



HAL
open science

The Cost of Installing a New Communication Schedule in a 6TiSCH Low-Power Wireless Network using CoAP (Extended Version)

Erwan Livolant, Pascale Minet, Thomas Watteyne

► **To cite this version:**

Erwan Livolant, Pascale Minet, Thomas Watteyne. The Cost of Installing a New Communication Schedule in a 6TiSCH Low-Power Wireless Network using CoAP (Extended Version). [Research Report] RR-8817, Inria. 2015, pp.56. hal-01239994v1

HAL Id: hal-01239994

<https://inria.hal.science/hal-01239994v1>

Submitted on 8 Dec 2015 (v1), last revised 9 Dec 2015 (v2)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Installing a New Schedule with CoAP in a 6TiSCH-based Wireless Sensor Network (Extended Version)

Erwan Livolant, Pascale Minet, Thomas Watteyne

**RESEARCH
REPORT**

N° 8817

November 2015

Project-Team Eva



Installing a New Schedule with CoAP in a 6TiSCH-based Wireless Sensor Network (Extended Version)

Erwan Livolant, Pascale Minet, Thomas Watteyne

Project-Team Eva

Research Report n° 8817 — November 2015 — 56 pages

Abstract: In this research report, we show how to install a new schedule of node activity in an IEEE 802.15.4e TSCH wireless sensor network using the CoAP and 6TiSCH standards. Through an illustrative example, we compute the number of messages transiting in the whole network for different methods compliant with the existing standards. With regard to these results, we can conclude that the cost of using these standards is prohibitive in a wireless sensor network in contrast to the cost of the OCARI ad hoc solution. To drastically reduce this number of messages, we propose different recommendations to improve the efficiency of standardized protocol.

Key-words: Wireless sensor network, IEEE 802.15.4e TSCH, CoAP, 6TiSCH, OCARI.

**RESEARCH CENTRE
PARIS – ROCQUENCOURT**

Domaine de Voluceau, - Rocquencourt
B.P. 105 - 78153 Le Chesnay Cedex

Installation d'un Nouvel Ordonnancement avec CoAP dans un Réseau de Capteurs Sans Fil Basé sur 6TiSCH (Version Etendue)

Résumé : Dans ce rapport de recherche, nous montrons comment installer un ordonnancement des activités des noeuds dans un réseau de capteurs sans fil IEEE 802.15.4e TSCH en utilisant les standards CoAP et 6TiSCH. A travers un exemple illustratif simple, nous calculons le nombre de messages véhiculés dans l'ensemble du réseau pour différentes méthodes compatibles avec les standards existants. Au vu des résultats obtenus, nous pouvons conclure que le coût d'utilisation de ces standards est prohibitif dans un réseau de capteurs sans fil, comparé au coût de la solution ad hoc OCARI. Afin de réduire ce nombre de messages, nous proposons différentes recommandations permettant d'améliorer l'efficacité des protocoles standardisés.

Mots-clés : Réseau de capteurs sans fil, IEEE 802.15.4e TSCH, CoAP, 6TiSCH, OCARI.

Contents

1	Context	6
2	Problem Statement	7
2.1	Models and assumptions	7
2.2	Notations	7
2.3	Properties	8
2.4	Problem definition	10
3	Different solutions based on the IETF standards versus an ad hoc solution	11
3.1	Illustrative example	11
3.2	Ad hoc solution	12
3.3	Update one field per cell at once	12
3.4	Send a patch to each node	13
3.5	Broadcast the entire schedule to all nodes	14
3.6	Broadcast a DIFF of the schedule to all nodes	16
3.7	Comparative evaluation	18
4	Recommendations	19
4.1	Short addresses at MAC layer	19
4.2	A cellId coding the slotOffset and channelOffset	19
4.3	Shorter syntax for the PATCH method	20
5	Conclusion	23
Appendix A Example of a CBOR transcription of the JSON document describing a patch		25
Appendix B Installing a schedule node by node with the PATCH method		25
B.1	First schedule without the node 13	25
B.1.1	PATCH request for install the schedule on node 2	25
B.1.2	PATCH request for install the schedule on node 3	26
B.1.3	PATCH request for install the schedule on node 4	28
B.1.4	PATCH request for install the schedule on node 5	29
B.1.5	PATCH request for install the schedule on node 6	29
B.1.6	PATCH request for install the schedule on node 7	29
B.1.7	PATCH request for install the schedule on node 8	30
B.1.8	PATCH request for install the schedule on node 9	30
B.1.9	PATCH request for install the schedule on node 10	30
B.1.10	PATCH request for install the schedule on node 11	31
B.1.11	PATCH request for install the schedule on node 12	31
B.2	second schedule with the node 13	31
B.2.1	PATCH request for update the schedule on node 2	31
B.2.2	PATCH request for update the schedule on node 3	32
B.2.3	PATCH request for update the schedule on node 4	32
B.2.4	PATCH request for update the schedule on node 5	33
B.2.5	PATCH request for update the schedule on node 6	34
B.2.6	PATCH request for update the schedule on node 7	34
B.2.7	PATCH request for update the schedule on node 8	34

B.2.8	PATCH request for update the schedule on node 9	34
B.2.9	PATCH request for update the schedule on node 10	35
B.2.10	PATCH request for update the schedule on node 11	35
B.2.11	PATCH request for update the schedule on node 12	35
B.2.12	PATCH request for update the schedule on node 13	35
B.3	PATCH requests installing the first schedule with cellId coding	35
B.3.1	PATCH request for install the schedule on node 2 with cellId coding . . .	35
B.3.2	PATCH request for install the schedule on node 3 with cellId coding . . .	37
B.3.3	PATCH request for install the schedule on node 4 with cellId coding . . .	38
B.3.4	PATCH request for install the schedule on node 5 with cellId coding . . .	39
B.3.5	PATCH request for install the schedule on node 6 with cellId coding . . .	39
B.3.6	PATCH request for install the schedule on node 7 with cellId coding . . .	40
B.3.7	PATCH request for install the schedule on node 8 with cellId coding . . .	40
B.3.8	PATCH request for install the schedule on node 9 with cellId coding . . .	40
B.3.9	PATCH request for install the schedule on node 10 with cellId coding . . .	41
B.3.10	PATCH request for install the schedule on node 11 with cellId coding . . .	41
B.3.11	PATCH request for install the schedule on node 12 with cellId coding . . .	41
B.4	PATCH requests installing the second schedule with cellId coding	42
B.4.1	PATCH request for update the schedule on node 2 with cellId coding . . .	42
B.4.2	PATCH request for update the schedule on node 3 with cellId coding . . .	42
B.4.3	PATCH request for update the schedule on node 4 with cellId coding . . .	43
B.4.4	PATCH request for update the schedule on node 5 with cellId coding . . .	43
B.4.5	PATCH request for update the schedule on node 6 with cellId coding . . .	44
B.4.6	PATCH request for update the schedule on node 7 with cellId coding . . .	44
B.4.7	PATCH request for update the schedule on node 8 with cellId coding . . .	44
B.4.8	PATCH request for update the schedule on node 9 with cellId coding . . .	44
B.4.9	PATCH request for update the schedule on node 10 with cellId coding . . .	45
B.4.10	PATCH request for update the schedule on node 11 with cellId coding . . .	45
B.4.11	PATCH request for update the schedule on node 12 with cellId coding . . .	45
B.4.12	PATCH request for update the schedule on node 13 with cellId coding . . .	45
B.5	PATCH requests installing the first schedule with a shorter syntax	46
B.5.1	PATCH request for install the schedule on node 2 with a shorter syntax . . .	46
B.5.2	PATCH request for install the schedule on node 3 with a shorter syntax . . .	47
B.5.3	PATCH request for install the schedule on node 4 with a shorter syntax . . .	48
B.5.4	PATCH request for install the schedule on node 5 with a shorter syntax . . .	49
B.5.5	PATCH request for install the schedule on node 6 with a shorter syntax . . .	49
B.5.6	PATCH request for install the schedule on node 7 with a shorter syntax . . .	50
B.5.7	PATCH request for install the schedule on node 8 with a shorter syntax . . .	50
B.5.8	PATCH request for install the schedule on node 9 with a shorter syntax . . .	50
B.5.9	PATCH request for install the schedule on node 10 with a shorter syntax . . .	51
B.5.10	PATCH request for install the schedule on node 11 with a shorter syntax . . .	51
B.5.11	PATCH request for install the schedule on node 12 with a shorter syntax . . .	51
B.6	PATCH requests installing the second schedule with a shorter syntax	52
B.6.1	PATCH request for update the schedule on node 2 with a shorter syntax . . .	52
B.6.2	PATCH request for update the schedule on node 3 with a shorter syntax . . .	52
B.6.3	PATCH request for update the schedule on node 4 with a shorter syntax . . .	53
B.6.4	PATCH request for update the schedule on node 5 with a shorter syntax . . .	53
B.6.5	PATCH request for update the schedule on node 6 with a shorter syntax . . .	54
B.6.6	PATCH request for update the schedule on node 7 with a shorter syntax . . .	54

B.6.7	PATCH request for update the schedule on node 8 with a shorter syntax .	54
B.6.8	PATCH request for update the schedule on node 9 with a shorter syntax .	55
B.6.9	PATCH request for update the schedule on node 10 with a shorter syntax	55
B.6.10	PATCH request for update the schedule on node 11 with a shorter syntax	55
B.6.11	PATCH request for update the schedule on node 12 with a shorter syntax	55
B.6.12	PATCH request for update the schedule on node 13 with a shorter syntax	55

1 Context

Wireless sensor networks (WSNs) can support various applications with strong requirements in terms of latency energy efficiency and reliability. To cope with these requirements, the IEEE 802.15.4e amendment [1] has been proposed with the Time Slotted Channel Hopping (TSCH) mode. This mode uses a time slotted medium access in a multichannel environment, performing channel hopping. A Slotframe, depicted in Figure 1, is a sequence of multichannel time slots allowing conflict-free transmissions in the wireless network. This Slotframe is periodically reproduced in the network. In this report, we focus on how to install the schedule used in this Slotframe.

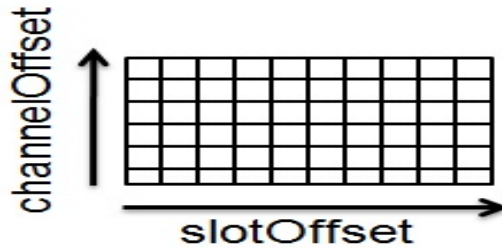


Figure 1: A Slotframe in a IEEE 802.15.4e TSCH network.

Examples of use cases include:

- instrumentation of a temporary industrial worksite,
- fire detection,
- intrusion detection in an industrial storage site,
- monitoring of air quality in a city,
- pollutant detection in a smart city,
- monitoring of the height snow in a ski resort,
- structure health monitoring in an aircraft....

All these applications are in charge of data gathering: data measured by wireless sensor nodes are transferred to a sink using multihop forwarding if needed. To improve the quality of data gathered, the WSN must meet the requirements previously given.

That is why, in the following we consider an IEEE 802.15.4e TSCH WSN. This WSN operates according to a schedule giving for each time slot and each channel the transmissions allowed. To ensure determinism of the medium accesses, the schedule must be conflict-free. We assume that this schedule is computed in a centralized way, usually by the sink. In this report, we want to evaluate the number of messages needed to install this schedule on all sensor nodes using the CoAP standard [2] with different methods. We compare these results with an ad hoc solution based on OCARI [3].

The remaining of this research report is organized as follows. In Section 2, we define the problem more precisely with the assumptions and models adopted. In Section 3, we compare different solutions based on the IETF standards with an ad hoc one based on OCARI. We propose some recommendations to reduce the number of messages in these standardized solutions in Section 4. Finally, we conclude in Section 5.

2 Problem Statement

We first define the problem considered in this research report.

2.1 Models and assumptions

A 6TiSCH WSN is based on the IEEE 802.15.4e TSCH [1] standard. On top of it, there is 6LoWPAN [4] that allows the use of IPv6 in WSNs. The transport protocol used is UDP. The RPL [5] routing protocol builds the destination-oriented directed acyclic graph (DODAG) rooted at the sink. This DODAG enables each node that is not the sink to know its parent in the routing tree. Finally CoAP [2] is used at the application layer. Hence, the protocol stack used by each wireless sensor node is depicted in Figure 2.

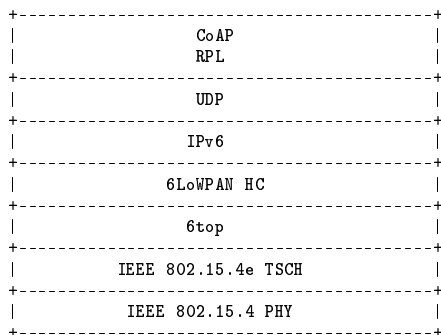


Figure 2: Protocol stack in a wireless node.

The schedule ruling medium accesses consists of a list of assignments. We assume that any schedule has a sequence number *ScheduleNumber* given by the scheduler and kept on each wireless sensor node. Each *assignment* of the schedule describes a transmission in the network. An assignment is a tuple $(SlotOffset, ChannelOffset, Tx, Rx)$, where Tx is the address of the transmitter and Rx is the address of the receiver.

2.2 Notations

In this paper, we adopt the following notations.

Let N_{assign} be the number of assignments in the new schedule.

Let $Gen(u)$ denote the number of messages locally generated by node u .

Let $Trans(u)$ be the number of transmissions of node u in the new schedule. This number includes the messages locally generated by u and the messages received from its children and forwarded to its parent.

Let $Depth(u)$ denote the depth of node u in the DODAG.

Let $Child(u)$ be the set of children of node u in the DODAG.

Let $B(u)$ be the number of blocks of size *BlockSize* needed to transfer the information related to cells involving node u .

Let B_{sched} be the number of blocks of size *BlockSize* needed to transfer the information related to the whole schedule.

Let *BlockSize* be the maximum size of blocks allowed in the useful payload. The value of *BlockSize* depends on the method chosen to transfer the schedule, as described in Section 3.

Let P be the number of nodes that are parents in the DODAG. This number includes the sink.

Let $CoAPcon$ denote a CoAP message that should be confirmed.

Let $CoAPack$ be the acknowledgment of a CoAP message.

Let $Nfield$ denote the number of fields that are updated per cell.

The different fields of a cell [6] that should be updated are:

- *SlotOffset* defines, with the ChannelOffset, the cell considered in the schedule,
- *ChannelOffset* defines, with the SlotOffset, the cell considered in the schedule,
- *LinkOption* specifies the role of the node: Transmitter (Tx) or Receiver (Rx);
- *NodeAddress*, denotes the node address of the remote node in the assignation received.

2.3 Properties

Since any node u has to transmit its own messages $Gen(u)$ as well as all messages it receives from its children, we get the following property:

Property 1 *Assuming that any node $u \neq sink$ generates locally $Gen(u)$ messages per slotframe, it needs to transmit $Trans(u)$ messages per slotframe, with*

$$Trans(u) = \sum_{v \in subtree(u)} Gen(v), \quad (1)$$

where $subtree(u)$ denotes all nodes belonging to the subtree rooted at u , including node u itself.

Taking into account the different communication protocols used by the wireless sensor nodes and given in Section 2.1, we can compute the available payload in a CoAP message, using the formats depicted in Figure 3 for the MAC frame and Figure 4 for a CoAP option.

Octets: 2	1	0/2	0/2/8	0/2	0/2/8	0/5/6/10/14	variable	2
Frame Control	Seq Num	Dest. PAN Id.	Dest. Address	Source PAN Id.	Source Address	Auxiliary Security Header	Frame Payload	FCS
Addressing fields								
MHR								
							MAC Payload	MFR

Figure 3: Format of a MAC frame.

More precisely, when the *Block* option [7] is used to fragment into blocks at the CoAP level, the size of the *Option Delta extended* field is taken equal to one octet, whereas the size of the *Option value* field depends on the block numbering used, as illustrated in Figure 5. The smallest size is 2 octets, meaning that blocks are numbered on 4 bits, leading to a maximum number of blocks of 16. If the number of blocks exceeds 16, then a numbering on 12 bits is used, leading to a maximum number of blocks of 4096. In our computation, we take into account the two possibilities.

Table 1 gives the size of the successive payloads available at each layer in the protocol stack.

First, the maximum allowed payload at the PHY layer in the IEEE 802.15.4 [8] standard is 127 bytes.

MAC header and footer require a total of 29 bytes in the context of the IEEE 802.15.4e TSCH mode. Indeed, authors of [9] determine that data messages must provide in their MAC header

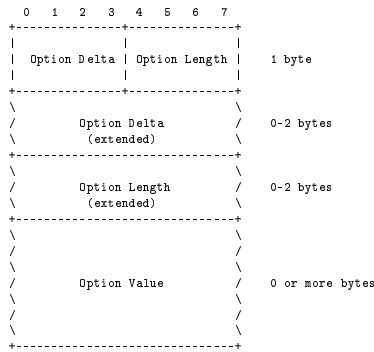


Figure 4: Format of a CoAP option.

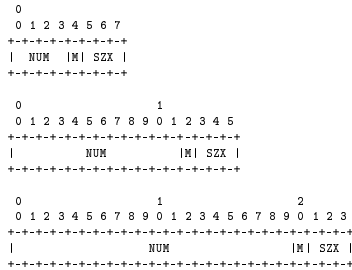


Figure 5: Option Value for the Block Option in CoAP.

Table 1: Payloads.

IEEE 802.15.4e TSCH	Frame Control	Seq Num	Dest PAN	Dest @	Src @	Aux Sec Header	MIC	Payload	FCS
	2	1	2	8	8	2	4	98	2
6LoWPAN - IP	LOWPAN_IPHC	Hop limit	Src @	Dest @	Payload				
	2	1	8	8	79				
6LoWPAN - UDP	LOWPAN_NHC	Port src + dest	Payload						
	1	1	77						
COAP without frag	Header	URI	Payload marker	Payload without block					
	4	11	1	61					
COAP with ≤ 16 frag	Header	URI	Option delta + Length	Option Delta extended	Option value	Payload marker	Payload with block		
	4	11	1	1	1	1	32		
COAP with ≤ 4096 frag	4	11	1	1	2	1	32		

fields 64-bit addresses for the destination and the source. Moreover, the Auxiliary Security header (AS) is coded with 2 bytes and the Message Integrity Code (MIC) is coded with 4 bytes. Finally, the MAC payload contains 98 bytes.

At the Network layer, the protocol requires 19 bytes for the IP compressed header (2 bytes), the Hop limit (1 byte) and the Source and Destination addresses coded on 8 bytes each.

At the Transport layer, the payload is reduced by 2 bytes corresponding to the UDP compressed header and the destination and source port coded together in 1 byte.

At the Application layer, the size of the applicative payload, depends on the CoAP option used. First, 4 bytes of header and 1 byte of payload marker are required. Moreover, the URI targeting the resource is defined as follows: `/mg/6t/hash` where `/mg/6t/` is the main path to the 6TOP management resources and `hash` is a hash of 30 bits (4 bytes) defining the rest of the path towards the targeted resource.

The three different possibilities studied are: without fragmentation, with less than 16 fragments and with less than 4096 fragments. In any case, we get a useful payload whose size is less than 61 bytes. For the *Block* option, it is specified in CoAP that the block size should be a power of two. As a consequence, the only possibility with this payload size is a block size of 32 bytes.

2.4 Problem definition

With the assumptions given in Section 2.1, the problem consists in computing the number of messages needed to install a new schedule, as a function of various parameters such as the number of wireless sensor nodes, the number of cells in the schedule... More precisely, the scheduler, usually the sink, that computed the new schedule uses CoAP messages to send to each node the fields values of the cells involving this node. Each wireless node in the wireless sensor network considered should know all the information related to on the one hand the cells in which it transmits, and on the other hand the cells in which it receives.

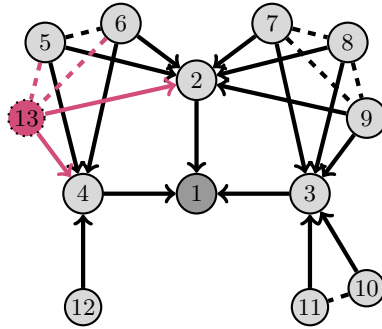
We will distinguish different solutions, depending on the entity updated each time (e.g. a cell or a list of cells), what is updated (e.g. the whole schedule or only the differences with regard to the current one), the type of communication used (e.g. broadcast, point-to-point).

3 Different solutions based on the IETF standards versus an ad hoc solution

3.1 Illustrative example

As an illustrative example, we consider the topology depicted in Figure 6a, where solid lines represent links in the DODAG rooted at node 1, and the dashed lines represent additional links (i.e. links that are not used to forward data to the sink). In this configuration, three channels are available for the WSN and the sink is assumed to be equipped with 3 radio interfaces. All nodes generate one message per SlotFrame, except nodes 2, 5 and 10 that generate 2 messages. We first assume that the WSN consists of 12 wireless nodes (i.e. node 13 is not present) and we evaluate the total number of control messages needed to install the first schedule, considering the five solutions described in the following sections. In a second step, we assume that node 13 joins the network, we then evaluate the total number of control messages needed to install the updated schedule.

Figures 6b and 6c depict the schedules computed by MODESA [10], with and without node 13. Without node 13, the schedule consists of 24 assignments distributed in 9 slots, whereas it contains 26 assignments distributed in 9 slots when node 13 is present. When node 13 joins the network, a new schedule is computed where 12 cells differ from the previous schedule, these cells are represented in black in Figure 6c.



(a) Network topology of the illustrative example.

CO \ SO	0	1	2	3	4	5	6	7	8
0	4→1	2→1	4→1	2→1	4→1	2→1	4→1	2→1	3→1
1	3→1	10→3	3→1	11→3	3→1	10→3	3→1	7→3	
2	5→2	6→4	2→1	12→4	9→2	5→4	8→2		

(b) First schedule computed by MODESA without the node 13.

CO \ SO	0	1	2	3	4	5	6	7	8
0	4→1	2→1	4→1	2→1	4→1	2→1	4→1	2→1	4→1
1	3→1	10→3	3→1	11→3	3→1	12→4	3→1	5→4	3→1
2	2→1	5→4	9→2	6→4	8→2	10→3	13→2	7→3	

(c) Second schedule computed by MODESA with the node 13.

Figure 6: The illustrative example.

Property 2 In both cases, the schedules computed by MODESA are optimal in term of number of slots. The minimum number of slots is equal to $\max(S_n, S_t)$ with $S_n = \lceil \sum_{u \neq \text{sink}} \text{Gen}(u)/g \rceil$ and

$S_t = Gen(Child1) + 2 \sum_{v \in Subtree(Child1), v \neq Child1} Gen(v) + \delta$ with $g = \min(n_{interf}, n_{channel}, n_{child})$ and $\delta = 1$ if the $(g + 1)^{th}$ child requires the same number of transmissions as $Child1$, the most transmitting child of the sink, and $\delta = 0$ otherwise. This property was proved in [11].

3.2 Ad hoc solution

The ad hoc solution presented in Figure 7 is based on the OCARI [12] protocol. OCARI schedules activities in the network in a cycle organized in four periods.

First, OCARI synchronizes the WSN in a multihop way by cascading beacons during the $[T0, T1]$ period. With this synchronized network, the "rendez-vous" $T1$ and $T2$ are known by every node of the network.

The period $[T1, T2]$ is dedicated to the control traffic used to collect network characteristics in order to compute a schedule for user data.

Then the $[T2, T3]$ period allows the user data gathering in multichannel.

Finally, the period $[T3, T0']$ is a sleep period where all nodes belonging to the network are setted in a deep sleep mode in order to save energy.

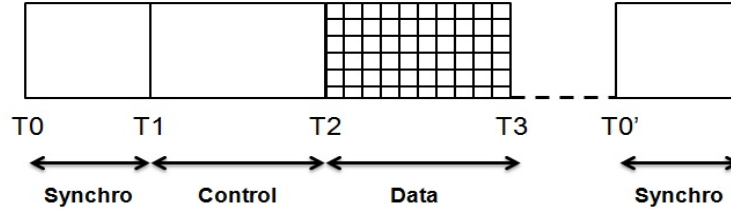


Figure 7: Cycle provided by the ad hoc solution

In this context, the beacons synchronizing the network are used to broadcast the schedule to all nodes. The OCARI protocol benefits of an association mechanism allowing the allocation of 16-bit short addresses to nodes. An assignation is then described by a tuple (SlotOffset, ChannelOffset, Tx, Rx) coded in 7 bytes: 2 bytes for SlotOffset, 1 byte for ChannelOffset, 2 bytes for Tx and 2 bytes for Rx.

In our previous illustrative example, the first schedule to install is made of 24 assignations and the second one is composed of 26 assignations. Therefore, we need to transmit $24 * 7 = 168$ bytes for the first schedule and $26 * 7 = 182$ bytes for the second. Since in our ad hoc solution the maximum payload available in a beacon is 80 bytes, we need 3 messages sent by each parent node, leading to a total of $3 * 4 = 12$ messages, since in our example $P = 4$ parent nodes.

3.3 Update one field per cell at once

In the naive solution, the scheduler transfers each cell field, one by one, for any assignation present in its new schedule, both on the transmitter and on the receiver if it is not the scheduler node itself. The update is made by a `POST` in CoAP that is acknowledged.

More precisely, to update a field cell on a node, the scheduler sends one `CoAPcon` message and receives one `CoAPack` acknowledgment. Since each assignation requires the update of N_{field} cell fields on each node involved (transmitter or receiver), the scheduler will send N_{field} `CoAPcon` messages and receive N_{field} `CoAPack` acknowledgments for each assignation where the receiver is not the scheduler node itself. To evaluate the total number of messages transiting in the network, we have to take into account the distance in number of hops of each node u to the scheduler node, that is $Depth(u)$. Any node $u \neq sink$ should be notified for the

$Trans(u)$ assignments for which it is transmitter. This generates a total of $Nfield \cdot (1CoAPcon + 1CoAPack) \sum_{u \neq sink} Trans(u) \cdot Depth(u)$ control messages in the network. Similarly, node u should be notified for the $\sum_{v \in Child(u)} Trans(v)$ assignments for which it is receiver. This generates a total of $Nfield \cdot (1CoAPcon + 1CoAPack) \sum_{u \neq sink} Depth(u) \sum_{v \in Child(u)} Trans(v)$ control messages in the network.

Hence we get the following total number of messages denoted $Nmsg(Naive)$:

$$Nmsg(Naive) = Nfield \cdot (1CoAPcon + 1CoAPack) \cdot \sum_{u \neq sink} Depth(u) \left(Trans(u) + \sum_{v \in Child(u)} Trans(v) \right) \quad (2)$$

Figure 8 describes a transcription of a POST request in CoAP to set the value 3 for the nodeAddress field of the cell defined by slotOffset = 3 and channelOffset = 1.

```
REQ:
POST
coap://<ip>:5683/mg/6t/cellList/nodeAddress/?slotOffset=3&channelOffset=1 3
```

Figure 8: A POST request in CoAP addressing the nodeAddress value of a cell.

Table 2 presents the number of messages required for first installing the schedule computed for a network of 12 nodes (see Figure 6a), then updating this schedule taking into account the new node 13. We can notice that it could be clearly prohibitive to use this method due to the strong overhead induced in a small network.

Table 2: Total number of messages required for installing and updating the schedules with the Naive method.

Node	without node 13			with node 13	
	Assignations	$Nfield$	Messages	Assignations * $Nfield$	Messages
2	8	4	64	$2*2 + 2*1$	$8 + 4$
3	9	4	72	$3*1$	6
4	7	4	56	$3*1 + 2*4$	$6 + 16$
5	2	4	32	$2*2$	16
6	1	4	16	$1*1$	4
7	1	4	16	$1*1$	4
8	1	4	16	$1*1$	4
9	1	4	16	$1*1$	4
10	2	4	32	$1*1$	4
11	1	4	16	0	0
12	1	4	16	$1*2$	8
13	-	-	-	$1*4$	16
Total	-	-	352	-	100

3.4 Send a patch to each node

In this second solution, the scheduler transfers at once a list of cells that must be updated. More precisely, it sends only the differences between the current schedule and the new one using the Patch option. The scheduler sends to each node u , $B(u)$ blocks containing the patches involving u either as transmitter or as receiver and receives an acknowledgment per block, as recommended in [6]. Hence the total number of messages transiting in the network is equal to:

$$Nmsg(Patch) = (1CoAPcon + 1CoAPack) \cdot \sum_{u \neq sink} B(u) \cdot Depth(u) \quad (3)$$

An example of such a patch is described in Figure 9. In this PATCH request written as recommended in [13], the two communications involving node 5 in the first schedule are presented as one transmission towards node 2 and another towards node 4.

```

REQ:
PATCH
coap://<ip>:5683/mg/6t/cellList
[
  {
    "op": "replace",
    "path": "/nodeAddress?slotOffset=0&channelOffset=2",
    "value": 2
  },
  {
    "op": "replace",
    "path": "/linkType?slotOffset=0&channelOffset=2",
    "value": 1
  },
  {
    "op": "replace",
    "path": "/nodeAddress?slotOffset=5&channelOffset=2",
    "value": 4
  },
  {
    "op": "replace",
    "path": "/LinkType?slotOffset=5&channelOffset=2",
    "value": 1
  }
]

```

Figure 9: Example of a PATCH request that installs the schedule on node 5.

Table 3 presents the number of messages required for installing and updating the schedule computed for our illustrative network (see Figure 6a). We can notice again that it could be prohibitive to use this PATCH method due to the strong overhead induced in a small network.

Table 3: Total number of messages required for installing and updating the schedules with the PATCH method.

Node	without node 13			with node 13		
	CBOR bytes	Blocks	Messages	CBOR bytes	Blocks	Messages
2	1049	33	66	397	13	26
3	918	29	58	208	7	14
4	918	29	58	533	17	34
5	263	9	36	269	9	36
6	132	5	20	67	3	12
7	132	5	20	70	3	12
8	132	5	20	67	3	12
9	132	5	20	67	3	12
10	263	9	36	70	3	12
11	132	5	20	0	0	0
12	132	5	20	136	5	20
13	-	-	-	132	5	20
Total	-	-	374	-	-	210

3.5 Broadcast the entire schedule to all nodes

In this solution, the scheduler broadcasts the whole schedule using a specific JSON document syntax. Each node u filters only the cells involving it as transmitter or receiver. To be sure that each node receives correctly the new schedule, the scheduler initially makes an *Observe* of the Schedule Number on each node. Hence, each time, this number is increased, the scheduler

is notified. This *Notify* is interpreted by the scheduler as an acknowledgment of the successful receipt of the schedule with the Sequence Number given.

To disseminate an *Observe* on all nodes except the sink requires P messages. This broadcast is done at network initialization and each time a new node joins the network.

To disseminate *Bsched* blocks of 32 bytes on all nodes except the sink requires $P \cdot Bsched$ messages.

To notify the scheduler of a change of the schedule number on all nodes except the sink requires $\sum_{u \neq sink} Depth(u)$ messages.

At each new schedule, a total number of control messages transiting in the network is:

$$Nmsg(Broadcast) = P \cdot Bsched + \sum_{u \neq sink} Depth(u) + \delta_P \cdot P \quad (4)$$

where δ_P is equal to one if at least a new node has joined the network since the last transmission of the *Observe* and 0 otherwise.

Figure 10 depicts the POST request with a JSON document describing the first schedule of our illustrative example. This request is broadcasted to all nodes of the network. The length of the CBOR transcription of the JSON document describing our schedule is 149 bytes. This CBOR transcription is divided into 5 fragments of 32 bytes, hence $Bsched = 5$. In our example topology, the number of parents broadcasting the schedule is $P = 4$ (node 1, 2, 3, 4) and the sum of depths of nodes is $\sum_{u \neq sink} Depth(u) = 19$. For this first schedule, as all nodes just joined the network $\delta_P = 1$. Finally, $msg(Broadcast) = 4 \cdot 5 + 19 + 1 \cdot 4 = 43$.

```

REQ:
POST
coap://<ip>:5683/mg/6t/schedule
{
  "ScheduleNumber": "1",
  "Schedule":
  [
    [0, 0, 4, 1],
    [0, 1, 3, 1],
    [0, 2, 5, 2],
    [1, 0, 2, 1],
    [1, 1, 10, 3],
    [1, 2, 6, 4],
    [2, 0, 4, 1],
    [2, 1, 3, 1],
    [2, 2, 2, 1],
    [3, 0, 2, 1],
    [3, 1, 11, 3],
    [3, 2, 12, 4],
    [4, 0, 4, 1],
    [4, 1, 3, 1],
    [4, 2, 9, 2],
    [5, 0, 2, 1],
    [5, 1, 10, 3],
    [5, 2, 5, 4],
    [6, 0, 4, 1],
    [6, 1, 3, 1],
    [6, 2, 8, 2],
    [7, 0, 2, 1],
    [7, 1, 7, 3],
    [8, 0, 3, 1]
  ]
}

```

Figure 10: A POST request broadcasting the first schedule to all nodes in the network.

Figure 11 depicts the same POST request for the second schedule of our illustrative example. This request is also broadcasted to all nodes of the network including the joining node 13. The

length of the CBOR transcription of the JSON document describing our schedule is 159 bytes. This CBOR transcription is divided into 5 fragments of 32 bytes, hence $B_{sched} = 5$. As in the previous case, the number of parents broadcasting the schedule is $P = 4$ but now the sum of depths of nodes is $\sum_{u \neq sink} Depth(u) = 21$. For this second schedule, node 13 just joined the network, hence $\delta_P = 1$. Finally, $msg(Broadcast) = 4 \cdot 5 + 21 + 1 \cdot 4 = 45$.

```

REQ:
POST
coap://<ip>:5683/mg/6t/schedule
{
  "ScheduleNumber": "2",
  "Schedule":
  [
    [0, 0, 4, 1],
    [0, 1, 3, 1],
    [0, 2, 2, 1],
    [1, 0, 2, 1],
    [1, 1, 10, 3],
    [1, 2, 5, 4],
    [2, 0, 4, 1],
    [2, 1, 3, 1],
    [2, 2, 9, 2],
    [3, 0, 2, 1],
    [3, 1, 11, 3],
    [3, 2, 6, 4],
    [4, 0, 4, 1],
    [4, 1, 3, 1],
    [4, 2, 8, 2],
    [5, 0, 2, 1],
    [5, 1, 12, 4],
    [5, 2, 10, 3],
    [6, 0, 4, 1],
    [6, 1, 3, 1],
    [6, 2, 13, 2],
    [7, 0, 2, 1],
    [7, 1, 5, 4],
    [7, 2, 7, 3],
    [8, 0, 4, 1],
    [8, 1, 3, 1]
  ]
}

```

Figure 11: A POST request broadcasting the updated schedule to all nodes in the network.

We can conclude that, with this broadcast method, the overhead for installing or updating a schedule is limited compared to the naive and patch methods in our example. However this overhead increases with the amount of traffic in the network. It can be prohibitive to broadcast a huge schedule particularly for a small update. We can then propose a variant of this broadcast method that only sends a DIFF of the schedule.

3.6 Broadcast a DIFF of the schedule to all nodes

In this solution, the scheduler broadcasts only the differences between the current schedule and the new one, using a specific JSON document syntax. Each node u filters only the cells involving it as transmitter or receiver.

To be sure that each node receives correctly the new schedule, the scheduler initially makes an *Observe* of the Schedule Number on each node. Hence, each time, this number is increased, the scheduler is notified. This *Notify* is interpreted by the scheduler as an acknowledgment of the successful receipt of the schedule with the Sequence Number given.

Figure 12 depicts the POST request with a JSON document describing the first schedule of our illustrative example. This request is broadcasted to all nodes of the network. The length of the CBOR transcription of the JSON document describing our schedule is 144 bytes. This

CBOR transcription is divided into 5 fragments of 32 bytes, hence $B_{sched} = 5$. In our example topology, the number of parents broadcasting the schedule is $P = 4$ (node 1, 2, 3, 4) and the sum of depths of nodes is $\sum_{u \neq sink} Depth(u) = 19$. For this first schedule, as all node just joined the network $\delta_P = 1$. Finally, $msg(Broadcast) = 4 \cdot 5 + 19 + 1 \cdot 4 = 43$.

```

REQ:
POST
coap://<ip>:5683/mg/6t/schedule
{
  "ScheduleNumber": "1",
  "Add":
  [
    [0, 0, 4, 1],
    [0, 1, 3, 1],
    [0, 2, 5, 2],
    [1, 0, 2, 1],
    [1, 1, 10, 3],
    [1, 2, 6, 4],
    [2, 0, 4, 1],
    [2, 1, 3, 1],
    [2, 2, 2, 1],
    [3, 0, 2, 1],
    [3, 1, 11, 3],
    [3, 2, 12, 4],
    [4, 0, 4, 1],
    [4, 1, 3, 1],
    [4, 2, 9, 2],
    [5, 0, 2, 1],
    [5, 1, 10, 3],
    [5, 2, 5, 4],
    [6, 0, 4, 1],
    [6, 1, 3, 1],
    [6, 2, 8, 2],
    [7, 0, 2, 1],
    [7, 1, 7, 3],
    [8, 0, 3, 1]
  ]
}

```

Figure 12: A POST request broadcasting the first schedule as a DIFF to all nodes in the network.

Figure 13 depicts the same POST request for the second schedule of our illustrative example. This request is also broadcasted to all nodes of the network including the joining node 13. The length of the CBOR transcription of the JSON document describing our schedule is 141 bytes. This CBOR transcription is divided into 5 fragments of 32 bytes, hence $B_{sched} = 5$. As in the previous case, the number of parents broadcasting the schedule is $P = 4$ but now the sum of depths of nodes is $\sum_{u \neq sink} Depth(u) = 21$. For this second schedule, node 13 just joined the network so $\delta_P = 1$. Finally, $msg(Broadcast) = 4 \cdot 5 + 21 + 1 \cdot 4 = 45$.

We can notice that the DIFF method requires the same overhead as the simple broadcast for the our illustrative network. However, when few transmissions have to be updated in a large scale network with a long schedule description, this DIFF method should clearly show its benefit.

```

REQ:
POST
coap://<ip>:5683/mg/6t/schedule
{
  "ScheduleNumber": "2",
  "Remove":
  [
    [0, 2, 5, 2],
    [1, 2, 6, 4],
    [2, 2, 2, 1],
    [3, 2, 12, 4],
    [4, 2, 9, 2],
    [5, 1, 10, 3],
    [5, 2, 5, 4],
    [6, 2, 8, 2],
    [7, 1, 7, 3],
    [8, 0, 3, 1]
  ],
  "Add":
  [
    [0, 2, 2, 1],
    [1, 2, 5, 4],
    [2, 2, 9, 2],
    [3, 2, 6, 4],
    [4, 2, 8, 2],
    [5, 1, 12, 4],
    [5, 2, 10, 3],
    [6, 2, 13, 2],
    [7, 1, 5, 4],
    [7, 2, 7, 3],
    [8, 0, 4, 1],
    [8, 1, 3, 1]
  ]
}

```

Figure 13: A POST request broadcasting the updated schedule as a DIFF to all nodes in the network.

3.7 Comparative evaluation

Table 4 shows a synthetic view of the overhead, in terms of number of messages, required to install or update a schedule for the small illustrative network presented in section 3.1. Available standardized mechanisms must be improved in order to be deployed in real implementation. In the next section, we propose three recommendations allowing a substantial performance saving.

Table 4: Number of messages needed to install or update schedules.

	install first schedule	update
Nmsg(Adhoc)	12	12
Nmsg(Naive)	352	100
Nmsg(Patch)	374	206
Nmsg(Broadcast)	43	45

4 Recommendations

We can propose some recommendations to decrease the number of messages needed to install a schedule as follows.

4.1 Short addresses at MAC layer

If an association mechanism could be implemented in a multi-hop fashion in order to ensure that the CPAN can do the mapping between Long and Short Addresses, 12 bytes could be saved at the MAC layer using the short addresses. The payload at MAC layer can then be extended to 110 bytes and the maximum applicative payload could be higher than 64 bytes (73 bytes exactly). So the block size relevant in this context should be 64 bytes, enhancing dramatically the performance of the PATCH method as it is shown in Table 5. Indeed, we need only 206 messages to install the first schedule with 64-byte blocks instead of 374 messages with 32-byte blocks and 104 messages to update the schedule instead of 210 with 32-byte blocks.

Table 5 presents the number of messages required for installing and updating the schedule computed for our illustrative network (see Figure 6a). We can notice a dramatic saving for this PATCH method compared to the previous results in section 3.4.

Table 5: Total number of messages required for installing and updating the schedules with the PATCH method.

Node	without node 13			with node 13		
	CBOR bytes	Blocks	Messages	CBOR bytes	Blocks	Messages
2	1049	17	34	397	7	14
3	918	15	30	208	4	8
4	918	15	30	533	9	18
5	263	5	20	269	5	20
6	132	3	12	67	1	4
7	132	3	12	70	1	4
8	132	3	12	67	1	4
9	132	3	12	67	1	4
10	263	5	20	70	1	4
11	132	3	12	0	0	0
12	132	3	12	136	3	12
13	-	-	-	132	3	12
Total	-	-	206	-	-	104

With the method consisting to broadcast a schedule to all nodes in the network, the saving in term of the number of messages is also significant. In the topology of our example, for the two solutions proposed (broadcast the whole schedule or broadcast a DIFF schedule), we obtain $msg(Broadcast) = 4 \cdot Bsched + 19 + 1 \cdot 4 = 35$ messages with $Bsched = 3$ for the first schedule. For updating the schedule with the presence of node 13, we obtain $msg(Broadcast) = 4 \cdot Bsched + 21 + 1 \cdot 4 = 37$ messages with $Bsched = 3$.

Table 6 shows that using short addresses can save around 45% of messages required to install the schedule with the Patch method and approximatively 50% of messages needed to update it. In the case of the broadcast method, the saving is lighter but still significant with a gain of 14% for installing the schedule and 17% for updating it.

4.2 A cellId coding the slotOffset and channelOffset

In order to compress the path descriptions used for the PATCH method, a coding of the tuple {slotOffset, channelOffset} should be used. Each cell must be uniquely identified by its *CellId*, where cells are numbered from slot 0, channel 0 up to channel 15 and then from slot 1, channel

Table 6: Number of messages needed to install or update schedules using short addresses at MAC layer.

	install first schedule	update
Nmsg(Adhoc)	12	12
Nmsg(Naive)	352	100
Nmsg(Patch)	206	104
Nmsg(Broadcast)	35	37

0 up to channel 15. This is equivalent to identify each cell by a sequence number equal to $SlotOffset * 16 + ChannelOffset$. This sequence number is coded on two bytes, where the 4 least significant bits are used to code the $ChannelOffset$ and the remaining 12 bits code the $SlotOffset$, enabling a maximum of 4096 slots in the SlotFrame.

Table 7 presents the number of messages required for installing and updating the schedule computed for our illustrative network (see Figure 6a). We can notice that this simple coding mechanism brings a greater gain compared to the previous results in section 3.4.

Table 7: Total number of messages required for installing and updating the schedules with the PATCH method using a cellId.

Node	without node 13			with node 13		
	CBOR bytes	Blocks	Messages	CBOR bytes	Blocks	Messages
2	729	12	24	275	5	10
3	636	10	20	132	3	6
4	636	10	20	371	6	12
5	181	3	12	132	3	12
6	92	2	8	44	1	4
7	92	2	8	45	1	4
8	92	2	8	44	1	4
9	92	2	8	44	1	4
10	183	3	12	44	1	4
11	92	2	8	0	0	0
12	92	2	8	44	1	4
13	-	-	-	92	2	8
Total	-	-	136	-	-	72

Figure 14 depicts the schedule description using the cellId coding in the context of the broadcasting method. In the case of our illustrative example, no saving can be noticed and more generally the cell coding impact seems negligible in this method.

Table 8 demonstrates that using a cellId coding, in addition with the short addresses, can save around 64% of messages required to install the schedule with the Patch method and more than 65% of messages needed to update it.

Table 8: Number of messages needed to install or update schedules.

	install first schedule	update
Nmsg(Adhoc)	12	12
Nmsg(Naive)	352	100
Nmsg(Patch)	136	72
Nmsg(Broadcast)	35	37

4.3 Shorter syntax for the PATCH method

In order to transmit lighter payloads, a more concise syntax for PATCH requests should be used. A such shorter syntax is presented in Figure 15. The CBOR transcription of this JSON document

```

REQ:
POST
coap://<ip>:5683/mg/6t/schedule
{
  "ScheduleNumber": "1",
  "Schedule":
  [
    [0, 4, 1],
    [1, 3, 1],
    [2, 5, 2],
    [16, 2, 1],
    [17, 10, 3],
    [18, 6, 4],
    [32, 4, 1],
    [33, 3, 1],
    [34, 2, 1],
    [48, 2, 1],
    [49, 11, 3],
    [50, 12, 4],
    [64, 4, 1],
    [65, 3, 1],
    [66, 9, 2],
    [80, 2, 1],
    [81, 10, 3],
    [82, 5, 4],
    [96, 4, 1],
    [97, 3, 1],
    [98, 8, 2],
    [112, 2, 1],
    [113, 7, 3],
    [128, 3, 1]
  ]
}

```

Figure 14: A POST request broadcasting the first schedule to all nodes in the network using a CellId.

transmitted in the PATCH request of node 5 is 26% shorter than the CBOR transcription using the syntax described in [13].

```

PATCH
coap://<ip>:5683/mg/6t/cellList
[
  {
    "o": "rpl",
    "p": "/nodeAddress?cellId=2",
    "v": 2
  },
  {
    "o": "rpl",
    "p": "/linkType?cellId=2",
    "v": 1
  },
  {
    "o": "rpl",
    "p": "/nodeAddress?cellId=82",
    "v": 4
  },
  {
    "o": "rpl",
    "p": "/LinkType?cellId=82",
    "v": 1
  }
]

```

Figure 15: Example of a PATCH request that installs the schedule on node 5 with a short syntax.

Table 9 presents the number of messages required for installing and updating the schedule computed for our illustrative network (see Figure 6a) with the recommended syntax. We

can notice that this shorter syntax brings a greater gain compared to the previous results in section 3.4.

Table 9: Total number of messages required for installing and updating the schedules with the PATCH method.

Node	without node 13			with node 13		
	CBOR bytes	Blocks	Messages	CBOR bytes	Blocks	Messages
2	537	9	18	203	4	8
3	468	8	16	96	2	4
4	468	8	16	275	5	10
5	133	3	12	96	2	8
6	68	1	4	32	1	4
7	70	1	4	33	1	4
8	68	1	4	32	1	4
9	68	1	4	32	1	4
10	135	3	12	32	1	4
11	68	1	4	0	0	0
12	68	1	4	32	1	4
13	-	-	-	68	1	4
Total	-	-	98	-	-	58

Table 10 shows that using a short syntax in PATCH requests, in addition with the two previous recommendations, can save approximatively 74% of messages required to install the schedule with the Patch method and around 72% of messages needed to update it.

Table 10: Number of messages needed to install or update schedules.

	install first schedule	update
Nmsg(Adhoc)	12	12
Nmsg(Naive)	352	100
Nmsg(Patch)	98	58
Nmsg(Broadcast)	35	37

5 Conclusion

In this research report, we describe three methods for the setting up of a scheduling in a IEEE 802.15.4e TSCH network, using all the provisions of IETF standards (6LoWPAN [4], CoAP [2], Block [7], CoMI [6] and Patch [13]).

We compare these methods with an ad hoc one. Considering the different results obtained, we may conclude that installing a new schedule using the IETF standards is very greedy in messages and consumes much bandwidth, energy and time.

That is why we offer three recommendations for limiting these overheads in terms of number of messages for the Patch method with a saving of more than 70% and for the Broadcast method with a gain of around 15%. Among these recommendations, a multihop association would lead to the highest benefit, but at the price of the largest impact on the specification and implementation. The other two recommendations have a more limited impact on performance but are easier to implement.

References

- [1] "IEEE Standard for Local and metropolitan area networks—Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) Amendment 1: MAC Sublayer," IEEE Std 802.15.4e-2012 (Amendment to IEEE Std 802.15.4-2011), Institute of Electrical and Electronics Engineers (IEEE), April 2012.
- [2] Z. Shelby, K. Hartke, and C. Bormann, "The Constrained Application Protocol (CoAP)," RFC 7252, RFC Editor, June 2014. <http://www.rfc-editor.org/rfc/rfc7252.txt>.
- [3] K. Al Agha, G. Chalhoub, A. Guitton, E. Livolant, S. Mahfoudh, P. Minet, M. Misson, J. Rahmé, T. Val, and A. van den Bossche, "Cross-layering in an industrial wireless sensor network: Case study of OCARI," *Journal of Networks*, vol. 4, no. 6, pp. 411–420, 2009.
- [4] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler, "Transmission of ipv6 packets over ieee 802.15.4 networks," RFC 4944, RFC Editor, September 2007. <http://www.rfc-editor.org/rfc/rfc4944.txt>.
- [5] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks," RFC 6550, RFC Editor, March 2012. <http://www.rfc-editor.org/rfc/rfc6550.txt>.
- [6] P. Van der Stok, A. Bierman, J. Schönwälder, and A. Sehgal, "CoAP Management Interface," Internet-Draft draft-vanderstok-core-comi-07, IETF Secretariat, July 2015. <http://www.ietf.org/internet-drafts/draft-vanderstok-core-comi-07.txt>.
- [7] C. Bormann and Z. Shelby, "Block-wise transfers in CoAP," Internet-Draft draft-ietf-core-block-18, IETF Secretariat, September 2015. <http://www.ietf.org/internet-drafts/draft-ietf-core-block-18.txt>.

-
- [8] “IEEE Standard for Local and metropolitan area networks—Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs),” IEEE Std 802.15.4-2011 (Revision of IEEE Std 802.15.4-2006), Institute of Electrical and Electronics Engineers (IEEE), September 2011.
- [9] X. Vilajosana and K. Pister, “Minimal 6tisch configuration,” Internet-Draft draft-ietf-6tisch-minimal-12, IETF Secretariat, September 2015. <http://www.ietf.org/internet-drafts/draft-ietf-6tisch-minimal-12.txt>.
- [10] R. Soua, P. Minet, and E. Livolant, “MODESA: an optimized multichannel slot assignment for raw data convergecast in wireless sensor networks,” in *31st IEEE International Performance Computing and Communications Conference, IPCCC 2012, Austin, TX, USA, December 1-3, 2012*, pp. 91–100, 2012.
- [11] R. Soua, P. Minet, and E. Livolant, “A distributed joint channel and slot assignment for convergecast in wireless sensor networks,” in *6th International Conference on New Technologies, Mobility and Security, NTMS 2014, Dubai, United Arab Emirates, March 30 - April 2, 2014*, pp. 1–5, 2014.
- [12] G. Chalhoub, E. Livolant, A. Guitton, A. van den Bossche, M. Misson, and T. Val, “Specifications and Evaluation of a MAC protocol for a LP-WPAN,” *Ad Hoc & Sensor Wireless Networks*, vol. 7, no. 1-2, pp. 69–89, 2009.
- [13] P. Van der Stok and A. Sehgal, “Patch method for constrained application protocol (coap),” Internet-Draft draft-vanderstok-core-patch-02, IETF Secretariat, October 2015. <http://www.ietf.org/internet-drafts/draft-vanderstok-core-patch-02.txt>.

Appendix A Example of a CBOR transcription of the JSON document describing a patch

```

84                                     # array(4)
a3                                     # map(3)
62                                     # text(2)
6f70                                  # "op"
67                                     # text(7)
7265706c616365                       # "replace"
64                                     # text(4)
70617468                              # "path"
78 29                                 # text(41)
2f6e6f64654164647265f73733f736c6f744f66667365743d30266368616e6e656c4f66667365743d32 # "/nodeAddress?slotOffset=0&channelOffset=2"
65                                     # text(5)
76616c7565                            # "value"
02                                     # unsigned(2)
a3                                     # map(3)
62                                     # text(2)
6f70                                  # "op"
67                                     # text(7)
7265706c616365                       # "replace"
64                                     # text(4)
70617468                              # "path"
78 26                                 # text(38)
2f6e69e6b547970653f736c6f744f66667365743d30266368616e6e656c4f66667365743d32 # "/linkType?slotOffset=0&channelOffset=2"
65                                     # text(5)
76616c7565                            # "value"
01                                     # unsigned(1)
a3                                     # map(3)
62                                     # text(2)
6f70                                  # "op"
67                                     # text(7)
7265706c616365                       # "replace"
64                                     # text(4)
70617468                              # "path"
78 29                                 # text(41)
2f6e6f64654164647265f73733f736c6f744f66667365743d35266368616e6e656c4f66667365743d32 # "/nodeAddress?slotOffset=5&channelOffset=2"
65                                     # text(5)
76616c7565                            # "value"
04                                     # unsigned(4)
a3                                     # map(3)
62                                     # text(2)
6f70                                  # "op"
67                                     # text(7)
7265706c616365                       # "replace"
64                                     # text(4)
70617468                              # "path"
78 26                                 # text(38)
2f6c69e6b547970653f736c6f744f66667365743d35266368616e6e656c4f66667365743d32 # "/LinkType?slotOffset=5&channelOffset=2"
65                                     # text(5)
76616c7565                            # "value"
01                                     # unsigned(1)

```

Appendix B Installing a schedule node by node with the PATCH method

B.1 First schedule without the node 13

B.1.1 PATCH request for install the schedule on node 2

```

PATCH
coap://<ip>:5683/mg/6t/cellList
[
  {
    "op": "replace",
    "path": "/nodeAddress?slotOffset=0&channelOffset=2",
    "value": 5
  },
  {
    "op": "replace",
    "path": "/linkType?slotOffset=0&channelOffset=2",
    "value": 2
  },
  {
    "op": "replace",
    "path": "/nodeAddress?slotOffset=1&channelOffset=0",
    "value": 1
  },
  {

```

```

    "op": "replace",
    "path": "/LinkType?slotOffset=1&channelOffset=0",
    "value": 1
  },
  {
    "op": "replace",
    "path": "/nodeAddress?slotOffset=2&channelOffset=2",
    "value": 1
  },
  {
    "op": "replace",
    "path": "/linkType?slotOffset=2&channelOffset=2",
    "value": 1
  },
  {
    "op": "replace",
    "path": "/nodeAddress?slotOffset=3&channelOffset=0",
    "value": 1
  },
  {
    "op": "replace",
    "path": "/linkType?slotOffset=3&channelOffset=0",
    "value": 1
  },
  {
    "op": "replace",
    "path": "/nodeAddress?slotOffset=4&channelOffset=2",
    "value": 9
  },
  {
    "op": "replace",
    "path": "/linkType?slotOffset=4&channelOffset=2",
    "value": 2
  },
  {
    "op": "replace",
    "path": "/nodeAddress?slotOffset=5&channelOffset=0",
    "value": 1
  },
  {
    "op": "replace",
    "path": "/LinkType?slotOffset=5&channelOffset=0",
    "value": 1
  },
  {
    "op": "replace",
    "path": "/nodeAddress?slotOffset=6&channelOffset=2",
    "value": 8
  },
  {
    "op": "replace",
    "path": "/linkType?slotOffset=6&channelOffset=2",
    "value": 2
  },
  {
    "op": "replace",
    "path": "/nodeAddress?slotOffset=7&channelOffset=0",
    "value": 1
  },
  {
    "op": "replace",
    "path": "/LinkType?slotOffset=7&channelOffset=0",
    "value": 1
  }
]

```

The 1049 bytes of the CBOR transcription of the JSON document describing the patch for install the schedule on node 2

B.1.2 PATCH request for install the schedule on node 3

PATCH

```

coap://<ip>:5683/mg/6t/cellList
[
  {
    "op": "replace",
    "path": "/nodeAddress?slotOffset=0&channelOffset=1",
    "value": 1
  },
  {
    "op": "replace",
    "path": "/linkType?slotOffset=0&channelOffset=1",
    "value": 1
  },
  {
    "op": "replace",
    "path": "/nodeAddress?slotOffset=1&channelOffset=1",
    "value": 10
  },
  {
    "op": "replace",
    "path": "/LinkType?slotOffset=1&channelOffset=1",
    "value": 2
  },
  {
    "op": "replace",
    "path": "/nodeAddress?slotOffset=2&channelOffset=1",
    "value": 1
  },
  {
    "op": "replace",
    "path": "/linkType?slotOffset=2&channelOffset=1",
    "value": 1
  },
  {
    "op": "replace",
    "path": "/nodeAddress?slotOffset=3&channelOffset=1",
    "value": 11
  },
  {
    "op": "replace",
    "path": "/linkType?slotOffset=3&channelOffset=1",
    "value": 2
  },
  {
    "op": "replace",
    "path": "/nodeAddress?slotOffset=4&channelOffset=1",
    "value": 1
  },
  {
    "op": "replace",
    "path": "/linkType?slotOffset=4&channelOffset=1",
    "value": 1
  },
  {
    "op": "replace",
    "path": "/nodeAddress?slotOffset=5&channelOffset=1",
    "value": 10
  },
  {
    "op": "replace",
    "path": "/LinkType?slotOffset=5&channelOffset=1",
    "value": 2
  },
  {
    "op": "replace",
    "path": "/nodeAddress?slotOffset=8&channelOffset=1",
    "value": 1
  },
  {
    "op": "replace",
    "path": "/linkType?slotOffset=8&channelOffset=1",
    "value": 1
  }
]

```

The 918 bytes of the CBOR transcription of the JSON document describing the patch for install the schedule on node 3

B.1.3 PATCH request for install the schedule on node 4

```
PATCH
coap://<ip>:5683/mg/6t/cellList
[
  {
    "op": "replace",
    "path": "/nodeAddress?slotOffset=0&channelOffset=0",
    "value": 1
  },
  {
    "op": "replace",
    "path": "/linkType?slotOffset=0&channelOffset=0",
    "value": 1
  },
  {
    "op": "replace",
    "path": "/nodeAddress?slotOffset=1&channelOffset=2",
    "value": 6
  },
  {
    "op": "replace",
    "path": "/LinkType?slotOffset=1&channelOffset=2",
    "value": 2
  },
  {
    "op": "replace",
    "path": "/nodeAddress?slotOffset=2&channelOffset=0",
    "value": 1
  },
  {
    "op": "replace",
    "path": "/linkType?slotOffset=2&channelOffset=0",
    "value": 1
  },
  {
    "op": "replace",
    "path": "/nodeAddress?slotOffset=3&channelOffset=2",
    "value": 12
  },
  {
    "op": "replace",
    "path": "/linkType?slotOffset=3&channelOffset=2",
    "value": 2
  },
  {
    "op": "replace",
    "path": "/nodeAddress?slotOffset=4&channelOffset=0",
    "value": 1
  },
  {
    "op": "replace",
    "path": "/linkType?slotOffset=4&channelOffset=0",
    "value": 1
  },
  {
    "op": "replace",
    "path": "/nodeAddress?slotOffset=5&channelOffset=2",
    "value": 5
  },
  {
    "op": "replace",
    "path": "/LinkType?slotOffset=5&channelOffset=2",
    "value": 2
  },
  {
    "op": "replace",
    "path": "/nodeAddress?slotOffset=6&channelOffset=0",
    "value": 1
  }
]
```

```

    },
    {
      "op": "replace",
      "path": "/linkType?slotOffset=6&channelOffset=0",
      "value": 1
    }
  ]

```

The 918 bytes of the CBOR transcription of the JSON document describing the patch for install the schedule on node 4

B.1.4 PATCH request for install the schedule on node 5

```

PATCH
coap://<ip>:5683/mg/6t/cellList
[
  {
    "op": "replace",
    "path": "/nodeAddress?slotOffset=0&channelOffset=2",
    "value": 2
  },
  {
    "op": "replace",
    "path": "/linkType?slotOffset=0&channelOffset=2",
    "value": 1
  },
  {
    "op": "replace",
    "path": "/nodeAddress?slotOffset=5&channelOffset=2",
    "value": 4
  },
  {
    "op": "replace",
    "path": "/LinkType?slotOffset=5&channelOffset=2",
    "value": 1
  }
]

```

The 263 bytes of the CBOR transcription of the JSON document describing the patch for install the schedule on node 5

B.1.5 PATCH request for install the schedule on node 6

```

PATCH
coap://<ip>:5683/mg/6t/cellList
[
  {
    "op": "replace",
    "path": "/nodeAddress?slotOffset=1&channelOffset=2",
    "value": 4
  },
  {
    "op": "replace",
    "path": "/linkType?slotOffset=1&channelOffset=2",
    "value": 1
  }
]

```

The 132 bytes of the CBOR transcription of the JSON document describing the patch for install the schedule on node 6

B.1.6 PATCH request for install the schedule on node 7

```

PATCH
coap://<ip>:5683/mg/6t/cellList
[
  {

```



```

    "op": "replace",
    "path": "/nodeAddress?slotOffset=7&channelOffset=1",
    "value": 3
  },
  {
    "op": "replace",
    "path": "/linkType?slotOffset=7&channelOffset=1",
    "value": 1
  }
]

```

The 132 bytes of the CBOR transcription of the JSON document describing the patch for install the schedule on node 7

B.1.7 PATCH request for install the schedule on node 8

```

PATCH
coap://<ip>:5683/mg/6t/cellList
[
  {
    "op": "replace",
    "path": "/nodeAddress?slotOffset=6&channelOffset=2",
    "value": 2
  },
  {
    "op": "replace",
    "path": "/linkType?slotOffset=6&channelOffset=2",
    "value": 1
  }
]

```

The 132 bytes of the CBOR transcription of the JSON document describing the patch for install the schedule on node 8

B.1.8 PATCH request for install the schedule on node 9

```

PATCH
coap://<ip>:5683/mg/6t/cellList
[
  {
    "op": "replace",
    "path": "/nodeAddress?slotOffset=4&channelOffset=2",
    "value": 2
  },
  {
    "op": "replace",
    "path": "/linkType?slotOffset=4&channelOffset=2",
    "value": 1
  }
]

```

The 132 bytes of the CBOR transcription of the JSON document describing the patch for install the schedule on node 9

B.1.9 PATCH request for install the schedule on node 10

```

PATCH
coap://<ip>:5683/mg/6t/cellList
[
  {
    "op": "replace",
    "path": "/nodeAddress?slotOffset=1&channelOffset=1",
    "value": 3
  },
  {
    "op": "replace",
    "path": "/linkType?slotOffset=1&channelOffset=1",

```

```

    "value": 1
  },
  {
    "op": "replace",
    "path": "/nodeAddress?slotOffset=5&channelOffset=1",
    "value": 3
  },
  {
    "op": "replace",
    "path": "/LinkType?slotOffset=5&channelOffset=1",
    "value": 1
  }
]

```

The 263 bytes of the CBOR transcription of the JSON document describing the patch for install the schedule on node 10

B.1.10 PATCH request for install the schedule on node 11

```

PATCH
coap://<ip>:5683/mg/6t/cellList
[
  {
    "op": "replace",
    "path": "/nodeAddress?slotOffset=3&channelOffset=1",
    "value": 3
  },
  {
    "op": "replace",
    "path": "/linkType?slotOffset=3&channelOffset=1",
    "value": 1
  }
]

```

The 132 bytes of the CBOR transcription of the JSON document describing the patch for install the schedule on node 11

B.1.11 PATCH request for install the schedule on node 12

```

PATCH
coap://<ip>:5683/mg/6t/cellList
[
  {
    "op": "replace",
    "path": "/nodeAddress?slotOffset=3&channelOffset=2",
    "value": 4
  },
  {
    "op": "replace",
    "path": "/linkType?slotOffset=3&channelOffset=2",
    "value": 1
  }
]

```

The 132 bytes of the CBOR transcription of the JSON document describing the patch for install the schedule on node 12

B.2 second schedule with the node 13

B.2.1 PATCH request for update the schedule on node 2

```

PATCH
coap://<ip>:5683/mg/6t/cellList
[
  {
    "op": "replace",

```

```

    "path": "/nodeAddress?slotOffset=0&channelOffset=2",
    "value": 1
  },
  {
    "op": "replace",
    "path": "/linkType?slotOffset=0&channelOffset=2",
    "value": 1
  },
  {
    "op": "replace",
    "path": "/nodeAddress?slotOffset=2&channelOffset=2",
    "value": 9
  },
  {
    "op": "replace",
    "path": "/LinkType?slotOffset=2&channelOffset=2",
    "value": 2
  },
  {
    "op": "replace",
    "path": "/nodeAddress?slotOffset=4&channelOffset=2",
    "value": 8
  },
  {
    "op": "replace",
    "path": "/nodeAddress?slotOffset=6&channelOffset=2",
    "value": 13
  }
]

```

The 397 bytes of the CBOR transcription of the JSON document describing the patch for update the schedule on node 2

B.2.2 PATCH request for update the schedule on node 3

```

PATCH
coap://<ip>:5683/mg/6t/cellList
[
  {
    "op": "replace",
    "path": "/channelOffset?slotOffset=5&channelOffset=1",
    "value": 2
  },
  {
    "op": "replace",
    "path": "/channelOffset?slotOffset=7&channelOffset=1",
    "value": 2
  },
  {
    "op": "replace",
    "path": "/channelOffset?slotOffset=8&channelOffset=0",
    "value": 1
  }
]

```

The 208 bytes of the CBOR transcription of the JSON document describing the patch for update the schedule on node 3

B.2.3 PATCH request for update the schedule on node 4

```

PATCH
coap://<ip>:5683/mg/6t/cellList
[
  {
    "op": "replace",
    "path": "/nodeAddress?slotOffset=1&channelOffset=2",
    "value": 5
  },
  {

```

```

    "op": "replace",
    "path": "/nodeAddress?slotOffset=3&channelOffset=2",
    "value": 6
  },
  {
    "op": "replace",
    "path": "/channelOffset?slotOffset=5&channelOffset=2",
    "value": 1
  },
  {
    "op": "replace",
    "path": "/nodeAddress?slotOffset=5&channelOffset=1",
    "value": 12
  },
  {
    "op": "replace",
    "path": "/nodeAddress?slotOffset=7&channelOffset=1",
    "value": 5
  },
  {
    "op": "replace",
    "path": "/linkType?slotOffset=7&channelOffset=1",
    "value": 2
  },
  {
    "op": "replace",
    "path": "/nodeAddress?slotOffset=8&channelOffset=0",
    "value": 1
  },
  {
    "op": "replace",
    "path": "/linkType?slotOffset=8&channelOffset=0",
    "value": 1
  }
]

```

The 533 bytes of the CBOR transcription of the JSON document describing the patch for update the schedule on node 4

B.2.4 PATCH request for update the schedule on node 5

```

PATCH
coap://<ip>:5683/mg/6t/cellList
[
  {
    "op": "replace",
    "path": "/slotOffset?slotOffset=0&channelOffset=2",
    "value": 1
  },
  {
    "op": "replace",
    "path": "/nodeAddress?slotOffset=1&channelOffset=2",
    "value": 4
  },
  {
    "op": "replace",
    "path": "/slotOffset?slotOffset=5&channelOffset=2",
    "value": 7
  },
  {
    "op": "replace",
    "path": "/channelOffset?slotOffset=7&channelOffset=2",
    "value": 1
  }
]

```

The 269 bytes of the CBOR transcription of the JSON document describing the patch for update the schedule on node 5

B.2.5 PATCH request for update the schedule on node 6

```
PATCH
coap://<ip>:5683/mg/6t/cellList
[
  {
    "op": "replace",
    "path": "/slotOffset?slotOffset=1&channelOffset=2",
    "value": 3
  }
]
```

The 67 bytes of the CBOR transcription of the JSON document describing the patch for update the schedule on node 6

B.2.6 PATCH request for update the schedule on node 7

```
PATCH
coap://<ip>:5683/mg/6t/cellList
[
  {
    "op": "replace",
    "path": "/channelOffset?slotOffset=7&channelOffset=1",
    "value": 2
  }
]
```

The 70 bytes of the CBOR transcription of the JSON document describing the patch for update the schedule on node 7

B.2.7 PATCH request for update the schedule on node 8

```
PATCH
coap://<ip>:5683/mg/6t/cellList
[
  {
    "op": "replace",
    "path": "/slotOffset?slotOffset=6&channelOffset=2",
    "value": 4
  }
]
```

The 67 bytes of the CBOR transcription of the JSON document describing the patch for update the schedule on node 8

B.2.8 PATCH request for update the schedule on node 9

```
PATCH
coap://<ip>:5683/mg/6t/cellList
[
  {
    "op": "replace",
    "path": "/slotOffset?slotOffset=4&channelOffset=2",
    "value": 2
  }
]
```

The 67 bytes of the CBOR transcription of the JSON document describing the patch for update the schedule on node 9

B.2.9 PATCH request for update the schedule on node 10

```

PATCH
coap://<ip>:5683/mg/6t/cellList
[
  {
    "op": "replace",
    "path": "/channelOffset?slotOffset=5&channelOffset=1",
    "value": 2
  }
]

```

The 70 bytes of the CBOR transcription of the JSON document describing the patch for update the schedule on node 10

B.2.10 PATCH request for update the schedule on node 11

The 0 bytes of the CBOR transcription of the JSON document describing the patch for update the schedule on node 11

B.2.11 PATCH request for update the schedule on node 12

```

PATCH
coap://<ip>:5683/mg/6t/cellList
[
  {
    "op": "replace",
    "path": "/slotOffset?slotOffset=3&channelOffset=2",
    "value": 5
  },
  {
    "op": "replace",
    "path": "/channelOffset?slotOffset=5&channelOffset=2",
    "value": 1
  }
]

```

The 136 bytes of the CBOR transcription of the JSON document describing the patch for update the schedule on node 12

B.2.12 PATCH request for update the schedule on node 13

```

PATCH
coap://<ip>:5683/mg/6t/cellList
[
  {
    "op": "replace",
    "path": "/nodeAddress?slotOffset=6&channelOffset=2",
    "value": 2
  },
  {
    "op": "replace",
    "path": "/linkType?slotOffset=6&channelOffset=2",
    "value": 1
  }
]

```

The 132 bytes of the CBOR transcription of the JSON document describing the patch for update the schedule on node 13

B.3 PATCH requests installing the first schedule with cellId coding

B.3.1 PATCH request for install the schedule on node 2 with cellId coding

```

PATCH

```

```

coap://<ip>:5683/mg/6t/cellList
[
  {
    "op": "replace",
    "path": "/nodeAddress?cellId=2",
    "value": 5
  },
  {
    "op": "replace",
    "path": "/linkType?cellId=2",
    "value": 2
  },
  {
    "op": "replace",
    "path": "/nodeAddress?cellId=16",
    "value": 1
  },
  {
    "op": "replace",
    "path": "/LinkType?cellId=16",
    "value": 1
  },
  {
    "op": "replace",
    "path": "/nodeAddress?cellId=34",
    "value": 1
  },
  {
    "op": "replace",
    "path": "/linkType?cellId=34",
    "value": 1
  },
  {
    "op": "replace",
    "path": "/nodeAddress?cellId=48",
    "value": 1
  },
  {
    "op": "replace",
    "path": "/linkType?cellId=48",
    "value": 1
  },
  {
    "op": "replace",
    "path": "/nodeAddress?cellId=66",
    "value": 9
  },
  {
    "op": "replace",
    "path": "/linkType?cellId=66",
    "value": 2
  },
  {
    "op": "replace",
    "path": "/nodeAddress?cellId=80",
    "value": 1
  },
  {
    "op": "replace",
    "path": "/LinkType?cellId=80",
    "value": 1
  },
  {
    "op": "replace",
    "path": "/nodeAddress?cellId=98",
    "value": 8
  },
  {
    "op": "replace",
    "path": "/linkType?cellId=98",
    "value": 2
  },
  {

```

```

    "op": "replace",
    "path": "/nodeAddress?cellId=112",
    "value": 1
  },
  {
    "op": "replace",
    "path": "/LinkType?cellId=112",
    "value": 1
  }
]

```

The 729 bytes of the CBOR transcription of the JSON document describing the patch for install the schedule on node 2

B.3.2 PATCH request for install the schedule on node 3 with cellId coding

PATCH

coap://<ip>:5683/mg/6t/cellList

```

[
  {
    "op": "replace",
    "path": "/nodeAddress?cellId=1",
    "value": 1
  },
  {
    "op": "replace",
    "path": "/linkType?cellId=1",
    "value": 1
  },
  {
    "op": "replace",
    "path": "/nodeAddress?cellId=17",
    "value": 10
  },
  {
    "op": "replace",
    "path": "/LinkType?cellId=17",
    "value": 2
  },
  {
    "op": "replace",
    "path": "/nodeAddress?cellId=33",
    "value": 1
  },
  {
    "op": "replace",
    "path": "/linkType?cellId=33",
    "value": 1
  },
  {
    "op": "replace",
    "path": "/nodeAddress?cellId=49",
    "value": 11
  },
  {
    "op": "replace",
    "path": "/linkType?cellId=49",
    "value": 2
  },
  {
    "op": "replace",
    "path": "/nodeAddress?cellId=65",
    "value": 1
  },
  {
    "op": "replace",
    "path": "/linkType?cellId=65",
    "value": 1
  },
  {
    "op": "replace",

```



```

    "path": "/nodeAddress?cellId=81",
    "value": 10
  },
  {
    "op": "replace",
    "path": "/LinkType?cellId=81",
    "value": 2
  },
  {
    "op": "replace",
    "path": "/nodeAddress?cellId=97",
    "value": 1
  },
  {
    "op": "replace",
    "path": "/linkType?cellId=97",
    "value": 1
  }
]

```

The 636 bytes of the CBOR transcription of the JSON document describing the patch for install the schedule on node 3

B.3.3 PATCH request for install the schedule on node 4 with cellID coding

```

PATCH
coap://<ip>:5683/mg/6t/cellList
[
  {
    "op": "replace",
    "path": "/nodeAddress?cellId=0",
    "value": 1
  },
  {
    "op": "replace",
    "path": "/linkType?cellId=0",
    "value": 1
  },
  {
    "op": "replace",
    "path": "/nodeAddress?cellId=18",
    "value": 6
  },
  {
    "op": "replace",
    "path": "/LinkType?cellId=18",
    "value": 2
  },
  {
    "op": "replace",
    "path": "/nodeAddress?cellId=32",
    "value": 1
  },
  {
    "op": "replace",
    "path": "/linkType?cellId=32",
    "value": 1
  },
  {
    "op": "replace",
    "path": "/nodeAddress?cellId=50",
    "value": 12
  },
  {
    "op": "replace",
    "path": "/linkType?cellId=50",
    "value": 2
  },
  {
    "op": "replace",
    "path": "/nodeAddress?cellId=64",

```

```

    "value": 1
  },
  {
    "op": "replace",
    "path": "/linkType?cellId=64",
    "value": 1
  },
  {
    "op": "replace",
    "path": "/nodeAddress?cellId=82",
    "value": 5
  },
  {
    "op": "replace",
    "path": "/LinkType?cellId=82",
    "value": 2
  },
  {
    "op": "replace",
    "path": "/nodeAddress?cellId=96",
    "value": 1
  },
  {
    "op": "replace",
    "path": "/linkType?cellId=96",
    "value": 1
  }
]

```

The 636 bytes of the CBOR transcription of the JSON document describing the patch for install the schedule on node 4

B.3.4 PATCH request for install the schedule on node 5 with cellID coding

```

PATCH
coap://<ip>:5683/mg/6t/cellList
[
  {
    "op": "replace",
    "path": "/nodeAddress?cellId=2",
    "value": 2
  },
  {
    "op": "replace",
    "path": "/linkType?cellId=2",
    "value": 1
  },
  {
    "op": "replace",
    "path": "/nodeAddress?cellId=82",
    "value": 4
  },
  {
    "op": "replace",
    "path": "/LinkType?cellId=82",
    "value": 1
  }
]

```

The 181 bytes of the CBOR transcription of the JSON document describing the patch for install the schedule on node 5

B.3.5 PATCH request for install the schedule on node 6 with cellID coding

```

PATCH
coap://<ip>:5683/mg/6t/cellList
[
  {
    "op": "replace",

```

```

    "path": "/nodeAddress?cellId=18",
    "value": 4
  },
  {
    "op": "replace",
    "path": "/linkType?cellId=18",
    "value": 1
  }
]

```

The 92 bytes of the CBOR transcription of the JSON document describing the patch for install the schedule on node 6

B.3.6 PATCH request for install the schedule on node 7 with cellID coding

```

PATCH
coap://<ip>:5683/mg/6t/cellList
[
  {
    "op": "replace",
    "path": "/nodeAddress?cellId=113",
    "value": 3
  },
  {
    "op": "replace",
    "path": "/linkType?cellId=113",
    "value": 1
  }
]

```

The 94 bytes of the CBOR transcription of the JSON document describing the patch for install the schedule on node 7

B.3.7 PATCH request for install the schedule on node 8 with cellID coding

```

PATCH
coap://<ip>:5683/mg/6t/cellList
[
  {
    "op": "replace",
    "path": "/nodeAddress?cellId=98",
    "value": 2
  },
  {
    "op": "replace",
    "path": "/linkType?cellId=98",
    "value": 1
  }
]

```

The 92 bytes of the CBOR transcription of the JSON document describing the patch for install the schedule on node 8

B.3.8 PATCH request for install the schedule on node 9 with cellID coding

```

PATCH
coap://<ip>:5683/mg/6t/cellList
[
  {
    "op": "replace",
    "path": "/nodeAddress?cellId=66",
    "value": 2
  },
  {
    "op": "replace",
    "path": "/linkType?cellId=66",
    "value": 1
  }
]

```

```

    }
]

```

The 92 bytes of the CBOR transcription of the JSON document describing the patch for install the schedule on node 9

B.3.9 PATCH request for install the schedule on node 10 with cellID coding

```

PATCH
coap://<ip>:5683/mg/6t/cellList
[
  {
    "op": "replace",
    "path": "/nodeAddress?cellId=17",
    "value": 3
  },
  {
    "op": "replace",
    "path": "/linkType?cellId=17",
    "value": 1
  },
  {
    "op": "replace",
    "path": "/nodeAddress?cellId=81",
    "value": 3
  },
  {
    "op": "replace",
    "path": "/LinkType?cellId=81",
    "value": 1
  }
]

```

The 183 bytes of the CBOR transcription of the JSON document describing the patch for install the schedule on node 10

B.3.10 PATCH request for install the schedule on node 11 with cellID coding

```

PATCH
coap://<ip>:5683/mg/6t/cellList
[
  {
    "op": "replace",
    "path": "/nodeAddress?cellId=49",
    "value": 3
  },
  {
    "op": "replace",
    "path": "/linkType?cellId=49",
    "value": 1
  }
]

```

The 92 bytes of the CBOR transcription of the JSON document describing the patch for install the schedule on node 11

B.3.11 PATCH request for install the schedule on node 12 with cellID coding

```

PATCH
coap://<ip>:5683/mg/6t/cellList
[
  {
    "op": "replace",
    "path": "/nodeAddress?cellId=50",
    "value": 4
  },
  {

```

```

    "op": "replace",
    "path": "/linkType?cellId=50",
    "value": 1
  }
]

```

The 92 bytes of the CBOR transcription of the JSON document describing the patch for install the schedule on node 12

B.4 PATCH requests installing the second schedule with cellID coding

B.4.1 PATCH request for update the schedule on node 2 with cellID coding

```

PATCH
coap://<ip>:5683/mg/6t/cellList
[
  {
    "op": "replace",
    "path": "/nodeAddress?cellId=2",
    "value": 1
  },
  {
    "op": "replace",
    "path": "/linkType?cellId=2",
    "value": 1
  },
  {
    "op": "replace",
    "path": "/nodeAddress?cellId=34",
    "value": 9
  },
  {
    "op": "replace",
    "path": "/LinkType?cellId=34",
    "value": 2
  },
  {
    "op": "replace",
    "path": "/nodeAddress?cellId=66",
    "value": 8
  },
  {
    "op": "replace",
    "path": "/nodeAddress?cellId=66",
    "value": 13
  }
]

```

The 275 bytes of the CBOR transcription of the JSON document describing the patch for update the schedule on node 2

B.4.2 PATCH request for update the schedule on node 3 with cellID coding

```

PATCH
coap://<ip>:5683/mg/6t/cellList
[
  {
    "op": "replace",
    "path": "/cellId?cellId=81",
    "value": 82
  },
  {
    "op": "replace",
    "path": "/cellId?cellId=112",
    "value": 113
  },
  {
    "op": "replace",

```

```

    "path": "/cellId?cellId=128",
    "value": 129
  }
]

```

The 132 bytes of the CBOR transcription of the JSON document describing the patch for update the schedule on node 3

B.4.3 PATCH request for update the schedule on node 4 with cellID coding

```

PATCH
coap://<ip>:5683/mg/6t/cellList
[
  {
    "op": "replace",
    "path": "/nodeAddress?cellId=18",
    "value": 5
  },
  {
    "op": "replace",
    "path": "/nodeAddress?cellId=50",
    "value": 6
  },
  {
    "op": "replace",
    "path": "/cellId?cellId=82",
    "value": 81
  },
  {
    "op": "replace",
    "path": "/nodeAddress?cellId=81",
    "value": 12
  },
  {
    "op": "replace",
    "path": "/nodeAddress?cellId=113",
    "value": 5
  },
  {
    "op": "replace",
    "path": "/linkType?cellId=113",
    "value": 2
  },
  {
    "op": "replace",
    "path": "/nodeAddress?cellId=128",
    "value": 1
  },
  {
    "op": "replace",
    "path": "/linkType?cellId=128",
    "value": 1
  }
]

```

The 371 bytes of the CBOR transcription of the JSON document describing the patch for update the schedule on node 4

B.4.4 PATCH request for update the schedule on node 5 with cellID coding

```

PATCH
coap://<ip>:5683/mg/6t/cellList
[
  {
    "op": "replace",
    "path": "/cellId?cellId=2",
    "value": 18
  },
  {

```

```

    "op": "replace",
    "path": "/nodeAddress?cellId=18",
    "value": 4
  },
  {
    "op": "replace",
    "path": "/cellId?cellId=82",
    "value": 113
  }
]

```

The 132 bytes of the CBOR transcription of the JSON document describing the patch for update the schedule on node 5

B.4.5 PATCH request for update the schedule on node 6 with cellID coding

```

PATCH
coap://<ip>:5683/mg/6t/cellList
[
  {
    "op": "replace",
    "path": "/cellId?cellId=18",
    "value": 50
  }
]

```

The 44 bytes of the CBOR transcription of the JSON document describing the patch for update the schedule on node 6

B.4.6 PATCH request for update the schedule on node 7 with cellID coding

```

PATCH
coap://<ip>:5683/mg/6t/cellList
[
  {
    "op": "replace",
    "path": "/cellId?cellId=113",
    "value": 114
  }
]

```

The 45 bytes of the CBOR transcription of the JSON document describing the patch for update the schedule on node 7

B.4.7 PATCH request for update the schedule on node 8 with cellID coding

```

PATCH
coap://<ip>:5683/mg/6t/cellList
[
  {
    "op": "replace",
    "path": "/cellId?cellId=98",
    "value": 66
  }
]

```

The 44 bytes of the CBOR transcription of the JSON document describing the patch for update the schedule on node 8

B.4.8 PATCH request for update the schedule on node 9 with cellID coding

```

PATCH
coap://<ip>:5683/mg/6t/cellList
[

```

```

    {
      "op": "replace",
      "path": "/cellId?cellId=66",
      "value": 34
    }
  ]

```

The 44 bytes of the CBOR transcription of the JSON document describing the patch for update the schedule on node 9

B.4.9 PATCH request for update the schedule on node 10 with cellID coding

```

PATCH
coap://<ip>:5683/mg/6t/cellList
[
  {
    "op": "replace",
    "path": "/cellId?cellId=81",
    "value": 82
  }
]

```

The 44 bytes of the CBOR transcription of the JSON document describing the patch for update the schedule on node 10

B.4.10 PATCH request for update the schedule on node 11 with cellID coding

The 0 bytes of the CBOR transcription of the JSON document describing the patch for update the schedule on node 11

B.4.11 PATCH request for update the schedule on node 12 with cellID coding

```

PATCH
coap://<ip>:5683/mg/6t/cellList
[
  {
    "op": "replace",
    "path": "/cellId?cellId=50",
    "value": 81
  }
]

```

The 44 bytes of the CBOR transcription of the JSON document describing the patch for update the schedule on node 12

B.4.12 PATCH request for update the schedule on node 13 with cellID coding

```

PATCH
coap://<ip>:5683/mg/6t/cellList
[
  {
    "op": "replace",
    "path": "/nodeAddress?cellId=98",
    "value": 2
  },
  {
    "op": "replace",
    "path": "/linkType?cellId=98",
    "value": 1
  }
]

```

The 92 bytes of the CBOR transcription of the JSON document describing the patch for update the schedule on node 13

B.5 PATCH requests installing the first schedule with a shorter syntax

B.5.1 PATCH request for install the schedule on node 2 with a shorter syntax

```

PATCH
coap://<ip>:5683/mg/6t/cellList
[
  {
    "o": "rp1",
    "p": "/nodeAddress?cellId=2",
    "v": 5
  },
  {
    "o": "rp1",
    "p": "/linkType?cellId=2",
    "v": 2
  },
  {
    "o": "rp1",
    "p": "/nodeAddress?cellId=16",
    "v": 1
  },
  {
    "o": "rp1",
    "p": "/LinkType?cellId=16",
    "v": 1
  },
  {
    "o": "rp1",
    "p": "/nodeAddress?cellId=34",
    "v": 1
  },
  {
    "o": "rp1",
    "p": "/linkType?cellId=34",
    "v": 1
  },
  {
    "o": "rp1",
    "p": "/nodeAddress?cellId=48",
    "v": 1
  },
  {
    "o": "rp1",
    "p": "/linkType?cellId=48",
    "v": 1
  },
  {
    "o": "rp1",
    "p": "/nodeAddress?cellId=66",
    "v": 9
  },
  {
    "o": "rp1",
    "p": "/linkType?cellId=66",
    "v": 2
  },
  {
    "o": "rp1",
    "p": "/nodeAddress?cellId=80",
    "v": 1
  },
  {
    "o": "rp1",
    "p": "/LinkType?cellId=80",
    "v": 1
  },
  {
    "o": "rp1",
    "p": "/nodeAddress?cellId=98",
    "v": 8
  },
  {

```

```

    "o": "rp1",
    "p": "/linkType?cellId=98",
    "v": 2
  },
  {
    "o": "rp1",
    "p": "/nodeAddress?cellId=112",
    "v": 1
  },
  {
    "o": "rp1",
    "p": "/LinkType?cellId=112",
    "v": 1
  }
]

```

The 537 bytes of the CBOR transcription of the JSON document describing the patch for install the schedule on node 2

B.5.2 PATCH request for install the schedule on node 3 with a shorter syntax

```

PATCH
coap://<ip>:5683/mg/6t/cellList
[
  {
    "o": "rp1",
    "p": "/nodeAddress?cellId=1",
    "v": 1
  },
  {
    "o": "rp1",
    "p": "/linkType?cellId=1",
    "v": 1
  },
  {
    "o": "rp1",
    "p": "/nodeAddress?cellId=17",
    "v": 10
  },
  {
    "o": "rp1",
    "p": "/LinkType?cellId=17",
    "v": 2
  },
  {
    "o": "rp1",
    "p": "/nodeAddress?cellId=33",
    "v": 1
  },
  {
    "o": "rp1",
    "p": "/linkType?cellId=33",
    "v": 1
  },
  {
    "o": "rp1",
    "p": "/nodeAddress?cellId=49",
    "v": 11
  },
  {
    "o": "rp1",
    "p": "/linkType?cellId=49",
    "v": 2
  },
  {
    "o": "rp1",
    "p": "/nodeAddress?cellId=65",
    "v": 1
  },
  {
    "o": "rp1",

```

```

    "p": "/linkType?cellId=65",
    "v": 1
  },
  {
    "o": "rpl",
    "p": "/nodeAddress?cellId=81",
    "v": 10
  },
  {
    "o": "rpl",
    "p": "/LinkType?cellId=81",
    "v": 2
  },
  {
    "o": "rpl",
    "p": "/nodeAddress?cellId=97",
    "v": 1
  },
  {
    "o": "rpl",
    "p": "/linkType?cellId=97",
    "v": 1
  }
]

```

The 468 bytes of the CBOR transcription of the JSON document describing the patch for install the schedule on node 3

B.5.3 PATCH request for install the schedule on node 4 with a shorter syntax

```

PATCH
coap://<ip>:5683/mg/6t/cellList
[
  {
    "o": "rpl",
    "p": "/nodeAddress?cellId=0",
    "v": 1
  },
  {
    "o": "rpl",
    "p": "/linkType?cellId=0",
    "v": 1
  },
  {
    "o": "rpl",
    "p": "/nodeAddress?cellId=18",
    "v": 6
  },
  {
    "o": "rpl",
    "p": "/LinkType?cellId=18",
    "v": 2
  },
  {
    "o": "rpl",
    "p": "/nodeAddress?cellId=32",
    "v": 1
  },
  {
    "o": "rpl",
    "p": "/linkType?cellId=32",
    "v": 1
  },
  {
    "o": "rpl",
    "p": "/nodeAddress?cellId=50",
    "v": 12
  },
  {
    "o": "rpl",
    "p": "/linkType?cellId=50",

```

```

    "v": 2
  },
  {
    "o": "rpl",
    "p": "/nodeAddress?cellId=64",
    "v": 1
  },
  {
    "o": "rpl",
    "p": "/linkType?cellId=64",
    "v": 1
  },
  {
    "o": "rpl",
    "p": "/nodeAddress?cellId=82",
    "v": 5
  },
  {
    "o": "rpl",
    "p": "/LinkType?cellId=82",
    "v": 2
  },
  {
    "o": "rpl",
    "p": "/nodeAddress?cellId=96",
    "v": 1
  },
  {
    "o": "rpl",
    "p": "/linkType?cellId=96",
    "v": 1
  }
]

```

The 468 bytes of the CBOR transcription of the JSON document describing the patch for install the schedule on node 4

B.5.4 PATCH request for install the schedule on node 5 with a shorter syntax

```

PATCH
coap://<ip>:5683/mg/6t/cellList
[
  {
    "o": "rpl",
    "p": "/nodeAddress?cellId=2",
    "v": 2
  },
  {
    "o": "rpl",
    "p": "/linkType?cellId=2",
    "v": 1
  },
  {
    "o": "rpl",
    "p": "/nodeAddress?cellId=82",
    "v": 4
  },
  {
    "o": "rpl",
    "p": "/LinkType?cellId=82",
    "v": 1
  }
]

```

The 133 bytes of the CBOR transcription of the JSON document describing the patch for install the schedule on node 5

B.5.5 PATCH request for install the schedule on node 6 with a shorter syntax

```

PATCH

```

```

coap://<ip>:5683/mg/6t/cellList
[
  {
    "o": "rpl",
    "p": "/nodeAddress?cellId=18",
    "v": 4
  },
  {
    "o": "rpl",
    "p": "/linkType?cellId=18",
    "v": 1
  }
]

```

The 68 bytes of the CBOR transcription of the JSON document describing the patch for install the schedule on node 6

B.5.6 PATCH request for install the schedule on node 7 with a shorter syntax

```

PATCH
coap://<ip>:5683/mg/6t/cellList
[
  {
    "o": "rpl",
    "p": "/nodeAddress?cellId=113",
    "v": 3
  },
  {
    "o": "rpl",
    "p": "/linkType?cellId=113",
    "v": 1
  }
]

```

The 70 bytes of the CBOR transcription of the JSON document describing the patch for install the schedule on node 7

B.5.7 PATCH request for install the schedule on node 8 with a shorter syntax

```

PATCH
coap://<ip>:5683/mg/6t/cellList
[
  {
    "o": "rpl",
    "p": "/nodeAddress?cellId=98",
    "v": 2
  },
  {
    "o": "rpl",
    "p": "/linkType?cellId=98",
    "v": 1
  }
]

```

The 68 bytes of the CBOR transcription of the JSON document describing the patch for install the schedule on node 8

B.5.8 PATCH request for install the schedule on node 9 with a shorter syntax

```

PATCH
coap://<ip>:5683/mg/6t/cellList
[
  {
    "o": "rpl",
    "p": "/nodeAddress?cellId=66",
    "v": 2
  },
]

```

```

    {
      "o": "rp1",
      "p": "/linkType?cellId=66",
      "v": 1
    }
  ]

```

The 68 bytes of the CBOR transcription of the JSON document describing the patch for install the schedule on node 9

B.5.9 PATCH request for install the schedule on node 10 with a shorter syntax

```

PATCH
coap://<ip>:5683/mg/6t/cellList
[
  {
    "o": "rp1",
    "p": "/nodeAddress?cellId=17",
    "v": 3
  },
  {
    "o": "rp1",
    "p": "/linkType?cellId=17",
    "v": 1
  },
  {
    "o": "rp1",
    "p": "/nodeAddress?cellId=81",
    "v": 3
  },
  {
    "o": "rp1",
    "p": "/LinkType?cellId=81",
    "v": 1
  }
]

```

The 135 bytes of the CBOR transcription of the JSON document describing the patch for install the schedule on node 10

B.5.10 PATCH request for install the schedule on node 11 with a shorter syntax

```

PATCH
coap://<ip>:5683/mg/6t/cellList
[
  {
    "o": "rp1",
    "p": "/nodeAddress?cellId=49",
    "v": 3
  },
  {
    "o": "rp1",
    "p": "/linkType?cellId=49",
    "v": 1
  }
]

```

The 68 bytes of the CBOR transcription of the JSON document describing the patch for install the schedule on node 11

B.5.11 PATCH request for install the schedule on node 12 with a shorter syntax

```

PATCH
coap://<ip>:5683/mg/6t/cellList
[
  {
    "o": "rp1",

```

```

        "p": "/nodeAddress?cellId=50",
        "v": 4
    },
    {
        "o": "rpl",
        "p": "/linkType?cellId=50",
        "v": 1
    }
]

```

The 68 bytes of the CBOR transcription of the JSON document describing the patch for install the schedule on node 12

B.6 PATCH requests installing the second schedule with a shorter syntax

B.6.1 PATCH request for update the schedule on node 2 with a shorter syntax

```

PATCH
coap://<ip>:5683/mg/6t/cellList
[
    {
        "o": "rpl",
        "p": "/nodeAddress?cellId=2",
        "v": 1
    },
    {
        "o": "rpl",
        "p": "/linkType?cellId=2",
        "v": 1
    },
    {
        "o": "rpl",
        "p": "/nodeAddress?cellId=34",
        "v": 9
    },
    {
        "o": "rpl",
        "p": "/LinkType?cellId=34",
        "v": 2
    },
    {
        "o": "rpl",
        "p": "/nodeAddress?cellId=66",
        "v": 8
    },
    {
        "o": "rpl",
        "p": "/nodeAddress?cellId=66",
        "v": 13
    }
]

```

The 203 bytes of the CBOR transcription of the JSON document describing the patch for update the schedule on node 2

B.6.2 PATCH request for update the schedule on node 3 with a shorter syntax

```

PATCH
coap://<ip>:5683/mg/6t/cellList
[
    {
        "o": "rpl",
        "p": "/cellId?cellId=81",
        "v": 82
    },
    {
        "o": "rpl",

```

```

    "p": "/cellId?cellId=112",
    "v": 113
  },
  {
    "o": "rpl",
    "p": "/cellId?cellId=128",
    "v": 129
  }
]

```

The 96 bytes of the CBOR transcription of the JSON document describing the patch for update the schedule on node 3

B.6.3 PATCH request for update the schedule on node 4 with a shorter syntax

```

PATCH
coap://<ip>:5683/mg/6t/cellList
[
  {
    "o": "rpl",
    "p": "/nodeAddress?cellId=18",
    "v": 5
  },
  {
    "o": "rpl",
    "p": "/nodeAddress?cellId=50",
    "v": 6
  },
  {
    "o": "rpl",
    "p": "/cellId?cellId=82",
    "v": 81
  },
  {
    "o": "rpl",
    "p": "/nodeAddress?cellId=81",
    "v": 12
  },
  {
    "o": "rpl",
    "p": "/nodeAddress?cellId=113",
    "v": 5
  },
  {
    "o": "rpl",
    "p": "/linkType?cellId=113",
    "v": 2
  },
  {
    "o": "rpl",
    "p": "/nodeAddress?cellId=128",
    "v": 1
  },
  {
    "o": "rpl",
    "p": "/linkType?cellId=128",
    "v": 1
  }
]

```

The 275 bytes of the CBOR transcription of the JSON document describing the patch for update the schedule on node 4

B.6.4 PATCH request for update the schedule on node 5 with a shorter syntax

```

PATCH
coap://<ip>:5683/mg/6t/cellList
[
  {

```



```

    "o": "rpl",
    "p": "/cellId?cellId=2",
    "v": 18
  },
  {
    "o": "rpl",
    "p": "/nodeAddress?cellId=18",
    "v": 4
  },
  {
    "o": "rpl",
    "p": "/cellId?cellId=82",
    "v": 113
  }
]

```

The 96 bytes of the CBOR transcription of the JSON document describing the patch for update the schedule on node 5

B.6.5 PATCH request for update the schedule on node 6 with a shorter syntax

```

PATCH
coap://<ip>:5683/mg/6t/cellList
[
  {
    "o": "rpl",
    "p": "/cellId?cellId=18",
    "v": 50
  }
]

```

The 32 bytes of the CBOR transcription of the JSON document describing the patch for update the schedule on node 6

B.6.6 PATCH request for update the schedule on node 7 with a shorter syntax

```

PATCH
coap://<ip>:5683/mg/6t/cellList
[
  {
    "o": "rpl",
    "p": "/cellId?cellId=113",
    "v": 114
  }
]

```

The 33 bytes of the CBOR transcription of the JSON document describing the patch for update the schedule on node 7

B.6.7 PATCH request for update the schedule on node 8 with a shorter syntax

```

PATCH
coap://<ip>:5683/mg/6t/cellList
[
  {
    "o": "rpl",
    "p": "/cellId?cellId=98",
    "v": 66
  }
]

```

The 32 bytes of the CBOR transcription of the JSON document describing the patch for update the schedule on node 8

B.6.8 PATCH request for update the schedule on node 9 with a shorter syntax

```

PATCH
coap://<ip>:5683/mg/6t/cellList
[
  {
    "o": "rp1",
    "p": "/cellId?cellId=66",
    "v": 34
  }
]

```

The 32 bytes of the CBOR transcription of the JSON document describing the patch for update the schedule on node 9

B.6.9 PATCH request for update the schedule on node 10 with a shorter syntax

```

PATCH
coap://<ip>:5683/mg/6t/cellList
[
  {
    "o": "rp1",
    "p": "/cellId?cellId=81",
    "v": 82
  }
]

```

The 32 bytes of the CBOR transcription of the JSON document describing the patch for update the schedule on node 10

B.6.10 PATCH request for update the schedule on node 11 with a shorter syntax

The 0 bytes of the CBOR transcription of the JSON document describing the patch for update the schedule on node 11

B.6.11 PATCH request for update the schedule on node 12 with a shorter syntax

```

PATCH
coap://<ip>:5683/mg/6t/cellList
[
  {
    "o": "rp1",
    "p": "/cellId?cellId=50",
    "v": 81
  }
]

```

The 32 bytes of the CBOR transcription of the JSON document describing the patch for update the schedule on node 12

B.6.12 PATCH request for update the schedule on node 13 with a shorter syntax

```

PATCH
coap://<ip>:5683/mg/6t/cellList
[
  {
    "o": "rp1",
    "p": "/nodeAddress?cellId=98",
    "v": 2
  },
  {
    "o": "rp1",
    "p": "/linkType?cellId=98",
    "v": 1
  }
]

```

The 68 bytes of the CBOR transcription of the JSON document describing the patch for update the schedule on node 13



**RESEARCH CENTRE
PARIS – ROCQUENCOURT**

Domaine de Voluceau, - Rocquencourt
B.P. 105 - 78153 Le Chesnay Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399