Low-power wireless sensor networks for the Internet of Things

IEEE 802.15.4

802.15.4 Introduction

Mainstream radio for IoT and sensor networks. Adopted by ZigBee, IETF network stack. Has two physical layers (2.4GHz, sub-GHz, UWB); no MAC layer by default, as it depends on the specific platform (so a lot of protocols appeared). -> Latency is less important than reliability in some cases (e.g. once-a-day logging).

2.4GHz band -> most popular (short range, tiny bits of data (127B), 250kbps) -> creates interference with BT and WiFi, since they stay on the same frequency.

Brief reminder on how WSN are made. First, we had a garbled mess of cables; then, the client-server architecture came, with a collector node, which limits bandwidth and flexibility: the farther a node is, the more energy is needed to be spent) -> we can move to a distributed architecture (whisper) with short range radios that are cheap, battery powered, spread over large areas, and with "word of mouth" they relay data through multi hop routing.

This idea was born in the US from the DoD but then spread to everywhere: wildlife monitoring, glacier monitoring, and so on.

WSN applications are classified by:

- goal (sense, sense-react)

> goal -> some network just gather data, other need to react appropriately: i.e. change the system at hand, since transmission is costly from a power and a latency point of view. e.g. hvac (aka smart thermostat) can be implemented both as a client-server approach (where data goes to a central PC), and with sensor talking to multiple actuators

- interaction pattern (1-1 unicast, 1-many, many-many, many-1)

> interaction -> nowadays mostly unicast (client-server). sensing is many to one (converging traffic); controls is one to many (central appliance contacting multiple nodes at once). Now many to many is common

- mobility (static, mobile node, mobile sink)

> mobility -> sometimes there is dinamicity in the nodes (if sensors on animals or robots etc) or the sinks (e.g. data mule going into proximity of the sensor)

- space (global/local)
- time (periodic/event-driven)

> space and time -> the place and the time when processing occurs Global applications are concerned with the system as a whole, regional only a fraction. Periodic applications are executed at regular intervals, event-driven perform processing once a certain condition is met. Example: a global and periodic application may be wildlife monitoring, a regional event-driven application may be a sensor that detects intrusion.

Reference architecture

```
|------------------------------|
| Application                  | --> sysmodules (t.sync, routing, locaalization)
|------------------------------|
| Programming abstraction      |
|------------------------------|
| Routing         |            |
| Localization    |            |
| Time synch.     |   System   |
| Storage         |  Services  |
| Programming     |            |
|------------------------------|
|  MAC  |          OS          |
|------------------------------|
| Radio |      HW (CPU/Mem..)  |
|------------------------------|
```

The situation is in reality a mess in which each component is built separately and therefore customized for the particular application. Components are often reused when they should not (maybe they are badly designed, or they interact poorly). This forces the programmer to focus also on low level logic (as in the OSI stack up to the MAC layer, whilist usually you just

build new L7 protocols and do not care about TCP or IP)

OS are not therefore usual ones, they just provide runtime support and no user interaction. Usually, programs are cross-compiled and linked with the OS library. Also, no multi-threading. A popular choice is Contiki today which is ported almost everywhere.

Either way, key networking layers in WSNs include MAC (as said before, needs to turn off as much as possible for the power) and routing (decides which route a packet should take through the multi hop network, best guarantees for reliability and energy and latency, also faults)

- MAC -> avoids packet collisions and turns on just when needed. Uses CSMA for preamble sampling and TDMA
- Routing -> different than usual: the nodes are not important as that singles, they just pick up and drop packets as they come. We need their collective features

Why is communicating a challenge? We need multi hop (since nodes may not be directly contactable, due to walls in between); we need to save energy (as both transmission AND reception do use a lot of battery, although transmission is worse on some, much better on other).

Moreover, low power links are unstable in time: what you see is not what you get. Environment, weather, movements, time of day, all affect packet reception rate. Also in space and frequency they are unstable.

Medium Access Control (MAC)

MAC Introduction

Part of the Data Link layer in the OSI Stack. In normal networks, latency, throughput and fairness are the key areas of interest. Interactive and data intensive applications are common in Wi-Fi/IP applications. On the other hand, WSNs don't care much about throughput, and have slightly different requirements.

First, we give a brief overview of general wireless medium access control (MAC) protocols.

CSMA:

- Carrier Seisten before you transmit, with back-off. Collisions may occur since transmission is not instantaneous. Also has the hidden terminal problem: in the case A, C are not in the range of each other and the receiver is in the middle, then B will be bombarded from both A and C
- Collision Avoidance -> request to send + clear to send messages to fix the previous problems, but more overhead (1 RTT before starting the communication)

TDMA:

- TDMA is schedule based, i.e., there is a priori syncronization of transmitters (unlike CSMA) and time is slotted into discrete intervals, used to organize communication. This includes header and footer data, for traffic control and contention periods.

In WSN, there are far too many MACs: most of them is not supported by hardware and must be explicitly developed or tailored to a specific application. Moreover, the main focus is lowering energy consumptions (mAs) at the cost of higher latency and less throughput. Moreover, it has to control the sleep cycle (i.e. deciding when the radio must be active, and minimizing that time), and control the total overhead for doing such a task.

In this cases, the protocol overhead becomes significant (as the payload gets smaller). Traffic also fluctulates wildly, being usually focused in short bursts (but complicating everything) in what are called event based applications.

First approaches (S/D/B-MAC, RandAcc)

S-MAC is an example of such a protocol. Nodes swap between listen and sleep modes. The goal is to reduce collisions and overhearing. Wake up times are roughly synchronized. Listening periods are divided between the time for exchanging SYNC and CTS/RTS messages. Also, it assumes that clock drift is negligible.

D-MAC, instead, is a slotted, cross-layer MAC. It is used for data gathering. Since multi-hop packets are usually slightly delayed, the active times are staggered according to the level in the spanning tree. This allows less time waste in collisions once the packet starts going upwards in the collection tree.

Random-Access is a protocol that is very flexible, but it is not energy efficient, due to a large amount of idle listening and collisions. Yet, having no clock synchronization means that its deployability is maximum and has no communication overhead, thus making it popular for real-world implementations that don't have rigorous system requirements.

B-MAC uses a different approach to the cause. The receiver usually does nothing and the sender carries out the hard work. The receiver does carrier sensing interleaved with sleeping. If energy is detected, it keeps listening - since there is a sender advertising its activity with a long preamble. Then, data is exchanged. The protocol is barebones: no ACKs, no RTS/CTS. The important bit is that, to ensure coordination, the preamble must be slightly longer than the sleep interval.

In order to check for the presence of energy in the channel, B-MAC uses *CCA* (Clear Channel Assessment) to determine the state (since there may just be environmental noise disturbing the channel). This functionality is provided by the radio chip hardware.

The throughput for B-MAC is high, but tends to get lower as there are more nodes and collisions. Moreover, power consumption is usually lower, due to the absence of sync messages.

LPL and overhead issues (X-MAC, BoX-MAC)

LPL does have some sources of overhead that must be taken into account. For example, idle listening (since radio receiving is much expensive, but must be done), overhearing (since we listen to anything, even messages not for us), and collisions (simultaneous transmissions interfere, RTS/CTS is very expensive but wasteful for small packets, usually we just retransmit)

To try to address these issues, two sort of Frankenstein protocols (X-MAC and BOX-MAC) were created, collectively referred as LPL.

In **X-MAC**, the destination is inserted in the preamble - to reduce overhearing - and the preamble itself is also shortened, so that it can be sent multiple times, with small spaces in between to listen for ACKs. This protocol greatly reduces the duty cycle (40% -> 10%) as the node density increases. The sleep interval remains the key parameter in this implementations.

BoX-MAC, on the other hand, just keeps sending the messages: as an ACK arrives, the data is marked as received.

However, how do we choose the sleep interval? We can inspect a checktime vs lifetime graph for that. As the check time increases, the lifetime hits an optimum and then descends slowly. The lifetime increases as the sample period increases (1-min < 5-min < 10-min ...). Thus, with small sleep intervals, the node wakes up too often and uselessly, while with bigger ones we have a greater chance of collision and the node spends more time listening to preambles. The actual choice depends on the system itself.

Finally, we have **low-power probing**. It uses a different approach. Since preambles occupy the medium a lot, we can have the receiver send a probe ("ready to listen") to the medium, telling tramsitters to start sending data. RI-MAC is a variant of this protocol, that makes the transmitter do short burts of listening phases and contemporaneously advertises his availability to listening. In other words: "You know I'm here if anyone wants to talk to me and I will trasnmit, else I'll go back to sleep". its a very short listne. Dually, this intention of transmission is no longer a long preamble, just a listening, such that sender only msut only transmit. In another sense, the wait is transfrered to the receiver.

A-MAC

A-MAC, the so-called "backcast" primitive, is a protocol very similar to LLP. It sends probes and receives reply with ACKs ("auto-ACKs", identical frames that are hardware generated). This is all done with a tight timing tolerance. ACKs collide non-destructively.

Usually, if you have two packets, should they overlap in collision, data is rendered useless. In this case, we can still get some information we if have:

- constructive interference: the packets are identical and overlap within 0.5 microseconds
- capture effect -> if the received signal is 3 dB stronger thatn the sum of all other received signals and arrives < 160 microseconds later than the first. This allows synchronization and the receiver can still decode it.

In pratice, this is what happens. The receiver probes with an ACK request, and waits several milliseconds until the ACK from the sender arrives (and no more than that few, in order to save energy!). Right after the ACK, it receives a data packet and the receiver probes the sender, which doubles as an ACK to the sender.

A-MAC can determine accurately if there is traffic (O(1)) due to those strict syncronizaztion timings. In case of frame collisions, the receiver re-sends a probe (as a nACK), in which hopefully one of the backoffs will help retransmission of data.

Finally, we can exploit the fact that we can separate control and data transfer on different channels, the latter of which is specified within the probe.

A-MAC is very efficient with collisions and has a high delivery rate, but again, the more nodes/receivers the more collisions.

Hybrid protocols

Unlike CSMA-like protocols, *TDMA* protocols rely on discrete slotting of frames to avoid collisions and overhearing, although it is usually expensive and made for 1-hop topologies, and requires strict time synchronization.

We can combine the *CSMA* and *TDMA* ideas into new protocols to get the best of both worlds. CSMA is totally anarchic and random and has no assumptions, is robust and flexible, and works best with low channel utilization. Vice versa, TDMA is more rigid and prone to sync issues, but is efficient with a lot of contenders.

Z-MAC (=Zebra) is a protocol with a striped behaviour: it works like CSMA (= low latency) with low channel utilization, and TDMA (= collision reduction) with high utilization on both cases. Time is divided into slots, but they are not as strict as TDMA. Indeed, slot owners have priority, but any node can use any unused slot. If the slot's owner drops it, then anyone else can grab it.

The protocol explicitly operates in HCL or LCL (high/low contention level. When notified of contention, it switches its state (e.g. if packets are lost or the noise is high). n HCL, send only during assigned slot. In LCL, wait to see if the owner is going to send, if not, wait a random back-off, then transmit.

IEEE 802.15.4 MAC

The official IEEE standard has several industry-standard protocols that each caters to a different use case. It has also been extended by a number of amendments - as the original algorithms were actually bad.

These algorithms target **"Personal" Area Networks** (now a relic) - whose nodes can be either **Full Function Devices** (which operates in any role) or **Reduced Function Devices** (i.e. light switches). The topologies are either star topologies (the classic one) with a PAN coordinator at the center, P2P topologies with RDDs exclusively as leaves (as a sort of mesh/cluster topology), and cluster tree topologies.

In 2006, the first standard was released and made as simple as possible. It comes in the flavors of **beacon-enabled** (synchronous/slotted) and **non-beacon-enabled**. (asynchronous/unslotted or sloppy). The former synchronizes the network with beacons and operates in contention mode, the latter assumes the radio is always on, and employs CCA to determine if there is a free channel.

In this mode, coordinators periodically broadcast a beacon. After this beacon, nodes know that there will be a *superframe*. It contains 16 slots: the first 9 are called *Contention Access Period*, in which nodes use CCA to try to send data packets and eventually back off. The last 7 are called *Contention Free Period*, in which nodes have their collision free data slots, that were previously agreed with the coordinator.

All superframes are syncronized - so this increases contention! In the first part, everyone is shooting for a shorter-than-normal window and actually reduces performance. Worse, the competition is for the "future": nodes ask for slots to be reserved in later superframes' CFPs, which is made of 7 slots - so with 7 or more contenders, reservations flow over to the next superframe, increasing latency and ultimately contention.

IEEE 802.15.4e

IEEE 802.15.4e was an amendment from 2012 and 2015 that introduces a Low Energy mode: *Receiver Initiated Transmission* (RIT === LPP) and *Coordinated Sampled Listening* (CSL === LPL with a twist: it learns the receiver patterns and syncronizes the device with the next transmission; also supports burst mode). Also features *Deterministic and Synchronous Multichannel Extension* (DSME), for the original beacon enabled mode, and a lot more. DSME is like beacon enabled, but uses a matrix instead of slots (supporting multi-channel operation) and groups multiple superframes. Also supports Guaranteed Time Slots. Finally, we have *Time Slotted Channel Hopping* (TSCH).

Coordinated Sampled Listening (CSL), as mentioned above, is similar to Low-Power Listening. Senders use a wakeup sequence before any non-synchronized transmission. As they receive packets from other senders, they can learn the sender's pattern and synchronize the next transmission.

Deterministic and Synchronous Multichannel Extension is a mode that runs similar to the original beacon-enabled mode. However, now the Contention Free Period supports multi-channel operation and allows the grouping of superframes into multi-superframes. DSME, sadly, does not have a standard for channel adaptation or hopping - which is protocol dependent.

TSCH

Time Slotted Channel Hopping is a sort of a modern TDMA suited for multi-hop mesh networks. It gives predictable latency, bandwidth. Communication slots are globally scheduled based on time and frequency channel. It supports multi-channel and channel hopping. These ideas were part of industrial standards that were centralized (with controllers at the center of star topologies).

The ideas at the core of TSCH were part of standards, each own with its own set of commercial devices. What draws them in common is that they are all centralized, with a controller at the virtual center of the topology, which must be kept updated.

In TSCH, there is a stark division between dedicated links (multi-booked slots for deterministic traffic, one receiver and one transmitter) and shared links, used for sporadic, unpredictable traffic such as discovery and routing messages. They falls back to CSMA/CA behavior. Slots are defined with microsecond-precision granularity, and they last 10 milliseconds: enough for sending and processing both the data packet and the ACK, plus some guard time for taking into account clock drifts.

For joining the PAN, wishful entrants must listen to *Enhanced Beacons*, sent by members of the PAN on multiple channels.

These frames contains information such as timeslotting, synchronization, when to listen for transmissions, and more. When the EB is received, the MAC layer notifies upper layers and the device can start sending data (and also advertising itself).

The schedule may be decided either in a centralized manner (where a single node builds and maintains the schedule) or in a distributed manner (where each node decides autonomously). The standard does not specify how the schedule is built, just the execution.

Within this protocol, a working group was setup in 2013 that tried to combine high reliability and low-energy consumption of IEEE 802.15.4e TSCH with the interoperability and integration offered by the IP stack - in particular, the IPv6 version. It includes a *6TOP* sublayer, that bridges the lower TSCH MAC and physical layers with the classic IP stack on top.

Routing

In traditional networks, routing is usually done with zero awareness from applications. In WSNs, routing - like MAC - must receive some kind of optimization for energy-related constraints, and usually operates s.t. it also servers some specific requirement of the application.

While the classic pattern was many-to-one (*convergecast*), in which the link quality was usually unstable and unpredictable and control traffic was minimized, modern WSNs also require alternative routing strategies such as one-to-many or even one-to-one.

What follows is not a comprehensive list of possible routing strategies, just a few examples.

First ones: LEACH, Directed Diffusion

Low-Energy Adaptive Clustering Hierarchy (LEACH) is a protocol specially designed for wireless sensor networks, that is divided in two phases: *setup* and *steady*. In the setup phase, the nodes elect randomly a clusterhead. The clusterheads are then responsible for advertising themselves, and other nodes choose their CH depending on signal strength. Clearly, clusterheads may be chosen poorly, affecting the protocol's efficiency. In the steady phase, nodes send clusterheads data, which are then are responsible for aggregating and sending their data to the base station. The base station may be far away, requiring more power to be reached. CHs are periodically and randomly changed, to ensure load balancing and uniform energy depletion across nodes.

Directed Diffusion is a data-centric approach in which sources name data with key-value pairs and sink flood interests based on those attributes. This creates a gradient in the network, in which the delivery follows the route estabilished by the gradient. This is very similar to content-based routing. Gradients are reinforced by sinks based on the quality of the data. It is designed for streaming data at a constant rate.

Tree-based routing

Probably the most common approach in WSNs routing is the *tree-based* one. First, the network is flooded with tree-building messages, and then on the reverse path the actual tree is built. Usually, primitives are provided for re-building the tree dynamically.

While for most application this approach works, other settings such as one-to-many interactions may require a different degree or reliability. Indeed, source-to-sink reliability drop sharply with the network diameter. To address this, (hardware) ACKs are usually sent at each HOP

One might ask how to choose its own parent. Usually, hop count is used, but it is not always reliable. We can therefore use *link quality estimators*, application information, and workload information. The former include:

- RSSI (**Received Signal Strength Indicator**) -> signal + floor noise
- LQI (**Link Quality Indicator**) -> correlation between received symbol and decoded symbol
- ETX (**Expected Tramsission Count**) -> number of retransmissions required before the receipt of a 1-hop packet is acknowledged

Another issue to be taken into account is that nodes closer to the sink die earlier. This is due to their increased workload. Some approaches have tried to fix this, such as load balancing, embedding bigger batteries in the topmost nodes, or using in-network aggregation (i.e. sending less data).

We define *network lifetime* as the amount of time the network survives, from the bootstrap until any node dies (partitioning the network), X% of the nodes are dead, or functionalities are disrupted.

Early tree-based protocols (MLQUI, CTP)

MintRoute / **MultihopLQI** is one of the first tree-based protocols, released in 2003. It is many-to-one. The root sends a periodic beacon that is re-propagated in order to build a tree for routing, based on LQI. We define the cost metric as:

$$\mathrm{m} = \mathrm{m}_{\mathrm{parent}} + \frac{1}{\mathrm{LQI}_{\mathrm{parent}}^3}$$

It is additive: grows with hops. Nodes choose their parent with lowest metric, and the sink metric is set to 0. Moreover,

nodes, keep a list of backup routes. It can also handle duplicates.

Collection Tree Protocol, instead, uses ETX as a link estimator and supports anycast over multiple sinks. It is much more efficient, sending less beacons and detecting loops fairly quickly.

One of their biggest differences lies in loop detection. In the former, loops are discovered at the source, and packets are dropped until a next hop is found. In the latter, loops are detected when there is a cost mismatch (the receiver's cost must be lower than sender's), so if it's higher, the packet is redirected back, increasing reliability.

Another difference is that CTP uses adaptive beaconing. While MLQI just sends beacons periodically, without tuning the period, CTP uses the idea of Trickle protocol, with minimum and maximum periods. If the topology is stable, the period is doubled up to the max. Else, if the topology changes, it is reset to the minimum. This vastly reduces the traffic in the network.

MUSTER

MUSTER is multi-source multi-sink routing protocol. This algorithm tries to overlap as many source-sink paths as possible, in order to build an "highway" of packets. This approach reduces redundant traffic, reduces the nodes involved along with the contention.

Nodes change their parent using a quality metric $Q(n, s)$that takes into account how good that parent is in serving as parent for the s-rooted tree. Q is defined on:

- the distance of the neighbor from s
- the number of source-sink paths that go through it
- the number of sinks served

The basic version of the protocol does not use load balancing. While lifetime is increased, the corresponding "highway" or "backbone" that is created dies first. With load balancing, lifetime is added to the quality metric. This allows full utilization of the network. However, the very final nodes next to the sink still end up dying first.

IETF RPL

IETF RPL (officially Routing Protocol for Low-Power and Lossy Networks as RFC 6550) is a standard, that encompasses multiple RFCs, protocols, and tries to bring IPv6 into WSNs.

6LoWPAN is part of this effort, that tries to fit IPv6 packets into small IEEE 802.15.4 packets (but actually other low-power protocol). It employs fragmentation and compression to reduce the size of the packets.

In a nutshell, RPL supports three traffic patterns:

- multipoint-to-point / many to one, used for sensing
- point-to-multipoint / one to many, used for commands and actuation
- point-to-point / one to one

Usually, the routing topology is a Destination-Oriented DAG (DODAG), a cycleless graph oriented towards a single root node.

In RPL MtP, this topology is maintained with DODAG Information Objects (DIOs), which are advertised by nodes and contain routing information (they use Trickle). DIOs support multiple backup routes.

RPL PtM, on the other hand, uses Destination Advertisement Objects (DAOs), which now propagate upwards and create downward routes (basically, the opposite of before). Two modes are supported: storing, and non-storing. Storing mode requires nodes to keep routing tables about their sub-DODAGs (which may be memory intensive), and non-storing only requires the root to do so.

RPL PtP, finally, uses a mix of the two before: again, with two modes, storing and non-storing.

RPL is very complex, due to the required IPv6 support. It has a large memory footprint, high system requirements (usually more than double than CTP), and typically generates large routing table: with 20 neighbors, the routing table may grow to as much as 50 entries.

Orchestra

Reliability is often a problematic factor in WSNs. While industrial solutions achieve up to 4 or 5 nine delivery (99.99%). On the other hand, low-power regular meshes for the IoT typically only reach 99%. Such a percentage may seem high, but it is not.

Orchestra is a protocol that tries to solve this problem. It is built around flexibility and *reliability*. It uses autonomous TSCH scheduling for RPL routing, trying to get the best of both worlds.

TSCH scheduing may be done in a centralized manner, decentralized, or auonomous. The first has a "base station" that collects statistics and deploys the schedule. The second employs negotiation between nodes. The third lets nodes infer

the schedule from the local state, and is the one used in Orchestra.

Orchestra has three slot types.

- rendez-vous: used for discovery, is contention based
- receiver-based shared: used for unicast, is contention based and has a single rx slot per node
- sender-based shared or dedicated: several rx slots per node, has less contention but is more wasteful

Up to N slots are grouped into slot frames. Orchestra allows per-traffic plane provisioning, in which multiple slot frames are used contemporaneously (using co-prime lengths, they can mix easily with only minor modifications with slotframe priority).

The reliability achieved is in the upwards of 99.996%, or 1 packet lost every 29398. This reliability is given by RPL's ability to repair the tree, but comes at a cost of higher duty cycle and latency. Indeed, the best trade-off is struck by ContikiMAC, which has shorter latency and very low energy consumption.

Opportunistic Forwarding

Opportunistic Forwarding was originally developed for mobile computing, and is opportunistic in the sense that rather than doing single unicast connections, it broadcasts them in order to reach the nearest neighbor. It forwards packets only if it overhears no one doing the same. This saves hops and time and allows packets to reach their destination quicker, always allowing the protocol to make progeress. This makes the topology look less like a tree and more like a DODAG, but still relies on a routing metric. Ultimately, it reduces duty cycle and latency.

Glossy and derivatives (LWB, Choas, Crystal)

As already said in A-MAC, 802.15.4 PHY can mitigate interference with:

- constructive interference: the packets are identical and overlap within 0.5 microseconds
- capture effect -> if the received signal is 3 dB stronger thatn the sum of all other received signals and arrives < 160 microseconds later than the first. This allows synchronization and the receiver can still decode it.

In particular, IEEE 802.15.4 uses Offset Quadrature Phase-Shift Keying (OQPSK), in which the real and imaginary parts of the signal (I, Q) have an offset T = 0.5 microseconds, hence the requirement.

Glossy is a protocol for fast and reliable (>99.99%) network flooding. It uses a precise global clock and software delays to account for inaccuracy. It works by flooding a tree with packets (with eventual retransmissions going upwards). It is very fast, taking a few milliseconds to complete, and does not require topology state.

The main number sin the protocol are the number *N* of allowed packet retransmissions and the *slot duration*.

While many concurrent transmissions and hops are bad, in Glossy we need a large power imbalance between concurrent transmitters in order to fully exploit the interference mitigations provided by the physical layers. Still, large packets are a problem, as retransmissions are expensive and it may be corrupted by the environment more easily.

Glossy is designed to support a single initiator of the network flood. However, it is often used to build higher level protocols, such as LWB, Chaos, and Crystal.

Low-Power Wireless Bus (LWB) is a protocol that schedules periodic transmissions as Glossy floods (but also other types of traffic). In each epoch, the network is active for a small percentage of the time: the controller provides the schedule, then the nodes send the data. Finally, there is a small contention window, and then the controller computes the new schedule.

Chaos is a protocol used to compute network-wide aggregates and relies on the capture effect. Actually, it uses Glossy just for the final part, to flood the network with the results. The rest of the protocol revolves around the fact that each node, every slot, learns new information, computes e.g. the maximum, and decides to trasmit.

Crystal is a novel network stack, designed for collection of sparse and aperiodic data. Its use may either be for event/alarm reporting, like condition monitoring, or for data prediction. It builds a transport protocol atop synchronous transmissions. This, however, brings a conflict. Indeed, timely and reliable dissemination of unpredictable updates clashes with reducing overhead in the control plane.

Crystal slots are designed as entire Glossy floods, again, supported by rebroadcast and constructive interference. It is very reliable. It also supports concurrent senders (i.e. it would be cool if we had just an update per epoch). Data slots T are contention slots in which multiple flood compete. Due to the capture effect, the sink will receive at least one flood almost 100% of the times.

Crystal is extended to withstand extreme interference, like in industrial settings, and allows it to fight or escape interference, for example with channel hopping.

Ultra-Low Power Wireless Sensor Networks

Adaptive lighting

We now shall see an example of a real-world implementation of WSN: adaptive lighting in road tunnels in Trento.

Pre-set tunnel lightning does not follow environment changes, may create discontinuity, and light is typically grossly overprovisioned, usually enough to target legislated levels. Usually, tunnels are fitted with external sensors that do not take into account the conditions inside the tunnel and never compute it.

A paper by Ceriotti, Picco, et al., proposed a WSN-based Closed-Loop Adaptive Lighting system. Its testbed was in Ponte Alto, and deployed in Cadine. The testbed tunnel was 260 m long, and was fitted with 40 WSN nodes (along with other equipment). These nodes were equipped with light and temperature sensors, a custom sensor board, 4x type D batteries, and an IP65-rated enclosure.

From an energy standpoints, savngs were in the 25-40% with regards to the open loop, and 35-50% with regards to the time-table. The battery lifetime did match the project requirements (1 year), but can something more be done?

An option considered was energy harvesting, using solar panels and wind turbines. However, the energy obtained as such was limited, difficult to exploit (as it had to be routed to the WSNs).

Data prediction

In classic periodic data reporting, the sink gathers all sensor readings of the WSN with perfect precision. In data prediction, however, the sink predicts sensor readings of the WSN, leading to less traffic at the cost of diminished accuracy. Depending on the scenario, some studies claim up to 99% less traffic generated. However, this does not easily translate to 99% saving.

A prediction model used in the tunnel WSN is the **Derivative-Based Prediction** (DBP). It is a simple linear prediction model, extremely easy to compute and implement. Moreover, only approximate values are used, as long as they generally fit the model. Indeed, both value and time tolerance are included in the model.

In the tunnel, this model caused a suppression of up to 99.7% of traffic. The data was approximated very well. During the tests, the value tolerance magnitude was calculated based on actual light values, and as it increased it meant that we were changing the model. Its strength lies in the capability of capturing trends, as long as in the short time it exhibits linear behavior. Moreover, it's the only one implementable in such types of devices.

With DBP, the delivery ratio was excellent regardless of the sleep interval, and the duty cycle was drammatically reduced, reaching "no-data" levels (different from 0!!!).

Further improvements

The staple WSN stack used has two fundamental limitations w.r.t. data prediction: idle listening at the MAC layer and topology maintenance at the routing layer. With common periodic traffic, they are necessary, but with data prediction they become redudant. Further improvements can be obtained by moving the stack to Crystal, which was explored in the previous section. Either way, data prediction does not equal a perfect solution, and must be supplemented to key changes in the network stack.

Low-power Wide-Area Networks (LPWAN)

The number of connected machine-to-machine devices is set to rise drammatically in the next few years, surpassing personal ones. In this context, the LPWAN is a new paradigm for the Internet of Things. It aims to conjugate low power (a few milliwatts) with long range, usually in the tens of kilometers.

The price is to pay is extremely low data rates (`0.1-100 kbps`) and high latency, sometimes even minutes. In this scenario, however, it is not a problem.

The long range enables the use of base stations and star topologies, which ease the necessity of routing protocols and pushes all the difficulty to the the base stations themselves. The MAC layer becomes very simple, with Aloha (!!) being a common choice, as most other protocols have an unsuitable overhead.

LPWAN radios have a very high receiver sensitivity, `-130 dBm`, while WiFi is usually `-110 dBm`. Moreover, the link budget (the difference between receiver sensitivity and maximum transmission power at the sender) is also very large (`150-160 dB`).

The used bands are usually sub-GHz: usually we talk about narrowband (< 25 kHz) and Ultra-arrow band (UNB, 100 Hz).

Spread spectrum and first derivatives (SigFox, NB-IoT)

The use of such bands enables the use of spread spectrum, in which the energy of the signal is spread over a wide frequency band, and transmission becomes a noise like signal. This makes the signal resilient to interference and jamming, requires more gain, and lowers efficiency. To encode the signal, novel techniques are used that are described in the next sections.

SigFox covers most of Europe with a handful of base stations, and is based on UNB. The total throughput is laughable (100

bps), and in order to conform to duty cycle limitations in Europe, the up and downlinks are limited to 14 x 120 and 4 x 8 B messages. Base stations can simultaneously receive an arbitrary number of messages.

NB-IoT is a software upgrade for GSM/GPRS and LTE, and was conceived as a variant of the LTE protocol. It is very powerful and extensive, can be installed without any specific hardware, but is not cheap. Since it's using an already existing protocol and infrastructure, it allows for relatively high throughput, packet sizes, and scalability. However, it is limited by the fact that works only where LTE is available.

LoRa and Chirp Spread Spectrum

LoRa is backed up by a large alliance (and partially proprietary), and it is composed of an architecture labeled "stars of stars". Packets are sent from end devices to multiple concentrators, which then use a different backbone (e.g. Ethernet) to send the data to the cloud, possibly removing duplicates.

Its modulation is very particular. Called *Chirp Spread Spectrum* (CSS), it is made of chirps, a signal with constantly increasing (or decreasing) frequencies. Within a frequency band, it overflows and wraps around when it reaches the end of the band. It is very resilient, and is inspired by real-world use by whales and dolphins.

The data is modulated by choosing the starting frequency. Itt has a very thorough physical layer configuration:

- transmission power: [2, 20] dBm
- bandwidth: 125, 250, 500 kHz
- coding rate: 4/x (4/5, 4/6, 4/7, 4/8 -> x bits encode 4 data bits;
- spreading factor: #bits encoded in a chirp [7, 12]. Thus, the duration of a chirp is T_sym = 2^SF / BW. This determines the "slope" of a chirp, and allows different chirsp to co-exist, increasing the channel capacity.

Spreading factor has a big impact on energy consumption, much more than transmission power. However, it is also significantly impacted by obstacles.

In practice, LoRa's high sensitivity and link budget allows exteme ranges, up to hundreds of km (max 700 km as of September 2017). In theory, up to 1400 km should be possible.

Finally, **LoRaWAN** is a public specification that defines system architecture and an Aloha MAC layer with three device classes.

- Class A -> longest lifetime, highest latency; RX for downlink only shortly after its uplink TX, and is supported by all devices, although usually required by battery-powered sensors
- Class B -> can also schedule downlink RX from the gateway. A typical case is battery-powered actuators
- Class C -> can continuously perform RX and TX with the lowest possible latency; typically mains-powered; typical case: mains-powered actuators and/or controllers

Other LPWAN-related efforts

IEEE 802.15.4 tried to join the game with amendments to the standard, *802.15.4g* Low-Data-Rate, Wireless, Smart Metering Utility Networks; 50kbps–1Mbps, ~1km, 1500 packets) and *802.15.4k* (Low Energy, Critical Infrastructure Monitoring Networks; up to 37.5kbps,~5km,2047B).

Both use sub-GHz and 2.4GHz ISM bands, and have been lately included into 802.15.4w.

Also IEEE 802.11 (WiFi) tried to publish a protocol, *802.11ah*, which uses a 900 MHz band, Wi-Fi speeds, up to 1 km range and low energy consumption (compared to WiFi). However, it has not been successful.

IETF has been working on its own on a comprehensive network stack. Moreover, we have seen cellular standards adapted for IoT, *Weightless*, a series of three standards, and the Dash7 Alliance. Currently, it is a niche, but both short-range and cellular radio solutions are expanding towards this range. However, the main issue still remains scalability.

Bluetooth

Bluetooth is a short range technology that was first made for avoiding cables in PANs. It was invented by Ericsson in 1994, now maintained by the Bluetooth SIG. Major updates to the technology include:

- 1999 (v1) -> basic rate
- 2004 (v2) -> enhanced data rate
- 2009 (v3) -> HS using WiFi bands
- 2010 (v4) -> LE which was a step in energy efficiency
- 2015 (v5) -> supports mesh and made for IoT

Bluetooth Classic (v1-v3)

It operates on the 2.4 GHz band. It uses 79 channels of 1 MHz each, and uses frequency hopping spread spectrum (1600 hops/s) with a pseudo-random sequence known only to senders and receivers. With v1.2, it also supports adaptive FHSS, which avoids channels already in use.

What makes Bluetooth different is that the entire layer is defined and standardized (from PHY to APP), and it provides profiles, which are standardized by the SIG. They include headset, mesh, file transfer, and more. With this profile, the exact configuration is provided for the network stack to operate correctly in that setting.

The architecture is very simple (due to the motivation, i.e. replacing cables): it is master-slave. The master is the PC, the slave is usually the tiny device. Networks are usually P2P, but can be star (piconet, 1 hop star with 7 slaves) or mesh (scatternet, multi-hop stars, but unused).

Versions 2 and 3 are based on the same architecture, but with increased data rates. v2 increased from 1 Mbit/s to 3 Mbit/s with a different modulation, while v3 increased it to 24 Mbit/s by negotiating a channel with Wi-Fi.

At the time, the focus of Bluetooth was multimedia streaming and transfer, but energy consumption was neglected.

Bluetooth Low Energy (BLE, v4)

BLE is a significant departure from Bluetooth Classic, although it was designed to co-exist with Classic. It is desigened with energy consumption in mind: there are 40 channels of 2 MHz. Three of these are used for device discovery (and chosen to avoid overlapping): advertising now lasts 1-3 ms and avoids using 32 channels.

The other interesting thing is that while B Classic was designed to expect connections, BLE does not require pairings and can be used for one-to-many connections.

The architecture comprises the following layers, from the topmost to the bottommost. Bluetooth standard specify all the stack layers.

- [1, 2 in Classic] Serial Port Profile (SPP) and RFCOMM : deals with serial-like point-to-point data transfer and related pairing;
- [1 in LE] Generic Attribute Profile (GATT) and Attribute Protocol (ATT): clients can query a server for its properties, via attribute-value pairs;
- [1 in LE] Generic Access Profile (GAP): enables applications to set the communication roles of the node;
- [2 in LE] Security Manager Protocol (SMP): two security modes, with different levels of encryption and data integrity;
- [3 in Classic and LE] Logical Link Control and Adaptation Protocol (L2CAP): multiplexes access from higher layers to the link layer.

```
Classic                 Dual                BLE
SPP                   SPP | GAP GATT      GAP GATT
RFCOMM              RFCOMM | SMP ATT       SMP ATT
L2CAP                    L2CAP             L2CAP
Link Manager   Link Man. | Link Layer     Link Layer
BR/EDR PHY        BR/EDR + LE PHY          LE PHY
```

Connections are either one-to-one oriented (like Classic) with a similar central-peripheral (master-slave) architecture relying on advertisement, or connectionless in which advertisements are non-connectable and we have broadcasters (TX) and observers (RX). While connections are secure, advertisements are not.

This allows the creation of mixed topologies, in which peripherals can have multiple roles at the same time in different connections: one can be central, or master, or peripheral for something else. Some of them will keep scanning without pairing.

Device discovery works as following. Broadcasters alternate between the three channels, sending advertisements (3 back-to-back packets) on each one at 10ms intervals, hoping that the observer will be listening in one of them. When the scan and the advertisement overlap in time, a connection can be estabilished. Meanwhile, observers will alternate scanning (on all three channels) and idling. The sum of the scan duration and idle duration is called scan interval.

There is no estabilished rule for setting these values: they are arbitrary, as long as they are a multiple of 0.625 ms. There are some constraints, like a minimum of 20ms for connectable advertisements, 100ms for non-connectable. Of course, the scan duration should be less than the scan interval. After receiving an advertisement, the observer can request a second one, allowing more data to be exchanged.

In practice, BLE is available on every smartphone, although sometimes with partial compliance.The scan/advertise mechanism has been used as a thin layer for COVID exposure apps (GAEN). Moreover, in the relative past there was a technology called *Proximity Beacons*, developed by Apple (iBeacon) and Google (eddystone), which enabled service triggering when nearby devices passed by.

Bluetooth 5 (v5)

It is basically BLE 5.0, but brings several enhancements from higher data rate (2 Mb/s), to longer ranges, to a tremendous increase in max nodes (from 7 up to 32767), and support for mesh networking. Version 5.1 also introduces support for localization.

It supports three connection types:

- LE 1M Uncoded = BLE, 1 Mb/s
- LE 2M Uncoded -> different physical modulation, 2 Mb/s
- LE 1M Coded -> 4x range, 250 kb/s or 2x range, 500 kb/s. Adds Forward Error Correction (FEC) and higher transmission power (up to 20 dBm).

Advertising has a bigger payload (254 B), uses all 40 channels, and has a shorter interval (20ms, same for connectable and non-connectable).

In Bluetooth 5, they target explicitly the notion of mesh. It adopts a publish-subscribe topic-based flooding model. Messages from publishers are disseminated to all nodes subscribing to the message topic. Nodes can implement one or more roles, *relay*, *proxy* (towards BLE 4.0 nodes), *low power* and *friend* (nodes are always scanning; to save energy, LP nodes turn off and ask their friends to store their data).

Continuous Neighbor Discovery (CND)

Netowrking in WSN may need nodes to know the position of other nodes in the network. If the nodes are static, then this info may be obtained as part of the protocol. Mobile nodes need a different approach.

Applications of CND include museums, proximity detection of people, COVID applications, children monitoring, and more.

Encounters are in general unplanned and unpredictable. This is a problem for WSN duty cycles, which are required to save as much energy as possible. No matter what, if we want to enforce discovery we will have to deplete the battery more. To make matters worse, global synchronization is usually not an option.

In literature, the dominant approach for an extended period of time was time slotting. Nodes wake up and sleep depending on a schedule (just like previous protocols). At the start and end of an awake slot, nodes send a beacon. When two awake slots of two different nodes overlap, discovery is triggered.

The problem becomes how to create the schedule of a timeslot (the number of times in which I'm active, divided by the total number of slots), aka a **scheduling** problem. Since we have no syncronization, we must deal with unsyncronized slots (perfect aligment is worse as it may cause collisions). We must also deal with asymmetric duty cycles (nodes may have less batteries)

Probabilistic protocols (BDay, Disco, U-Connect, Searchlight)

The **birthday protocols** (2001), inspired by the birthday paradox (the probability that 2 people in a room share a birthday is usually high), are a family of protocols that try to apply this notion to neighbor discovery. Nodes randomly select a slot to wake up in with a given probability. While the probabilities are quite high, there are no guarantees that nodes may be discovered.

Disco (2008) builds on the previous idea by using prime numbers. A node i chooses two primes, p1i and p2i, and wakes up each p1ith and p2ith slots. Two nodes i, j are guaranteed to discover each other within:

$$\min(p_{1i} \times p_{1j}, p_{2i} \times p2j, p_{1i} \times p_{2j}, p_{2i} \times p_{1j})$$

This is guaranteed by the Chinese Remainder Theorem. This also guarantees asymmetric duty cycles, although nodes choosing smaller primes will spend more energy trying to ping (e.g. 3 vs 107). This usually results in the same primes being chosen by all nodes.

U-Connect (2010), instead, employs only one prime p. Each node is in discovery mode each p-th slot, then transmits for (p+1)/2 slots every p^2 slots. This approach guarantees discovery within p^2 slots, but does not take into account collisions.

Searchlight (2012) was one of the first protocol to take a complementary approach between birthday and prime-based protocols. The idea is to have a deterministic discovery with a pseudo random component.Consider two nodes with the same duty cycle. Select a fixed period t, and keep a slot fixed (anchor slot), that repeats periodically. Thus, we have something like this:

```
(A)  | A | p | p | p |   |   | A | p | p | ...
(B)  |   |   | B | p | p | p |   |   |   | B | ...
```

We now add a probe slot (p in the picture). Its objective is to meet the anchor slot of the other node, and searches in the range 1 to t/2 (so half until the next anchor slot). In this semi-period, we can just move the probe around rather than probing all of them at once. We can now use two approaches:

- Sequential probing: the position of the probe slot is incremented by one each time. The overlap is guaranteed within 1/2 * t^2 slots.
- Random probing: the position is randomly chosen. It improves average case performance.

When slots are not aligned, it is sufficient to have one overlap to allow discovery. With this, we can reduce the number of active slots by almost 1/2. On the other hand, if the slots are too aligned, we need to let slots overflow a little bit (this also helps with clock drift).

Many of the aapproaches don't go into the details of what happens into a slot. Slot sizes are usually 10ms, but this is arbitrary, and is impossible to implement in some LP radios. Moreover, it does not take into account collisions.

BLEnd

BLEnd (2017) is a protocol that tries to bring predictable and real-life performance to low-power neighbor discovery. Three observations:

- slots are not fundamental, what matters is only the fact that when I'm sending a beacon that signals my presence, someone else is listening - i.e. the schedule;
- unidirectional discovery is usually enough, no need to have both devices match and waste energy;
- supporting domain experts requires a SLA (service level agreement) that takes collisions into account and is based on the network size and situation. This allows to have more realistic performance estimates.

Given these, we can find two simple elements in a schedule: a beacon ||, and a scan |----|. In a given epoch, there will be a schedule in which scans and beacons will be mixed. In *unidirectional discovery* (**U-BLEnd**), nodes will be active for half of the time (saving a lot of energy). In bidirectional discovery, the schedule is *fully extended* (**F-BLEnd**). We also have a third mode, **B-BLEnd**, in which the next scan timing is stowed in the beacon, and we can let the target make an estimate on when we will be available. This will allow him to schedule an extra beacon that will speed up discovery.

BLE has some annoying constraints that kinda corner the protocol into some small details, like:

- BLE listeners must receive the entire beacons
- BLE adds random slack to advertising intervals

Moreover, advertising on three channel enables discovery by more nodes at the cost of a potential waste of time and energy.

Given a domain expert’s neighbor discovery needs, the goal is to determine the values of L, A, and E that parameterize BLEnd, in order to achieve a target discovery latency (Λ) at a certain probability, while minimizing energy consumption. With only two nodes, discovery is guaranteed in an epoch. With more, beacons can collide, and the random slacks will help in the following epochs.

Judging the protocol at a scale, if a network has 100 nodes, the probability that they collide with each other increases drammatically. Vice-versa, if a node is alone it needs to send many beacons in order to hope to get discovered. This means that less beacons are needed and the scan interval is larger, when the network is large.

Localization

The previous section talked about the notion of proximity. However, proximity detection requires only an estimation of distance. Localization provides the (x,y,z) position of the target in a coordinate system, and in some settings, a far more accurate estimation is needed.

To begin with, we assume some reference points, called *anchors*, exists in a known point in the relevant space. We perform localization by determining the relation of a given node w.r.t. these anchors. Once the relation is known, then localization can be computed.

Localization may be distinguished between *with devices* (the target for localization) or *device-free* (such as the ones used by radars).

- RFID
  - good for proximity;passive tags can be easily embedded in “things” and are battery-less
  - poor ranging accuracy; requires dedicated infrastructure (readers); short-range (typically <1m, but up to tens of m);
- Wi-Fi
  - ubiquitous indoor; localization + communication
  - energy-hungry; coarse (m) positioning accuracy;
- BT
  - ubiquitous on personal and IoT devices; localization + communication; energy-efficient;
  - coarse (m) positioning accuracy; short-range (tens of m)
- UWB
  - supports ranging at PHY level; good accuracy (<10cm); localization + communication; long range (~300m)
  - requires dedicated infrastructure (anchors); energy-hungry (more than BLE but less than WiFi)
- Ultrasound
  - speed of sound (~343 m/s) is lower than light, enables looser time synchronization; precise (cm);

    - energy-hungry; short-range (few m)
- Infrared
    - typically used for proximity applications; low-cost and low-energy,
    - short-range and requires line of sight
- IMU (inertial measurement unit)
    - measures acceleration and orientation using accelerometers and gyroscopes
    - suffers from accumulated error
- laser
    - very precise (mm to cm) and long-range (hundreds of m)
    - very energy-hungry; very directional
- optical
    - time synchronized cameras capture 2D images that are then triangulated. Has sub-mm accuracy for fast moving objects
    - extremely expensive, short range, needs line of sight, requires processing power, markers, calibration, and is susceptible to environmental noise.

Computing the position

In anchor-based localization, **lateration** refers to the fact that nodes can determine their own position using circle Euclidean equations. This results in a linear system of equations with n unknowns and n equations (one for each anchor). Can be solved with a least-squares method.

Fingerprinting, on the other hand, relies on the fact that each position has its own characteristic power signature (RSS). Used with BLE and Wi-Fi, it requires a large database of measurements, and depends on the environment itself.

Dead reckoning predicts the position of a moving node based on previously estimated positions or w.r.t. fixed points, using Kalman filters. It requires no previous infrastructure, but suffers from an accumulated error over time. It is used often in conjunction with GPS to mitigate the error.

Measuring quantities

Depending on the technology and/or positioning technique at hand, the localization system measures different quantities. Those typically used in RF systems are:

- **Received Signal Strength Indicator** (*RSSI*) -> since radio signal attenuates over distance, it can be used as a measure
- **Time of Flight** (*TOF*) -> measures the time a signal travels from the transmitter to the receiver. Distance can be estimated by multiplying the TOF by speed of light.
    - One-way ranging: with synchronized clock, t1 - t0
    - Two-way ranging: the most common technique
    - Single-side TWR: does not require clock synchronization. Given a "ping-pong", calculate it with 1/2 * (t4 - t1 - (t3 - t2))
    - Double-side TWR: further ping-pongs to reduce dependency on clock synchronization
- **Time Difference of Arrival** (*TDOA*) -> also known as multilateration. It is the time difference between the arrival of two signals, and is similar to how inter-aural time works in the brain.
- **Angle of Arrival** (*AOA*): with appropriate eqipment, uses triangulation between signals

Finally, the **Geometric Dilution of Precision** (*GDOP*) is a metric used to characterize the performance of an anchor-based localization when geometric constraints between anchors are imposed.

In practice each radio uses a different technology. Wi-Fi and Bluetooth use RSSI and fingerprint, but some access points use *channel state information* and Wi-Fi RTT (that enables calculation of TOF).

BT 5.1 supports AoA and AoD (angle of arrival and angle of departure). It requires a bulky and expensive antenna to acquire the angle information, but allows cm-level accuracy.

Ultra-wideband ranging

UWB are the hot trend in low-power networks. Ultra-wideband is a technology for transmitting information across a wide bandwidth (>500 MHz), and at very high frequencies (3-10 GHz). The signal propagates with extremely short pulses, often in the order of 1-2 nanoseconds. This allows the employment of UWB as a very accurate localization source, with cm accuracy, low power, and nowadays small footprint (chips have been included into high-end smartphones now). Finally, UWB can better disambiguate multipath and determine the signal arrival time.

The DW1000 is a UWB transceiver chip, compliant with IEEE 802.15.4-2011. It supports six overlapping channels (two of 900 MHz, others of 500 MHz) and fair data rates (6.8 Mb/s at 100 m to 110 Kb/s at 600 m).

The UWB radio works by sending impulses, and the energy of the signal is spread across its large bandwidth. Timestamping can occur at the first pulse of the packet header (*PHR*), after the start frame delimiter (*SFD*). However,

ranging can occur on any packet as long as the relevant flag is set. Pulse repetition frequency (*PRF*) is set to 16 or 64 MHz.

With UWB, localization can be done with *two-way ranging* (with N anchors, send 2N messages, very scalable) or *TDOA* (a single message, but requires synchronization). The latter's time synchronization must be very tight, and usually employs an anchor as a time reference for the others. Such anchors, knowing the distance between them and the clock drift, can set their clocks with ns accuracy.

Finally, UWB supports concurrent ranging. It relies on simulataneous responses from multiple nodes with a single polling beacon. When each response arrives, the sender calculates the time of arrival by verifying the relative peak. This enables significant energy savings, as just two messages are required. However, it is not supported by hardware and requires a sophisticated, low-level firmware.

Social distancing in IoT

In the era of COVID-19, several applications have been developed to employ localization between mobile devices to check for social distancing. With people always in motion, it is hard to continously discover neighbors while accurately estimating their positions.

Janus is a protocol that adds BLEnd's network discovery capabilities to UWB ranging. Moreover, BLE coordinates ranging exchanges, to avoid collisions. It is very energy efficient, due to the smart alternation of BLE scans and UWB ranging.

Recent advances and trends

Cross-Technology Communications (CTC)

Nowadays, several networks are already deployed, each with its own characteristics. There exist some researches that try to let Wi-Fi, ZigBee and BT work together. A possible solution is using a gateway as a translator, but it is not cheap and neither flexible.

Some efforts to create CTC at the packet level, thus "translating" the packets between different networks, are being done. For example, **WeBee** translates Wi-Fi packets to ZigBee packets: the sender emulates ZigBee signals and adds padding to the packet to match the WiFi one. ZigBee receivers discard the padding and read the corresponding frame. The issue comes from translating ZigBee time domain signals to WiFi frequency domain signals. This emulation brings inevitable errors.

Another approach is by using binary phase shift of the signal, which is key to the ZigBee receiver.

Wake-up radios

Wake-up radios are a recent technology that allows to wake up a sleeping device by sending a signal. They complement the main node's radio, and provide lower power consumption and always-on functionality. They are not used as communication and do not require MAC layer coordination.

Key requirements include:

- low power consumption, in the order of microwatts
- low latency, tens of ms for waking up the main radio
- resiliency to noise from other wake-up calls
- range in the tens of meteres
- fair data rates, a good balanceto reduce time on air but also to increase range
- 5% the cost of the main radio

Wake-up calls can be distinguished between *directed* and *broadcast*. The former target node IDs in the signals, and leads to significant savings in energy consumption, the latter "wakes up all the neighborhood", and is simpler to implement.

Moreover, we can distinguish between *in-band* (same frequency as the main radio) and *out-of-band* (requires different transceiverse); *active* (requiring an energy source) or *passive* (that use energy harvesting, such as a plant fuel cell).

Ambient backscatter

In conventional radio system, energy sources provide the radio capabilities for generating the signal. In **backscatter communication**, transmission occurs by reflecting and properly modulating a signal generated by the receiver.

Such an example is *_RFID*. It can be:

- passive -> has no battery, is cheap, and has a short range (cm) - most common;
- active -> has a battery, is expensive, and has a long range (m) - that's not backscatter...;
- semi-passive -> uses a battery only for the circuitry of the reader, and this power is used as a backscatter for the recipient.

RFID can also be fully programmable: it employs an-on board small MCU, and employs the reader's backscatter like a passive RFID. However, it also harvests energy from the reader to power the MCU itself.

The basic idea of **ambient backscatter** is to use the environment's signals, which are well present (TV, Wi-Fi, cellular...). This enables truly battery-free, infrastructure-free devices. To communicate using ambient backscatter, the sender changes the antenna properties of an already present signal, such as the one from the TV. The TV receiver will ignore the signal, discarding it as noise, but the receiver will be able to decode it. In the last years, a few prototypes have appeared, also using Wi-Fi signals.

Visible Light Communications (VLC)

Visible Light Communications (VLC) is a new technology that modulates "visible" light to transmit information. LEDs are usually used, switched on and off quickly, and are unnoticed by the human eye. They strike their own set of advantages: high bandwidth and data rates, zero interference with other signals, no wall penetration (if needed for security), efficiency.

Nowadays, they are in early stages in development, as they are too big to be deployed in the real world. However, some prototypes have been developed, and are being tested in the field. OpenVLC, for example, has a mote-like form factor and has a 4m range with 400 Kb/s.

Conclusion

Each technology strikes different tradeoffs, catering for different application scenarios in IoT. Hardware vendors offer boards and chips equipped with multiple radios nowadays: for exampke, an iPhone 11 has BLE, Wi-Fi, UWB along with regular cellular connectivity, and maybe in the future even LoRa will appear. Each of them has their set of advantages and disadvantages. The question now is how to use these radios together, rather than to find a clear winner in a field that is still evolving.