TIBUCONPROJECT

TIBUMAC - an energy efficient and flexible communication protocol

TIBUCON is financially supported by the European Commission under the 7FP.

Self Powered Wireless Sensor Network for HVAC System Energy Improvement Towards Integral Building Connectivity

This document describes TIBUMAC, an energy efficient and flexible communication protocol targeted to EH based WSANs. The protocol is composed of several mechanisms that take into account the special constraints typically found in EH systems, besides the common issues related to WSANs. Multiple strategies can be implemented over the mechanisms in order to meet the energy constraints and changeable requirements of different applications. The protocol dynamically adapts to the application demands, always keeping the energy consumption profile as low as possible.

TIBUMAC OVERVIEW

TIBUMAC is a flexible communication protocol that defines a series of configurable mechanisms targeted to EH applications needs. As there could be no nodes where the network load is concentrated, TIBUMAC must ensure that each node acts within its capabilities while optimizing the network overall functionalities

TIBUMAC is conceived for the efficient gathering of distributed information to one point. This is a common WSAN architecture where the nodes are distributed in a tree-like fashion with the gateway or sink at the top. In this type of system the nodes relationships are described in terms of parent/child links. A parent is a node that acts as a link between one or more nodes and the gateway, and retransmits data from/towards it. A node's descendants are the group of nodes formed by its children, its children's children and so on. A node can have one or more children but just one parent. Note that the gateway's descendants are all the remaining nodes of the network. An example of this architecture is shown in the following figure, Fig. 1. Although the main traffic load is expected to be towards the gateway, the protocol also supports data traffic from the gateway towards the network.

The TIBUMAC protocol is based on a slotted TDMA scheduler. Each cycle is divided into different phases and each phase into slots. There are two types of slots: **Tiny-Slots (TSlots)** and **Big-Slots (BSlots)**, 1.25ms and 5ms long respectively. All the slots in a phase are of the same type.

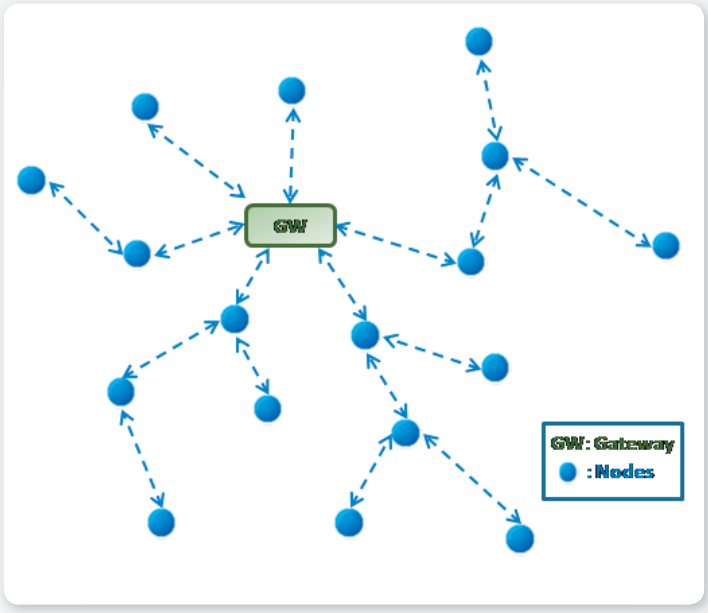

Fig. 1: Common WSN architecture.

The **Basic Cycle** defines the periodic behaviour of the protocol. A node can interact with the network at different periods but they will be multiples of the Basic Cycle duration. For instance, nodes can select greater cycle periods for data communications, called **Data Cycles**.

Each node in the network has an identifier that acts as an index for the scheduler. These indexes are

denoted **Global Indexes (GIdx)** and are assigned by the gateway. As will be described later, the assignment rules for GIdx is outside the scope of TIBUMAC. The mechanism for GIdx assignment is implemented at TIBUMAC layer but the assignment policies are set at application level.

Actually, although TIBUMAC is referred as a MAC protocol, it performs some duties related to the network layer. The topology is fixed to a tree architecture turning the routing into an intrinsic mechanism of the protocol.

There are three parameters that are constant in TIBUMAC. These values must be hard coded and cannot be changed after the initial deployment:

- **Maximum Node Number**: the number of nodes that can connect to the network is limited by the number of GIdx available. In TIBUMAC this number is 250 (represented in 1 Byte) with the remaining numbers (251-255) reserved for special purposes.
- **Data Interchange Cycle Basic Period**: this parameter defines the periodic behaviour of the protocol. A node can interact with the network at different rates but they will be multiples of the basic cycle duration.
- **Maximum Hop Number**: the duration of some phases is determined by the current network hop number. As all the phases of a given cycle must be performed within one cycle period time, the hop number must be bounded.

The structure of the TIBUMAC firmware is shown in Fig. 2.

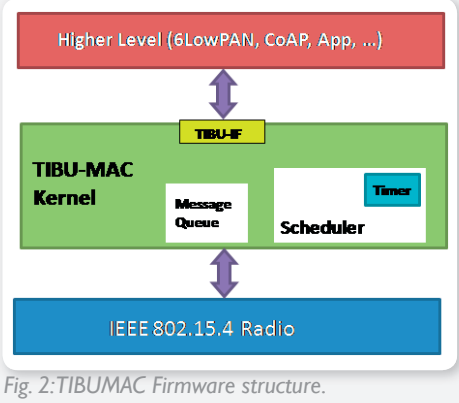

Fig. 2:TIBUMAC Firmware structure.

Node Types

From the point of view of TIBUMAC there are just two types of elements in a network: **the gateway and normal nodes**. A network is started by the gateway and nodes connect to it until the whole network is created.

The gateway is constantly powered and its available energy is supposed to be infinite. On the other hand, the functional behaviour of the nodes may change depending on their available resources.

However, from the application point of view there is a distinction between normal nodes:

- **Device**: a device is a node with sensors or some external control duty. Its tasks are the maintenance of the network connection and to perform some application dependant activity periodically. It may or may not have descendants depending on its resources.
- **Router**: a router is a node without a specific application level activity. Its only task is to retransmit as much information as possible. These nodes are located in places that benefit the Energy Harvesters used and do not have to expend energy in sensor readings or other duties.

There may be a third node type: the commissioner. This node is a battery powered device which synchs with a given network and advertises it in a channel. The use of **the commissioner** is only expected to be used for assistance in the deployment and validation of the network.

IEEE 802.15.4

TIBUMAC uses a specific configuration of the IEEE 802.15.4 standard header. The possible configuration of the Frame Control is shown in the Fig. 3 and the message format in Fig. 4. The node address is always 16 bits long (short address) and only the source address field is used. Its actual nature (source or destination) depends on the specific message type. The address of a node is formed by the current Hop number in the MSB and GIdx in the LSB.

Frame Type	Security Enabled	Frame Pending	AR	PAN ID Compression	Dest. Address Mode	Frame Version	Source Address Mode
001	0	0	0	0	00	01	10

Fig. 3: IEEE 802.15.4 Frame Control Format for TIBUMAC.

	Frame Length	Frame Control Field	Frame Sequence Number	PAN ID	Address	Payload	Frame Check Sum
Bytes:	1	2	1	2	2	N	2

Fig. 4: IEEE 802.15.4 message format for TIBUMAC.

The Dispatch Byte

IEEE 802.15.4 does not contain any field to identify the type of the payload. 6LowPAN introduces the "Dispatch Byte" in the first byte of the payload in order to make this differentiation.

The values where the two most significant bytes are both '0' are reserved for use outside 6LowPAN. Thus, TIBUMAC uses the Dispatch value '0x0A' to identify the protocol.

PROTOCOL PHASES

In TIBUMAC the network tasks are distributed in different phases, always scheduled in the same order. However, the participation in each phase is not mandatory. Not all the phases have the same duration. In some cases the duration is stretched or reduced depending on the current hop number. In these cases, a phase is divided into sub-phases with a fixed slot number (related to the available GIdx) and type.

The schedule of phase sequence is shown in the next diagram, Fig. 5:

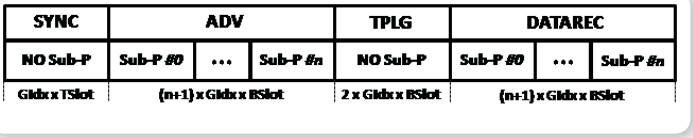

Fig. 5: TIBUMAC Phase schedule.

Advertising Phase [ADV]

The ADV phase is one of the most complex tasks of TIBUMAC. It is involved in the management of the topology, synchronization, and data transfer from the gateway.

- The number of ADV sub-phases is equal to the current number of hops plus one. Thus it will change as the network hop number increases or decreases.
- The slots used in the ADV phase are Big-Slots.
- The ADV header can change in length and shape depending on the options used. The size of the ADV header ranges from 10 to 25 bytes.

The format of the ADV message is shown in Fig. 8 and Fig. 9.

	Dispatch Byte	Frame Control	Channel Control	Sync. Info
Bytes:	1	1	1	3

Fig. 6: SYNC message format.

	Dispatch Byte	Frame Control	Path Cost	Channel Control	Max. Hop	Children Mng.	Neigh. Mng.	ACK Mng.	Indx. Mng.	Payload	Sync. Info
Bytes:	1	2	2	1	1	0/1	0/3	0/2	0/9	N	3

Fig. 8: ADV message format.

Version	Command	Reserved
00	00	0000

Fig. 7: SYNC message Frame Control field format.

Version	Command	Consistent	Desc. Allowed	Children Info	Neigh. Info	ACK Info	Indx. Info	Reserved	Header Seq.
00	01	0/1	0/1	0/1	0/1	0/1	00-11	0	xxxx

Fig. 9: ADV message frame control format.

Synchronization Phase [SYNC]

In the SYNC phase a node can send a SYNC message indicating its GIdx, Hop and synchronization information. Additionally the channel for the multi-channel hop management is sent.

- The SYNC phase is optional and is only scheduled in certain situations.
- The number of slots used in this phase is fixed to the available GIdx and are of Tiny-Slots type.
- The format of the SYNC message is shown in Fig. 6 and Fig. 7.

The gateway sends its ADV message in slot '0' of the sub-phase '0'. Any other node in the network sends its ADV message in the sub-phase corresponding to their Hop minus 1 and in the slot assigned to their GIdx. The state and data of the whole network is refreshed every Basic Cycle.

The *Consistent* flag is used to indicate that an ADV message is consistent with the ADV message sent by the gateway. A node will set this flag if an ADV with the Consistent flag has been heard this cycle. The gateway always as this flag set.

The *Descendants* Allowed flag is set if a node can accept more descendants. If a node sets this flag to '0' all its children MUST clear the flag.

The *Children* Info field is used to manage the children that have lost the connection with the parent.

The *Neighbour* Info flag indicates whether the Neighbour Management field is present (shown in Fig. 10).If the ACK Info flag is set the ACK Management field is present in the header.

The Index Info field is used to determine the format of the Index Management field (shown in Fig. 11).

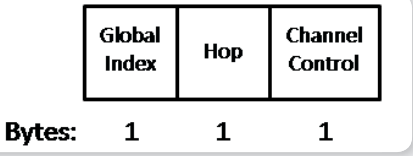

Fig. 10: Neighbour Management field format.

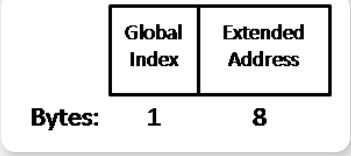

Fig. 11: Index Management field format.

Version	Command	Type	Reserved
00	10	xx	00

Fig. 12:TPLG message frame control format.

The Header Sequence field is incremented each time any other field of the header changes. It is used by a node to detect possible changes in the options when a packet has been lost. When the sequence number of a parent and a child are not equal, the parent has two options. The easier one is to send the next ADV header with all the fields expanded. The most efficient (and complex) option would be to maintain a history of the changes and expand only the fields modified within the cycles lost by the children.

Value	Definition
00	No Index data.
01	GIdx valid.
10	GIdx invalid.
11	Reserved.

Table 1: Index Info field values.

Topology Phase [TPLG]

The topology management phase is composed of 2x GIdx slots. Two consecutive BSlots are assigned to each GIdx. The first BSlot is assigned to the reception of REQ messages. These messages are used for the request of GIdx or for asking a node to establish a connection link. The REQ BSlot is divided in four TSLots to allow random access.

The second BSlot is for the receiver node to answer the requests that can be directly confirmed or rejected by the receiver such as the establishment of a communication link.

This phase is optional and it will only be used by nodes that have capacity for more descendants and by nodes that want to establish a new connection.

The format of the TPLG Frame Control Field is shown in Fig. 12.

The Type field specifies the exact purpose of the message. Table 2 shows the possible values.

The IDX-REQ message is used by a node to request a Global Index to the network or to check the validity of the owned GIdx.

Value	Definition
00	IDX-REQ, request for a Global Index.
01	PARENT-REQ, request for a parent.
10	CONFIRM, confirmation of a link establishment.
11	Reserved.

Table 2:Values of TPLG types.

The format is shown in Fig. 13.

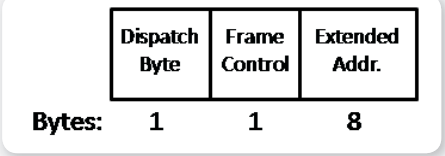

Fig. 13: IDX-REQ message format.

The GIdx field of the address is set to 0xFF if a new GIdx is requested and to the corresponding value if the GIdx is to be checked. *The Extended Addr.* field carries the extended address of the node that makes the request.

The PARENT-REQ topology message is used by a node that wants to establish a link with a specific parent. The format is shown in Fig. 14.

The Extended Addr. field carries the extended address of the node that makes the request.

	Dispatch Byte	Frame Control	Extended Addr.
Bytes:	1	1	8

Fig. 14: PARENT-REQ message format.

The CONFIRM topology message is the answer of a PARENT-REQ request and it is sent by a node only if the link is accepted. The format is shown in Fig. 15.

	Dispatch Byte	Frame Control	Extended Addr.	PcRC	PcF	ScRC	PcL	ScL
Bytes:	1	1	8	1	1	1	1-16	1-16

Fig. 15: CONFIRM message format.

The Extended Addr. field must be set to the accepted node's extended address. The remaining fields include information about the current channel hopping sequence. This information is divided in the fields Primary Channel List (PcL), Secondary Channel List (ScL), Primary Channel Repetition Counter (PcRC), Primary Channel Frequency (PcF) and Secondary Channel Repetition Counter (ScRC). Their purpose and meaning is explained further into this document. The PcL and ScL field lengths depends on the actual number of channels used for each list. If any of the lists is shorter than 16 elements the end of list is marked with a value of 0xFF.

Data Recollection Phase [DATAREC]

In the DATAREC phase the data of each node is retransmitted to the gateway in a multi-hop fashion. Each node sends its data to its parent which in turn makes a retransmission towards its own parent. This process is repeated until the gateway is reached. Apart from the data, information for the management of the network is sent. This information includes topology, channel hop and packet sequence number management data

The structure of the DATAREC message is shown in Fig. 16 and Fig. 17.

		Local Info			Original Info				
	Dispatch Byte	Frame Control	Next Msg. Mng.	Indx. Mng.	Orig. Addr	Parent GIdx	Channel Control	Max. Desc.	Payload
Bytes:	1	2	0-6	0/9	1	1	1	1	N

Fig. 16: DATAREC message format.

Version	Command	Next Msg. Info	Period Info	Indx. Info	Reserved	Header Seq.
00	11	00-11	00-11	0/1	000	xxxx

Fig. 17: DATAREC message frame control format.

The size of the DATAREC message header ranges from 7 to 22 bytes, controlled by the Frame Control field. The fields grouped under the Local Info tag are generated each hop by the node sending the message. The fields under Original Info are used to retransmit the data from the descendants. When a parent receives a packet from a child it retransmits this section of data.

The Next Message Management field is used to determine the GIdx of the next message(s) to be sent by the node (in this cycle). A node can advertise up to three messages in advance, depending on the selected length for the field.

The Period Info field determines the rate at which the node will send data to the parent. The possible values are described in the Table 4.

	Next Node Info		
	Global Index	Hop	...
Bytes:	1	1	

Fig. 18: Next Message Management field format.

The *Index Info* flag indicates whether the Index Management field is present. This field is shown in Fig. 11.

Value	Definition
00	No Next Nodes.
01	1 Node present (Length = 2 bytes).
10	2 Nodes present (Length = 4 bytes).
11	3 Node present (Length = 6 bytes).

Table 3: Next Message Info field values.

Value	Definition
00	Communication frequency = 1 cycle.
01	Communication frequency = 2 cycles.
10	Communication frequency = 4 cycles.
11	Communication frequency = 8 cycles.

Table 4: Period Info field values.

The *Header Sequence* field must be set to the last value received from the parent.

The number of sub-phases is equal to the number of current hops plus one. The node starts sending its own DATAREC message in sub-phase '0' and is retransmitted in the following sub-phases. Suppose the following topology with 6 elements and 2 hops (Fig. 19): the gateway (GIdx = 0) is the father of the nodes GIdx1 and GIdx2. GIdx1 is the father of nodes GIdx3 and GIdx4. GIdx2 is the father of GIdx5. In the sub-phase '0' all the nodes (with the exception of the gateway) send their DATAREC message in the corresponding slot. In the sub-phase '1', GIdx1 retransmit the messages of nodes GIdx3 and GIdx4; node GIdx2 does the same thing with the DATAREC message from GIdx5.

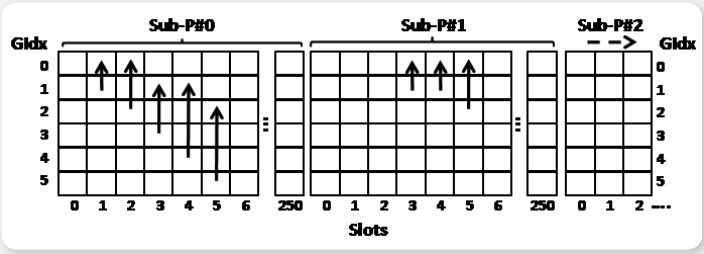

Fig. 19: DATAREC message schedule.

It is important to note that the last sub-phase is unused in normal operation. If a node connects to a network and adds a new hop, this sub-phase allows the node to send its data in that cycle.

TIBUMAC PROTOCOL DESCRIPTION

The different phases and messages that are the base of the TIBUMAC protocol have been defined. This section describes the protocol policies, tasks, and different states a node follows to find, connect, and, in short, to interact with a TIBUMAC network.

Fig. 20 shows and example of the state machine a node may follow to interact with the network. In this case, the node takes into account the available energy in order to decide the best state.

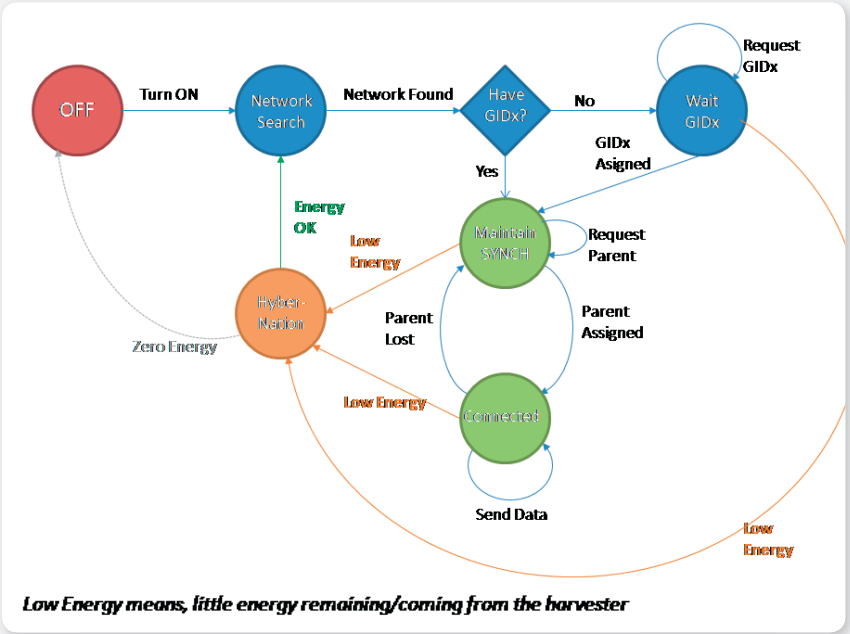

Fig. 20:Transitions between operation modes.

Operation Modes

There are various mechanisms that can be ignored or turned off if there is not enough energy:

- A node can decide not to send SYNC messages even if the network demands them.
- A node can decide not to send ADV messages even if the network demands them.
- A node can dynamically change the amount of descendants it has and dismiss part or all of its children.
- A node can reduce the frequency of communication with its father.

If a node has not enough energy to stay connected to the network, it changes to a state of "hibernation". In this situation a node does not communicate with the network (the radio is always off) and remains in low activity or sleep state. However, the node will maintain specific information like the cycle synchronization, the accumulated oscillator drift and surrounding nodes' data.

A node must ensure that it has enough remaining energy to stay in hibernation state for a minimum period of time. In the case of EH applications, this period of time should be enough to restore a good energy level and avoid the total depletion of the stored energy. If this requirement is not met, the depletion of the stored energy will produce a device reset losing valuable information such as network synchronization.

Global Indexes Allocation

Before a node establishes a connection with a network it needs a GIdx. In order to request a GIdx a node sends a TPLG message, the Type field set to IDX-REQ and the Global Index field set to 0xFF (invalid GIdx value). The node that receives the request will retransmit it via the DATAREC message's Index Management field.

When the request reaches the gateway, it assigns a unique GIdx to the node and transmits it in the ADV message's Index Management field. The Extended Address field is set to the IEEE 802.15.4 extended address of the node and the Global Index field to the assigned GIdx.

When the requesting node hears the ADV message it stores the assigned GIdx and initializes the connection procedure.

If a node wants to check the validity of its GIdx it can send a IDX-REQ message with the same procedure as to request a new GIdx. However, the Global Index field is set to the current value. If the GIdx is still valid, the gateway will send the ADV message Index Management field with the address and GIdx of the node. If the GIdx is already reserved for another node the Global Index field will be set to 0xFF (invalid GIdx value).

Topology Management

In order to establish a connection link a TPLG message of type PARENT-REQ must be sent to the selected (possible) new parent. Upon reception of the TPLG CONFIRM message, the node checks whether the extended address coincides with its own. In that case the node stores the parent GIdx for future use.

The Children Management field of the ADV header is used to advertise a child that has lost connection with their parent. A parent will set the Children Management field to the GIdx of the lost child. If the parent needs to remove all their children, it will set the value to 255. A node also keeps track of the number of continuous cycles without hearing its parent. If this number exceeds a security threshold, the node considers that the link has been lost and starts a connection procedure. A parent also keeps track of its children's state. If more than ChildLostCyclesMax cycles pass and no DATAREC message from a child has been received, the parent will erase the child from its list. If the child's DATAREC message Parent Indx field is not equal to the parent's GIdx, the child is also removed.

If a node cannot have more descendants, whether it is because it does not have enough energy or because it is in the last hop, it will clear the ADV message's control field flag Descendants Allowed and will skip the Topology Phase.

Network Dynamic Topology

Each node sends information about its surrounding nodes in the ADV message's Neighbour Management field. This allows a node to hear a limited number of nodes each cycle and consider a new parent.

A node can decide to change its parent at any time. In order to choose the best parent, the node will take into account the LQI and the ADV message Path Cost field of the involved nodes. This value is application dependant but it should take into account the Path Cost of its parent, the LQI of the link with the parent and the remaining energy.

Channel Hop

To reduce the data packet lost due to channel interferences (especially from Wi-fi) various channel hopping mechanisms are implemented:

- Periodically a node can change the channels it uses to communicate with its children.
- A node can decide to change the channel it uses to communicate with its children if the link quality changes frequently or there are high packet lost rates.
- A channel change can be requested from the application domain.

The ADV header Channel Control field is used to perform a channel hop. The Channel Counter is used to advert the cycles remaining for the next channel hop. The Next Channel field indicates the channel the hop will be made to.

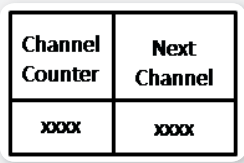

Fig. 21: ADV message Channel Control field.

In order to maintain the same channel across cycles, the Channel Counter field is set to '0' and the *Next Channel* field to the current channel.

The *Channel Counter* minimum value must be higher than the ParenLostCyclesMax threshold.

Nodes can maintain blacklisting tables and use different set of channels for each link. This is especially useful in large deployments where the surrounding conditions can vary significantly for each node cluster.

When a parent sends a confirmation message to a new child, it includes information about the current channel hopping sequence. This information is divided in the fields Primary Channel List (PcL), Secondary Channel List (ScL), Primary Channel Repetition Counter (PcRC), Primary Channel Frequency (PcF) and Secondary Channel Repetition Count (ScRC).

The PcL is used to reduce the time a node needs to detect a network. Primary channels and secondary channels are alternated in order to assure the occurrence of a primary channel each few Basic Cycles. The PcRC and PcF indicate, respectively, the number of consecutive Basic Cycles each primary channel is advertised and the channel hops between primary channels occurrences. The ScRC contains the number of consecutive Basic Cycles advertising the same secondary channel. Fig. 22 shows various examples with different values. For these examples it is assumed that ScL is composed of all the channels not present in PcL.

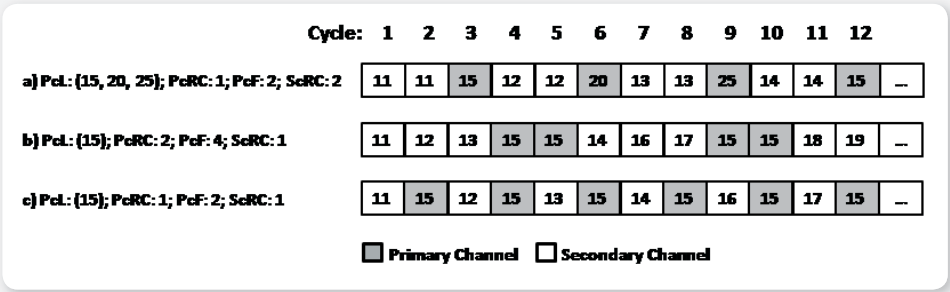

Fig. 22: Channel hopping sequence examples.

Network Search

When a node is not synchronized, it needs to find the channel and the network cycle start in order to create a connection. The usual approach is simple "channel sniffing", based on opening the radio and waiting for activity in a channel for a certain amount of time. This process is repeated for every available channel. Searching for a new network is a very high energy consuming procedure.

TIBUMAC uses the concept of primary and secondary channels. The protocol allows for primary channels to be used each few secondary channels in a predictive schedule effectively reducing the search space.

When a node reaches the point where it does not have enough energy to maintain the connection with the network it should enter the sleep state. In this state the node saves as much energy as possible until there is enough energy to establish a new connection. The node may not lose synchronization but will accumulate an oscillator drift. The reconnection mechanism is similar to the one used the first time a node connects to a network. The differences are that the node will use the accumulated drift to adjust the channel sniffing and that the stored list of neighbour nodes will be available. In this way the reconnection is much more efficient than a connection from scratch as there is no need for a full channel sniffing.

When the node receives a SYNC or ADV message it synchs with the network and starts the process to establish a connection. In any case, once the node is synchronized, its activity can be lowered to the minimum while still interacting with the network.

Network Lost

There are two situations in which a node can lose connectivity with the network: The most common is when a node does not have enough energy and enters the hibernation state. In this state a node saves as much energy as possible until it has accumulated the amount of energy necessary to establish a connection again. The node will not lose synchronization but will accumulate an oscillator drift. The reconnection mechanism is similar to the one used the first time a node connects to a network. The differences are that the node will use the accumulated drift to adjust the channel sniffing and that it will have stored a list of surrounding nodes. In this way the reconnection is more efficient than a connection from scratch.

On the other hand, if the node lost the network due to other reason (usually because the energy was totally depleted), the network synchronization is totally forgotten. In this case, a node does not have any option other than to start a complete channel sniffing as if it was the first time.

Message Sequence Management

To minimize the network traffic in both directions (from the gateway to the network and vice versa), a sequence number and confirmation mechanism is used. This is preferred to a packet by packet basis ACK in order to reduce the needed slot duration of each transaction.

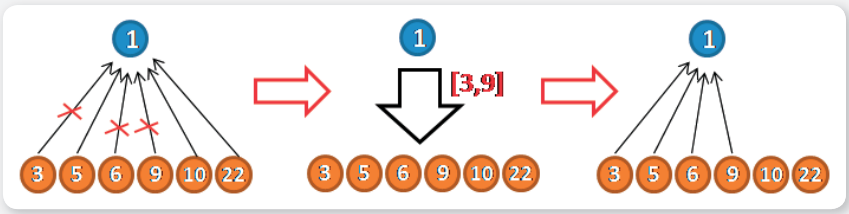

Fig. 23: Message confirmation example.

Each time the gateway has new information for the network, it sends it in the ADV message payload and increments the Frame Sequence Number field of the IEEE 802.15.4 header by '1'. The remaining nodes will send in the DATAREC message the last sequence number received. If the sequence numbers of all the nodes of the network and the sequence number of the gateway are equal, the data packet will not be sent again.

To confirm the message towards the gateway, each node will send in the ADV message ACK Management field the lowest and highest GIdx of the descendants for which the DATAREC messages have been lost. Consider a parent with 6 children with GIdx 3, 5, 6, 9, 10 and 22 (Fig. 23). Suppose that the messages from the nodes 3, 6 and 9 are lost in a disastrous cycle. In the next cycle the affected parent will send the ADV ACK Management field with the values [3,9] so the nodes 3,6 and 9 will send its data again. Note that the node 5 has also sent its data although it was not necessary.

TIBUMAC LIMITATIONS

Network Maximum Hop Number

The maximum hop limit is a function of the maximum number of GIdx and the cycle period. All the phases of a cycle must be performed within the same cycle. This obvious rule implies that the sum of the duration of all phases must be short enough to fit into a cycle. As the length of the phases (in slots) depends on the available GIdx (which is a constant) and some phases stretch with the number of hops, this second parameter must be bounded.

Network Maximum Node Number Limits

TIBUMAC uses 1 Byte for the GIdx. This means that the maximum number of nodes supported is 249 ('0' and values above '250' are reserved). This drawback is easily resolved changing the message structure. For example the whole IEEE 802.15.4 header Addr parameter can be used to store the GIdx and allow a range of 65535 different values (minus the reserved ranges). The hop can be codified in the protocol message headers.

Node Physical Limits

The number of messages a node can retransmit, and hence the number of descendants a node can have, not only depends on the available energy. A node must be capable of storing the messages before sending them, so there is a memory dependency. This means that if there are nodes implemented into different platforms, they may have different maximum descendants values even if they have the same energy level.

Network Static Cycle

The network cycle duration is constant because the nodes must know a priori how much time they must search in each channel before assuming there is not any network there. Furthermore, a variable network cycle must be advertised incrementing the message payload.

In any case, the network cycle defines the data refreshing time resolution. A node can decide to refresh its data at a lower frequency.

Network Search Power Consumption

Searching for a network is the most power demanding activity in TIBUMAC. That is why there is a big effort in maintaining the network synchronization once there is a connection and after if the network connection is lost.

CONCLUSIONS AND NEXT STEPS

TIBUMAC is able to dynamically change its behaviour to adapt to different energy consumption profiles or other application needs while keeping reliable communication. Different policies can be applied on top of the mechanisms (best parent function, neighbour advertisement, message ACK, channel hop, etc…) to implement application specific behaviours. Nodes powered by means of EH technologies could configure TIBUMAC to stay in low power mode or to modify the communication options (throughput, descendants number, etc…) as the available energy level changes. Additionally, nodes with different energy capacities and energy input profiles can coexist as the communication load is balanced over the network.

Future work will be focused on changing the tree topology towards a more flexible full-mesh topology. This will allow TIBUMAC to reach a wider range of applications. Additionally, the protocol will be revised to reduce the memory needs (due to intermediate buffers used for retransmissions) of the nodes. The support of more variable cycle times is also being considered in order to be able to deploy networks with very different throughput needs. Applications with no synchronization capabilities or sporadic transmissions will also be taken into account.

Finally, the study of different strategies/policies over the different mechanisms is an open issue and should be further researched to find their impact on the overall performance.

TIBUMAC LIBRARY OVERVIEW

The TIBUMAC communication library implements the necessary mechanisms to communicate with a TIBUMAC network. Once the library is configured and started, it takes the control of the communication interface and manages all the tasks related to the TIBUMAC protocol. Fig. 24 shows the inner structure of the TIBUMAC library. The scheduler controls the different states the node goes through and plans the tasks to be performed (e.g. open/close the radio, send a message, etc). As the library goes through all the different processes and states, the application is informed by means of events. A software timer is used for the management of the local time and synchronized tasks. The received/sent messages are managed via an inner queue in order to be pre-processed by the library before giving them to the application or sending them to the network.

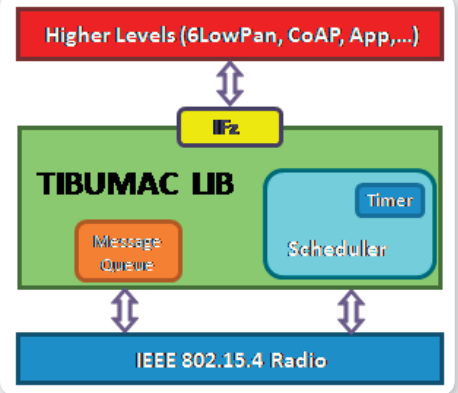

Fig. 24: TIBUMAC Firmware structure.

If the TIBUMAC library is configured as a Base Station, the device will start sending ADV messages immediately and will wait for requests. In the case of a normal node, the state machine is more complex. Fig. 20 summarizes the state machine of a node. The diagram assumes that the energy level is the major driver of the application behaviour. After the library is started a search for a valid network is performed. When a network is found, a request for a Global Index is issued. If a GIdx has been already assigned, a suitable parent is chosen and the connection process starts. Note that, the moment a network is found, the library performs the tasks in synchronization and the activity is lowered to the minimum possible. When the application decides that the connection with the network cannot be kept (e.g. due to energy constrains), the library can be instructed to go to a hibernation state. In this state all the activity is cancelled but network information (such as synchronization) is stored in order to perform a much more efficient reconnection.

Library Limitations

The TIBUMAC library is provided with a specific configuration. This configuration is hard coded and cannot be changed by the user:

- **Maximum Node Number:** the maximum number of nodes that can connect to one network is 50, including the Base Station.

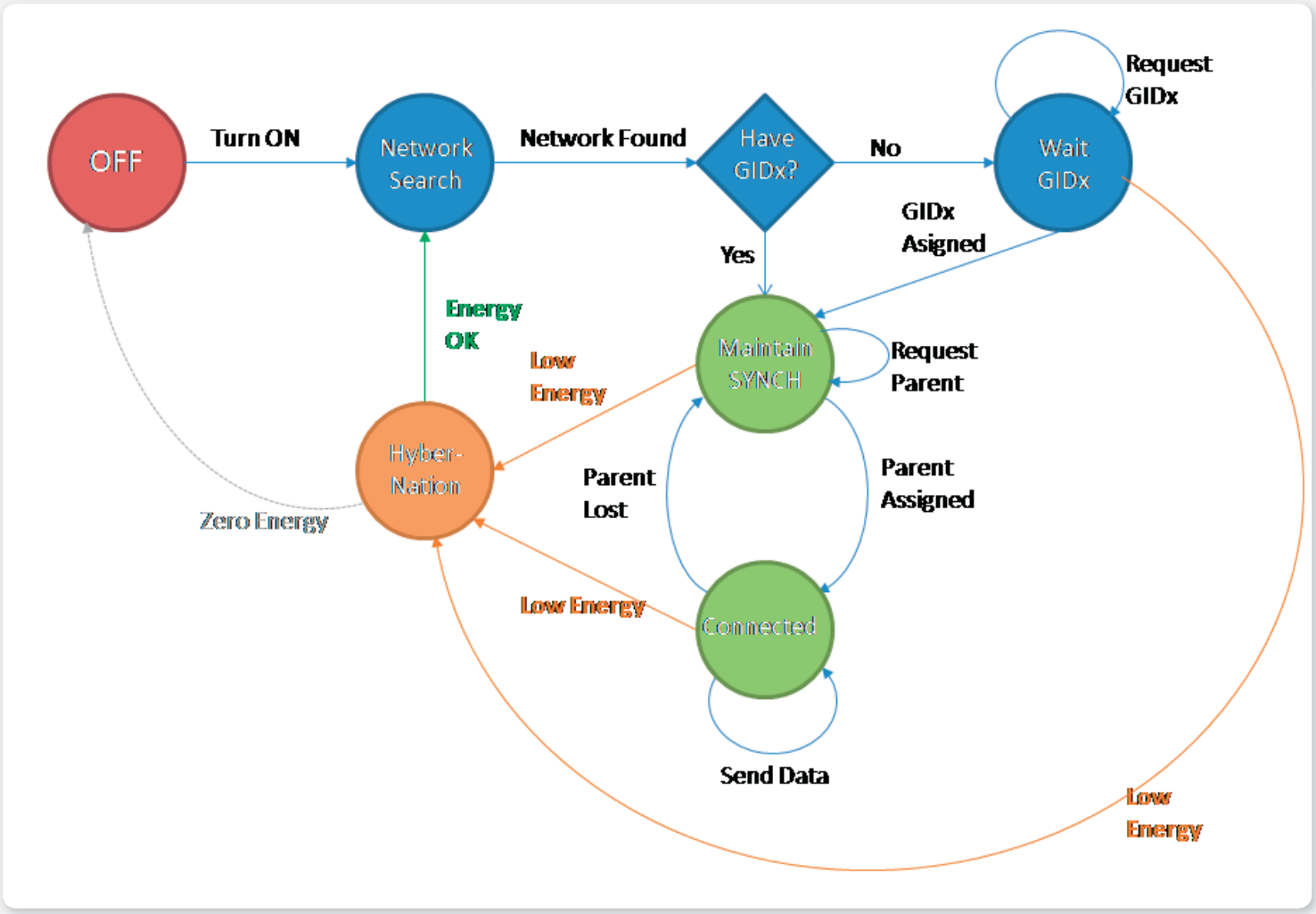

Fig. 25: Transitions between operation modes.

- **Maximum Descendant Number:** the maximum number of descendants for the Base station is 50, but nodes have a limit of 10 descendants.
- **Cycle Basic Period:** the basic period of the network has been set to 30 seconds.
- **Data Cycle Period**: the data interchange cycle of the nodes has been fixed to 1 Basic Cycle, i.e. 30 seconds.
- **No Flow Control:** the native traffic flow control of TIBUMAC has been disabled.

Additionally, the library makes use of some of the microcontroller's peripherals and may interfere with the application on the node.

- **Sleep Timer**: the sleep timer is used by the library to keep the timeline of the scheduler. The interrupt is already handled by an internal function and cannot be overwritten. However, the library offers software timers to the application to somewhat compensate for this drawback.
- **Radio**: the radio is fully used by the library (including interrupts) and its direct use by the application (once the communication stack has been started) is totally discouraged.
- **DMA#0**: the DMA buffer '0' is used internally by the library and it is not available for the application.

The sleep timer and radio interrupts are used during time critical tasks, thus their interrupt priorities should be set to the highest and second highest (respectively). This is already done within the TIBUMAC library initialization.

TIBUMAC LIBRARY EXAMPLE

The TIBUMAC example is composed of the TIBUMAC library and a very simple example application. The IDE used is the IAR Workbench 8051 v8.10. The example project contains two configurations for use with the Texas Instruments cc2530_EM and cc2530-cc2590_EM platforms.

Files

The files provided with the example workspace are listed in Table 5.

File	Description
include\GIdxTableMng.h	Header of the Global Index Table management functions
include\hal.h	TI cc2530 HAL definitions
include\main.h	Header of the main file
source\GIdxTableMng.c	Functions for the global Index management table
source\main.c	Main file with the example application
TIBUMAC_Stack\include\TIBUMAC.h	Header of the TIBUMAC stack
TIBUMAC_Stack\TIBUMACLib_cc2530.r51	TIBUMAC stack library for the cc2530_EM
TIBUMAC_Stack\TIBUMACLib_cc2530_cc2590.r51	TIBUMAC stack library for the cc2530-cc2590EM
TPowerModeCtrl\include\TPowerModeCtrl.h	Header for the power management functions of the cc2530
TPowerModeCtrl\cc2530PM.r51	Library with the power mode functions for the cc2530

Table 5: Example application file list.

Example Application Description

The Base Station sets a timer to be fired at a semi-random period. Each time the timer is fired the number of allowed descendants is changed. Additionally, a message is sent to the network in order to activate/deactivate a LED on the nodes. On the other hand, the nodes gather local metrics and send them to the Base Station. The "main.h" header contains definitions that are used to configure the TIBUMAC Library. These parameters are summarized in the Table 6.

Parameter	Description
NODE_TYPE	The type of the node. Either a Base Station or a Node
NODE_SERIAL	The serial Number of the node
NODE_NWK_ID	The Network Identifier
NODE_PARENT_VALID_LQI_THRESHOLD	The minimum LQI level to consider a neighbour as a possible parent.

Table 6: Example application parameter list.

PROJECT INFORMATION

COORDINATOR CONTACT

Piotr Dymarski
Mostostal Warszawa S.A.
Konstruktorska 11A,
02-673 Warsaw, Poland

www.mostostal.waw.pl
e-mail: p.dymarski@mostostal.waw.pl

Phone: (+48) 22 548 56 46
Fax: (+48) 22 548 56 22

PROJECT DETAILS

- Project Acronym: **TIBUCON**
- Project Reference: **260034**
- Start Date: **2010-09-01**
- Duration: **36 months**
- Project Cost: **2.46 million euro**
- Contract Type: **Collaborative project** (generic)
- End Date: **2013-08-31**
- Project Status: **Execution**
- Project Funding: **1.59 million euro**
- Website: **www.tibucon.eu**
- Website: **www.tibucon.info**

Scan the 2D barcode to visit the TIBUCON project website

Scan the 2D barcode to visit the TIBUCON landing page

PROJECT PARTNERS

Mostostal Warszawa S.A. www.mostostal.waw.pl Poland

Tekniker www.tekniker.es Spain

Katholieke Hogeschool Kempen www.khk.be

University of Southampton www.ecs.soton.ac.uk United Kingdom

E&L Architects www.el-architects.pl Poland

Giroa-Dalkia www.dalkia.es Spain

IK4 Tekniker www.tekniker.es Spain