L}$$

IMAGE

Depending on the duty cycle, the lower → the higher battery saving.

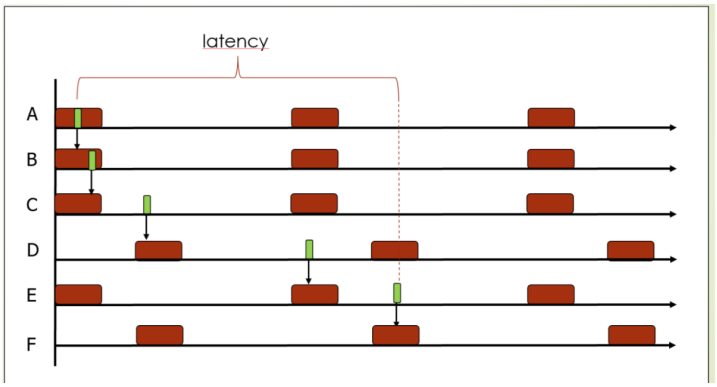
We can see that the sensor's activity is extremely repetitive because stopping for 95% of the time the graph follows the same curve, suggesting to not keep everything active all the time.

Better energy efficiency → LOWER DUTY CYCLE

1. Turn off the processor LOCAL DECISION
2. stop radio signal INVOLVES NETWORK, you don't contribute anymore

in WSNs are implemented strategies like sensor synchronization and turning off the radio signal when not needed

12) SMAC



Intro for energy efficiency and arbitrations there are 3 ways:
① synchronization of nodes through S-MAC
② preamble sampling (B-MAC)
③ polling (802.15.4)

SMAC synchronizing nodes, they can be turned on simultaneously
⇒ low duty cycle, inactive most of the time
"there's no network

LOCAL SYNCHRONIZATION

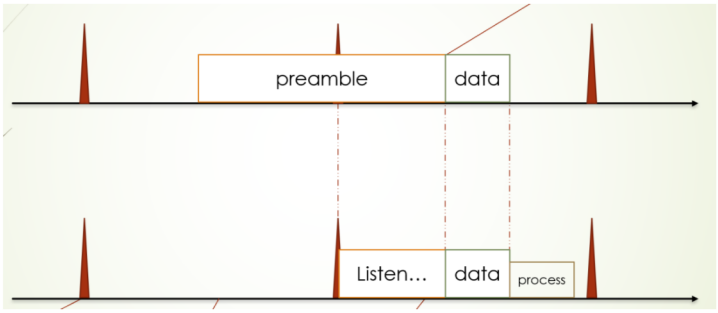
Node alternates listen and sleep periods
adjacent sensors synchronize their listen period through SYNC periodical packets
- if sensing someone, use same period
- o/w choose one

} in any case, advertise neighbors

In SMAC, sense the channel before transmitting: if channel is busy, transmit in the next period

The image(↑) is about latency in a multihop path
packet will have to wait (worst case) for the listen period of each intermediate

So the latency for the green msg sent from A to F
A, B, C share the same listen period → send istantly
D, F different one, C will wait for D



BMAC

A sender may send anytime because the packet has a very LONG PREAMBLE in the header.

The listener will periodically check for preambles (preamble sampling)

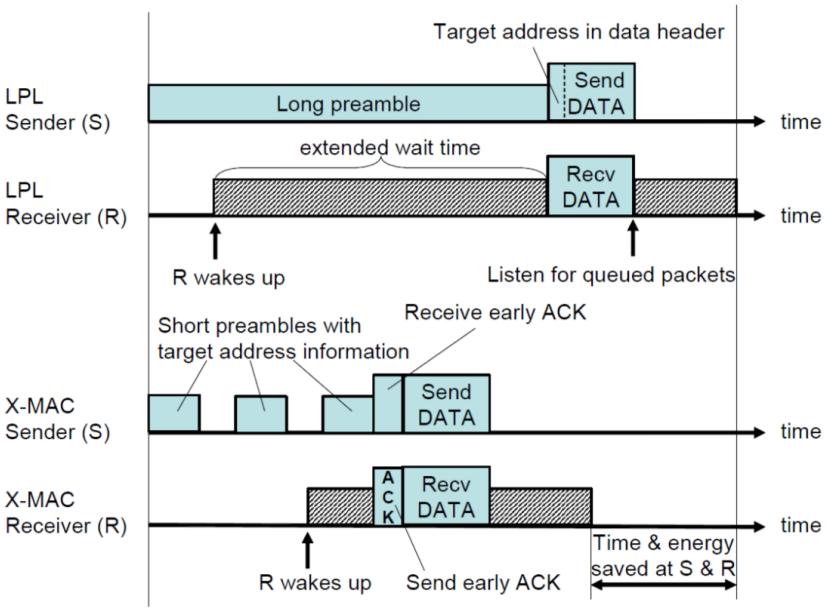
If there's one, the sensor will maintain radio on

Idea: spend more in transmission, save energy in reception

Preamble must be longer than the sleep period!

PRO
- it's not a network organization protocol
- transparent to higher layers

CONS
- preamble sampling's cost isn't negligible
- may cost more than synchronizing



XMAC

evolution of BMAC

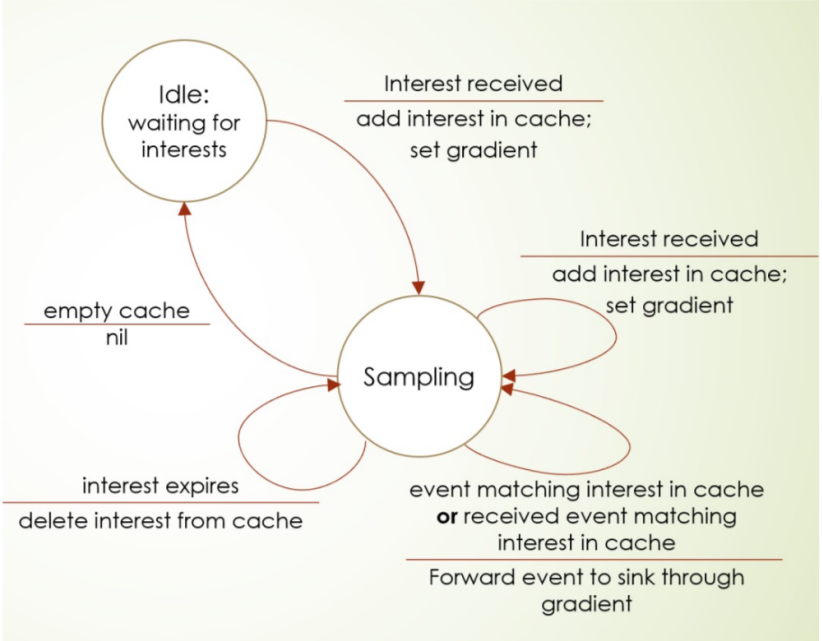
↑ reduce the impact of long preambles

contains the receiver's ID
can check if it's for him

Receiver can interrupt the preamble

Image After noticing that the preamble is for R,
R sends an early ACK and can then receive data
=> that left time is saved in comparison of long preambles

15) Directed Diffusion



NETWORK PROTOCOLS

Intro sensor networks are DATA-CENTRIC and which is attribute-based → ADDRESS-BASED

⇓

energy efficiency is a key factor. Usual protocols won't work because Routing Tables would be too big (same for headers/put dim!)

This causes:

① **Implosion problem**
consume unnecessary resources to send/receive already sent/received data

② **overlap problem**
two sensors may cover part of the same area

[post oral exam: gradient is next hop!]

Image DIRECTED DIFFUSION

all communications are for named data ← using attribute-value pairs

The SINK disseminates a sensing task for a certain named data ≡ INTEREST ← periodically *

This insertion sets up a metric → Gradient

All the data matching the interest will flow towards the sink (multiple paths)

(one (or some) may be reinforced by the sink)

* 1st broadcast is for exploration, next for interest's refreshes (no reliability) ≡ spam

Nodes cache and forward received interests
have a lifetime ↓ ↳ aggregate them on same attributes
- timestamp
- area
- ...

The gradient is associated to a DIRECTION and to a DATA RATE used to route data matching the interest towards the sink.

The image (↑) is the state machine

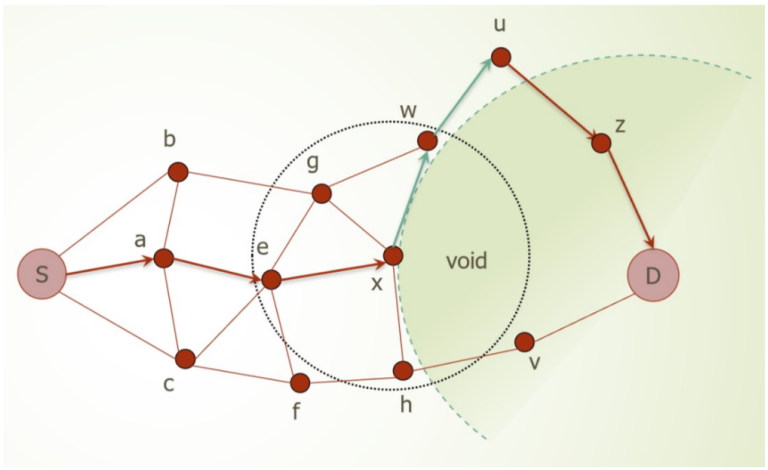
Idle receive an interest → add interest in cache and set gradient (+ lifetime) → Sampling State

if cache's empty ↩

can still receive interests
- if it matches a cached one forward it to the sink through the gradient

Note DirDif is simple and scalable
It can be used on low complexity data (can be aggregated)

BUT nodes closer to sink consume the most of energy because all data goes through them



Greedy Perimeter Stateless Routing

ASSUMPTIONS

- nodes are put in a 2D space
- nodes know their positioning
- nodes know neighbors positioning
- source knows coordinates of dest (specified in the header)

⇓ simple, scalable

How Greedy Forwarding

A ⟶ B, for next node, choose the neighbor closest TO destination

How Perimeter Forwarding

Right (or Left) Hand Rule. Return to greedy if you find a node closer to destination with respect to the node when you changed mode.

"Primo antiorario rispetto a dove arrivi"

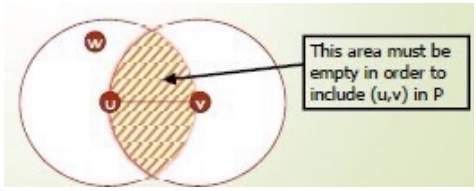
(in the image: U returns to greedy because Z is closer than X to D)

⚠ doesn't work if the graph is non planar (≡ arcs may cross)

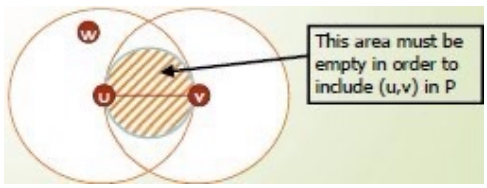
⇒ Planar graph P obtained from G ① G connected ⇒ P connected
② P = G \ {edges}
③ P computed through a distributed alg.

by either

- Relative Neighborhood Graph

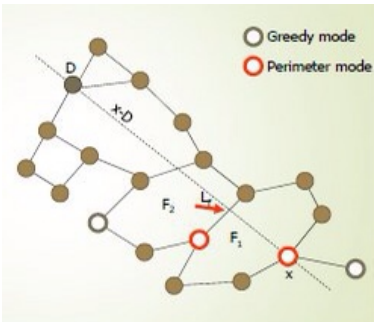


- Gabriel Graph



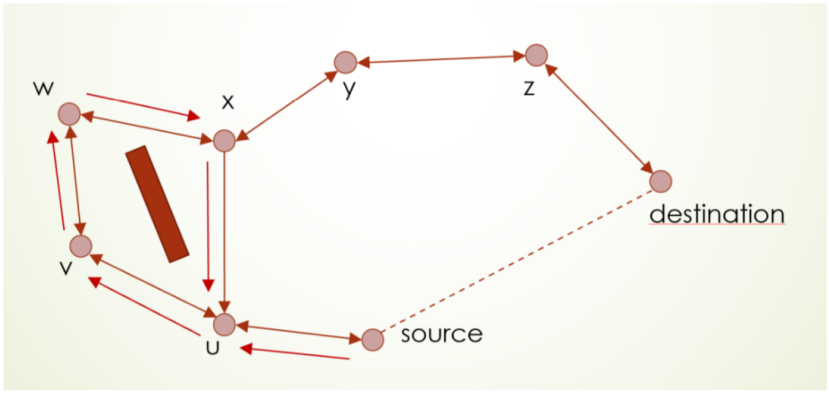
⇓

A planar graph has 2 faces
- INTERIOR delimited by arcs
- EXTERIOR outside, unbounded



Perimeter Mode ⇒ RHR to reach to an arc that crosses x-D.

GPSR needs updated infos about neighbors and to planarize the graph at each topology change ← impossible ⇒ proactive approach
periodically communicate position to neighbors. Excessive changes in Topology → FORCE PLANARIZATION



Non-bidirectional links cause failures in GPSR

Source —perimeter mode→ u → v

x ← w

x remains in Perimeter Mode because it's farther than u (where the mode switch happened), so instead of going to y it returns to u LOOP

To solve this GPSR uses MUTUAL WITNESS

extends the planarization alg of GPSR making all links bidirectional

IT MAY NOT DETECT CROSS LINKS ∴ non planar

⇓

CROSS LINK DETECTION PROTOCOL (CLDP)

Remove the links that cross

How? each node sends a probe to all its outgoing links
↳ goes on with RHR

each node controls the coordinates of the traversed node

⇒ if there's a links that cross an already existing link

remove either the crossed or the crossing link

↑ may result in network disconnection

⇒ add a PROBE that counts the number of times it crosses a link

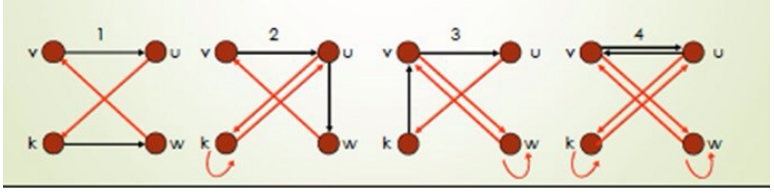
- #times = 1 ⇒ can be removed
- else

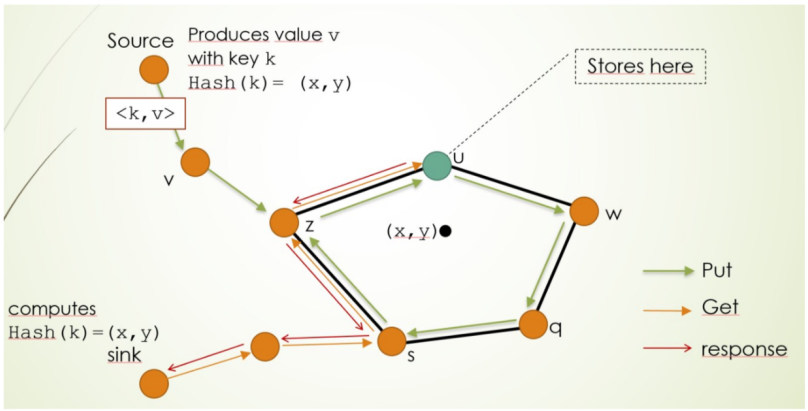
① uk

② uv

③ uk

④ ???





DCS – GHT

Data Centric Storage – Geographic Hash Table ← like P2PBC!

events are indicated by keys and values are stored through names in the network.

Queries are made by a key towards the nodes that store that key

2 operations

- PUT calculates hash(k) and returns coordinates (x,y) then send <k,v> to (x,y) with GPSR
  - <k,v> stored in the closest node
  - !!! every node knows neighbors positioning
- GET calculates hash(k) again and sends a request to that point using GPSR

Due to mobility/node failures, something may be unavailable

⇒ Perimeter Refresh Protocol (PRP)

Select a HOME NODE for the key k

Replicate <k,v> in all nodes in the perimeter around (x,y)

Home Node creates REFRESH packets to preserve consistency
{ contains <k,v>

takeover timeout

If Home Node moves and Refresh Pkt reaches a node closer to (x,y) than Home Node → eligible as new Home Node

→ if this happens, new Home Node will generate a new Refresh Pkt.

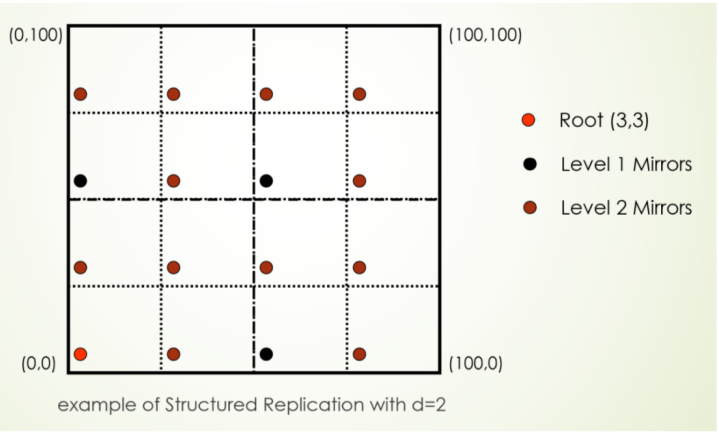
Refresh Pkt resets timeout

But any Home Node may fail, so when the timeout expires replicas will generate Refresh Pkts too

There are also lifetime timers

DEATH_TIMEOUT > refresh, takeover timeout

19) DCS-GHT



example of Structured Replication with d=2

DCS-GHT Structured Replication

A node may be overburdened if too many values for K are produced

⇒ use a hierarchy

hash(K) is the hierarchy's root

For each key K there are associated:

- a root
- $4^{depth} - 1$ mirrors

A node stores the pair <K,V> in the closest mirror of hash(K)

⇒ the GET queries the root and (possibly) all mirrors

In the image↑

an example of StrRepl with depth d=2 (level 1 and 2)

queries will reach root (3,3) which will forward the query to ALL level 1 mirrors (3 in total) that will forward to level 2s.

"each mirror will forward to descendent mirrors"

⇒ TRADEOFF storage was the problem, solved ✓

but communication overhead

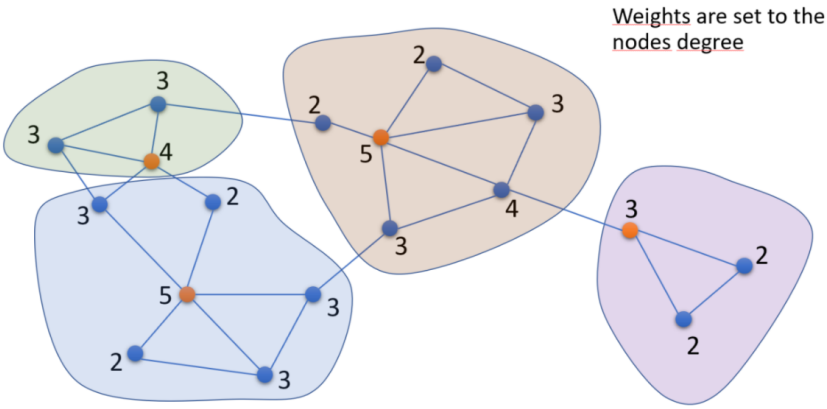
Furthermore, you have no more control over the data replication degree

possible improvements: better load balancing

scalability

use labels instead of coordinates

TODO physical and virtual coordinates



DMAC

Intro → to create a hierarchy

clustering in WSN → Cluster Head, subgroups

This way nodes that necesitates routing's capacity are the Heads only (they will forward to dest node)

⇒ physical and logical links

⇒ K-neighbor-clustering

limit the distance CH↔nodes to (K) hops — SMALL (1,2,3)

Nodes that allow communications between different clusters → gateways

Overlaps with phys or it's a path of phys links, length H

they have the phys link!

To choose the CH ≡ find the Dominating Set

which is the set of nodes for a 1-neighbor-clustering

to get a ⇒ CONNECTED dominating set

a DomSet where K=1 and H=1

finally ⇒ MINIMUM ConnDomSet (NP-HARD) → use DMAC

DISTRIBUTED AND MOBILITY ADAPTIVE CLUSTERING

Clustering alg where every node has a weight (many metrics) which represent its willingness to become CH

Each node has an ID and a status {CH, OrdinaryNode, Undecided Node} ← starting one

How ① each node sends an HELLO msg

② After receiving all of them, each node decides its status on a DECISION RULE

if the weight is the max among its CH and UN → become CH

⇒ all neighbors become ON

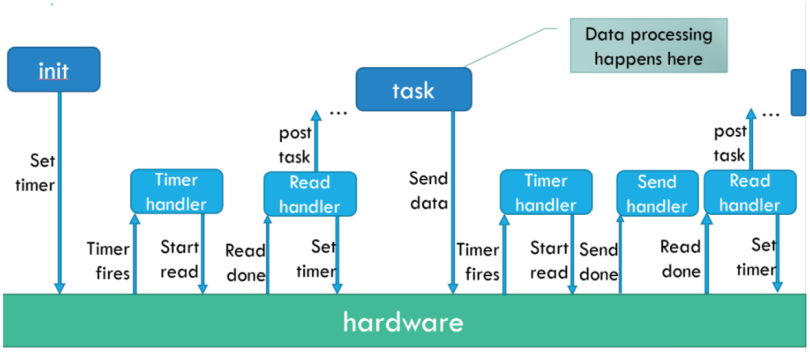
③ An ON decides its cluster based on the ASSOCIATION RULE

≡ associate to the max weight neighbor's CH ≡ every node is either CH or distance=1 from CH

not a MINIMUM DomSet, but $2 \leq H \leq 3$

EXTRA JOIN • start with UN → • send Hello msg → • neighbors will answer → → • ON if a neighbor is CH, o/w new cluster and become its CH. JOIN may alterate metrics, possible (costly) reconfigurations (just do them periodically)

21) embedded programming



An embedded system is a system designed and built to solve one (or a few) specific task, based on microcontrollers

Its job is CYCLIC, through low level code (system calls) that interacts with Hardware with COMMANDS and INTERRUPTS to:

- reading from transducers
- controlling actuators
- taking a decision
- (optional) communicate with other devices

The memory is very small, 2 strategies:

① Event Loop (Arduino)

Single thread that repeats its loop without ever suspending the thread
If you need to "wait", use sleep() in the code

② Event-based programming (TinyOS)

Through COMMANDS, EVENTS (abstractions of interrupts) and TASKS
interrupts are handled as soon as they arrive by an event handler
tasks can be activated by events

Tasks are used for long computations and are implemented with a FIFO queue ≡ executed sequentially

In the image (↑)

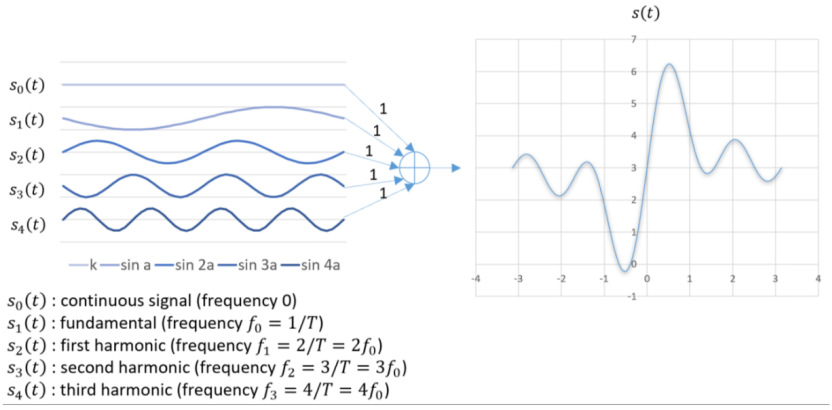
Init sets the timer, the timer fires (event) → handled to reach a read Handler that generates a task and repeat

process data here

The task will send data when it's ready

Fourier series

$$\frac{1}{2}a_0 + \sum_{n=1}^{\infty}(a_n \cos nt + b_n \sin nt)$$



$s_0(t)$: continuous signal (frequency 0)
$s_1(t)$: fundamental (frequency $f_0 = 1/T$)
$s_2(t)$: first harmonic (frequency $f_1 = 2/T = 2f_0$)
$s_3(t)$: second harmonic (frequency $f_2 = 3/T = 3f_0$)
$s_4(t)$: third harmonic (frequency $f_3 = 4/T = 4f_0$)

The Fourier's serie decomposes a signal as the sum of an infinite number of continuos functions, oscillating at different frequencies.

There's a change of coordinates: domain changes from time_domain to frequence domain

These functions define the base of decomposition that for the Fourier's serie is given through trigonometric functions

About the formula

$\frac{1}{2}a_0$ is the CONTINUOS SIGNAL

a_n and b_n are the ARMONICS

cos (/sin) nt are the AMPLITUDE OF THE ARMONICS

"Given a continuos signal, it's periodic in $-\pi \dots \pi$

$s_0(t)$ is flat

$s_1(t)$ is given by $\frac{1}{T}$, T signal period

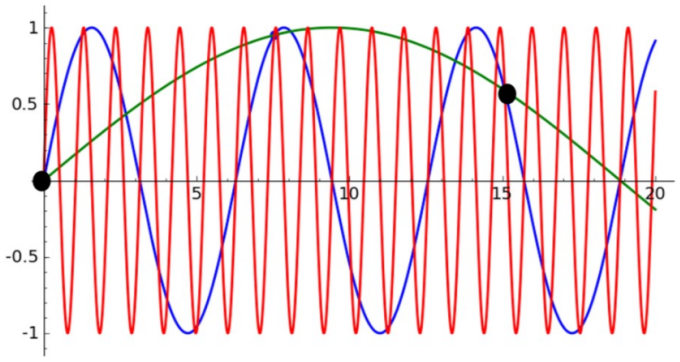
$s_{2/3/4}$ is $s_1 * 2/3/4$

summing these with weigth = 1 we get the img on the right

summing to ∞

it becomes a sinusoid

22) sampling and aliasing



convert a signal from ANALOGIC ↓ DIGITAL

means that we take the analogic signal and we transform it into a sequence of integer numbers that can be transmitted by a digital device.

How ① sampling

take the signal and extract a sequence of values that it assumes in discrete interval of times and return this stream of numbers

② quantization

take that stream and transform it in numbers that can be managed by a digital device

⇒ LOSS OF INFORMATION

Sampling doesn't imply loss of info under some hypothesis

Sampling Theorem frequence of Nyquist

2 times the max freq of the signal

↑ under this condition, you can reverse the process through INTERPOLATION

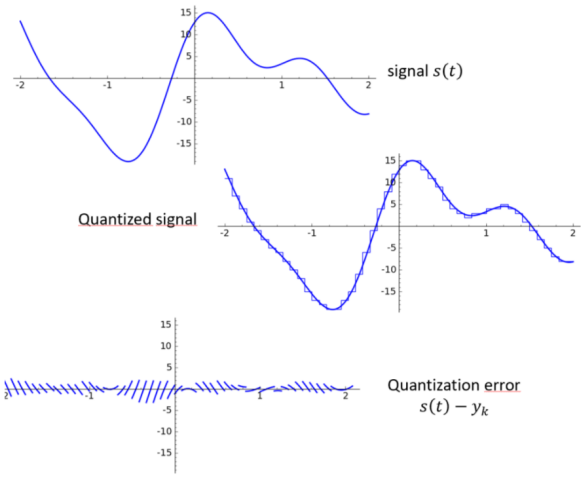
The image is about ALIASING.

When we don't respect the Sampling Theorem, we can't distinguish different signals once sampled (img: 3 different signals pass through the 2 same points)

Extra

the Sampling Theorem has an hypothesis, that the function to sample must have the Fourier's transf. equal to 0 outside of a certain range of frequencies

23) quantization



Numbers that come from sampling are real numbers, but these are usually expensive for Arduino, ...

⇒ Quantization approximate the value

$[0, 2^r-1]$

r is the number of bit per sample
quantization rate

This means there will be errors with respect to the sample

furthermore • OVERLOAD PROBLEM

signals higher than 2^r-1 ALL will be MAX value

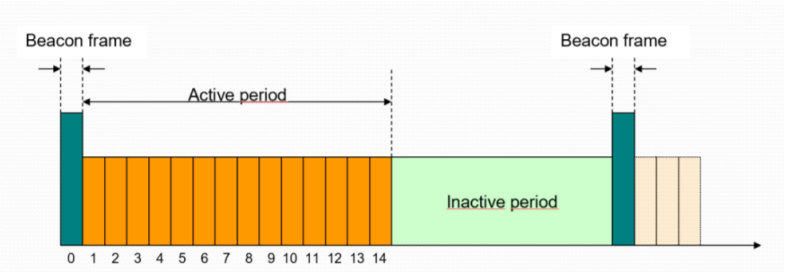
like overflow and underflow

• GRANULARITY

when we represent a series of values in the x-axys with the same value on the y-axys

Image take the signal, divide it in intervals and transform it between $[0, 2^r-1]$ and get the digital version of it (lower graph)
3rd grafe is the committed error

24) IEEE 802.15.4



IEEE 802.15.4 is the specification of the physical and MAC layers for low rate wireless Personal Area Network (PAN)
- infrastructureless
- short-range

⓪ security mechanisms

① PHYSICAL LAYER

Data Service
receiving and Transmission of Physical Protocol Data Unit (PPDU)
used to report the result of transmissions to the upper layer

Management Service
→ [de]activation of the radio transceiver
→ management
- energy detection used to discover a free channel and to do carrier sensing
- link quality indicator used in multi-hop routing, it indicates the quality of received packets. Based on Energy Detection and Signal To Noise Ratio ← SNR how efficiently you can understand a signal from a noise
Estimated value is forwarded To the network
- channel assessment to understand if a channel is busy, in 3 ways:
① using energy detection ② using carrier sense ③ combining ① and ② with AND/OR

② MAC LAYER

Data Service used To Transmit and receive MAC Protocol Data Units (MPDU)
Management Service synchronization, channel access, time-slots management and device association

In this layer, there are 2 types of devices...
① Reduced Function Device (RFD), they implement a reduced set of MAC's layer functionalities and have small capabilities
② Full Function Device use full MAC layer and coordinate a set of RFDs possibly the PAN Too
... and two types of Topologies: Star, P2P

continues ↓

Image for the channel access there are two possibilities:

WITH SUPERFRAME (Star Topology) or without (P2P)

Time is divided in slots $Slot_0$ is the Beacon Frame
sent by PAN coordinator

superframe has an ACTIVE and an INACTIVE period

in the ACTIVE all the communications happen "wow rly"

up to 16 slots

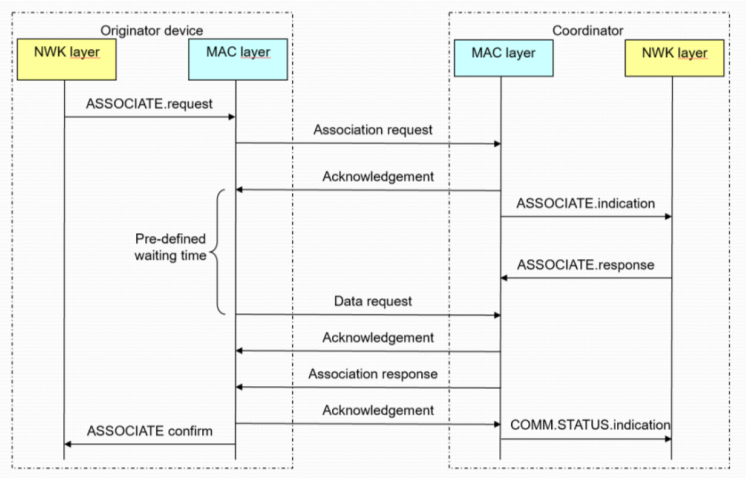
- Content Access Period (up to 15)
  in these, devices try to acquire the channel via CSMA-CA
  if acquired, it's maintained until the end of frame
- Contention Free Period
  optional, used in low latency applications
  divided in Guaranteed Time slots (assigned by PAN coord)
  it can't occupy all time slots because CAP
  is used for network maintainance

Between two consecutive transmissions a node must wait
an Inter-Frame Spacing (wide Frame → long IFS, short → short IFS)

Extra

without superframe, there are no beacons nor slots
Coordinator is the one that receives and
forwards data to end device, which do polling

25) IEEE 802.15.4



for the DATA Transfer, 3 ways:

- End Device → Coordinator
- Coordinator → End Device
- P2P

all 3 with or without Beacon

WITH Beacon

- ED→C C will send Beacon. ED will check if its GTS and will use it else it will try to guarantee it itself with CSMA-CA
  coordinator may send an optional ACK
- C→ED ED sleeps and sometimes waves up to poll during a CAP slot
  C will send istantly an ACK
  in one of the next CAP slots it will receive the DATA
  ED will answer with ACK so C can remove the msg from the list
- P2P ED can't communicate between themselves.
  If it's the case of one of the previous ones, use these ↑
  Otherwise it's a C→C (two routers or a coordinator and a router)
  Sender will first synchronize with dest's beacon (like an ED)
  ← how isn't specified in the standard

WITHOUT Beacon

- ED→C ED will Transmit data to C. C will ACK
- C→ED ED polls through CSMA-CA C will ACK and will Transmit pending msgs if present, otherwise an empty msg
  ED will ACK so C can remove msgs from its list
- P2P each device can communicate with any other in-range device ⇒ they must be synchronized
  not specified again

Image there are 4 primitives in the MAC layer

DATA.REQUEST invoked by upper layer to send a msg to another device

- CONFIRM returns the result of a REQ transmission to the upper layer
- INDICATION used by MAC layer when it receives a msg from the physical layer to pass it to the upper layer
- RESPONSE returns the result of an IND request

ASSOCIATION PROTOCOL

Invoked by an ED when it wants to associate with a PAN that has already been identified by a preliminary execution of the SCAN service (with Beacon)

ASSOCIATE.REQUEST

takes some parameters like PAN identifier, coordinator's address,- and sends an association request msg to the coordinator
→ ACK is sent

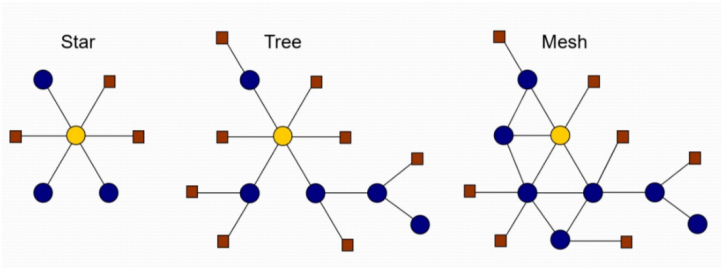
From the coordinator's side, the message is sent to the NETWORK layer with ASSOCIATE.INDICATION

If the request is accepted, the network layer generates a 16-bit address and it inserts it into ASSOCIATE.RESPONSE
↑ will be used to identify the joining ED

Then after some pre-defined time, it'll start all the DATA.* part

ED will make a request to the coordinator
C will ACK and then ASSOCIATION.RESPONSE
ED will ACK and respond with ASSOCIATION.CONFIRM to the network layer
C answers with COMM.STATUS.INDICATION that indicates that the association is concluded (success or error code)

Extra security layer based on symmetric keys given by upper layer. encryption of data and keys. Frame's integrity is calculated
Devices have an access control list



Network layer – topologies

3 types of devices
- coordinator of the network } FFD
- router
- end device — RFD

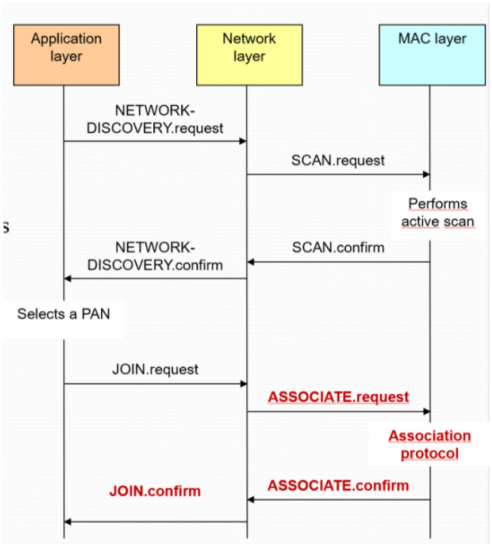
Star the same as 812.15.4, uses the super-frame.
Coordinator and other all around him

Tree can use the super-frame.
Coordinator and both routers/EDs connected to C or Rs

Mesh doesn't use super-frame.
"All routers are connected to each other"
in their range!

P2P fashion

27) ZigBee



Before being able to communicate in the network, a ZigBee device must form a network or join an existing one.

extra network formation

starts with NETWORK-FORMATION.REQUEST invoked by the device that'll become coordinator.

Using the SCAN service (MAC layer), it searches a channel that doesn't conflict with others.

Then it selects a non-already-in-use PAN identifier.

through the SET service (MAC) set the PAN identifier, the device addresses and an address for itself (16 bit)

Finally, START service to initialize the PAN

JOIN(↑) 2 ways:
- DIRECT-JOIN a router/coord requests for a device to enter in their PAN
- ASSOCIATION

It does a Network Discovery to see what PAN exist, selects one then invokes the primitive JOIN.REQUEST (contains the PAN-identifier and a FLAG to indicate if it wants to join as a router or ED)

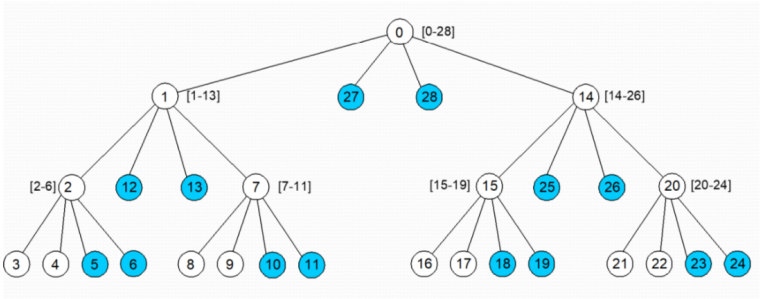
In the Network Layer, when the JOIN.REQ is received, choose the parent node of the requester

if Star Topology → coordinator Tree → coordinator or router

The association protocol obtains from this parent a 16-bit address that is sent through ASSOCIATE.CONFIRM

Then JOIN.CONFIRM to say if it's gone good or not

28) Zigbee



Parent-children relations create a tree structure where

- the coordinator is the root
- routers are internal nodes or leaves
- end devices are always leaves

This way we can assign network addresses (16 bit)

Coordinator is statically configured by 3 parameters:

- R_m max num of routers that any router may have as childs
- D_m max num of end devices " " " "
- L_m depth of the tree

(↑) in blue end devices

whites \ (0) are routers

$R_m = 2$ $D_m = 2$ $L_m = 3$

Full tree with these params

Why there's a network address and a MAC address?

If I send a packet from A to E (A, B, C, D, E)

network address is always Source=A, Dest=E

MAC address there are pairs A,B B,C C,D D,E

29) Zigbee-Binding

Src Addr (64 bits)	Src EP	Cluster ID	Dest Addr (16/64 bits)	Addr/Grp	Dest EP
0x3232...	5	0x0006	0x1234...	A	12
0x3232...	6	0x0006	0x796F...	A	240
0x3232...	5	0x0006	0x9999	G	_
0x3232...	5	0x0006	0x5678...	A	44

Intro Routing can be made in 3 ways: Broadcast, Tree (like the usual one, you know how addresses are split. There's beacon's scheduling to prevent that the Beacon of a router collides with others routers' beacons) and Mesh (AODV based and if the sender is an end device then it'll send it to its parent (Coord or Router) if it's a Coord or Rout there's no need to forward because they have an internal Routing Table, follow that entry: Address Dest, Next Hop, Entry Status { active, inactive, discovery under way, discovery failed }

If there's no entry → Route Discovery

The Two Topologies may coexist, because a router can maintain both info.

→ APPLICATION LAYER

application framework (up to 240 APO, 0 is reserved for ZDO) (Application Object, ZigBee Device Object) that manages the application services and the...

→ APPLICATION SUPPORT SUBLAYER (provides binding and discovery services)

defines:

- ENDPOINTS identified in each node by a number [1, 240]
  on a ZigBee node multiple apps can run (1 for each APO)
- CLUSTERS
  protocol that implements an application's functionality, defining commands (change state) and attributes (view state) via 16-bit ident.
- APPLICATION PROFILES
  ↓ profile ID
  it's the specification of the application's behavior.
  A PROFILE specifies a set of devices and clusters via 16-bit ident.
  Different profiles may coexist
  Sent data is tagged to know source
  ↓ device ID
  has two purposes: allow ZigBee tools to be effective and have a human-readable display

- offers DATA SERVICE
  allows to exchange msgs (request → ACK)
- and BINDING
  it allows an endpoint to connect to other endpoints in other nodes (unidirectional!)

The message can be sent in

- DIRECT ADDRESSING routing saying what's the dest APO based on <endpoint dest, address dest>
- INDIRECT ADDRESSING (?) implicitly specify the destination of msgs through Binding Tables

2 primitives BIND/UNBIND create/remove an entry in the BT it's inside the coord and/or routers: it can be updated through specific requests of ZDO

ZDO is at endpoint 0, it handles bindings of network and nodes.

Furthermore, DEVICE DISCOVERY obtain others devices' addresses

SERVICE DISCOVERY query based on Cluster's ID and addresses to research services

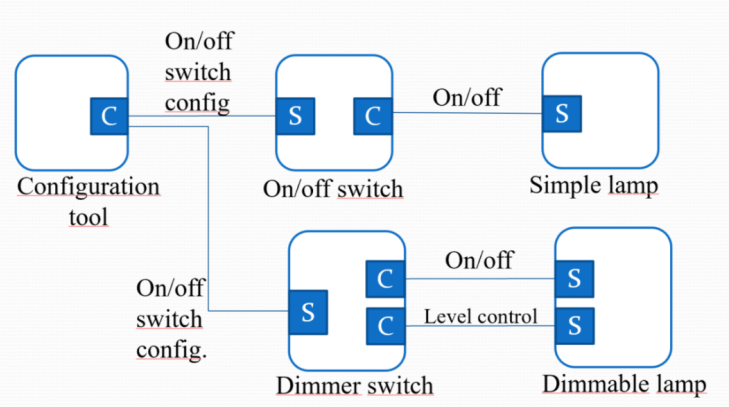
Extra Also Address Map Table, associate Netw addr (16 bit) to MAC address (64 bit)

Image

A request from cluster 6 from endpoint 5 on node 0x3232 (DIRECT) will generate 3 requests

① node 1234 at endpoint 12
② in broadcast to group 9999 (G is "group", so broadcast)
③ request to 5678 at endpoint 44

30) Zigbee cluster library



Clusters
collections of commands & attributes [...]

→ server: the device who stores the attributes

→ client: the device who manipulates them

commands are in a specific format
include Header & Payload

READ/WRITE (attr)

CONFIGURE/READ (report)
↳ periodic, whenever an attr/config changes

DISCOVER_ATTRIBUTES()

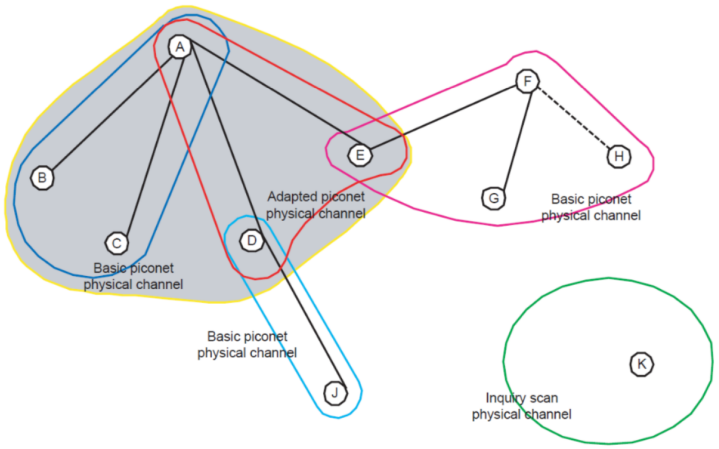
Image

Config. Tool configures these 2 switches

Client chooses if on/off and lamp will act accordingly

Below, same thing but also another attribute (level control)
↳ so handled by another client & server

31) Bluetooth



BR / EDR

Bluetooth BaseRate / Enhanced DataRate

operates at 2.4 GHz

up to 3Mbps

forms PICONETS (a group of synchronized devices with a shared clock and a Frequency Hopping Pattern)

A physical link may be seen as many logical links

↑ on these there is synchro, async and isochronous data

This type of BT is connection-oriented

⇒ once connected, the link remains up even if no data is shared

⇒ there are modes that allow the device to sleep to reduce power consumption

About LowEnergy (LE) it's optimized for an ultra low power consumption because of very small pacuets ≡ reduce current to transmit ∧ reduce receiving time

operates at 2.4 GHz Too, but only up TO 1Mbps

it uses both FDMA and TDMA ← Frequency / Time Division Multiple Access

on over 40 phys channels

↳ divided in EVENTS, ADVERTISING and CONNECTION

designed to send little chuncks of data, not files!

2 topologies Point To Point and Point To Multipoint

be master in one, slave in others

Two or more stations that share the same channel form a

PICONET: a station is the master, others are slaves. A device may be part of more PICONETS!

↑ synchronizes all

in BR/EDR

up to 7 active slaves, others are connected in Parked Mode (up to 255, can't Transmit but kept synchronized)

Piconets that coexist in the same area and overlap

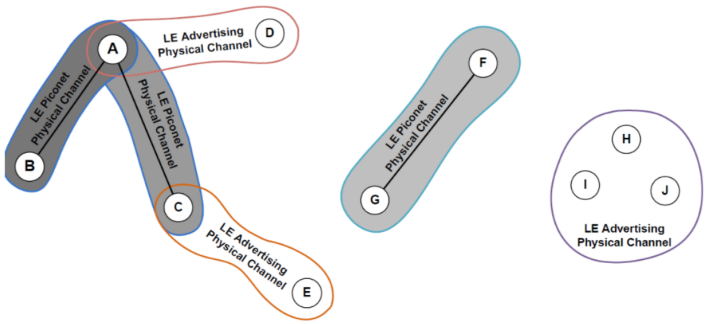
!!! scatternets

no routing protocol already specified here

in LE

no limits (except the physical one on resources)

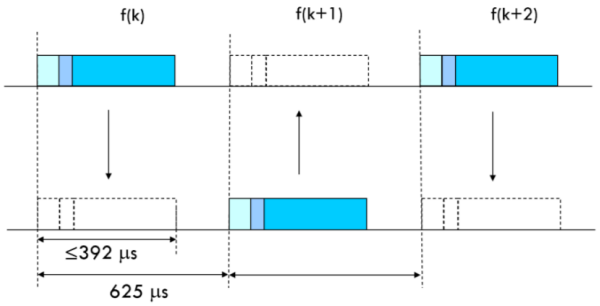
32) Bluetooth



Point To Point! see ↑

Advertising see ↓

33) Bluetooth



How to access the channel in a Piconet

The physical channel is built by the Baseband layer (up to 79 available freqs) depending on the master's address.

Channel access is done by Time Division Duplex (TDD)

The channel is divided in time slots of 625 µs

The time between 393-625 is used for freq. hopping

→ Master uses odd time slots

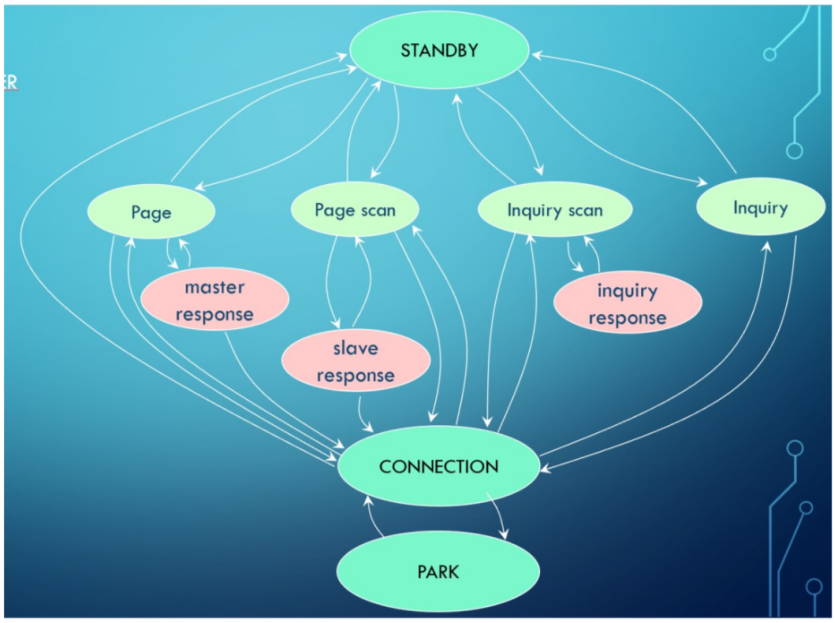
→ Slaves use even ones (one by one!)

Messages are exchanged with packets (transmitted in a different frequency hop)

→ Baseband Layer

- Frequency Hopping Management to avoid frequency interference
  - BR/EDR: freqs range from 2.402 Ghz to 2.480 Ghz (79 channels)
    signals have a rate of 1600 jumps/sec (follows a pattern)
  - LE: a sequence of 37 pseudo-random numbers
- Channel Management
  - power control
  - link control
  - access to the channel

34) Bluetooth



State Machine

3 main statuses

standby, connection, park

Standby the device isn't part of a PICONET

Connection the device is part of a PICONET and can exchange packets

substatuses: ACTIVE, SNIFF and HOLD mode

park the device is part of a PICONET but can't actively participate

very low duty cycle

other substatuses

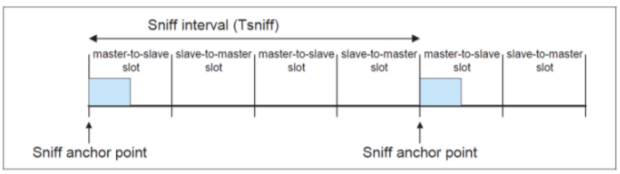
- PAGE used by master to activate a slave's connection (when not synchronized yet) periodically sent until a msg from the slave is received.
- PAGE SCAN used by the slaves to look for a PICONET
- PAGE RESPONSE M/S goes in this status when a PAGE msg is positively received by S/M. Master will send to the slave the PICONET's parameters (clock, number seq, ...)

master/slave

- INQUIRY used to find other devices. While in this status, always send an INQUIRY msg to collect devices' addresses, names, ...
- INQUIRY SCAN used by devices that want to be discovered (they listen for INQUIRY msgs)
- INQUIRY RESPONSE (optional) answer an INQUIRY msg by sending the previously requested params

35) Bluetooth – Sniff mode

connection substatuses



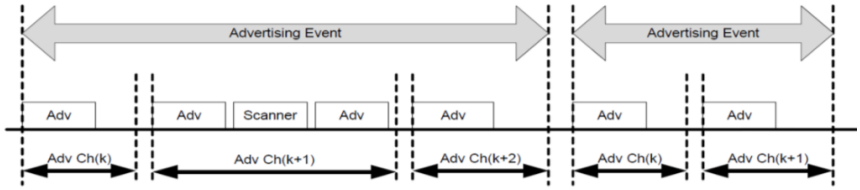
active the node is actively part of the PICONET
In this mode, the Master schedules the transmissions, while slaves listen for possible packets from Master
if a slave isn't interested for a packet, it may go into sleep mode until next transmission from master

sniff(↑) the duty cycle is reduced to save energy and master will transmit only in specific programmable slots

hold slaves keep synchronizing, reducing duty cycle, and use this time to attend to communications of another PICONET or to enter low power mode

extra in park status, nodes receive another address to leave the old addres to active ones

36) Bluetooth Low energy



There are two types of events:

Advertising

packets are transmitted by a device called ADVERTIZER towards devices that receive the event, called SCANNERs.

Advertising Event is the one where data is exchanged

A scanner may answer to an Advertiser and receive an answer

Connection

These events are used to construct PICONETs.

Advertizer publishes a "connectable advertizing event"

Initiator receives the event and starts the connection protocol

Initiator becomes the Master of the PICONET, so advertizer is a slave.

Extra: A LE device may operate in 4 roles:

① CONNECTION BASED

peripheral device (connectable advertizer, can operate as a slave in a connection)

initiator device (Listening for advertizers and "initiate connections" and operates on them)

② ONE DIRECTIONAL COMMUNICATION

broadcaster (non connectable advertizer) temperature sensor

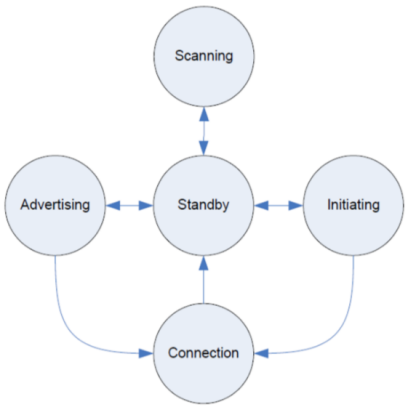
observer (scan for advertisements, can't initialize connections) remote display

Advertizers emit advertiments called Beacon.

With Beacons a device may run for years even with little battery.

In Beacon applications, the device roles are PERIPHERAL and BROADCASTER

(the [non] connectable depends on a flag)

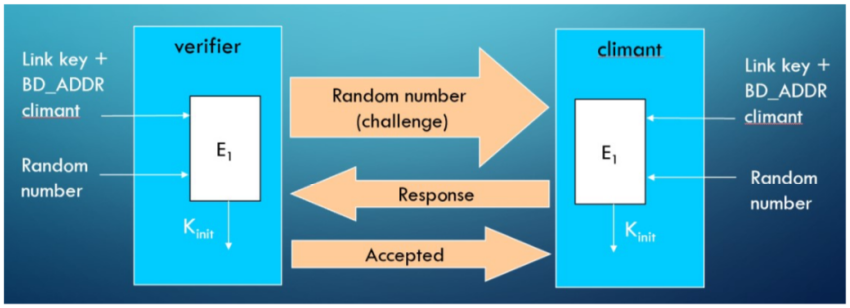


State Machine of Baseband Layer in LE

state

- standby doesn't transmit nor receive
- advertising device transmits advertiments and is possibly listening for answers (for advertizers!)
- scanning device listens for advertisements (for scanners!)
- initiating to connect with peripheral devices. Listen for advertiments from a specific device and answer to these packets to start a connection (for initiators!)
- connection device is known as being in a connection In this state, Master and Slaves are defined

38) bluetooth



Intro

in Baseband Layer 2 addresses are used

- MAC address (48 bit)
- network address, in PICONET (3 bit for active slave, 8bit for parked)

- Link manager layer to check connection between two devices.
  BT BR/EDR uses Link Management Protocol, BT LE uses Link Layer Protocol
- Host Controller Interface is an interface used to access BT's hardware.
- above HCI → Logical Link Control and Adaptation Protocol (L2CAP)
  it does multiplexing of the packets, segmentation and manages QoS.
- RFCOMM simulates the serial port
- Audio Protocol to manage audio streams (Forward Error Correction for eventual errors)
- Telephony Control Protocol Specification defines the necessary signaling to start a voice/data communication
- Talking with L2CAP there's
  - GAP → Generic Access Profile
  - GAT → Generic Attribute Profile

  } they define the device's profile in Terms of the services he offers
  - ATT → Attribute Protocol defines the access method to services

Furthermore, GAP defines a base profile that every BT device must implement

GAT is mandatory in BT LE and optional in BR/EDR. It's used by ATT to define how the device send and receive messages and defines profiles (set of primary or secondary services)

ATT is mandatory in BT LE and optional in BR/EDR. It defines the roles of server (the one that exposes the set of attributes and their value) and client (the one that finds, reads and writes on attributes)

on a value you can:

Push(v)	Pull(v)	Broadcast(v)	Set(v)	Get(v)
when a server sends data when this has changed	client requests data when it needs	sends data to every listening device	used to configure the server	to obtain the value of an attribute

Image ↑ BT LE is less secure!

There are 4 access models (pairing)

① JUST_WORKS accept the connection (without PIN, ---)

② NUMERIC_COMPARISON 6 digits must be the same on two devices

③ OUT_OF_BAND use alternative channels (∉ BT freqs) To find devices and send cryptographic code

④ PASS_KEY_ENTRY used when a device has an input but no display and another device has an output and a display.
shown output must be inserted in the input device

About security

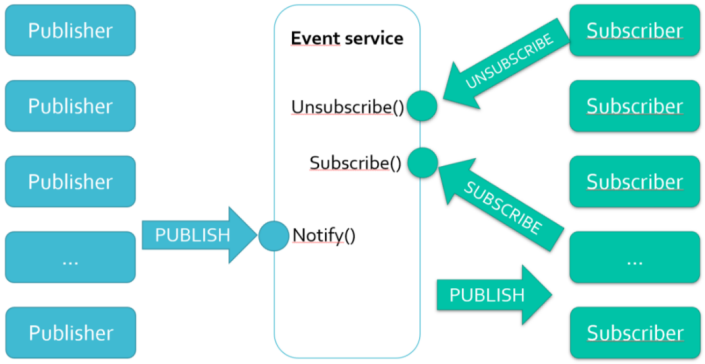
- use an unique ID (48 bits)
- a private key (128 bit) for authentication ≡ Link Key generated and sent in the pairing phase, so when the channel is created through one of the ④ ↖
- a private key for data encryption (8-128 bit, updated each session)
- random number (128 bit)

Image is SECURE SIMPLE PAIRING, an authentication mechanism based on a challenge-response to verify that they both know the same Link Key.

In the pairing phase, create a safe channel between the two devices
⇒ building the initialization ping (K_{init} in img)

Verifier has the Link Key + Bluetooth Device address that he wants to connect To. Then Random Number for → K_{init}

39) Publish/subscribe & MQTT



MQTT Message Queueing Telemetry Passing is a protocol that implements the Publish Subscribe mechanism.

It's lightweight because it has low bandwidth and low packet overhead

Main Actors

client: publishers and subscribers

server: event-service (broker)

Pubs produce events received by broker who forwards to subs that expressed interests for certain events

4 operations in the protocol

① PUBLISH ② SUBSCRIBE ③ UNSUBSCRIBE ④ NOTIFY when an event is generated

This leads to
- space decoupling (pubs and subs don't need to know eachother)
- time decoupling (no need to be executed in the same moment)
- synchronization decoupling (operations on pubs/subs aren't stopped while working)

⇒ approach is scalable

broker's operation can be parallelized (event-driven, just replicate the broker)

Messages are filtered on the broker depending on:
- topic the thing to what the client subscribes to
- content client subscribes to queries like "Temp >40°"
- type both content and structure, like object-oriented

⇒ Assumptions: ① pubs and subs must agree on topics before
② pubs doesn't assume that someone will read its messages
③ Broker can store msgs for offline subscribers, but they must be connected via persistent connection and be already subscribed to the topic.

So, MQTT is asynchronous, callbacks-based, very simple

can be synchro by using the synchro API

↑ complexity on the broker's side

Each message has a topic (for future extensions, topic's hierarchy must be well designed)

Connection Protocol (client → broker)

① CONNECT msg contains
- Client's ID (unique string)
- clean session [opt] (false → persistent connection) — store msgs ui
- username & password [opt, if needed]
- will flag [opt] (if a client gracefully disconnects, broker notifies it)
- KEEP_ALIVE [opt] (an interval of time in seconds, every t seconds sends a control pkt to the broker, =0 → no)

② CONNECT_ACK with flags — and why not
if connection was accepted or NOT and SESSION_PRESENT indicating if in the broker there's already an existing session for that client

Once connected, a client may publish msgs ⊃ topic + payload (binary, text, xml/json)

⇒ PUBLISH msg contains
- Packet ID (int)
- Topic Name
- Quality Of Service (0,1,2)
- Payload
- RETAIN flag — if the msg must be stored as the last known value for that Topic
- DUP flag — when the msg is a duplicate of a previous one

When Broker receives this msg, send ACK if QoS > 0. Then process + send msg to subscribers

⇒ SUBSCRIBE msg contains
- Packet ID (int)
- List of Pairs <Topic, QoS>

Broker will answer with SUB_ACK, containing Packet ID (same as the one sent) and RETURN_CODE (one for each topic you have subscribed To. 128 if fail, 0, 1 or 2 are success and the guaranteed QoS for that message)

UNSUBSCRIBE is the same and UNSUB_ACK too (no RETURN_CODE ?)

Topics may be organized in a hierarchy, usually separated by /.
Wildcards can be used to represent groups (+, #, ...)

Topics that start with $ can't be published because they're used for internal MQTT's statistics. other than this, no specific rules (but best practices!)

There are 3 levels of Quality of Service

QoS is an agreement between the pub and the sub.

① AT_MOST_ONCE = 0
it's a "delivery best-effort".
msgs don't have an ACK and aren't stored in the broker

② AT_LEAST_ONCE = 1
msgs are stored by the broker until they're sent to all subs.
so msgs are sent A_L_O, maybe more pubs will receive PUB_ACKs

③ EXACTLY_ONCE = 2
slowest, but guarantees that the msg is sent once because it uses the 2-way-handshake

P=Publisher B=Broker
P PUBS and waits for PUB_REQ
↓
B receives the PUB and sends PUB_REQ maintaing a reference to the msg until it receives PUB_REL
↓
P receives PUB_REQ and sends PUB_REL maintaing a ref TO PUB_REQ
↓
B receives PUB_REL and sends PUB_COMP deleting the ref
↓
P receives PUB_COMP and deletes ref.

QoS from P to B and B to S are independent

No rules here too, but best practice

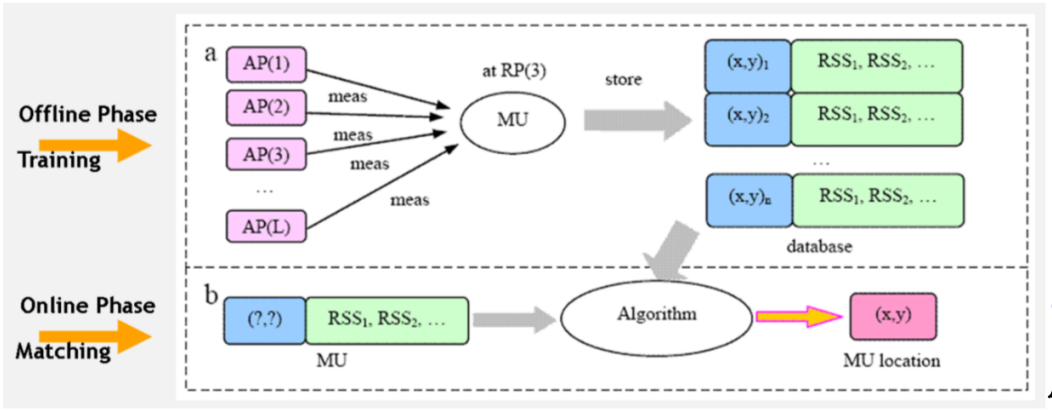
- use 0 when connection is stable and msgs are frequently updated (no need to store msgs for offline subs)
- use 1 when you need all the msgs, but you have to handle duplicates
- use 2 when 1 can't handle duplicates

Persistent sessions
are used so that a reconnecting sub doesn't need to resubscribe to topics and are associated to the Client ID defined in CONNECT and contains:
- All subscriptions
- All unconfirmed msgs with QoS 1 or 2
- All pending msg, QoS 1 or 2

⇒ client must store its persistent state

RETAINED msgs are used to immediately update subs status. Good when a topic isn't frequently updated

Broker may be a limit in distributed apps with many point2point communications and is a single



2 types of location

① PHYSIQUE 2D-3D coordinates

② SYMBOLIC info in natural language "close to the kitchen"

Divided in ABSOLUTE ∧ RELATIVE

ABS to REL is possible

Outdoor positioning is covered by GPS

↳ indoor? 2 topologies ① Remote Positioning the unit to localize is mobile and acts as a Transmitter. The units that measures locations are fixed (anchors). The localization algorithm is executed by Fixed Location Manager

② Self positioning the unit to localize is mobile and it does itself the measurements ∧ localization algorithm. It receives signals from fixed anchors (transmitters with known position)

2 more are inferred • indirect remote positioning ≃ self, but pos sent to a Remote Location Manager

• indirect self positioning ≃ remote, Locting sends the pos to mobile

3 types of metrics

① TIME

- Time Of Arrival higher propagation time, higher distance (directly proportional)
  Issue requires synchronization [in 2D you need 3 measurements from 3 anchors] 1-1-1
- Difference Of Arrival position is calculated with pairs of sensors
- Round Trip Time Of Flight anchor and transmitter are the same thing. The device is a transponder (receive signal and send it back)

② ANGLE (works well with Line Of Sight LOS)

- Angle Of Arrival location's obtained doing the intersection of different pairs of angles. 2D → 2 pairs, 3D → 3 Needs directional antennas (costly!)

③ SIGNAL

- Received Signal Strength distance is based on how powerful the signal comes
- Received Signal Phase we assume that transmitter sends sinusoidal signals. Based on the received phase of signal. Doesn't work for dist > wave length ∧ needs LOS

Finally 2 approaches:

- Range-Free Appr ignore the signal and do everything depending on the anchors' position (crucial placement)
- Range-Based Appr consider the signal and calculate the position

Image

Offline phase is about collecting data (vectors of <x,y> positions and signals) from anchors (AP). They do measures from the Mobile Unit (MU)

used to generate Tuples RSS1, RSS2 for each POINT OF THE GRID (points decided via euristics or neural network)

Online phase is where you match RSS to existing fingerprints (probabilistically) in this phase the MU will match the tuples with the fingerprints he stored in OFFLINE, getting its localization through the localization algorithm