IPP HURRAY!

www.hurray.isep.ipp.pt

Technical Report

# Implementation of the i-GAME mechanism in IEEE 802.15.4 WPANs

André CUNHA

Anis KOUBAA

Mário ALVES

TR-060707

Version: 1.0

Date: 24-07-2006

# Implementation of the i-GAME mechanism in IEEE 802.15.4 WPANs

André CUNHA, Anis KOUBAA, Mário ALVES

IPP-HURRAY!

Polytechnic Institute of Porto (ISEP-IPP)

Rua Dr. António Bernardino de Almeida, 431

4200-072 Porto

Portugal

Tel.: +351.22.8340509, Fax: +351.22.8340509

E-mail: arec@isep.ipp.pt, akoubaa@dei.isep.ipp.pt, mjf@isep.ipp.pt

http://www.hurray.isep.ipp.pt

## Abstract

This technical report describes the implementation details of the Implicit GTS Allocation Mechanism (i-GAME), for the IEEE 802.15.4 protocol. The i-GAME was implemented in nesC/TinyOS for the CrossBow MICAz mote, over our own implementation of the IEEE 802.15.4 protocol stack. This document provides the implementation details, including a description of the i-GAME software interfaces.

# Contents

## Figures

## Tables

# 1 Introduction

This technical report describes the implementation details of the Implicit GTS Allocation Mechanism (i-GAME) in IEEE 802.15.4 for Time-Sensitive Wireless Sensor Networks or [1]. The i-GAME mechanism was implemented over our own implementation of the IEEE 802.15.4 protocol stack [2], using the nesC [3] programming language and the TinyOS [4] operating system in the MICAz motes [5]. The i-GAME mechanism was originally proposed in [1].

The "IEEE 802.15.4-compliant" MICAz motes [5] operate in the 2,4 GHz ISM band and feature a 16 Mhz Atmel ATMega128L microcontroller (with 128 kB of program Flash) [6] and a Chipcon CC2420 802.15.4 radio transceiver (allowing a maximum 250 kbps bit rate) [7].

# 2 i-GAME Implementation Details

## 2.1 Implementation Architecture

The i-GAME mechanism is implemented both in the MAC and in the Network layers.

Comparing with the standard IEEE 802.15.4, the i-GAME mechanism in the MAC layer just needs to change the management of the beacon GTS descriptors, they have to be included in the beacon in a round robin sequence. The implicit GTS descriptors are managed by the i-GAME Admission Control by issuing the MLME_iGAME.response. This primitive is implemented in the MAC layer updates the GTS descriptors, either by removing or adding. The MAC layer maintains a list with the descriptors characteristics. Also the MAC layer needs to know how to build an implicit GTS request frame triggered by the Network layer when it calls the MLME_iGAME.request.primitive. If the request is successfully transmitted, acknowledge and received by the PAN coordinator, the MAC layer will signal the Network layer with the MLME_iGAME.confirm with the status of SUCCESS otherwise with the status of the problem occurred.

In the Network layer is the implementation of the i-GAME functions or the i-GAME Admission Control functions. The functions implemented for the admission control are the i-GAME management algorithm (*uint8_t iGAME_management_algorithm(Flow F)* ) and the admission control algorithm (*uint8_t admission_control()).* These functions are explained with detail in Section 2.3.

The Network layer implements the MLME_iGAME.indication primitive used by the MAC layer to indicate a request for an implicit GTS request. When the Network layer is signalled by this primitive it will call the i-GAME management algorithm function with the flow characteristics of the implicit GTS request. After the processing of the iGAME Admission Control, the Network layer will signal the MAC layer thru the MLME_iGAME.response indicating the flow characteristics, the current implicit time slots allocated and the status of the management algorithm. The status can be the flow refused or the flow accepted. Depending of this status the MAC layer will take the appropriate actions.

The MLME_iGAME.confirm primitive is also implemented in the Network Layer and is used by the MAC to inform the status of the MLME_iGAME.request.

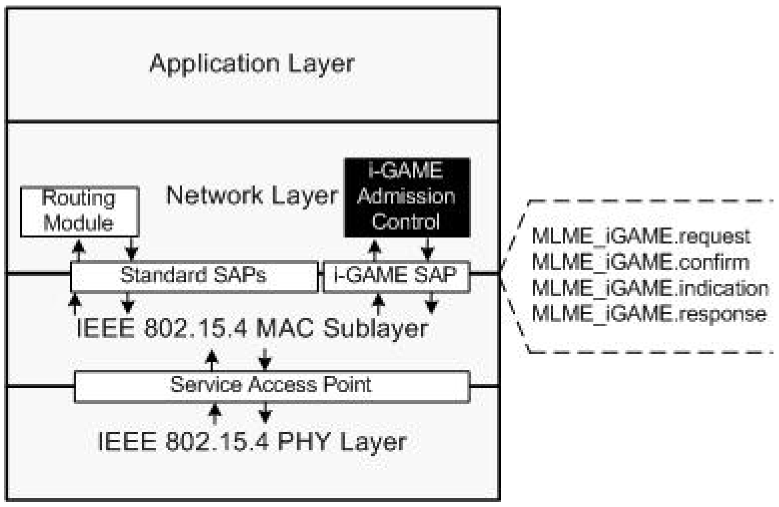


**Figure 1- Protocol Layer Architecture for the i-GAME implementation**

## 2.2 Structure definitions

### *2.2.1 GTS characteristics*

The GTS characteristics field is 8 bit in length and shall be formatted as illustrated in the next figure.

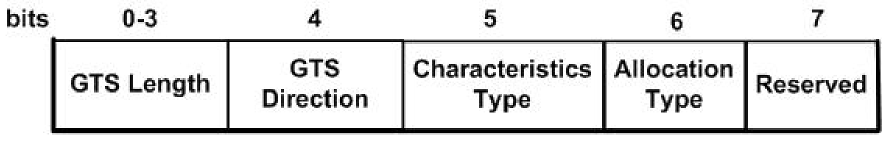


**Figure 2 - GTS Characteristics field format**

The next table describes the fields and valid ranges of values for the GTS Characteristics field.

| Name | Valid Range | Description |
|---|---|---|
| GTSLength | 0 to 7 | Number of superframe slots being requested for the GTS |
| GTS Direction | 1 or 0 | GTS direction is defined relative to the direction of data frame transmissions by the device. Is 1 if the GTS is to be a receive-only GTS and 0 if the GTS is to be a transmit-only GTS. |
| Characteristics Type | 1 or 0 | Is 1 if the characteristics refer to a GTS allocation or 0 if the characteristics refer to a GTS deallocation. |
| Allocation Type | 1 or 0 | Is 1 if the command is a GTS allocation or 0 if the request is an explicit allocation. |

**Table 1 - GTS characteristics fields descriptions**

### *2.2.2 Flow Specifications*

The Flow Specification format is 16 bits in length and shall be formatted as illustrated in the next figure. The burst size and the arrival rate fields should be expressed by four bits each (16 classes for each field). The Delay Requirement field is expressed by five bits (32 classes).

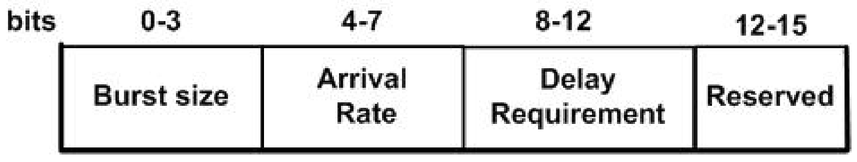


**Figure 3 – Flow Specification Field Format for i-GAME**

The next table describes the fields and valid ranges of values for the Flow Specification field. The PAN coordinator must define a fixed range for each value (class) of the corresponding field. These patterns should be known in advance by all nodes associated to the PAN before initiating an implicit allocation.

| Name | Valid Range | Description |
|---|---|---|
| Burst size | Refer to Table 2 | Burst size of the implicit GTS allocation |
| Arrival Rate | Refer to Table 3 | Arrival rate of the implicit GTS allocation |
| Delay Requirement | Refer to Table 4 | Delay requirement if the implicit GTS allocation |

**Table 2 – Flow Specification fields descriptions**

### *2.2.3 Burst size, Arrival Rate and Delay Requirement Class values*

| Value | Burst size |
|---|---|
| 0 | 10 |
| 1 | 15 |
| 2 | 20 |
| 3 | 25 |
| 4+ | 127 |

**Table 3 - Burst size Class**

| Value | Arrival Rate |
|---|---|
| 0 | 600 |
| 1 | 1200 |
| 2 | 2400 |
| 3 | 48000 |
| 4+ | 9600 |

**Table 4 – Arrival Rate Class**

| Value | Delay Requirements |
|---|---|
| 0 | 300 |
| 1 | 350 |
| 2 | 400 |
| 3 | 1000 |
| 4+ | 2000 |

**Table 5 – Delay Requirements Class**

## 2.3 i-Game Implementation Code

This section presents the i-GAME algorithms and the respective nesC implementations. The i-GAME Admission Control functions are exclusively implemented in the Network layer of the PAN coordinator, these function are:

- i-GAME Management Algorithm – On the signal indication of the MLME-iGAME.indication primitive in the Network layer, the Management Algorithm is called with the Flow specification of the implicit GTS request. This function will manage the time slots used for the implicit GTS allocation based on the result of the admission control algorithm. Next we will present the algorithm and the nesC code implementation. For more details refer to [1]
    - i-GAME Management Algorithm

```
type Flow = (id, b, r, D) //traffic specification and delay requirement
type FlowSetType = (Fi , where Fi requests a time slot in the CFP)
int N = 0; // the number of flow sharing a GTS
int k = 1; // the number of shared time slot
FlowSetType FlowSet; Flow F;
On (arrival of a new flow F) do {
N = N + 1;
```

```
8       if (admission_control (k, N, FlowSet, F) == false) {
9           if (k == 7) { //the maximum number of GTSs is reached
10              reject_request(F);
11              N = N - 1; break;
12          }
13          else { // k < 7
14              k = k + 1; //increase the length of the CFP
15              goto line 8;
16          }
17      }
18      else {
19      accept_request(F); //accept the new flow to share the GTS
20      FlowSet_Add(FlowSet, F); //add the new flow to the GTSset
21      }
```

- i-GAME Management nesC Implementation

```
uint8_t iGAME_management_algorithm(Flow F)
{       N = N + 1;
        flowset[N - 1] = F;
        while(admission_control() == FLOW_REFUSED) {
            if (k == 7 ) {  //the maximum number of GTSs is reached
                //Flow rejected
                N = N - 1;
                return FLOW_REFUSED;
            } else {    k = k + 1;   }
        }  //accept the new flow to share the GTS
        return FLOW_ACCEPTED;  }
```

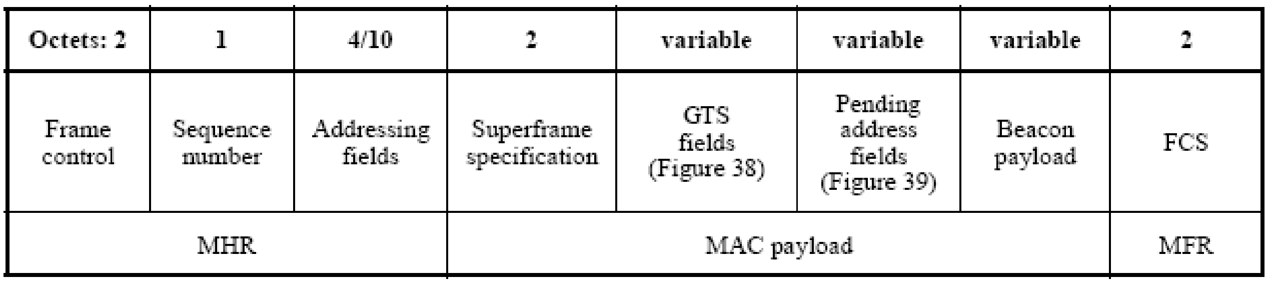
- i-GAME Admission Control Function – This function is called by the management algorithm and will return the value of FLOW_ACCEPTED or FLOW_REFUSED stating whether to accept or not the new flow requesting an implicit GTS. For more details refer to [1]
  - i-GAME Admission Control Function Algorithm

```
RTS = guaranteed bandwidth by one time slot
Ts = time slot duration
boolean admission_control (int k, int N, FlowSetType FlowSet, Flow F)
{
        boolean adm_crt = true;
        if (k <= N) {
            p = ceil (N / k);
            q = N – p * k – 1;
            for (int i = 1, i++; i<=N)
                if (( Di < (( bi / (k * RTS / N)) + ( p * Bi – q * Ts))) or  (ri>k*RTS/N))
                    adm_crt = false;
        } else { //the case (k>N) is considered as explicit allocation
            adm_crt = false;
        }
```

- i-GAME Admission Control nesC Implementation

```
uint8_t admission_control()
{
        uint8_t adm_crt;
        uint8_t p;
        uint8_t q;
        uint8_t i;
        adm_crt=FLOW_ACCEPTED;
        if ( k <= N)
        {       if ( (N % k) != 0)
                {       p = (N  / k) + 1;
                }else{  p = (N  / k);       }
                q = (p * k) + 1 - N ;
                for (i=0; i < N;i++) {
                   if ( ( iGAME_get_delay_requirements_ms(flowset[i].delay_class) <
                   (( iGAME_get_burst_size_byte(flowset[i].burst_class) * 8) / (k * (Rts / 1000) / N) ) + ( p * Bi - q * Ts ))
                    || (iGAME_get_arrival_rate_bps(flowset[i].rate_class) > (k * Rts / N)))
                    {       adm_crt = FLOW_REFUSED;    }
                }
      }  else
      {  adm_crt = FLOW_REFUSED;   }
return adm_crt;
}
```

## 2.4 Beacon generation – GTS Fields



| Octets: 2 | 1 | 4/10 | 2 | variable | variable | variable | 2 |
|---|---|---|---|---|---|---|---|
| Frame control | Sequence number | Addressing fields | Superframe specification | GTS fields (Figure 38) | Pending address fields (Figure 39) | Beacon payload | FCS |
| MHR | | | MAC payload | | | | MFR |

**Figure 4 – Beacon Frame Format**

The PAN coordinator manages the GTS descriptors by receiving a MLME_iGAME.response primitive from the Network Layer informing a new allocation or dealocation, and the number of time slots available in the CFP for the implicit allocations. On the creation of the beacon the PAN coordinator will build the descriptors distributing the allocated GTS in the available time slots using a round robin distribution.

# 3 i-GAME Implementation - MLME-iGAME Primitives

## 3.1 MLME-iGAME. request

The MLME-iGAME.request primitive allows a device to send a request to the PAN coordinator to allocate a new implicit GTS.

### *3.1.1 Semantics of the service primitive*

The semantics of the MLME-iGAME.request primitive is as follows:

```
MLME-iGAME.request  (
                            GTSCharacteristics,
                            Flow_Specification,
                            SecurityEnable
                            )
```

| Name | Type | Valid Range | Description |
|---|---|---|---|
| GTSCharacteristics | iGAME-GTS characteristics | Refer to section 2.6.2 | The characteristics of the GTS request. |
| Flow_Specification | Flow | Refer to section 2.6.2 | The characteristics of the flow requested. |
| SecurityEnable | Boolean | TRUE or FALSE | TRUE if security is enabled for this transfer or FALSE Otherwise |

**Table 6 - MLME-iGAME.request parameters**

### *3.1.2 When Generated*

The MLME-iGAME.request primitive is generated by the next higher layer and issued to its MLME to request the allocation of a new implicit GTS or to request the deallocation of an existing implicit GTS.

The GTSCharacteristics parameter specifies whether the request is for the allocation of a new implicit GTS request or for the deallocation of an existing GTS. The GTSCharacteristics of an implicit GTS must have the Allocation Type set to 1 otherwise it’s an explicit GTS allocation. If the characteristics type field of the GTSCharacteristics parameter is equal to 1, the remaining fields in the GTSCharacteristics will specify the desired characteristics of the new GTS. If the characteristics type field of the GTSCharacteristics parameter is equal to 0, the remaining fields in the GTSCharacteristics will specify the length and direction of the GTS that the device wishes to deallocate.

The Flow_Specification parameter specifies the payload of the implicit GTS request. The fields of the Flow_Specification will specify the desired characteristics of the data flow that will be used in the GTS.

### 3.1.3 Effect on receipt

On receipt of the MLME-iGAME.request primitive the MLME, of the device, attempts to generate an implicit GTS request command with the information contained in this primitive and, if successful, send it to the PAN coordinator.

If *macShortAddress* is equal to 0 x fffe or 0 x ffff, the device is not permitted to request a GTS. In this case, the MLME issues the MLME-iGAME.confirm primitive containing a status of NO_SHORT_ADDRESS.

If the request command cannot be send due to the failure of the slotted CSMA algorithm, the MLME issues the MLME-iGAME.confirm primitive containing a status of CHANNEL_ACCESS_FAILURE

To transmit the implicit GTS request command frame, the MLME issues the PLME-SET- TRX-STATE.request primitive with a state of TX_ON to the PHY in order to enable its transceiver. On receipt of the PLME-SET-TRXSTATE.confirm primitive with a status of either SUCCESS or TX_ON the implicit GTS request command frame is transmitted using the slotted CSMA algorithm. The transmission of the frame is accomplished by issuing the PD-DATA.request primitive. On the receipt of the PD-DATA.confirm primitive, the MLME enables is receiver in preparation for the acknowledgment.

After sending the implicit GTS request command the MLME will wait for the reception of an acknowledgment. If that does not occur the MLME will try to send the request again until it reaches the *aMaxFrameRetries* attemps. If this happen the MLME will issue the MLME-iGAME.confirm primitive with the status of NO_ACK.

When the implicit GTS request is acknowledge the PAN coordinator will issue the MLME-iGAME.indication primitive with the characteristics and the flow specifications of the implicit GTS to the next higher layer. The iGAME Admission Control will analyze the data received from the MLME and will issue the MLME-iGAME.response primitive with the characteristics and the flow descriptor to the MLME. The status of the response will be FLOW_ACCEPTED If the iGAME Admission Control can allocate the GTS otherwise it will be FLOW_REFUSED and the implicit GTS request is denied.

Then, the MLME will generate a GTS descriptor with the short address and the characteristics of the device if the status of the MLME-iGAME.response is FLOW_ACCEPTED. If the status is FLOW_REFUSED the PAN coordinator will generate the GTS descriptor with the short address of the device but with the start slot of zero.

The device will wait for the confirmation via a GTS descriptor specified in a beacon frame from its PAN coordinator.

If a descriptor appears that matches the characteristics requested the implicit GTS was successfully allocated. The MLME of the device requesting the GTS issues the MLME-iGAME.confirm primitive with a status of SUCCESS and a GTSCharacteristics parameter with a characteristics type equal to 1. The device may now start using the GTS.

If the descriptor appears with a start slot of 0, the PAN coordinator has denied the request. In this case, the device requesting the GTS issues the MLME-iGAME.confirm primitive with a status of DENIED, indicating that the GTSCharacteristics parameter is to be ignored.

The security parameters are not used and they are out of the scope of this implementation.

## 3.2 MLME-iGAME.confirm

The MLME-iGAME.confirm primitive reports the result of a request to allocate a new implicit GTS.

### 3.2.1 Semantics of the service primitive

The semantics of the MLME-iGAME.confirm primitive is as follows:

```
MLME-iGAME.confirm  (
                        GTSCharacteristics,
                        Flow_Specification,
                        Status
                        )
```

| Name | Type | Valid Range | Description |
|---|---|---|---|
| GTSCharacteristics | iGAME-GTS characteristics | Refer to section 2.6.2 | The characteristics of the GTS request. |
| Flow_Specification | Flow | Refer to section 2.6.2 | The characteristics of the flow requested. |
| Status | Enumeration | SUCCESS, DENIED, NO_SHORT_ADDRESS, CHANNEL_ACCESS_FAILURE, NO_ACK | The status of the implicit GTS request. |

**Table 7 - MLME-iGAME.confirm parameters**

### *3.2.2 When generated*

The MLME-iGAME.confirm primitive is generated by the MLME and issued to its next higher layer with the status result to a previously issued MLME-iGAME.request.

### *3.2.3 Effect on receipt*

On receipt of the MLME-iGAME.confirm primitive the next higher layer is notified of the result to a previously issued MLME-iGAME.request and the device may now start using the GTS

## 3.3 MLME-iGAME.indication

The MLME-iGAME.indication primitive indicates that an implicit GTS request was requested.

### *3.3.1 Semantics of the service primitive*

The semantics of the MLME-iGAME.confirm primitive is as follows:

```
MLME-iGAME.indication   (
                        DevAddress,
                        GTSCharacteristics,
                        Flow_Specification,
                        SecurityUse,
                        ACLEntry
                        )
```

| Name | Type | Valid Range | Description |
|---|---|---|---|
| DevAddress | Device Address | 0 x 0000-0x fffd | The short address of the device that requests the allocation |
| GTSCharacteristics | iGAME-GTS characteristics | Refer to section 2.6.2 | The characteristics of the GTS request. |
| Flow_Specification | Flow | Refer to section 2.6.2 | The characteristics of the flow requested. |
| SecurityUse | Boolean | TRUE or FALSE | An indication of whether the received frame is using security. This value is set to TRUE if the security enable subfield was set to 1 or FALSE if the security enabled subfield was set to 0. |
| ACLEntry | Enumeration | 0 x00 – 0x 08 | The *macSecurityMode* parameter value from the ACL entry associated with the sender of the data frame. This value is set to 0x08 if the sender of the data frame was not found in the ACL. |

**Table 8 - MLME-iGAME.indication parameters**

### *3.3.2 When generated*

The MLME-iGAME.indication primitive is generated by the MLME of the PAN coordinator to its next higher layer after the reception of an implicit GTS command. The MLME will issue this primitive with the GTSCharacteristics and the Flow specifications to the iGAME Admission Control.

### *3.3.3 Effect on receipt*

On receipt of the MLME-GTS.indication primitive the next higher layer is notified with the the GTSCharacteristics and the Flow specifications of the request. The iGAME Admission Control will evaluate the implicit GTS request and it will issue the MLME-iGAME.response primitive with the result of the evaluation.

## 3.4 MLME-iGAME.response

The MLME-iGAME.response primitive is used to initiate a response to a MLME-iGAME.indication primitive.

### *3.4.1 Semantics of the service primitive*

The semantics of the MLME-iGAME. response primitive is as follows:

```
MLME-iGAME. response        (
                            Flow_Description,
                            Shared_Time_Slots,
                            Status
                            )
```

| Name | Type | Valid Range | Description |
|---|---|---|---|
| Flow_Description | Flow | Refer to section 2.6.2 | The characteristics of the flow requested. |
| Shared_Time_Slots | iGAME-GTS characteristics | Refer to section 2.6.2 | Number of time slots available in the CFP for implicit allocations. |
| Status | Flow | FLOW_ACCEPTED, FLOW_REFUSED, | The status of the implicit GTS request. |

**Table 9 - MLME-iGAME.response parameters**

### *3.4.2 When generated*

The MLME-iGAME.response primitive is generated by the next higher layer of the PAN coordinator to its next higher layer after the reception of an implicit GTS command.

### *3.4.3 Effect on receipt*

On receipt of the MLME-iGAME.response primitive the MAC layer is notified of the result of the previous generation of the MLME-iGAME.indication primitive with the requested implicit GTS information.

The MAC layer is informed of the result of the iGAME Management Algorithm together with the GTSCharacteristics of the requested GTS and the number of available timeslots for implicit GTS allocation. If the flow is accepted, the status of the MLME-iGAME.response primitive is FLOW_ACCEPTED the MAC layer will update the number of available timeslots and the list of the GTS descriptors otherwise it will do nothing.

## 3.5 Implicit GTS management message sequence charts

The next figures illustrate the sequence of messages necessary for successful implicit GTS management. The first demonstrate the message flow for the case in which the device initiates the implicit GTS allocation and the second shows the message flow for the two cases for which an implicit GTS deallocation occurs, first, by a device (a) and, second, by the PAN coordinator (b).

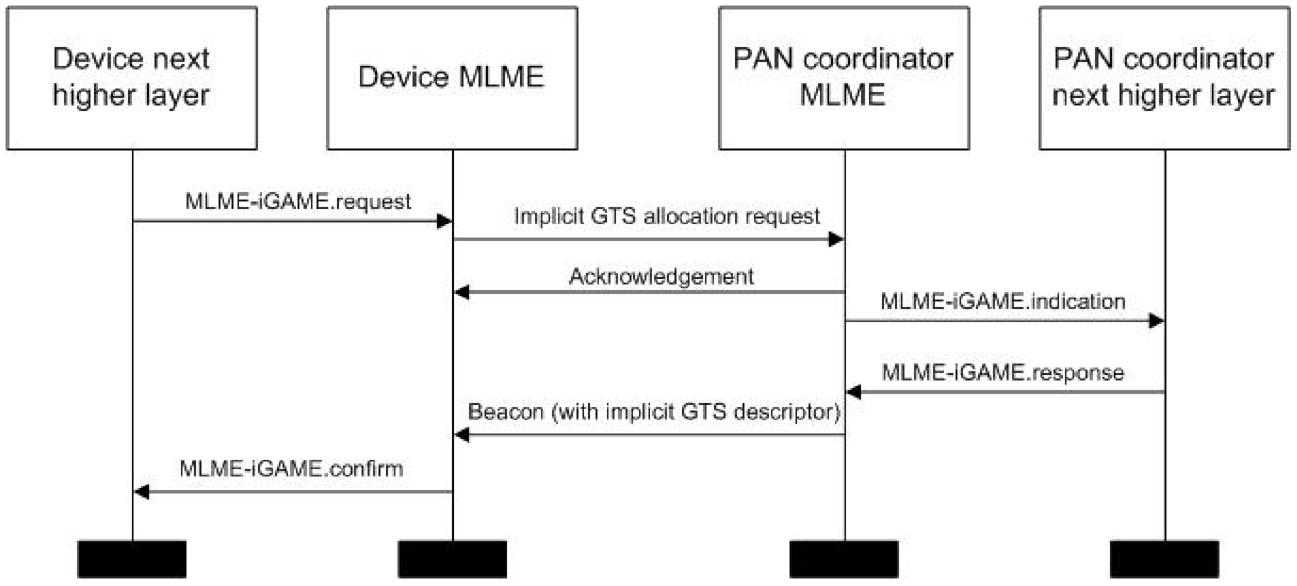


**Figure 5 - Message sequence chart for implicit GTS allocation initiated by a device**

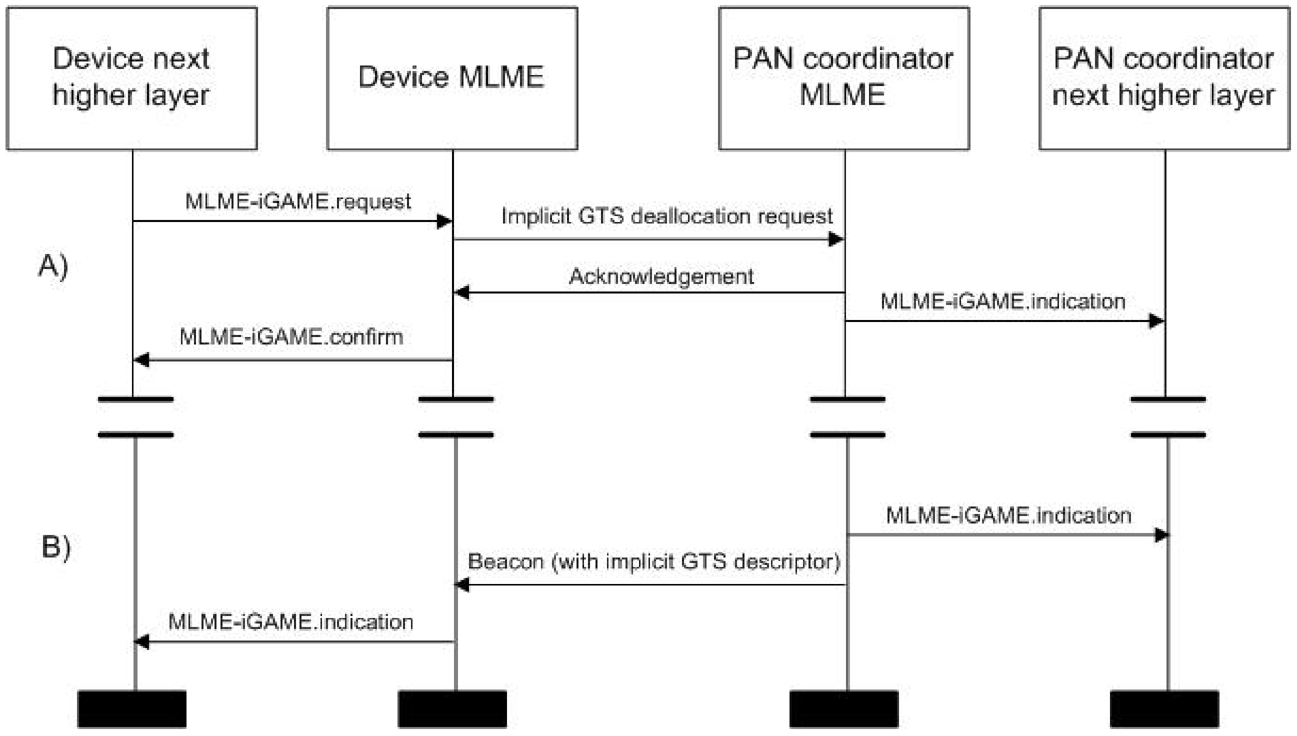


**Figure 6 - Message sequence chart for GTS deallocation initiated by a device (A) and the PAN coordinator (B).**

## References

[1] Anis KOUBAA, Mário ALVES and Eduardo TOVAR, “i-GAME: An Implicit GTS Allocation Mechanism in IEEE 802.15.4 for Time-Sensitive Wireless Sensor Networks”, published in proceedings of the 18th Euromicro Conference on Real-Time Systems (ECRTS’06), Dresden, Germany.

[2] IEEE 802.15.4 Standard-2003, "Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)", IEEE-SA Standards Board, 2003.

[3] David Gay, Phil Levis, Rob von Behren, Matt Welsh, Eric Brewer, and David Culler, “The nesC language: A holistic approach to network embedded systems”, in PLDI’03.

[4] www.tinyos.net

[5] Crossbow Technologies INC. http://www.xbow.com

[6] ATmega128L 8-bit AVR Microntroller Datasheet, Atmel ref: 2467M-AVR-11/04, http://www.atmel.com

[7] Chipcon, SmartRF CC2420 Datasheet (rev 1.3), 2005. http://www.chipcon.com