Mobile and Cyber-Physical Systems

Contents

Wireless Networks

A wireless network is a network of mobile hosts connected by wireless links. Such networks can operate using an infrastructure with base stations or without any centralized coordinator as an ad-hoc network.

A base station is a device typically connected to a wired network and it's responsible as a relay between the wireless hosts in its area and an outer network. Other than the different architectures, wireless networks could be classified by the number of hops needed to deliver a package.

Multiple issues are specifics of the wireless medium, for instance:

- Decreased signal strength, the radio signal attenuates quicker as it propagates respect to the wired attenuation.
- Interference from other sources, the standard wireless frequencies are shared by multiple wireless devices, and possibly by other tools (engines, microwave ovens, etc.)
- Multipath propagation, the radio signal reflects off objects or ground, arriving at destination at slightly different times.

Two possible metrics to validate the effect of these issues are the signal-to-noise ratio (SNR) and in the bit-error-rate (BER). These problematic characteristic of the medium also produce multiple challenges for wireless communication:

- Limited knowledge, a terminal cannot hear from all the others in the network, producing the hidden/exposed terminal scenarios.
- Limited terminals, the battery life, the memory, the computational power and the transmission range of a device are limited.
- Mobility/Failure of terminals, that can move in the range of different base-stations or move away from each other.
- Privacy, undermined by the easiness eavesdropping of ongoing communication.

CSMA/CD

In a wired network a single channel is available for all the stations. If frames are sent simultaneously the resulting signal is garbled, generating a collision detectable by all the hosts.

Given these context many different protocols were developed, culminating in the

widely adopted Carrier-sense multiple access with collision detection (CSMA/CD) protocol:

- When a station has a frame to send, it listens to the channel to check if it's busy, waiting until it becomes idle.
- If a collision occurs the station waits a random amount of time and repeats the procedure.
- If a station is communicating and detects a collision it aborts immediately. (CD)

The random amount of time is selected by the binary exponential backoff technique. This method uses contention slots equal to the double of the time necessary for the farthest stations to communicate. After the i collision each station waits a number of contention slots comprised in the interval $[0, 2^i - 1]$, up to a maximum of $i = 16$ after which the failure is reported to upper levels.

In CSMA/CD the interferences are detected by the transmitter, while in the wireless networks what matters are interferences at the receiver. Using CSMA/CD in wireless networks leads to two known problematic scenarios: the hidden terminal and the exposed terminal.

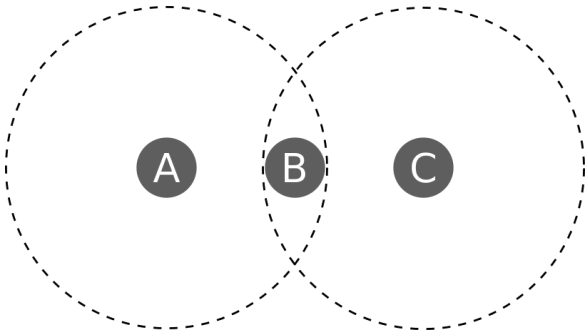


Figure 1: Hidden terminal

In the hidden terminal problem there are three stations, `A`, `B` and `C`. Suppose that `A` and `C` want simultaneously to send a packet to `B`, and that `A` is not in the range of `C` and vice versa. Both the senders would sense the medium as free, while `B` will only receive noise due to the collision in its range.

Suppose now that `A` and `C` are both in the range of `B` and that `B` is sending a packet to `A`. In the exposed terminal scenario using CSMA/CD `C` must refrain to send any packet, while there could be a station `D` out of the range of `B` to which `C` could communicate without any problem.

MACA protocol

The Multiple Accesses with Collision Avoidance (MACA) protocol offers a simple solution to overcome the technical limitations of CSMA/CD in wireless networks.

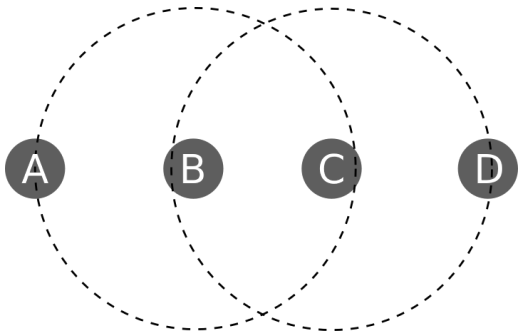


Figure 2: Exposed terminal

The basic idea concerns stimulating the receiver using a short frame (Ready-to-send, RTS) into transmitting a short frame in turn (Clear-to-send, CTS). Once the CTS has been received from the sender it can start transmitting the data frame.

The other stations hearing the CTS should refrain from requesting the media during the transmission of the data frame, of which the length is known since it is contained both in the RTS and the CTS. This is sufficient to solve both the hidden and the exposed terminal issues.

If a station receives multiple RTS detects the collision and doesn't respond with a CTS. The sending stations noticing that no CTS has been received assume a collision at the receiver and so they start binary exponential backoff before retrying.

The MACAW protocol offers various improvements over MACA:

- An ACK frame is introduced to acknowledge a successful data frame.
- Carrier sensing is required to keep a station from transmitting RTS when a nearby station is also transmitting an RTS to the same destination.
- Exponential backoff is run for each separate pair source/destination and not for the single station.
- A mechanisms to exchange information among stations and recognize temporary congestion problems is introduced.

IEEE 802.11

IEEE 802.11 is part of the IEEE 802 set of LAN protocols, and specifies the set of media access control and physical layer protocols for implementing wireless local area networks in various frequencies, including but not limited to 2.4 GHz, 5 GHz, and 60 GHz frequency bands. Introduced in the late 90s, different versions have been released during the years, improving the physical layers but leaving the unchanged the use of CSMA/CA in the MAC sublayer.

In the 802.11 architecture a group of stations operates under a given coordination function, this function can be decentralized (Distributed Coordination Function, DCF) or controlled by a unique station (Point Coordination Function, PCF). The presence of a base station, know as access point (AP), requires the other stations to communicate with each other by channeling all the traffic through the centralized AP. The access point can provide connectivity with other APs and so other groups of stations via a fixed infrastructure. In the absence of an AP, 802.11 supports ad-hoc networks which are as always under the control of a single coordination function, but without the aid of an infrastructure network.

The DCF must be implemented by all stations, and DCF and PCF can be active at the same time in the same cell.

Distributed Coordination Function

In the DCF the carrier sensing is performed both at the physical and the virtual level, and the channel is marked busy if either one of this two indicators is considered to be active:

Physical carrier sensing detects the presence of other active nodes by analyzing all the detected packets and detecting any activity in the channel due to other sources. Virtual carrier sensing is performed by sending duration information in the header of an RTS, CTS or data frame and keeping the channel virtually busy up to the end of the data frame transmission.

The access to the medium is controlled through the use of growing inter-frame spaces (IFS): Short IFS, Point IFS and Distributed IFS.

This is the essence of the DCF based communication:

- The source senses the channel until it becomes idle, then waits DIFS and checks if the channel is still idle.
- The source sends the data, all the stations detecting the frame set a Network Allocation Vector (NAV) to mark the channel virtually busy for the time required for the data to be transmitted and the ACK to be received.
- The destination waits SIFS after receiving the data and then it sends the ACK.

A collision could happen because of multiple stations sending a data frame in the same instant, in that case the whole frame is sent before using binary exponential backoff to resend. The contention time depends on the physical layer, anyway the backoff can be interrupted if the channel is sensed as busy and then the timer reprises after DIFS when it becomes idle again.

To avoid this situation, the RTS/CTS mechanism already seen in MACA can be implemented for all the possible frames or only for the enough big ones, in this case the other stations set multiple shrinking NAVs for the RTS, CTS and the data frame.

Also fragmentation can be useful to improve reliability, the frames are sent in sequence but each one waits for the acknowledgement of the previous. In the case a fragments isn't correctly delivered the transmission reprises from that last one and not from the start, even in this case the NAV is specialized for the different fragments length other than for the RTS/CTS. It should be noticed that if present, the RTS/CTS mechanism is used only for the first fragment.

Point Coordination Function

In this operative mode the communication is under the control of the point coordinator (PC) that performs polling and enables stations to transmit without contending for the channel. As previously stated this method must coexist with DCF.

According to a repetition interval, `CFP_RATE`, the PC periodically waits PIFS and sends a beacon frame to notify the start of a portion of time allocated for contention-free traffic. The contention-free portion has a maximum time duration `CFP_MAX_DURATION` shared between all the stations, but the actual length is determined inside the beacon frame by the PC. The contention-free period is always closed by a Contention Free End (CFE) frame.

Inside the contention-free period the communication are always initialized by the PC station, that can send data and/or a poll frame to a station, this can be done multiple times inside the same contention-free window. After having received a beacon frame each station sets a NAV for the duration of the contention-free period, in that period they can only respond to data received using SIFS+ACK, or send data to any other station if polled by the PC.

With this model PC can also send to a non-PCF aware station that only has DCF, since this station will respond with an ACK, also messages can be fragmented as in DCF.

Frame format specification

A frame of the 802.11 protocol most contain other important informations other than the obvious source and destination addresses and the data buffer.

The version of the protocol is needed, since given the interoperability at the MAC layer different protocols can coexist in the same cell. There are three types of frames: management (association/dissociation, synchronization, authentication), control (handshaking, acknowledgement) and data (data transmission, possibly combined with polling and ACK in PCF). Also the subtype of the frame may be specified to identify special frames like RTS, CTS and ACK.

Further informations regard the fragmentation, the encryption, the estimated time to complete the transmission and the CRC for error detection.

Cellular networks

A cellular network is composed by a set of a cells, each one one covering a distinct geographical region and connected by a Mobile Switching Center (MSC) that also manages the call setup and handles mobility. A cell is composed by a base station (BS), that has analogous functions to an access point of 802.11, and its mobile users.

The radio spectrum used for the mobile-to-BS communication can be handled in two different ways:

- Combined FDMA/TDMA, where the spectrum is divided in frequency channels, and each channel into time slots.
- CDMA, Code Division Multiple Access.

A Base Station System (BSS) is composed by multiple base stations grouped by a Base Station Controller. It should be noticed that up to 3G the data communication was parallel in respect to the voice one, that still used old circuit switching techniques.

Mobility

Each host in a network holds a permanent address in its home network, this address can always be used to reach the host. The home agent is the entity that will perform mobility functions on behalf of the host when the host is not in the home network but in another visited network. In a visited network a care-of-address is assigned to the host, and a foreign agent also have to cooperate with the home agent to perform mobility functions.

The mobility could be handled by the routing mechanism, letting the routers advertise permanent address of mobile-nodes-in-residence via usual routing table exchange, but this cannot scale to millions of mobile devices like it's required nowadays.

The alternative is to let the end-systems handle the mobility. The first step is the registration of a mobile-device in a visited network: the mobile contacts the foreign agent and receives the care-of-address, after that the home agent is notified that its mobile device is resident in the visited network.

After the registration the communication can be routed in two ways: indirected or directed.

In indirect routing, a correspondent communicates to the home network that is responsible to forward the packets to the visited network, this replies directly to the correspondent. It should be noticed that the foreign agent functions may be done by the mobile itself. Also the mobile user can maintain on going connections by simply notifying the home network of the new position by the next foreign agent. This technique is obviously inefficient if the correspondent and the host are in the same visited network, because of the necessary triangularization.

In direct routing the correspondent gets the address of the visited network from the home agent and then it communicates directly with the mobile device in the visited network. In this way the communication is not transparent, also if the mobile device changes visited network chaining is used. In chaining the first visited network is called the anchor, and all of the traffic must be forwarded to the anchor even when the mobile changes visited network.

Global System for Mobile communications

In the context of GSM cellular networks, the home network is the network of the provider the host is subscribed to (TIM, Vodafone, etc.) that maintains a database called home location register (HLR). A similar database is kept by the visited network containing all the actual connected devices, called visitor location register (VLR).

In GSM indirect routing is used, so the correspondent packets pass trough the home network and the visited network, consulting respectively the HLR and the VLR.

Inside the same network, so under the same Mobile Switching Center the mobile could change Base Station System using a procedure called handoff initialized by the old BSS. There are various reasons to perform handoff, like signal power issues and load balance. Anyway the GSM standard only defines how to perform handoff and not the reasons why it should be performed.

The handoff protocol is the following:

1. Old BSS inform MSC of impending handoff, providing a list of BSS
2. MSC choses one BSS and notifies it to the selected one
3. New BSS allocates radio channel for the upcoming mobile device
4. New BSS notifies, through the MSC, that it's ready
5. Old BSS notify the mobile device to perform handoff to the new BSS
6. Mobile device contact the new BSS to activate the new channel
7. Mobile notifies the MSC trough the new BSS that the handoff is complete
8. Old BSS releases the resources after the MSC notifies it of the outcome of the overall operation

Handoff could also be performed between different MSCs, in that case chaining mechanism is used between all the traversed MSC.

Managing mobility in ad-hoc

In ad-hoc networks we can recognize different goals to achieve:

- Quickly adaptation to topology changes
- No centralization
- Loop free routing
- Load balancing among different routes in case of congestion

- Supporting asymmetric communications
- Low overhead and memory requirements
- Security

The proactive approach attempts to maintain consistent and updated routing informations over the network. Instead in the reactive approach the route to a destination is discovered only when it's desired by the source node.

In the following discussion we will represent each host as a node in a graph, a directed edge from a node to another represents the faculty to directly send a packet between these hosts.

Dynamic Source Routing

Dynamic Source Routing, DSR, is a protocol proposed in 1994 and based on the following assumptions:

- The nodes are cooperative, all of them want to participate fully in the network protocol and will forward packets for other nodes.
- The network diameter is small, the number of hops needed to travel from any node at the extreme edge of the network is small (5-10) but greater than 1.
- Corrupted packets can be recognized and discarded by their destination.
- The nodes may move in the network at any time without notice, but their speed is moderate with respect to the packet transmission latency, meaning that in general the topology change should be infrequent and unlikely to occur during the route discovery.

The route discovery (RD) is the mechanism by which a node wishing to send a packet obtains a route to the destination. The other basic mechanism of DSR is the route maintenance (RM) by which the source route is able to detect that topology has changed and so that the route for the required destination is no longer available.

Suppose now that a node `S` wants to send a packet to a node `D`. The source node searches for a route to `D` in its cache, if it finds it places the route in the header of the new packet and sends it, otherwise it start a route discovery protocol sending a route request (RREQ) via local broadcast. Each copy of the RREQ contains the unique IDs of the source and the destination, and the list of nodes through which the particular RREQ has been forwarded.

When a node different from `D` receives the RREQ it checks if it already received the same request or if it is already in the accumulated path, in that case it discards it, otherwise it appends its unique ID and local broadcast the updated RREQ. When `D` receives the RREQ returns a reply to S (RREP) including a copy of the accumulated route record, if the destination has not a path to the original source it has to start a route discovery possibly combined with the RREP, this is obviously not needed if all the links are bidirectional. The target can receive

the same RREQ from different paths, it can then independently chose one of them based on latency, number of hops, energy levels or other metrics.

While the packets to be sent wait for a route discovery to be completed they are kept in a send buffer, they can expire and be deleted after a timeout, also they can be evicted with some policy (FIFO) to prevent send buffer from overflowing. Also the source node occasionally starts a route discovery for the nodes in the buffer. This is done by using exponential backoff to delay route discoveries for the same destination and avoid overflowing the network in case the target is unreachable.

When sending a packet along a route each node is responsible of the receipt of the packet at the following hop. If a device detects that a route link is down a route error is sent back to `S`, that then removes all the routes containing the broken link from the cache. The route error is also reported to the upper layers.

A possible improvement of the protocol consist in storing overheard routing information. This means that a node not only caches the results of a route discovery, but also the route accumulated in a route request/reply or in the header of a data packet. Also it's possible to reply to route requests using cached routes, sharing the internal knowledge of the node without local broadcasting the request up to the target. In this case many nodes may have a route, so to avoid collisions and to favour shortest routes an host must wait for a random period before replying to the route request, this period is computed as following:

$$D = L(H - 1 + R)$$

Where H is the hop-length of the route to be returned, R is a random number between 0 and 1, and L is a constant at least twice the maximum wireless link propagation delay.

Further improvements are:

- Limit the time-to-live of RREQ
- Automatically cut out intermediate hops that are no longer needed
- Cache broken links to prevent their use in other routes

The DSR packets are standard IP packets that uses special flags in the option header.

Ad-Hoc On-Demand Distance Vector (AODV)

AODV is a protocol with the same goals of DSR, but that also integrates unicast, multicast and broadcast messages. The assumptions are similar to the ones for DSR, the nodes must be cooperative but also the links must be bidirectional, this could be enforced by pruning unidirectional links.

The protocol uses one route table for unicast and one for multicast. Each table contains at most one route for each destination, maintaining the next hop and a

precursor. If the entry is not used within a lifetime timer it is removed.

Each node also maintains a monotonic sequence number, this value is increased each time a node detects a change in its neighborhood. Each multicast group maintains a separate sequence number.

When a node `S` originates a packet for a destination `D` non present in it's routing table it broadcasts a route discovery message RREQ and sets a timer to wait for a reply. Since flooding the network with RREQ could be expensive a growing TTL is used by the source to try different route requests up to a fixed constant maximum, when a route is established the distance to the destination is recorded to set the initial TTL in the next route discovery for the same destination. The RREQ contains:

- IP address of source and its current sequence number, which is incremented
- IP address of the destination and its last known sequence number
- Broadcast ID, which is incremented
- Hop count initially set to zero

When an intermediate node, known also as rely, receives a RREQ checks if it has already received it by comparing both IP source and broadcast ID with a cache of all received RREQ in the near past. If the request is new it is processed by inserting in its routing table the reverse route for the source node, to be able to forward to it a RREP if it is received. This reverse route entry has a lifetime, if not used it is deleted to prevent stale info in the cache.

A rely node is able to directly respond to a RREQ via unicast if it contains a unexpired entry for the destination in its route table with a sequence number at least great as the one included in the RREQ, this is needed to avoid loops. If a rely node is not able to respond to RREQ it increments the hop count field in the RREQ and broadcasts the packet to its neighbors.

When the RREQ finally reaches the destination a RREP is generated, containing more or less the same information of the RREQ, but also the route lifetime and the sequence number of the destination which is incremented by one.

If a rely node receives a RREP sets up a forward path for the destination in its route table and forwards the RREP by updating the hop count with its distance from the destination.

Each node periodically sends to its neighbors special `HELLO` packets to inform its neighbors of its presence in the neighborhood, technically this packages are unsolicited RREP with TTL equal to 1 to prevent resend. For what concerns route maintenance each node back-propagates RERR packets to all its precursors, marking the unreachable destination with an infinite distance.

Multicast

Each multicast group has a leader and a bidirectional multicast tree, a message to a group is routed using the tree starting from the first node in the group

reached. A multicast group has a sequence number maintained by the leader.

When a node wishes to join a group or to send data to a group it starts a route request including the known sequence number for the group. Only the members of the multicast tree can respond to such request.

Dynamic Manet On Demand Routing (DYMO)

The DYMO protocol tries to simplify AODV and merge the good features of both DSR and AODV. The assumptions of the protocol are the same of DSR and AODV: cooperative nodes, bidirectional symmetric links, recognition of corrupted packets and slow mobility nodes.

In DYMO route discovery and route maintenance work in similar way as in AODV, using sequence numbers to prevent loops but RREQ and RREP messages are used for creating table entries for all intermediate nodes, not only for source and destination. Like DSR instead `HELLO` packets are not used, favoring timers.

The sequence numbers are incremented when a source node generates a new RREQ, when a destination node answers with a RREP or when an intermediate node adds its information in a routing packet.

Also when a node is rebooted it must not set its sequence number to 0, since this could produce loops due to old entries in other node's tables, so the sequence number should be kept in persistent memory if possible. If there is not persistent memory the node should abstain from the communication, waiting for all the possible timers regulating the deletion of the path that may contain it.

Wireless sensor networks

A Wireless Sensor Network is a peculiar ad-hoc wireless network characterized by the fact that its nodes are small autonomous sensors with low power. The sensors are equipped with little amounts of memory storage and computational power, other than the radio transceivers needed to perform communication.

Energy consumption

Of the many issues in the WSN design one of the most important is the energy consumption, since the sensors are battery-powered and so they need efficient solutions. The wireless communication is obviously costly and so the radio should be turned off as much as possible, this should be applied to all the components in a sensor. In any case switching on and off the components consumes power as well, so a balance must be found.

The activities of a sensor are mostly repetitive: sense, process, store, transmit and receive. Each sensor alternates a period of activity and a period of inactivity, this defines a duty cycle.

Formally the duty cycle of a component μ is the fraction of the period in which the component is active.

$$\sum_{s \in S_\mu} dc_\mu^s = 1$$

The energy cost of any component μ can be computed as:

$$E_\mu = \sum_{s \in S_\mu} c_\mu^s \cdot dc_\mu^s$$

The total energy cost for the sensor is obtained as:

$$E = \sum_{\mu} E_\mu$$

Given an initial capacity B_0 for the battery of the sensor it's possible to compute the ideal lifetime of a sensor as:

$$n = \frac{B_0}{E}$$

Actually the battery is an object that loses its capacity during the time, so also the battery leaks must be formalized:

$$n = \frac{B_0 - L(n)}{E}$$

Given that the overall battery leaks L also depends on the lifetime the formulation of a problem leads to the following recurrence equation, where the term ϵ represents the leaks per cycle and $\gamma = 1 - \epsilon$ is used to simplify the notation.

$$B_n = B_{n-1} \cdot (1 - \epsilon) - E$$
$$B_n = B_0 \cdot \gamma^n - E \cdot \frac{\gamma^n - 1}{\gamma - 1}$$

The lifetime n is so the first value for which $B_n \leq 0$.

Reducing the duty cycle is a good solution to reduce the energy consumption and improve the efficiency. Anyway it should be noticed that while turning off the processor is a local decision, turning off the radio requires a global decision, since when the node is down the sensor can't contribute to the network, that as we already have seen is a requirement of multiple cooperative routing algorithms. Because of this the decisions about the state of the radio are usually directly managed by the MAC protocols for the WSN.

Multi-hop communication and mobility

Assuming a wireless channel model without interference we can define P_A as the power used by node A to send a message and P_B^r as the intensity of the received signal at node B and β as the minimal intensity to correctly receive the message.

$$P_B^r = \frac{P_A}{PL(A,B)}$$
$$P_B^r > \beta$$

The path loss PL is assumed to be proportional to $d(A,B)^\alpha$, where d is the euclidean distance between the nodes and α is a constant depending on the environment, normally assumed in the range $[2,6]$.

Assuming $\alpha = 2$ and using the law of cosines it's possible to state that $D(A,B)^2 > D(A,C)^2 + D(C,D)^2$ when $\gamma \geq \frac{\pi}{2}$, or equivalently when the node `C` is in the circle of the figure 3. Because of this two short links should be preferred when sending a message from `A` to `B`.

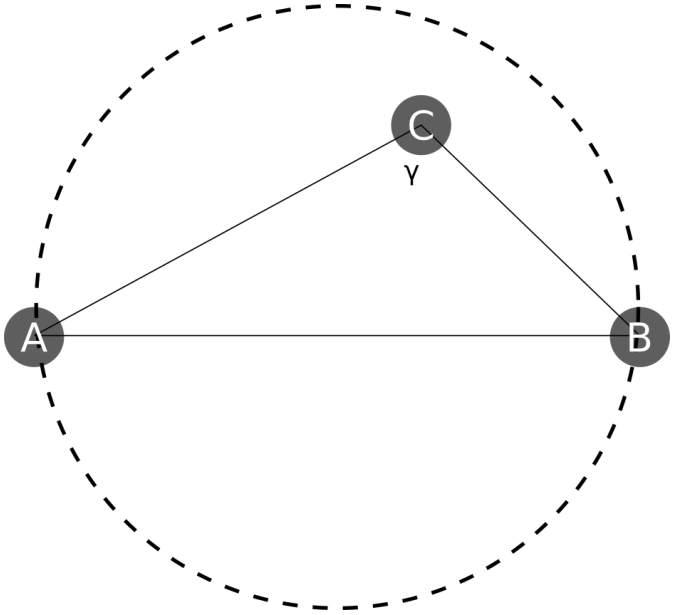


Figure 3:

Figure 4:

Another possible problem in multi-hop communication is given by the interferences. Assume that the node `A` sent a packet to node `B`, the packet is correctly received if and only if for any other node `C` that transmits simultaneously:

$$d(C, B) \geq (1 + \Delta)d(A, B)$$

Where Δ is a constant that depends on the features of the radio.

For this reason in scenario like the one in figure 4 is always better to use multiple short links than a wider one, since in this way we can allow other communications of other sensors. This can be proved by using Hölder inequality.

MAC Protocols

In a WSN the use of MAC protocols is not confined to media arbitration, but it is also useful for energy efficiency. The objectives are to reduce the radio duty cycle and to maintain network connectivity, so the tradeoff is between the energy consumption of each sensor and the performances of the whole network.

S-MAC

This protocol exploits local node synchronization, alternating listen and sleep periods when the sensor is not able to detect any incoming message. The synchronization of the nodes is a desirable condition since if the nodes are synchronized they can turn on the radios simultaneously.

Adjacent sensors synchronize their listen periods by means of periodical local broadcasts of `SYNC` packets that contain the schedule of the sensor. If a sensor detects adjacent sensors with a predefined listen period it uses the same period, otherwise it chooses its own period advertising it to the neighbors. Even after the initialization of the sensor it's possible to revert to something else's schedule, if its own schedule is not shared with anybody else.

Since a sensor can receive a packet only in its listen period, the sender may need to turn on its radio also outside its listen period, so it should know all the listen periods of its neighbors. Before sending a packet the media is sensed, if busy it tries again in the slot, the collision avoidance mechanism used is based on RTS/CTS as in IEEE 802.11.

While travelling across a multihop path a packet may have to wait in the worst case for the listen period of each intermediate node, accumulating latency at each hop. This situation should be mitigated by the fact that hopefully a certain number of sensors will converge towards the same schedule, even though there is not any guarantee of convergence. In fact depending on the topology it may be impossible for a sensor to have a listen period compatible with all its neighbors.

B-MAC

In B-MAC a node can send whenever it wants using packets that contain a very long preamble in the header. The receiver activates its radio periodically to check if there is a preamble on the air, this activity is called preamble sampling and it is based on a low-power-listening mode. Obviously the preamble should be longer than the sleep period, because of this B-MAC costs more in transmission but save energy in reception since the preamble sampling can be very short and cheap.

The energy consumption can be modeled as follows. Assume that f_c is the frequency of checking the medium for each sensor and that the medium is sensed for t_c seconds, then the duty cycle in a second of time is:

$$dc^{\text{check}} = f_c \cdot t_c$$

Given a packet length in seconds of t_d and a preamble length in seconds of t_p, the duty cycle for a transmitter is:

$$dc^{\text{tx}} = f_d \cdot (t_p + t_d)$$

Assuming ϵ to be the fraction of the preamble that has been captured by the receiver, its duty cycle is:

$$dc^{\text{rx}} = f_d \cdot (t_p \cdot \epsilon + t_d)$$

Now assuming that power in Watt required to transmit to be p_{tx} and to receive p_{rx} it is possible to compute the energy spent in Joule using for t seconds.

$$E = \sum_{s \in \{\text{check,tx,rx,none}\}} p_s \cdot dc^s$$

$$E(t) = t \cdot E$$

This simple and transparent protocol may turn to be expensive in the long run since the preamble sampling is not negligible.

X-MAC

This protocol is an evolution of B-MAC aimed to reduce the impact of long preambles. The transmitter fragments the preamble and inserts the ID of the receiver in each of this fragments. The correct receiver of the packet is so able to interrupt the preamble to request the packet.

The protocol BoX-MAC is a further development of X-MAC that sends the actual packet instead of a preamble, so that the receiver only has to respond with an acknowledgement to stop the repetition of the message.

Polling

Polling is an asymmetrical MAC technique that is combined with synchronization.

One master node issues periodic beacons, while several slaves can keep the radio off whenever they want. If the master receives a message for a slave it stores the message and advertises it in the beacon, when the slave turns on the radio it waits for the beacon and recognizing that there is a pending message it requests it to the master.

Network Protocols

Sensor networks are mostly data centric. The identification of a node is less meaningful than its capabilities, nonetheless traditional routing protocols are not practical because of the large routing tables and the size of packet headers. There are also two main issues in the network level: the implosion problem due to flooding based dissemination and the overlap problem due to sensors covering an overlapping geographical region.

It's important to use data centric routing protocols that are able to aggregate informations and are location aware to handle this characteristic scenario.

Directed diffusion

Directed diffusion is a data-centric network protocol targeted to perform distributed sensing of environmental phenomena. The sensor network is programmed to respond to location and movement-aware queries. All of the data in the network is named data generated by sensors as attribute-value pairs. The device that is in charge of querying the network is called sink, directed diffusion protocol works also in presence of multiple sinks.

To resolve a query the sink disseminates a sensing task in the network using messages called interests. Interests are composed by a sequence of attribute-value pairs that describe the task, for example the type of data requested, the interval of events, the duration of the whole sensing request and the region of interested sensors.

The nodes receiving an interest may forward the interest to a subset of neighbors if they haven't already received the same request, this is checked by using a unique ID assigned to the interest. If the node accepts an interest it must set up a gradient that contains a copy of the interest and the sensor from which the interest originally came. The interests are kept in an internal cache where they expire according to a timer.

When a sensor detects an event matching with an interest in cache starts sampling the event at the largest sampling rate of the corresponding gradients. The sensor sends sampled data to the neighbors interested, by eventually lowering the rate. The neighbors forward the data if and only if a corresponding interest, within

a gradient, is still in the cache. This behaviour could lead to the implosion problem, since the same message could arrive at different ratios at the sink.

Since each sensor is associated with a unique ID, it's possible for the sink to reinforce the paths to a subset of specific sensor to improve the quality of received that. To improve the quality of received data the reinforcement of an interest are required with higher sampling rate, this also is useful to contrast the implosion problem. Reinforcement is usually done after a first broadcast dissemination, so once the sink already started receiving data matching the exploratory interest.

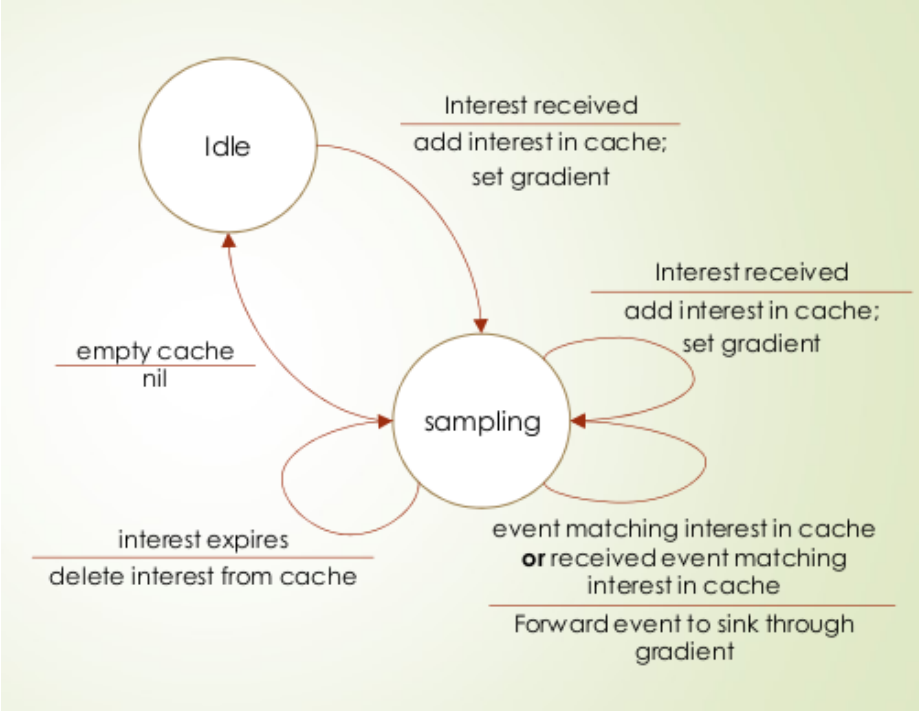


Figure 5: State machine of a sensor in directed diffusion

The protocol can be easily represented using a two-state machine as in figure 5, and it's suitable for implementation on low-end devices. It's a scalable and robust protocol that results to be effective in applications that do not require complex data aggregation/preprocessing. The implicit assumption of directed diffusion is that the overall system can survive to lost messages, considering the they will flow constantly, and to redundant messages.

The tree structure constructed by the sink is scalable but it's not completely fair for what concerns power consumption. In a grid-connected network the nodes closer to the sink are those that consume more energy. Also in an arbitrary connected network bottlenecks are common, especially between nodes that are close to sink.

Greedy Perimeter Stateless Routing (GPSR)

This protocol tries to overcome the limitations of directed diffusion and of other similar protocols, by providing arbitrary node-to-node routing and assuming limited resources and small communication overhead. The protocol is highly scalable since there is no use for route discovery and few control packets are used, without the need of large route caches.

The nodes in the network must be aware of their position and of the position of their neighbors in a two dimensional space, so that when sending a message the source knows only the geographical coordinates of the destination.

There are two different modes in which a package can traverse the network: greedy forwarding and perimeter forwarding.

Greedy forwarding In greedy forwarding, considering a packet from a node x to a destination D, the next hop is chosen as the node most closer to D.

$$y = \arg\min_{y} d(y, D) < d(x, D)$$

Greedy forwarding fails if the packets encounters a void, that means that the actual node x is closer to the destination than all of its neighbors, but it can't reach D anyway.

Perimeter forwarding Perimeter forwarding is used when greedy forwarding fails, routing around the void using a rule (usually Right-Hand-Rule RHR) to define rotation order, exploring the edges of the polygon enclosing the void. In practice suppose that the packet lands on node y from node x, using RHR the next edge to follow is the first counterclockwise edge from (x, y).

GPSR switches back to greedy forwarding whenever it finds a node that is closer than x to the destination, where x is the last node visited before switching to perimeter forwarding.

The graph of the WSN is a non-planar embedding of a graph, so edges may cross and the RHR may take a degenerate tour that does not trace the boundary of a closed polygon. For this reason, GPSR applies the perimeter mode to a planar graph P obtained from the Relative Neighborhood Graph and the Gabriel Graph of the original non-planar graph G. If G is connected then P is connected, also P is obtained from G by removing edges and it is computed with a distributed algorithm executed along with the perimeter mode packet forwarding.

Relative neighborhood graph The relative neighborhood graph of G is the set of all and only the edges that describe the minimal path for each pair of nodes.

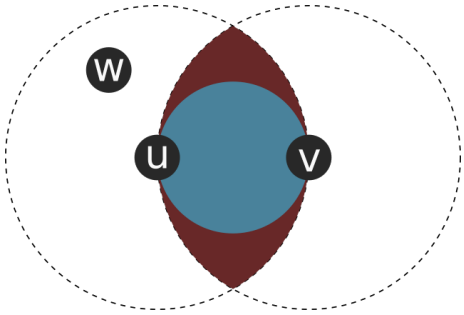


Figure 6: To select (u, v) in RNG, w must be out of the red area. To select it in the GG, w must be out of the blue area.

$$(u,v) \in P \iff (u,v) \in G \wedge d(u,v) \leq \max_{\forall w \in N(u) \cup N(v)} d(u,w) + d(w,v)$$

Gabriel graph The Gabriel graph of G is the set of all and only the edges that describe the minimal quadratic path for each pair of nodes.

$$(u,v) \in P \iff (u,v) \in G \wedge d(u,v)^2 \leq \max_{\forall w \in N(u) \cup N(v)} d(u,w)^2 + d(w,v)^2$$

The Gabriel graph keeps more links than relative neighborhood graph, and actually RNG is a subgraph of GG, anyway both are suitable to use for perimeter mode in GPRS. A visualization of the conditions for RNG and GG is in figure 6, the node w to be searched is called "witness".

A planar graph has two types of faces: the interior faces from the closed polygonal regions bounded by the graph edges, and the exterior face from the outer boundary of the graph.

When in perimeter mode GPSR uses the right hand rule to reach an edge which crosses the line passing in x and D and is closer to D than x. Using that edge, the packet moves to the adjacent face crossed by the line, at this time the current edge e_0 and the point of intersection between the line and the edge L_f are stored.

If D is reachable from x then GPSR guarantees to find a route, otherwise the packet will reach the internal face containing the coordinates of D or the external face. Not finding the node D the packet will tours around the face until it passes again through edge e_0, at that point the packet is discharged.

GPSR relies on updated information about the position of the neighbors, so it needs a freshly planar version of the graph to avoid performance degradation. Performing planarization at each topology change is not a good idea, since nodes may move within a node's transmission range, continuously changing the

selection of links by GG or RNG. A proactive approach is instead used: nodes periodically communicate their position to their neighbors. These beacon packets are used to keep updated the list of neighbors and to force planarization when the changes are excessive.

Planarization could fail due to unidirectional links caused for example by obstacles or by non circular transmission ranges. As already observed without planarization loops may arise.

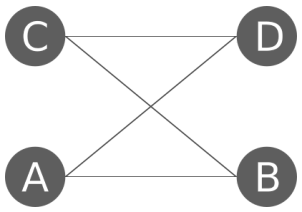


Figure 7: Cross links using mutal witness. (a, d) uses b as witness and (b, c) uses d.

Mutual witness The mutual witness protocol is an extension of the planarization algorithms. When analyzing the edge (u, v) the edge is removed only if the witness w is visible to both of the nodes, so not only the distance is considered as in standard GG or RNG creation. Anyway the procedure is not perfect, since as seen in figure 7 non planar graphs can result from the use of the mutual witness rule.

Cross Link Detection Protocol CLDP operates on the full graph, so without preliminary planarization. Each node probes each of its links to see if it is crossed (in a geographic sense) by one or more other links. A probe initially contains the locations of the endpoints of the link being probed, and traverses the graph using the right-hand rule. When a node receives a probe it controls the coordinates of all the nodes that the probe traversed, if it finds a link crossing the current link it records the information in the probe. When the probe return to the source it may decide to remove one of the crossing links.

However a link removal may result in a network disconnection, for this reason the probe counts the number of times it crosses a link. If a link had been crossed only once then it can be removed, since it means that there exist a loop and thus it is possible to reach any node in the loop by an alternative path.

Link removal may require additional communications between nodes. Assume that node w is testing its outgoing node L that crosses link L', to reduce the overhead CLDP uses some rules:

- If nor L neither L' can be removed then both links are kept
- If both links can be removed then w removes L, since it requires less communication

- If L' cannot be removed, L is removed, and vice versa

Data-Centric Storage

In a wireless sensor network there is an important tradeoff between the transmission and the storage of the informations. There are three possible ways to store data from a WSN:

1. The sensed information is sent to the sink, that then stores it outside of the WSN.
2. The information is locally kept in the sensor, it's up to the sink to flood a query in the whole WSN to retrieve it.
3. Data centric storage, where different nodes in the WSN are responsible of storing different data types, then the sink can retrieve the whole data or a summary per type.

To compare this three techniques assume now that there are $T = Q$ different types of events or queries in the network, D_{tot} is the number of events detected in a fixed amount of time and D_q is the same for a fixed type of data. Given n nodes in the network the number of packets needed to transmit an information is $O(n)$ for flooding the WSN and $O(\sqrt{n})$ for unicast. Actually there are two metrics to consider, the total usage is the total number of packets in the WSN and hotspot usage that is the maximum number packets sent or received by a given node, for instance the sink.

Type	Total	Hotspot
External storage	$D_{tot} * \sqrt{n}$	D_{tot}
Local storage	$Q \cdot n + Q \cdot D_q \cdot \sqrt{n}$	$Q + Q \cdot D_q$
DCS list	$D_{tot} \cdot \sqrt{n} + Q \cdot \sqrt{n} + Q \cdot D_q \cdot \sqrt{n}$	$Q + Q \cdot D_q$
DCS summary	$D_{tot} \cdot \sqrt{n} + 2 \cdot Q \cdot \sqrt{n}$	$2 \cdot Q$

In data-centric storage the events are named keys and corresponding data are stored by names in the network, queries are directed to the node that stores events of that key. There are so two operations supported by DCS: `Put(k,v)` and `Get(k)`. Both of these operations depend on a geographic hash function $h(k)$ that is able to produce geographic coordinates (x, y).

Perimeter refresh protocol Mobility or failure of a sensor may result in unavailability of a stored value, to provide persistence and consistency a possible solution is the perimeter refresh protocol. Assume from now on that the node u has been selected as the home node for the key k, since it's the nearest node to (x, y).

PRP ensures persistence by selecting a perimeter of u computed by GPSR and replicating all the known values of the key k in all the nodes of the perimeter.

Consistency is ensured since u constantly generates refresh packets to destination (x, y), if this packets reach a new destination v then v is elected as home node for the key k and replicates the data in itself and its perimeter. Also the replica nodes in the perimeter checks on the home node using refresh packets, all these behaviours are regulated by timers.

Structured replication The DCS should easily scale to multiple keys without overloading the nodes, structured replication is a way to load balance between multiple nodes. To each key k are associated a root and $4d - 1$ mirrors, then the node stores data in its closest mirror of $h(k)$. The retrieval of a value involves querying the root and possibly all mirrors of its mirrors.

Physical and virtual coordinates

Traditional routing protocols for ad-hoc networks are not practical because of large routing or path caches and the size of packet headers, the geographic routing appears so to be a good option. The coordinates can be obtained by equipping nodes with GPS, but this solution comes with an additional cost and it's not always feasible. When no GPS system is available there are different ways to approximate the physical coordinates of the nodes.

Identify boundaries The first task is to identify the boundary nodes of the network:

- Choose at random two bootstrap nodes
- Let the bootstraps broadcast `HELLO` packets in the network
- Each node is able to determine its hop distance from the two bootstrap nodes, the nodes with maximum hop distance are classified as boundary nodes.

Position boundaries Now that the boundary nodes are known it's possible to approximate their coordinates:

- Each boundary node broadcasts `HELLO` packets in the network
- Each boundary node is able to determine its hop distance with all of the other boundary nodes, this is called perimeter vector.
- Each boundary node broadcasts its perimeter vector to the entire network.

Assume now that $h(i, j)$ is the hop-distance between the two boundary nodes i, j and $d(i, j)$ their euclidean distance, their positions can be found as the position that minimize the following optimization problem.

$$\sum_{i,j \in p} (h(i, j) - d(i, j))^2$$

Position other nodes All of the other nodes in the network can now approximate their position (x_i, y_i) in the network by using their neighbors N_i with the following iterative process:

$$x_i = \sum_{k \in N_i} \frac{x_k}{|N_i|}, \quad y_i = \sum_{k \in N_i} \frac{y_k}{|N_i|}$$

Routing with recursive virtual coordinate An alternative is to use full virtual coordinates without trying to approximate physical coordinates, one of this is the RRVC protocol. Given two anchors A, B in a network any other node position is described as the hop-distance from the anchors, the anchors also partition the networks between the nodes nearer to A and those nearer to B. This partition can be iterated to improve the coordinates precision.

Clustering in WSN

Clustering imposes a hierarchy to a WSN whose organization is otherwise flat. In this hierarchy it's possible to identify cluster heads that are dynamically assigned by the clustering protocol. The cluster heads set up and maintain the logical backbone of the network and coordinate the nodes to improve scalability and efficiency.

A hierarchy in the network simplifies several network protocols, in particular routing and route discovery. For instance routing a packet from a node $u \to v$ is simpler using cluster heads $CH(\cdot)$. First using intra-cluster communication the message is sent $u \to CH(u)$, then using inter-cluster from $CH(u) \to CH(v)$ and finally $CH(v) \to v$.

The clusters are constructed over the graph of logical links, in general to improve the communication logical links should overlap physical links as much as possible. If a logical link does not have a physical link, it is implemented by a path of physical links as short as possible.

The clusters should also have a similar size, ideally the same size although this is often not possible. The cluster heads have an overhead that depends on the size of the cluster, a similar size of clusters means a similar overhead across the network.

Given the dynamic nature of WSN and ad-hoc networks, stability is a desired feature for a clustering protocol. Slight changes in the network topology should result in slight changes in the clustering or in no change at all.

k-neighbor clustering Using k-neighbor clustering the hop-distance between a node and its cluster head is limited as in:

$$\forall u \in N.\, h(u, CH(u)) < k$$

For instance with $k = 1$ all the logical links within the cluster are also physical links, this is the case of Bluetooth and Zigbee. Also for inter-cluster links a similar rule is applied to guarantee small paths of physical links.

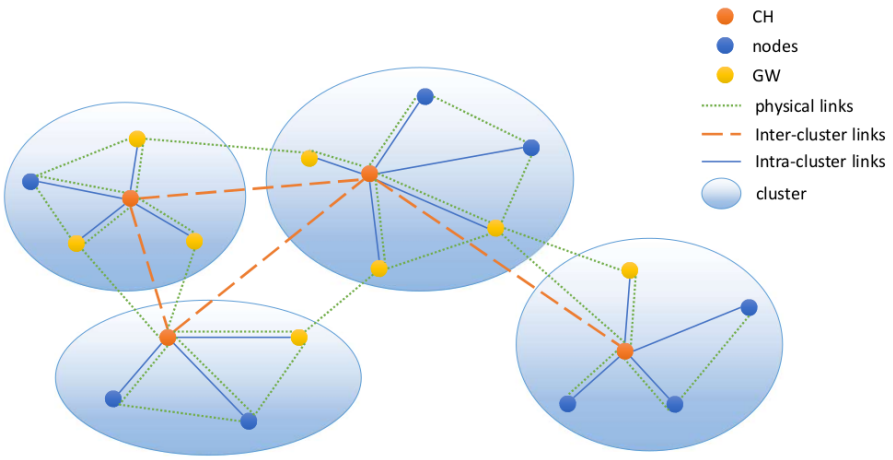


Figure 8: Example of k-neighbor clustering.

$$\forall u, v \in N. h(CH(u), CH(v)) < h, \quad h \geq k$$

The nodes that provide connectivity between different clusters are called gateways. An example of k-neighbor clustering with $k = 1, h = 2$ can be seen in figure 8.

Dominating set A dominating set $D \subseteq N$ is the set of nodes such that

$$\forall u \in N \setminus D. \exists v \in D. (u, v) \in E$$

A dominant set is a feasible set of CH for a 1-neighbor clustering of the network.

Connected dominating set A connected dominating set $C \subseteq N$ for G is a set of nodes such that C is a dominating set and the subgraph $G' \subseteq G$ induced by C is connected. Selecting the set of cluster heads as the CDS of G corresponds to cluster with $k = 1, h = 1$ and so to obtain a connected backbone.

Minimum connected dominating set In general there exists several CDS for a graph, the MCDS problem consists in finding one of minimum cardinality, that is the smallest set of cluster heads for k-neighbor clustering with $k = 1, h = 1$. This problem in NP-hard.

The cluster heads may keep their radios on to sustain the communication of the entire network, allowing other nodes to save energy. The small size of the CDS is particularly desirable in this case: only a few nodes consume more energy, most of the nodes save energy.

Distributed and mobility adaptive clustering DMAC is a simple clustering algorithm where each node in the network has a weight, which represents its willingness to become a cluster head. Weights may depend on the combination of several parameters of a node, like its battery charge, its ID or its position.

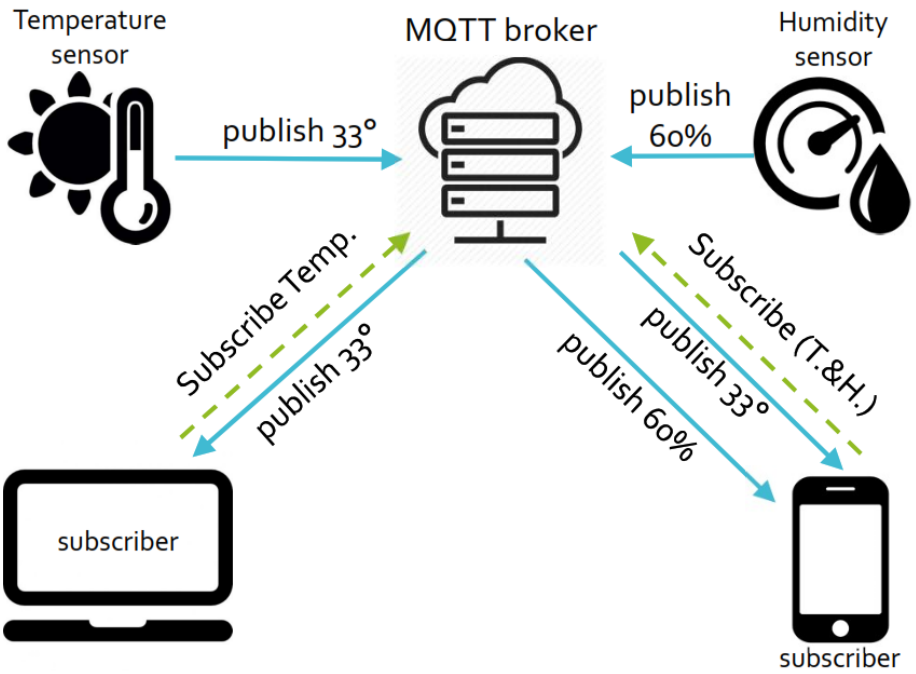
Each node can be in three possible states: cluster head (CH), ordinary node (ON) or undecided (UN). When DMAC starts each node broadcasts to its neighbors an `HELLO` message which contains the node ID, its weight and its status. If a node has the maximum weight among its neighbors then it becomes a cluster head, all of its neighbors become ordinary nodes. This procedure enforces $k = 1$, hence the set of CH is a dominating set, and $2 \leq h \leq 3$.

When a node joins an already clustered network it should set its status to UN and sends an hello message to its neighbors that will consequently update their status. Other than for topology changes the clusters should be updated for each weight change.

Chain reactions could arise when a node elects itself cluster head, and it's more likely as more nodes join the network. This is in general undesirable since involved nodes have to reconfigure themselves and this induces additional overhead.

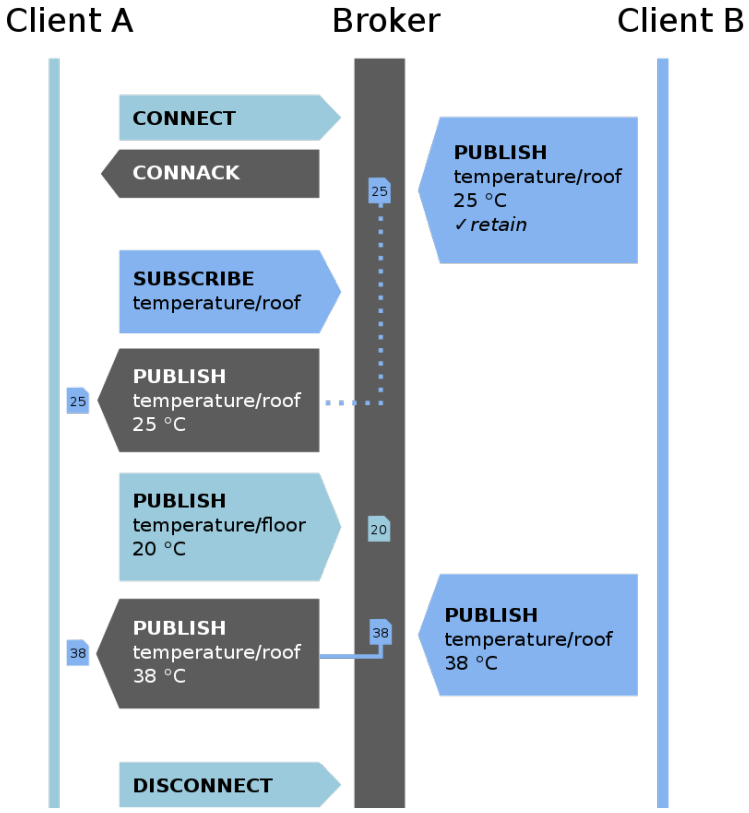
MQTT

The Message Queuing Telemetry Transport (MQTT) is a communication protocol, lightweight and reliable. It's based on the publish/subscribe paradigm. It is lightweight because it needs a small code footprint, low network bandwidth, and lower packet overhead that guarantees better performance than HTTP. The main actors: the publisher, subscriber and the event service (knowns as broker). The first two, are clients and don't know each other meanwhile the broker knows both. The publisher and subscriber are fully decoupled in time, space and synchronization. Decoupling is guaranteed by the presence of the broker that acting as an intermediary, it receives all incoming messages from the publishers, filters them and distributes all messages to the subscribers and also manages the requests of subscription/unsubscription.



Compared to the client-server architecture it allows greater scalability, by parallelizing the broker it is possible to connect millions of devices.

The publisher and the subscriber need to know broker hostname/IP and port to publish/subscribe messages. Messages can be filtered on a certain subject or topic, the content of the message (for example a specific query) or the type of data. Publishers and subscribers need to agree on the topics beforehand. Also, MQTT provides QoS to ensure reliability in the delivery of messages, there are three levels of QoS. Communication between the actors provides different kinds of messages:



The CONNECT message, sent by clients to the broker, contains: - ClientID: a string that uniquely identifies the client at the broker. It can be empty: the broker assigns a clientID and it does not keep a status for the client (the parameter Clean Session must be TRUE) - Clean Session (optional): a boolean value that determines if the client requests a persistent session: if it's FALSE the broker will store all subscriptions and missed messages, otherwise the broker cleans all information of the client of the previous session. - Username/Password (Optional) - Will flags (optional): If and when the client disconnects unexpectedly, the broker will notify the other clients of the disconnection. - KeepAlive (optional): The client commits itself to send a control packet (or a ping message) to the broker within a keep-alive interval.

The CONNECTACK message, response for the CONNECT message: - Connect Acknowledgement flags: confirms whether the connection was successful or not. - Session Present: indicates whether the broker has already a persistent session of the client

After connection, a client can publish messages. Each message contains a topic and a payload that contains the data.

The PUBLISH message: - PacketId: an integer, it's 0 if the QoS level is 0. -

TopicName: a string possibly structured in a hierarchy with «/» as delimiters, for example: «home/bedroom/temperature». - QoS: 0, 1 or 2. - RetainFlag: tells if the message is to be stored by the broker as the last known value for the topic. If a subscriber collects later, it will get this message. - Payload: the actual message in any form. - DupFlag: indicates that the message is a duplicate of a previous, un-acked message. Meaningful only if the QoS level is > 0.

SUBSCRIBE message: - PacketId: an integer - Topic1: a string (see publish messages) - QoS1: 0, 1 or 2

The last 2 fields are repeated in a list for all the topics the sender wants to subscribe to.

SUBACK message: - PacketId: the same integer of SUBSCRIBE message - ReturnCode: one for each topic subscribed

Topics are strings that are organized in a hierarchy (topic levels) each level is separated by a «/», for example: home/firstfloor/bedroom/presence. Using wildcard extends the flexibility of this system: - '+' is used to subscribe to an entire set of elements for a specific level of the hierarchy - '#' is used to subscribe to all publisher under a level of the hierarchy Topics that begin with a «$» are reserved for internal statistics of MQTT and they cannot be published by clients.

QoS

As said above in the MQTT protocol is provides a QoS system which is an agreement between publisher, broker and subscriber. There are three levels of QoS:

- QoS 0 (At most one): at this level the delivery uses the "best-effort" method, there aren't ACK messages and the broker doesn't store messages for offline clients.
- QoS 1 (At least one): messages are numbered and stored by the broker until they are delivered to all subscribers. "At least one" means that each message is delivered at least once to the subscribers. There are ACK messages.
- QoS 2 (Exactly one): it's the highest QoS level in MQTT and the slowest. It guarantees that each message is received exactly once by the subscriber and uses a double two-way handshake.

Downgrade of QoS

As we already mentioned, the QoS definition and levels between the client that sends (publishes) the message and the client that receives the message are two different things. The QoS levels of these two interactions can also be different. The client that sends the PUBLISH message to the broker defines the QoS of the message. However, when the broker delivers the message to recipients (subscribers), the broker uses the QoS that the receiver (subscriber) defined

during the subscription. For example, client A is the sender of the message. Client B is the receiver of the message. If client B subscribes to the broker with QoS 1 and client A sends the message to the broker with QoS 2, the broker delivers the message to client B (receiver/subscriber) with QoS 1. The message can be delivered more than once to client B, because QoS 1 guarantees delivery of the message at least one time and does not prevent multiple deliveries of the same message.

Packet identifiers are unique per client

The packet identifier that MQTT uses for QoS 1 and QoS 2 is unique between a specific client and a broker within an interaction. This identifier is not unique between all clients. Once the flow is complete, the packet identifier is available for reuse. This reuse is the reason why the packet identifier does not need to exceed 65535. It is unrealistic that a client can send more than this number of messages without completing an interaction.

Persistent session

If QoS is 1 or 2 the broker keeps persistent information about the state of the communication with clients. This information includes: the topics of the client, all messages that were not confirmed and those that were arrived when the client was offline. To achieve a persistent session at connection time the flag "cleanSession" must be set on FALSE. A particular type of persistent message is the retained messages which are normal messages with flag "retainedFlag" set on TRUE, their peculiarity consists that only the last sent message published on a certain topic will be stored. Retained messages make sense for unfrequent updates of a topic, for instance, the device status update (ON/OFF).

Last will & testament

Last Will & Testament is a type of message sent by the broker to notify other clients about the ungraceful disconnection of a client, for instance I/O error, KeepAlive message missing or a simple network disconnection. Like all the other messages last will is a normal message with topic, retained flag, QoS and a payload. To activate it is necessary to specify it at CONNECT time requiring a specific behavior about its last will.

Alternatives to MQTT

In some cases the need for a centralized broker can be limiting in distributed IoT applications with diffused point-to-point communications, for instance, its overhead may easily become not compatible with end-devices capabilities as the network scales up or the fact that the broker is a single point of failure and it is essential to keep the network alive. Moreover, MQTT relies on TCP, which is not particularly cheap for lowend devices that, require much more resources than UDP for example, it uses connections that need to be established and maintained

and it impacts on the batteries (and thus on the lifetime of the devices). An alternative to MQTT is Constrained Application Protocol (CoAP), which is a specialized web transfer protocol, useful in a constrained network for IoT, designed for machine-to-machine (M2M) applications such as smart energy and building automation. It's based on the Client/Server paradigm, uses the REST model, works similarly to HTTP but is based on UDP and provides a more light header suitable for devices that has small amounts of ROM and RAM.