6TiSCH
Internet-Draft
Intended status: Informational
Expires: January 5, 2016

D. Dujovne, Ed.
Universidad Diego Portales
LA. Grieco
Politecnico di Bari
MR. Palattella
University of Luxembourg
N. Accettura
University of California Berkeley
July 4, 2015

6TiSCH On-the-Fly Scheduling
draft-dujovne-6tisch-on-the-fly-06

Abstract

This document describes the environment, problem statement, and goals of On-The-Fly (OTF) scheduling, a Layer-3 mechanism for 6TiSCH networks. The purpose of OTF is to dynamically adapt the aggregate bandwidth, i.e., the number of reserved soft cells between neighbor nodes, based on the specific application constraints to be satisfied. When using OTF, softcell reservation is distributed: through the 6top interface, neighbor nodes negotiate the cell(s) to be (re)allocated/deleted, with no intervention needed of a centralized entity. This document aims at defining a module which uses the functionalities provided by the 6top sublayer to (i) extract statistics and (ii) determine when to reserve/delete soft cells in the schedule. The exact reservation and deletion algorithm, and the number and type of statistics to be used in the algorithm are out of scope. OTF deals only with the number of softcells to be reserved/deleted; it is up to 6top to select the specific soft cells within the TSCH schedule.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The IEEE802.15.4e standard [IEEE802154e] was published in 2012 as an amendment to the Medium Access Control (MAC) protocol defined by the IEEE802.15.4-2011 [IEEE802154] standard. The Timeslotted Channel Hopping (TSCH) mode of IEEE802.15.4e is the object of this document.

On-The-Fly (OTF) scheduling is a 1-hop protocol with which a node negotiates the number of soft cells scheduled with its neighbors, without requiring any intervention of a centralized entity (e.g., a PCE). This document describes the OTF allocation policies and methods used by two neighbors to allocate one or more softcells in a distribution fashion. It also proposes an algorithm for estimating the required bandwidth (BW). This document defines the interface between OTF and the 6top sublayer ([I-D.wang-6tisch-6top]), to collect and retrieve statistics, or allocate/delete soft cells. It also defines two threshold values for bounding the number of triggered 6top allocate/delete commands. This document defines a framework; the algorithm and statistics used are out of scope. This draft follows the terminology defined in [I-D.ietf-6tisch-terminology] and addresses the open issue related to the scheduling mechanisms raised in [RFC7554].

2. Allocation policy

OTF is a distributed scheduling protocol which increases/decreases the bandwidth between two neighbor nodes (i.e., adding/deleting soft cells) by interacting with the 6top sublayer. It retrieves statistics from 6top, and uses that information to trigger 6top to add/delete softcells to a particular neighbor. The algorithm which decides how many softcells to add/delete is out of scope. For example, OTF might decide to add a cell if some queue of outbound frames is overflowing. Similarly, OTF can delete cells when the queue has been empty for some time. OTF only triggers 6top to add/delete the soft cells, it is the responsibility of the 6top sublayer to determine the exact slotOffset/channelOffset of those cells. In this document, the term "cell" and "soft cell" are used interchangeably.

OTF is a Layer-3 Mechanism, and as such, it operates on L3 links, on the best effort track, i.e. with TrackID=00, as defined in [I-D.wang-6tisch-6top]. Inside an intermediate node, a track is uniquely associated with only one bundle: the outgoing bundle. For an IP link, the bundle is identified by the peer mac address. For instance (macA, macB, TrackID=00) will be the bundle associated to the L3 link between node A and node B. The cells on the best effort track can be used for forwarding any packet in the queue, regardless of the specific L2 bundle (and thus, end-to-end L2 track) the packet belongs to. OTF manages the global bandwidth requirements between two neighbor nodes; per-track management is currently out of scope.

OTF is prone to schedule collisions. Nodes might not be aware of the cells allocated by other pairs of nodes. A schedule collision occurs when the same cell is allocated by different pairs in the same interference space. The probability of having allocation collision may be kept low by grouping cells into chunks (see [I-D.ietf-6tisch-terminology] and [I-D.ietf-6tisch-architecture] for more details). The use of chunks is outside the scope of this current version of the OTF draft.

The "allocation policy" is the algorithm used by OTF to decide when to increase/decrease the bandwidth allocated between two neighbor nodes in order to satisfy the traffic requirements. These requirements can be expressed in terms of throughput, latency or other constraints.

This document introduces the following parameters for describing the behavior of the OTF allocation policy:

SCHEDULEDCELLS: The amount of soft cells scheduled in a bundle on the best effort track between two neighbors.

REQUIREDCELLS: Number of cells requested by OTF to 6top, a non-negative value. How this is computed is out of the scope. It MAY be an instantaneous request, or a value averaged on several measurements.

OTFTHRESHLOW: Threshold parameter introducing cell over-provisioning in the allocation policy. It is a non-negative value expressed as number of cells. Which value to use is application-specific and out of OTF scope.

OTFTHRESHHIGH: Threshold parameter introducing cell under-provisioning in the allocation policy. It is a non-negative value expressed as number of cells. Which value to use is application-specific and out of OTF scope.

The OTF allocation policy compares the number of required cells against the number of scheduled ones, using the OTF threshold for bounding the signaling overhead due to negotiations of new cells. In details:

```
                SCHEDULEDCELLS
    <------------------------------------->
   +---+---+---+---+---+---+---+---+---+---+---+---+---+---+
   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
   +---+---+---+---+---+---+---+---+---+---+---+---+---+---+
                       |<----------------->|<------------->|
                       |   OTFTHRESHLOW    | OTFTHRESHHIGH |
                       |                   |               |
   REQUIREDCELLS       |                   |               |
   +---+---+---+       |                   |               |     ADD
   |   |   |   |       |                   |               |     SOME
   +---+---+---+       |                   |               |     CELLS
                       |                   |               |
           REQUIREDCELLS                   |               |
   +---+---+---+---+---+---+---+           |               |     DO
   |   |   |   |   |   |   |   |           |               |     NOTHING
   +---+---+---+---+---+---+---+           |               |
                       |                   |               |
                   REQUIREDCELLS           |               |
   +---+---+---+---+---+---+---+---+---+---+---+           |     DO
   |   |   |   |   |   |   |   |   |   |   |   |           |     NOTHING
   +---+---+---+---+---+---+---+---+---+---+---+           |
                       |                   |               |
                       |   REQUIREDCELLS   |               |
   +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+ REMOVE
   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   | SOME
   +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+ CELLS
```

Figure 1: Relation among the OTF parameters used for triggering 6top add/remove soft cell commands

1. If REQUIREDCELLS is greater than (SCHEDULEDCELLS + OTFTHRESHHIGH), OTF asks 6top to add one or more soft cells to the bundle on the best effort track.

2. If REQUIREDCELLS is greater or equal than (SCHEDULEDCELLS - OTFTHRESHLOW), and it is lower than or equal to (SCHEDULEDCELLS + OTFTHRESHHIGH), OTF does not perform any bundle resizing, since the scheduled cells are sufficient for managing the current traffic conditions.

3. If REQUIREDCELLS is lower than (SCHEDULEDCELLS - OTFTHRESHLOW), OTF asks 6top to delete one or more soft cells from the bundle on the best-effort track.

When both OTFTHRESHLOW and OTFTHRESHHIGH are equal to 0, any discrepancy between REQUIREDCELLS and SCHEDULEDCELLS triggers a 6top negotiation of soft cells. Other values for the thresholds, different from 0, reduce the number of triggered 6top negotiations.

The number of soft cells to be scheduled/deleted for bundle resizing is out of the scope of this document and implementation-dependant.

3. Allocation method

Beyond the allocation policies that describe the approach used by OTF for fulfilling the node bandwidth requests, the OTF framework also includes the Allocation Method that specify how OTF issues commands to the 6top sublayer. As specified in [I-D.wang-6tisch-6top], 6top provides a set of commands that allows OTF to allocate/delete soft cells. Such commands are used by the OTF soft cell allocation method.

With the soft cell allocation method, OTF can ask 6top to reserve one (or N > 1) soft cell(s) on the best effort L3 boundle, between two neighbor nodes. The 6top layer allocates and maintains these cells. If a L3 bundle with TrackID=00 was already reserved between the same pair of neighbors, 6top translates the OTF request into a bundle resize request. The newly allocated cell increases the size of the already existing bundle. Similarly, when OTF realizes there is a reduction of traffic exchanged between the two neighbors, it may asks 6top to delete a softcell (or N > 1) from the best effort track, i.e. to decrease the size of the best effort L3 bundle. If no bundle with TrackID=00 exists when 6top receives the OTF request, then the 6top softcell create command generates a new bundle of size 1.

4. Cell and Bundle Reservation/Deletion

In order to reserve/delete softcells, OTF interacts with 6top sublayer. To this aim OTF uses the following set of commands offered by 6top: CREATE.softcell, and DELETE.softcell. When creating (deleting) a softcell, OTF specifies the track the cell belongs to (i.e., best effort track, TrackID=00), but not its slotOffset nor the channelOffset. If at least one cell on the best effort L3 bundle already exists, the CREATE.softcell and DELETE.softcell, translate into INCREASE and DECREASE the bundle size, respectively. 6top is responsible for picking the specific cell to be added/deleted within the bundle. Before being able to do so, source and destination nodes go through a cell negotiation process. This process is out of scope of 6top and OTF. By using the CREATE.softcell command, OTF can ask 6top to add multiple softcells on the best effort L3 bundle. Following OTF request, 6top either (i) creates a new bundle, if no cells were reserved already on the best effort track, or (ii) increases the L3 bundle size of the already existing best-effort bundle. By using the DELETE.softcell command, OTF can ask 6top to delete cells from the best effort bundle.

OTF provides a policy for 6top to generate CREATE/DELETE.softcells commands, policy that is out of 6top scope [I-D.wang-6tisch-6top]. Such policy is not the only one that can be used by 6top. Others may be defined in the future.

5. Getting statistics and other information about cells through 6top

Statistics are kept in 4 data structures of 6top MIB: CellList, MonitoringStatusList, NeighborList, and QueueList.

CellList provides per-cell statistics. From this list, an upper layer can get per-bundle statistics. OTF may have access to the CellList, by using the CoAP-YANG Model, but actually cell-specific statistics are not significant to OTF, since softcells can be re-allocated in time by 6top itself, based on network conditions.

MonitoringStatusList provides per-neighbor and slotframe statistics. From it an upper layer (e.g., OTF) can get per bundle overview of scheduling and its performance. Such list contains information about the number of hard and soft cells reserved to a given node with a specific neighbor, and the QoS (that can be expressed in form of different metrics: PDR, ETX, RSSI, LQI) on the actual bandwidth, and the over-provisioned bandwidth (which includes the over-provisioned cells). 6top can use such list to operate 6top Monitoring Functions, such as re-allocating cells (by changing their slotOffset and/or channelOffset) when it finds out that the link quality of some softcell is much lower than average. Unlike 6top, OTF does not operate any re-allocation of cells. In fact, OTF can ask for more/ less bandwidth, but cannot move any cell within the schedule. Thus, the 6top Monitoring function is useful to OTF, because it can provide better cells for a given bandwidth requirement, specified by OTF. For instance, OTF may require some additional bandwidth (e.g. 2 cells in a specific slotframe) with PDR = 75%; then, 6top will reserve 3 slots in the slotframe to meet the bandwidth requirement. In addition, when the link quality drops to 50%, 6top will reserve 4 slots to keep meeting the bandwidth requirement. Given that OTF operates on the global bandwidth between two neighbor nodes, it does not need to be informed from 6top about cells’ re-allocation.

NeighborList provides per-neighbor statistics. From it, an upper layer can understand the connectivity of a pair of nodes, e.g. based on the queue length increase, OTF may ask 6top to add some cells, in order to increase the available bandwidth.

QueueList provides per-Queue statistics. From it, an upper layer can know the traffic load. OTF, based on such queue statistics (e.g., average length of the queue, average age of the packet in queue, etc.) may trigger a 6top CREATE.softcell (DELETE.softcell) command for increasing (decreasing) the bandwidth and be able to better serve the packets in the queue.

6. Events triggering algorithms in OTF

The Algorithms running within OTF MUST be event-oriented. As a consequence, OTF requires to connect the algorithms with external events to trigger their execution. The algorithm also generates one or more events when it is executed, such as a new soft cell allocation. Both type of events, the one which triggers the algorithm and the ones which are generated by the execution of the algorithm are called OTF events. We define the following elements:

A set of parameters P(E): parameters used to define E and its triggering conditions;

a set of triggering variables V(E): variables that can trigger the event;

a set of triggering conditions C(E): conditions to satisfy on the variables V(E) to trigger E;

a set of process handlers H(E): handlers required to respond and process the triggering conditions C(E).

To illustrate how P(E), V(E), C(E) and H(E) can be used to define a real event, the allocation policy described in Sec. 2 is considered hereby.

P(E) consists of the OTFTHRESHLOW and OTFTHRESHHIGH parameters (P1 and P2, respectively);

V(E) consists of the REQUIREDCELLS and SCHEDULEDCELLS parameters (V1 and V2, respectively);

C(E) consists of the following conditions:

C1: V1 > V2+P2

C2: V1 <=V2-P1

H(E) consists of the following handlers (one handler for each triggering condition)

H1(C1): OTF asks 6top to add one or more soft cells to the L3 best effort bundle.

H2(C2): OTF asks 6top to delete one or more soft cells from the L3 best effort bundle.

7. Bandwidth Estimation Algorithms

OTF supports different bandwidth estimation algorithms that can be used by a node in a 6TiSCH network for checking the statistics provided by 6top and the actual bandwidth usage. By doing so, one can adapt (increase or decrease) the number of scheduled soft cells for a given pair of neighbors (e.g., parent node and its child), according to their specific requirements. OTF supports several bandwidth estimation algorithms numbered 0 to 255 in the OTF implementation. The first algorithm (0) is reserved to the default algorithm that is described below. By using SET and GET commands, one can set the specific algorithm to be used, and get information about which algorithm is implemented.

The steps of the default bandwidth estimation algorithm, running over a parent node, are listed hereafter:

Step 1: Collect the bandwidth requests from child nodes (incoming bundle soft cell allocation from 6top-to-6top negotiation).

Step 2: Collect the node bandwidth requirement from the application (self/local traffic, from the application soft cell pending requests).

Step 3: Collect the current outgoing scheduled bandwidth (outgoing traffic).

Step 4: If (outgoing < incoming + self) then SCHEDULE soft cells to satisfy bandwidth requirements.

Step 5: If (outgoing > incoming + self) then DELETE the soft cells that are not used.

Step 6: Return to step 1.

The default bandwidth estimation algorithm introduced in this document adopts a reactive allocation policy, i.e., it uses

OTFTHRESHLOW = 0 and OTFTHRESHHIGH = 0; the histeresys is not enabled and the allocation/deallocation follows directly. The algorithm is triggered either by Step 4 or Step 5.

8. OTF external CoAP interface

In order to select the current OTF algorithm and provide functional parameters from outside OTF, this module uses CoAP with YANG as the data model. The algorithm number and the parameters MUST be invoked in different CoAP calls.

The path to select the algorithm is '6t/e/otf/alg' with A as the algorithm number.

```
         +------------------------------------------+
 Header  | POST                                     |
         +------------------------------------------+
 Uri-Path| /6t/e/otf/alg                            |
         +------------------------------------------+
 Options | CBOR( {AlgNo: 123} )                     |
         +------------------------------------------+
```

Figure 2: Algorithm number POST message

To obtain the current algorithm number:

```
         +------------------------------------------+
 Header  | GET                                      |
         +------------------------------------------+
 Uri-Path| /6t/e/otf/alg                            |
         +------------------------------------------+
 Options | Accept: application/cbor                 |
         +------------------------------------------+
```

Figure 3: Algorithm number GET message

An example is: 'coap://[aaaa::1]/6t/e/otf/alg'

The current algorithm parameter path is '6t/e/otf/alg/par'.

```
          +-------------------------------------------+
 Header   | POST                                      |
          +-------------------------------------------+
 Uri-Path| /6t/e/otf/alg/par                          |
          +-------------------------------------------+
 Options | CBOR( {Par: 0x1234} )                      |
          +-------------------------------------------+
```

Figure 4: Algorithm number POST message

An example follows: 'coap://[aaaa::1]/6t/e/otf/alg/par'

9. Acknowledgments

Special thanks to Prof. Kris Pister for his valuable contribution in designing the default Bandwidth Estimation Algorithm, and to Prof. Qin Wang for her support in defining the interaction between OTF and 6top sublayer.

Thanks to the Fondecyt 1121475 Project, to INRIA Chile "Network Design" group and to the IoT6 European Project (STREP) of the 7th Framework Program (Grant 288445).

10. References

10.1. Informative References

[I-D.ietf-6tisch-terminology]
Palattella, M., Thubert, P., Watteyne, T., and Q. Wang, "Terminology in IPv6 over the TSCH mode of IEEE 802.15.4e", draft-ietf-6tisch-terminology-04 (work in progress), March 2015.

[I-D.ietf-6tisch-architecture]
Thubert, P., "An Architecture for IPv6 over the TSCH mode of IEEE 802.15.4", draft-ietf-6tisch-architecture-08 (work in progress), May 2015.

[RFC7554] Watteyne, T., Palattella, M., and L. Grieco, "Using IEEE 802.15.4e Time-Slotted Channel Hopping (TSCH) in the Internet of Things (IoT): Problem Statement", RFC 7554, May 2015.

[I-D.wang-6tisch-6top]
Wang, Q., Vilajosana, X., and T. Watteyne, "6TiSCH Operation Sublayer (6top)", draft-wang-6tisch-6top-00 (work in progress), October 2013.

10.2. External Informative References

[IEEE802154e]
IEEE standard for Information Technology, "IEEE std. 802.15.4e, Part. 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) Amendament 1: MAC sublayer", April 2012.

[IEEE802154]
IEEE standard for Information Technology, "IEEE std. 802.15.4, Part. 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks", June 2011.

Authors' Addresses

Diego Dujovne (editor)
Universidad Diego Portales
Escuela de Informatica y Telecomunicaciones
Av. Ejercito 441
Santiago, Region Metropolitana
Chile

Phone: +56 (2) 676-8121
Email: diego.dujovne@mail.udp.cl

Luigi Alfredo Grieco
Politecnico di Bari
Department of Electrical and Information Engineering
Via Orabona 4
Bari 70125
Italy

Phone: 00390805963911
Email: a.grieco@poliba.it

Maria Rita Palattella
University of Luxembourg
Interdisciplinary Centre for Security, Reliability and Trust
4, rue Alphonse Weicker
Luxembourg L-2721
LUXEMBOURG

Phone: (+352) 46 66 44 5841
Email: maria-rita.palattella@uni.lu

Nicola Accettura
University of California Berkeley
Berkeley Sensor & Actuator Center
490 Cory Hall
Berkeley, California 94720
USA

Email: nicola.accettura@eecs.berkeley.edu

6TiSCH
Internet-Draft
Intended status: Informational
Expires: January 7, 2016

Q. Wang, Ed.
Univ. of Sci. and Tech. Beijing
X. Vilajosana
Universitat Oberta de Catalunya

July 6, 2015

6TiSCH Operation Sublayer (6top) Interface
draft-ietf-6tisch-6top-interface-04

Abstract

This document defines a generic data model for the 6TiSCH Operation Sublayer (6top), using the YANG data modeling language. This data model can be used for network management solutions defined by the 6TiSCH working group.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document defines a generic data model for the 6TiSCH Operation Sublayer (6top), using the YANG data modeling language defined in [RFC6020]. This data model can be used for future network management solutions defined by the 6TiSCH working group. This data model gives access to metrics (e.g. cell state), TSCH configuration and control procedures, and support for the different scheduling mechanisms described in [I-D.ietf-6tisch-architecture]. The 6top sublayer addresses the set of management information and functionalities described in [I-D.ietf-6tisch-tsch].

For example, network formation in a TSCH network is handled by the use of Enhanced Beacons (EB). EBs include information for joining nodes to be able to synchronize and set up an initial network topology. However, [IEEE802154e] does not specify how the period of EBs is configured, nor the rules for a node to select a particular node to join. 6top offers a set of commands so control mechanisms can be introduced on top of TSCH to configure nodes to join a specific node and obtain a unique 16-bit identifier from the network. Once a network is formed, 6top maintains the network's health, allowing for nodes to stay synchronized. It supplies mechanisms to manage each node's time source neighbor and configure the EB interval. Network layers running on top of 6top take advantage of the TSCH MAC layer information so routing metrics, topological information, energy consumption and latency requirements can be adjusted to TSCH, and adapted to application requirements.

TSCH requires a mechanism to manage its schedule; 6top provides a set of commands for upper layers to set up specific schedules, either explicitly by detailing specific cell information, or by allowing 6top to establish a schedule given a bandwidth or latency requirement. 6top is designed to enable decentralized, centralized or hybrid scheduling solutions. 6top enables internal TSCH queuing configuration, size of buffers, packet priorities, transmission failure behavior, and defines mechanisms to encrypt and authenticate MAC slotframes.

As described in [morell04label], due to the slotted nature of a TSCH network, it is possible to use a label switched architecture on top of TSCH cells. As a cell belongs to a specific track, a label header is not needed at each packet; the input cell (or bundle) and the output cell (or bundle) uniquely identify the data flow. The 6top sublayer provides operations to manage the cell mappings.

2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

3. 6TiSCH Operation Sublayer (6top) Overview

6top is a sublayer which is the next-higher layer for TSCH (Figure 1), as detailed in [I-D.ietf-6tisch-architecture]. 6top offers both management and data interfaces to an upper layer, and includes monitoring and statistics collection, both of which are configurable through its management interface. The detail of 6top-sublayer is described in [I-D.wang-6tisch-6top-sublayer]

Protocol Stack

```
+-----------------------------------+
| PCEP | CoAP |      | 6LoWPAN |    |
| PCC  | DTLS | PANA |    ND   |RPL |
+------------------------------------------+
| TCP  |     UDP     |     ICMP     | RSVP |
+------------------------------------------+
|                 IPv6                     |
+------------------------------------------+
|               6LoWPAN HC                 |
+------------------------------------------+
|                 6top                     |
+------------------------------------------+
|           IEEE802.15.4e TSCH             |
+------------------------------------------+
|              IEEE802.15.4                |
+------------------------------------------+
```

Figure 1

6top distinguishes between hard cells and soft cells. It therefore requires an extra flag to all cells in the TSCH schedule, as detailed in Section 3.1.

When a higher layer gives 6top a 6LoWPAN packet for transmission, 6top maps it to the appropriate outgoing priority-based queue, as detailed in Section 3.2.

Section 4 contains a generic data model for the 6top sublayer, described in the YANG data modeling language.

3.1. Cell Model

[IEEE802154e] defines a set of options attached to each cell. A cell can be a Transmit cell, a Receive cell, a Shared cell or a Timekeeping cell. These options are not exclusive, as a cell can be qualified with more than one of them. The MLME-SET-LINK.request command defined in [IEEE802154e] uses a linkOptions bitmap to specify the options of a cell. Acceptable values are:

b0 = Transmit

b1 = Receive

b2 = Shared

b3 = Timekeeping

b4-b7 = Reserved

Only Transmit cells can also be marked as Shared cells. When the shared bit is set, a back-off procedure is applied to handle collisions. Shared behavior does not apply to Receive cells.

6top allows an upper layer to schedule a cell at a specific slotOffset and channelOffset, in a specific slotframe.

In addition, 6top allows an upper layer to schedule a certain amount of bandwidth to a neighbor, without having to specify the exact slotOffset(s) and channelOffset(s). Once bandwidth is reserved, 6top is in charge of ensuring that this requirement is continuously satisfied. 6top dynamically reallocates cells if needed, and over-provisions if required.

6top allows an upper layer to associate a cell with a specific track by using a TrackID. A TrackID is a tuple (TrackOwnerAddr,InstanceID), where TrackOwnerAddr is the address of the node which initializes the process of creating the track, i.e., the owner of the track; and InstanceID is an instance identifier given by the owner of the track. InstanceID comes from upper layer; InstanceID could for example be the local instance ID defined in RPL.

If the TrackID is set to (0,0), the cell can be used by the best-effort QoS configuration or as a Shared cell. If the TrackID is not set to (0,0), i.e., the cell belongs to a specific track, the cell MUST not be set as Shared cell.

6top allows an upper layer to ask a node to manage a portion of a slotframe, which is named as chunk. Chunks can be delegated explicitly by the PCE to a node, or claimed automatically by any node that participates to the distributed cell scheduling process. The resource in a chunk can be appropriated by the node, i.e. the owner of the chunk.

Given this mechanism, 6top defines hard cells (which have been requested specifically) and soft cells (which can be reallocated dynamically). The hard/soft flag is introduced by the 6top sublayer named as CellType, 0: soft cell, 1: hard cell. This option is mandatory; all cells are either hard or soft.

3.1.1. hard cells

A hard cell is a cell that cannot be dynamically reallocated by 6top. The CellType MUST be set to 1. The cell is installed by 6top given specific slotframe ID, slotOffset, and channelOffset.

3.1.2. soft cells

A soft cell is a cell that can be reallocated by 6top dynamically. The CellType MUST be set to 0. This cell is installed by 6top given a specific bandwidth requirement. Soft cells are installed through the soft cell negotiation procedure described in [I-D.wang-6tisch-6top-sublayer].

3.2. Data Transfer Model

Once a TSCH schedule is established, 6top is responsible for feeding the data from the upper layer into TSCH. This section describes how 6top shapes data from the upper layer (e.g., RPL, 6LoWPAN), and feeds it to TSCH. Since 6top is a sublayer between TSCH and 6LoWPAN, the properties associated with a packet/fragment from the upper layer includes the next hop neighbor (DestAddr) and expected sending priority of the packet (Priority), and/or TrackID(s). The output to TSCH is the fragment corresponding to the next active cell in the TSCH schedule.

6top Data Transfer Model

```
                        |
                        | (DestAddr, Priority, Fragment)
                        |
  +----------------------------------------+
  |                  I-MUX                 |
  +----------------------------------------+
    |       |       |       |    ....   |
    |       |       |       |           |
  +---+   +---+   +---+   +---+       +---+
  |   |   |   |   |   |   |   |       |   |
  |Q1 |   |Q2 |   |Q3 |   |Q4 |       |Qn |
  |   |   |   |   |   |   |   |       |   |
  +---+   +---+   +---+   +---+       +---+
    |       |       |       |           |
    |       |       |       |           |
  +----------------------------------------+
  |                  MUX                   |
  +----------------------------------------+
                       |
                       |
                     +---+
                     |PDU|
                     +---+
                       |
                       | TSCH MAC-payload
                       |
```

Figure 2

In Figure 2, Qi represents a queue, which is either broadcast or unicast, and has an assigned priority. The number of queues is configurable. The relationship between queues and tracks is configurable. For example, for a given queue, only one specific track can be used, all of the tracks can be used, or a subset of the tracks can be used.

When 6top receives a packet to transmit, the I-MUX module selects a queue in which to insert it. If the packet's destination address is a unicast (resp. broadcast) address, it will be inserted into a unicast (resp. broadcast) queue.

The MUX module is invoked at each scheduled transmit cell by TSCH. When invoked, the MUX module goes through the queues, looking for the best matching frame to send. If it finds a frame, it hands it over to TSCH for transmission. If the next active cell is a broadcast cell, it selects a fragment only from broadcast queues.

How the MUX module selects the best frame is configurable. The following rules are a typical example:

The frame's layer 2 destination address MUST match the neighbor address associated with the transmit cell.

If the transmit cell is associated with a specific track, the frames in the queue corresponding to the TrackID have the highest priority.

If the transmit cell is not associated with a specific track, i.e., TrackID=(0,0), frames from a queue with a higher priority MUST be sent before frames from a queue with a lower priority.

Further rules can be configured to satisfy specific QoS requirements.

4. Generic Data Model

This section presents the generic data model of the 6top sublayer, using the YANG data modeling langage. This data model can be used for future network management solutions defined by the 6TiSCH working group. The data model consists of the MIB (management information base) defined in 6top, and part of the PIB (personal area network information base) defined in [IEEE802154e] and [IEEE802154].

4.1. YANG model of the 6top MIB

```
module ietf-6top {

  namespace
    "urn:ietf:params:xml:ns:yang:ietf-6top";
  prefix
    "ietf";


   contact
       "WG Web:   <http://tools.ietf.org/wg/6tisch/>
        WG List:  <mailto:6tisch@ietf.org>

        WG Chair: Pascal Thubert
                  <mailto:pthubert@cisco.com>

        WG Chair: Thomas Watteyne
                  <mailto:thomas.watteyne@inria.fr>

        Editor:   Qin Wang
                  <mailto:wangqin@ies.ustb.edu.cn>";
```

```
    description
      "Data model for the 6top sublayer";
    organization
         "IETF 6TiSCH Working Group";
    revision 2015-06-16 {
      description
        "v4 revision.";
      reference
        "draft-ietf-6tisch-6top-interface";
    }


    typedef nodeaddresstype {
      type uint64;
      description
        "The type to store a node's address. It can be a 64-bit EUI;
         or the short address defined by 6top, constrained by TSCH
         macNodeAddress size, 2-octets. If using TSCH as MAC, the
         higher 6 octets should be filled with 0, and lowest 2-octets
         is neighbor short address";
    }
    typedef asntype {
      type string {
        length "0..5";
      }
      description
        "The type to store ASN. String of 5 bytes";
    }

    list Version {
      key "major minor";
      description
        "Provides a unique identification for the set of resources
         defined in this draft. Provides a major and minor version
         number that may be accessible independently";

      leaf major {
        type uint8;
        description
          "major revision number";
      }
      leaf minor {
        type uint8;
        description
          "minor revision number";
      }
    }
```

```
    list SlotframeList {
      key "SlotframeID";
      min-elements 1;
      description
        "List of all of the slotframes used by the node.";

      leaf SlotframeID {
        type uint8;
        mandatory true;
        description
          "Equal to SlotframeHandle defined in TSCH";
        reference
          "IEEE802154e";
      }
      leaf NumOfSlots {
        mandatory true;
        type uint16 {
          range "1..max";
        }
        description
          "indicates how many timeslots in the slotframe";
      }
    }

    list CellList {
      key "CellID";
      min-elements 1;
      unique "SlotframeID SlotOffset ChannelOffset";
      description
        "List of scheduled cells of a node with all of its neighbors,
         in all of its slotframes.";



      leaf CellID {
        type uint16;
        description
          "Equal to Linkhandle in the linkTable of TSCH";
        reference
          "IEEE802154e";
      }
      leaf SlotframeID {
        type leafref {
          path "/SlotframeList/SlotframeID";
        }
        description
          "SlotframeID, one in SlotframeList, indicates the slotframe
           the cell belongs to.";
```

```
        reference
          "IEEE802154e";
      }
      leaf SlotOffset {
        type uint16;
        description
          "Defined in IEEE802154e.";
        reference
          "IEEE802154e";
      }
      leaf ChannelOffset {
        type uint16;
        description
          "Defined in IEEE802154e. Value range is 0..15";
        reference
          "IEEE802154e";
      }
      leaf LinkOption {
        type bits {
          bit Transmit {
            position 0;
          }
          bit Receive {
            position 1;
          }
          bit Share {
            position 2;
          }
          bit Timekeeping {
            position 3;
          }
        }
        description
          "Defined in IEEE802154e.";
        reference
          "IEEE802154e";
      }
      leaf LinkType {
        type enumeration {
          enum NORMAL;
          enum ADVERTISING;
        }
        description
          "Defined in IEEE802154";
        reference
          "IEEE802154";
      }
      leaf CellType {
```

```
        type enumeration {
          enum SOFT;
          enum HARD;
        }
        description
          "Defined in 6top";
      }
      leaf NodeAddress {
        type nodeaddresstype;
        description
          "specify the target node address.";
      }
      leaf TrackID {
        type leafref {
          path "/TrackList/TrackId";
        }
        description
          "A TrackID is one in the TrackList, pointing to a tuple
           (TrackOwnerAddr,InstanceID) , where TrackOwnerAddr is the
           address of the node which initializes the process of
           creating the track, i.e., the owner of the track; and
           InstanceID is an instance identifier given by the owner of
           the track.";
      }
      container Statistic {
        description
           "The Statistic container";
        leaf NumOfStatistic {
          mandatory true;
          type uint8;
          description
            "Number of statistics collected on the cell";
        }
        list MeasureList {
          key "StatisticsMetricsID";
          min-elements 1;
          description
             "The list of measures.";
          leaf StatisticsMetricsID{
            type leafref {
              path "/StatisticsMetricsList/StatisticsMetricsID";
            }
            description
              "An index of StatisticsMetricList, which defines how
               to collect data and get the statistic value";
          }
          leaf StatisticsValue{
            type uint16;
```

```
            config false;
            description
              "updated by 6top according to the statistics method
               specified by StatisticsMetricsID";
          }
        }
      }
    }

    list MonitoringStatusList {
      key "MonitoringStatusID";
      min-elements 1;
      unique "SlotframeID NodeAddress";
      description
        "List of the monitoring configuration and results per
         slotframe and neighbor. Basically, it is used for Monitoring
         Function of 6top to re-allocate softcells or initial the
         softcell negotiation process to increase/decrease number of
         softcells. Upper layer can use it also.";

      leaf MonitoringStatusID {
        type uint16;
        description
             "The monitoring status ID.";
      }
      leaf SlotframeID {
        type leafref {
          path "/SlotframeList/SlotframeID";
        }
        description
          "SlotframeID, one in SlotframeList, indicates the slotframe
           being monitored";
        reference
          "IEEE802154e";
      }
      leaf NodeAddress {
        type nodeaddresstype;
         description
             "The lead node address";
      }
      leaf EnforcePolicy {
        type enumeration {
          enum DISABLE;
          enum BESTEFFORT;
          enum STRICT;
          enum OVERPROVISION;
        }
        default DISABLE;
```

```
        description
          "Currently enforced QoS policy. DISABLE-no QoS;
           BESTEFFORT- best effort policy is used; STRICT- Strict
           Priority Queueing; OVERPROVISION- cell overprovision";
      }
      leaf AllocatedHard {
        type uint16;
        config false;
        description
          "Number of hard cells allocated";
      }
      leaf AllocatedSoft {
        type uint16;
        config false;
        description
          "Number of soft cells allocated";
      }
      leaf OverProvision {
        type uint16;
        config false;
        must "../EnforcePolicy <> DISABLE ./";
        description
          "Overprovisioned cells. 0 if EnforcePolicy is
           DISABLE";
      }
      leaf QoS {
        type uint16;
        config false;
        description
          "Current QoS including overprovisioned cells, i.e. the
           bandwidth obtained including the overprovisioned cells.";
      }
      leaf NQoS {
        type uint16;
        config false;
        description
          "Real QoS without over provisioned cells, i.e. the actual
           bandwidth without taking into account the overprovisioned
           cells.";
      }
    }

    list StatisticsMetricsList {
      key "StatisticsMetricsID";
      min-elements 1;
      unique "SlotframeID SlotOffset ChannelOffset NodeAddress";
      description
        "List of Statistics Metrics used in the node.";
```

```
      leaf StatisticsMetricsID {
        type uint16;
        description
             "The metrics ID for statistics.";
      }
      leaf SlotframeID {
        type leafref {
          path "/SlotframeList/SlotframeID";
        }
        description
          "SlotframeID, one in SlotframeList, specifies the
           slotframe to which the statistics metrics applies to.
           If empty, applies to all slotframes";
        reference
          "IEEE802154e";
      }
      leaf SlotOffset {
        type uint16;
        description
          "Specific slotOffset to which the statistics metrics applies
           to. If empty, applies to all timeslots";
        reference
          "IEEE802154e";
      }
      leaf ChannelOffset {
        type uint16;
        description
          "Specific channelOffset to which the statistics metrics
          applies to. If empty, applies to all channels";
        reference
          "IEEE802154e";
      }

      leaf NodeAddress {
        type nodeaddresstype;
        description
          "If NodeAddress is empty, applies to all neighbor nodes.";
      }

      leaf Metrics {
        type enumeration {
          enum macCounterOctets;
          enum macRetryCount;
          enum macMultipleRetryCount;
          enum macTXFailCount;
          enum macTXSuccessCount;
          enum macFCSErrorCount;
          enum macSecurityFailure;
```

```
          enum macDuplicateFrameCount;
          enum macRXSuccessCount;
          enum macNACKcount;
          enum PDR;
          enum ETX;
          enum RSSI;
          enum LQI;
        }
        description
          "The metric to be monitored. Include those provided by
           underlying IEEE 802.15.4e TSCH -- see table 4i (2012).
           PDR,ETX,RSSI,LQI are maintained by 6top. ";
      }
      leaf Window {
        type uint16;
        description
          "measurement period, in Number of the slotframe size";
      }
      leaf Enable {
        type enumeration {
          enum DISABLE;
          enum ENABLE;
        }
        default DISABLE;
        description
          "indicates the StatisticsMetric is active or not";
      }
    }

    list EBList {
      key "EbID";
      min-elements 1;
      description
        "List of information related with the EBs used by the node";

      leaf EbID {
        type uint8;
        description
          "The EB id.";

      }
      leaf CellID {
        type leafref {
          path "/CellList/CellID";
        }
        description
          "CellID, one in CellList, indicates the cell used
          to send EB";
```

```
      }
      leaf Peroid {
        type uint16;
        description
          "The EBs period, in seconds, indicates the interval between
           two EB sendings";
      }
      leaf Expiration {
        type enumeration {
          enum NEVERSTOP;
          enum EXPIRATION;
        }
        description
          "NEVERSTOP- the period of the EB never stops; EXPIRATION-
           when the Period arrives, the EB will stop.";
      }
      leaf Priority {
        type uint8;
        description
          "The joining priority model that will be used for
           advertisements. Joining priority MAY be for example
           SAME_AS_PARENT, RANDOM, BEST_PARENT+1 or
           DAGRANK(rank).";
      }
    }

    container TimeSource {
      description
        "specify the timesource selection policy and some relative
        statistics. ";
      leaf policy {
        type enumeration {
          enum ALLPARENT;
          enum BESTCONNECTED;
          enum LOWESTJOINPRIORITY;
        }
        default LOWESTJOINPRIORITY;
        description
          "indicates the policy to choose timesource.
           ALLPARENT- choose from all parents;
           BESTCONNECTED- choose the best-connected node;
           LOWESTJOINPRIORITY- choose the node with lowest
           priority in its EB.";
      }
      leaf NodeAddress {
        type nodeaddresstype;
        description
          "Specifies the address of selected time source neighbors.";
```

```
      }
      leaf MinTimeCorrection {
        type uint16;
        config false;
        description
          "measured in microsecond";
      }
      leaf MaxTimeCorrection {
        type uint16;
        config false;
        description
          "measured in microsecond";
      }
      leaf AveTimeCorrection {
        type uint16;
        config false;
        description
          "measured and computed in microsecond";
      }
    }

    list NeighborList {
      key "NodeAddress";
      description
        "statistics per communication link. ";
      leaf NodeAddress {
        type nodeaddresstype;
        description
          "Specifies the address of the neighbor.";
      }
      leaf RSSI {
        type uint8;
        config false;
        description
          "The received signal strength";
      }
      leaf LinkQuality {
        type uint8;
        config false;
        description
          "The LQI metric";
      }
      leaf ASN {
        type asntype;
        config false;
        description
          "The 5 ASN bytes, indicates the most recent
          timeslot when a packet from the neighbor was received";
```

```
      }
    }

    list QueueList {
      key "QueueId";
      min-elements 1;
      description
        "List of Queues, including configuration and statistics.";
      leaf QueueId {
        type uint8;
        description
          "Queue Identifier";
      }
      leaf TxqLength {
        type uint8;
        description
          "The TX queue length in number of packets";
      }
      leaf RxqLength {
        type uint8;
        description
          "The RX queue length in number of packets";
      }
      leaf NumrTx {
        type uint8;
        description
          "Number of allowed retransmissions.";
      }
      leaf Age {
        type uint16;
        description
          "In seconds. Discard packet according to its age
           on the queue. 0 if no discards are allowed.";
      }
      leaf RTXbackoff {
        type uint8;
        description
          "retransmission backoff in number of slotframes.
           0 if next available timeslot wants to be used.";
      }
      leaf StatsWindow {
        type uint16;
        description
          "In second, window of time used to compute stats.";
      }
      leaf QueuePriority {
        type uint8;
        description
```

```
           "The priority for this queue.";
       }
       list TrackIds {
         key "TrackID";
         leaf TrackID{
           type leafref {
             path "/TrackList/TrackId";
           }
           description
             "The TrackID, one in TrackList, indicates the Track is
              associated with the Queue.";
         }
         description
              "The track IDs.";
       }
       leaf MinLenTXQueue {
         type uint8;
         config false;
         description
           "Statistics, lowest TX queue length registered
            in the window.";
       }
       leaf MaxLenTXQueue {
         type uint8;
         config false;
         description
           "Statistics, largest TX queue length registered
            in the window.";
       }
       leaf AvgLenTXQueue {
         type uint8;
         config false;
         description
           "Statistics, avg TX queue length registered
           in the window.";
       }
       leaf MinLenRXQueue {
         type uint8;
         config false;
         description
           "Statistics, lowest RX queue length registered
            in the window.";
       }
       leaf MaxLenRXQueue {
         type uint8;
         config false;
         description
           "Statistics, largest RX queue len registered
```

```
          in the window.";
      }
      leaf AvgLenRXQueue {
        type uint8;
        config false;
        description
          "Statistics, avg RX queue length registered
           in the window.";
      }
      leaf MinRetransmissions {
        type uint8;
        config false;
        description
          "Statistics, lowest number of retransmissions registered
           in the window.";
      }
      leaf MaxRetransmissions {
        type uint8;
        config false;
        description
          "Statistics, largest number of retransmissions registered
           in the window.";
      }
      leaf AvgRetransmissions {
        type uint8;
        config false;
        description
          "Statistics, average number of retransmissions registered
           in the window.";
      }
      leaf MinPacketAge {
        type uint16;
        config false;
        description
          "Statistics, in seconds, minimum time a packet stayed in
           the queue during the observed window.";
      }
      leaf MaxPacketAge {
        type uint16;
        config false;
        description
          "Statistics, in seconds, maximum time a packet stayed
           in the queue during the observed window.";
      }
      leaf AvgPacketAge {
        type uint16;
        config false;
        description
```

```
          "Statistics, in seconds, average time a packet stayed in
           the queue during the observed window.";
      }
      leaf MinBackoff {
        type uint8;
        config false;
        description
          "Statistics, in number of slotframes, minimum Backoff
           for a packet in the queue during the observed window.";
      }
      leaf MaxBackoff {
        type uint8;
        config false;
        description
          "Statistics, in number of slotframes, maximum Backoff
           for a packet in the queue during the observed window.";
      }
      leaf AvgBackoff {
        type uint8;
        config false;
        description
          "Statistics, in number of slotframes, average Backoff
           for a packet in the queue during the observed window.";
      }
    }

    list LabelSwitchList {
      key "LabelSwitchID";
      description
        "List of Label switch’ configuration on the node";

      leaf LabelSwitchID {
        type uint16;
        description
           "The label switch ID.";

      }
      list InputCellIds {
        key "CellID";
        leaf CellID{
          type leafref {
            path "/CellList/CellID";
          }
          description
            "The CellID, indicates the Rx cell on which the packet
             will come in.";
        }
        description
```

```
                "The input cell IDs.";
        }
        list OutputCellIds {
          key "CellID";
          leaf CellID{
            type leafref {
              path "/CellList/CellID";
            }
            description
              "The CellID, indicates the Tx cell on which the received
               packet should be sent out.";
          }
          description
               "The output cell IDs.";

        }
        leaf LoadBalancingPolicy {
          type enumeration {
            enum ROUNDROBIN;
            enum OTHER;
          }
          description
            "The load-balancing policy.
             ROUNDROBIN- Round Robin algorithm is used for
             forwarding scheduling.";
        }
      }

      list TrackList {
        key "TrackId";
        min-elements 1;
        unique "TrackOwnerAddr InstanceID";
        description
          "List of the tracks through the node. At lease the best effort
          track is existing";

        leaf TrackId {
          type uint16;
          description
            "Track Identifier, named locally. It is used to refer to the
             tuple (TrackOwnerAddr, InstanceID).";
        }
        leaf TrackOwnerAddr {
          type uint64;
          description
            "The address of the node which initializes the process of
             creating the track, i.e., the owner of the track;";
        }
```

```
      leaf InstanceID {
        type uint16;
        description
          "InstanceID is an instance identifier given by
           the owner of the track. InstanceID comes from
           upper layer; InstanceID could for example be
           the local instance ID defined in RPL.";
      }
    }

    list ChunkList {
      key "ChunkId";
      description
        "List of the chunks assigned to the node.";

      leaf ChunkId{
        type uint16;
        description
          "The identifier of a chunk";
      }
      leaf SlotframeId{
        type leafref {
          path "/SlotframeList/SlotframeID";
        }
        description
          "SlotframeID, one in SlotframeList, indicates the
           slotframe to which the chunk belongs";
      }
      leaf SlotBase {
        type uint16;
        description
          "the base slotOffset of the chunk in the slotframe";
      }
      leaf SlotStep {
        type uint8;
        description
          "the slot incremental of the chunk";
      }
      leaf ChannelBase {
        type uint16;
        description
          "the base channelOffset of the chunk";
      }
      leaf ChannelStep {
        type uint8;
        description
          "the channel incremental of the chunk";
      }
```

```
      leaf ChunkSize {
        type uint8;
        description
          "the number of cells in the chunk. The chunk is the set
           of (slotOffset(i), channelOffset(i)),
           i=0..Chunksize-1,
           slotOffset(i)= (slotBase + i * slotStep) % slotframeLen,
           channelOffset(i) = (channelBase + i * channelStep) % 16";
      }
    }

    list ChunkCellList {
      key "SlotOffset ChannelOffset";
      description
        "List of all of the cells assigned to the node via the
         assignment of chunks.";

      leaf SlotOffset{
        type uint16;
        description
          "The slotoffset of a cell which belongs to a Chunk";
      }
      leaf ChannelOffset{
        type uint16;
        description
          "The channeloffset of a cell which belongs to a chunk.";
      }
      leaf ChunkId {
        type leafref{
          path "/ChunkList/ChunkId";
        }
        description
          "Identifier of the chunk the cell belongs to";
      }
      leaf CellID{
        type leafref {
          path "/CellList/CellID";
        }
        description
          "Initial value of CellID is 0xFFFF. When the cell is
           scheduled, the value of CellID is same as that in
           CellList";
      }
      leaf ChunkCellStatus {
        type enumeration {
          enum UNSCHEDULED;
          enum SCHEDULED;
        }
```

```
        description
          "The Cell status in a Chunk.";

      }
    }

    container TSCHSpecificPIBAttributes {
      description
        "TSCH specific MAC PIB attributes.";
      reference
        "table 52b in IEEE802.15.4e-2012.";

      leaf macMinBE {
        type uint8;
        description
          "defined in Table 52b of IEEE802.15.4e-2012,
           The minimum value of the backoff exponent (BE) in the
           CSMA-CA algorithm or the TSCH-CA algorithm. default:
           3-CSMA-CA, 1-TSCH-CA";
      }
      leaf macMaxBE {
        type uint8;
        description
          "defined in Table 52b of IEEE802.15.4e-2012,
           The maximum value of the backoff exponent (BE) in the
           CSMA-CA algorithm or the TSCH-CA algorithm. default:
           5-CSMA-CA, 7-TSCH-CA";
      }
      leaf macDisconnectTime {
        type uint16;
        description
          "defined in Table 52b of IEEE802.15.4e-2012,
           Time (in Timeslots) to send out Disassociate frames
           before disconnecting, default: 0x00ff";
      }
      leaf macJoinPriority {
        type uint8;
        description
          "defined in Table 52b of IEEE802.15.4e-2012,
           The lowest join priority from the TSCH Synchronization
           IE in an Enhanced beacon, default: 1";
      }
      leaf macASN {
        type asntype;
        description
          "defined in Table 52b of IEEE802.15.4e-2012,
           The Absolute Slot Number, i.e., the number of slots
           that ha elapsed since the start of the network.";
```

```
      }
      leaf macNoHLBuffers {
        type enumeration {
          enum TRUE;
          enum FALSE;
        }
        description
          "defined in Table 52b of IEEE802.15.4e-2012,
           If the value is TRUE, the higher layer receiving the
           frame payload cannot buffer it, and the device should
           acknowledge frames with a NACK; If FALSE, the higher
           layer can accept the frame payload. default: FALSE";
      }
    }

    list TSCHmacTimeslotTemplate {
      key "macTimeslotTemplateId";
      min-elements 1;
      description
        "List of all timeslot templates used in the node.";
      reference
        "table 52e in IEEE802.15.4e-2012.";

      leaf macTimeslotTemplateId {
        type uint8;
        description
          "defined in Table 52e of IEEE802.15.4e-2012.
           Identifier of Timeslot Template. default: 0";
      }
      leaf macTsCCAOffset {
        type uint16;
        description
          "The time between the beginning of timeslot and start
           of CCA operation, in microsecond. default: 1800";
      }
      leaf macTsCCA {
        type uint16;
        description
          "Duration of CCA, in microsecond. default: 128";
      }
      leaf macTsTxOffset {
        type uint16;
        description
          "The time between the beginning of the timeslot and
           the start of frame transmission, in microsecond.
           default: 2120";
      }
      leaf macTsRxOffset {
```

```
        type uint16;
        description
          "Beginning of the timeslot to when the receiver shall
           be listening, in microsecond. default: 1120";
      }
      leaf macTsRxAckDelay {
        type uint16;
        description
          "End of frame to when the transmitter shall listen for
           Acknowledgment, in microsecond. default: 800";
      }
      leaf macTsTxAckDelay {
        type uint16;
        description
          "End of frame to start of Acknowledgment, in
           microsecond.
           default: 1000";
      }
      leaf macTsRxWait {
        type uint16;
        description
          "The time to wait for start of frame, in microsecond.
           default: 2200";
      }
      leaf macTsAckWait {
        type uint16;
        description
          "The minimum time to wait for start of an
           Acknowledgment, in microsecond. default: 400";
      }
      leaf macTsRxTx {
        type uint16;
        description
          "Transmit to Receive turnaround, in microsecond.
           default: 192";
      }
      leaf macTsMaxAck {
        type uint16;
        description
          "Transmission time to send Acknowledgment,in
           microsecond. default: 2400";
      }
      leaf macTsMaxTx {
        type uint16;
        description
          "Transmission time to send the maximum length frame,
           in microsecond. default: 4256";
      }
```

```
      leaf macTsTimeslotLength {
        type uint16;
        description
          "The total length of the timeslot including any unused
           time after frame transmission and Acknowledgment,
           in microsecond. default: 10000";
      }
    }

    list TSCHHoppingSequence {
      key "macHoppingSequenceID";
      min-elements 1;
      description
        "List of all channel hopping sequences used in the
         nodes";
      reference
        "Table 52f of IEEE802.15.4e-2012";

      leaf macHoppingSequenceID {
        type uint8;
        description
          "defined in Table 52f of IEEE802.15.4e-2012.
           Each hopping sequence has a unique ID. default: 0";
      }
      leaf macChannelPage {
        type uint8;
        description
          "Corresponds to the 5 MSBs (b27, ..., b31) of a row
           in phyChannelsSupported. Note this may not correspond
           to the current channelPage in use.";
      }
      leaf macNumberOfChannels {
        type uint16;
        description
          "Number of channels supported by the PHY on this
           channelPage.";
      }
      leaf macPhyConfiguration {
        type uint32;
        description
          "For channel pages 0 to 6, the 27 LSBs(b0, b1, ...,
           b26) indicate the status (1 = to be used, 0 = not to
           be used) for each of the up to 27 valid channels
           available to the PHY. For pages 7 and 8, the 27 LSBs
           indicate the configuration of the PHY, and the channel
           list is contained in the extendedBitmap.";
      }
      leaf macExtendedBitmap {
```

```
        type uint64;
        description
          "For pages 7 and 8, a bitmap of numberOfChannels bits,
           where bk shall indicate the status of channel k for
           each of the up to numberOfChannels valid channels
           supported by that channel page and phyConfiguration.
           Otherwise field is empty.";
      }
      leaf macHoppingSequenceLength {
        type uint16;
        description
          "The number of channels in the Hopping Sequence.
           Does not necessarily equal numberOfChannels.";
      }
      list macHoppingSequenceList {
        key "HoppingChannelID";
        leaf HoppingChannelID {
          type uint16;
          description
            "channels to be hopped over";
        }
        description
         "The hopping sequence";

      }
      leaf macCurrentHop {
        type uint16;
        config false;
        description
          "Index of the current position in the hopping sequence
           list.";
      }
    }

    container SecurityAttributes{
      description
          "The Security Attributes Container.";

      leaf-list K1{
        type uint8;
        config true;
        min-elements 16;
        description
           "The key is used to authenticate EBs.
           The default value of the key is
           36 54 69 53 43 48 20 6D 69 6E 69 6D 61 6C 31 35
           ,i.e. 6TiSCH minimal15.";
        }
```

```
      leaf EBSecurityLevel{
          type enumeration {
             enum NONE;
             enum MIC-32;
          }
          description
             "The security level respective to the EB key.";
        }

      list K2List{
        key "NodeAddress";
        description
          "The keys are shared with neighbors, used to authenticate
          and/or encrypt DATA, ACKNOWLEDGEMENT, MAC COMMAND frame.";

        leaf NodeAddress {
           type nodeaddresstype;
           description
              "Specifies the address of the neighbor(s) sharing the key.
               The address could be broadcast/unicast address.";
        }
        leaf-list K2{
           type uint8;
           min-elements 16;
           config true;
           description
             "The key is used as shared key with
             the specific neighbor";
        }
        leaf SecurityLevel{
           type enumeration {
              enum NONE;
              enum MIC-32;
              enum MIC-64;
              enum MIC-128;
              enum ENC;
              enum ENC-MIC-32;
              enum ENC-MIC-64;
              enum ENC-MIC-128;
           }
           description
              "The security level respective to
              the specific shared key.";
        }
      }
    }
  }
```

4.2. Yang Model for the Security aspects of 6top

The [I-D.ietf-6tisch-architecture] and [I-D.richardson-6tisch--security-6top] define the attributes needed to secure network bootstraping and joining and authentication processes. The SecurityAttributes container in the included yang model above contains attributes that are exposed by 6top interface to enable access and configuration to the security mechanisms carried out by 6top management entity.

5. References

5.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

5.2. Informative References

[RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

[I-D.ietf-6tisch-tsch] Watteyne, T., Palattella, M., and L. Grieco, "Using IEEE802.15.4e TSCH in an IoT context: Overview, Problem Statement and Goals", draft-ietf-6tisch-tsch-06 (work in progress), March 2015.

[I-D.ietf-6tisch-architecture] Thubert, P., "An Architecture for IPv6 over the TSCH mode of IEEE 802.15.4", draft-ietf-6tisch-architecture-08 (work in progress), May 2015.

[I-D.ietf-6tisch-terminology] Palattella, M., Thubert, P., Watteyne, T., and Q. Wang, "Terminology in IPv6 over the TSCH mode of IEEE 802.15.4e", draft-ietf-6tisch-terminology-04 (work in progress), March 2015.

[I-D.ietf-6tisch-minimal] Vilajosana, X. and K. Pister, "Minimal 6TiSCH Configuration", draft-ietf-6tisch-minimal-09 (work in progress), June 2015.

[I-D.wang-6tisch-6top-sublayer]
Wang, Q., Vilajosana, X., and T. Watteyne, "6TiSCH Operation Sublayer (6top)", draft-wang-6tisch-6top-sublayer-01 (work in progress), July 2014.

[I-D.ietf-6tisch-coap]
Sudhaakar, R. and P. Zand, "6TiSCH Resource Management and Interaction using CoAP", draft-ietf-6tisch-coap-03 (work in progress), March 2015.

[I-D.richardson-6tisch--security-6top]
Richardson, M., "6tisch secure join using 6top", draft-richardson-6tisch--security-6top-04 (work in progress), November 2014.

5.3. External Informative References

[IEEE802154e]
IEEE standard for Information Technology, "IEEE std. 802.15.4e, Part. 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) Amendment 1: MAC sublayer", April 2012.

[IEEE802154]
IEEE standard for Information Technology, "IEEE std. 802.15.4, Part. 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks", June 2011.

[OpenWSN] Watteyne, T., Vilajosana, X., Kerkez, B., Chraim, F., Weekly, K., Wang, Q., Glaser, S., and K. Pister, "OpenWSN: a Standards-Based Low-Power Wireless Development Environment", Transactions on Emerging Telecommunications Technologies , August 2012.

[morell04label]
Morell, A., Vilajosana, X., Lopez-Vicario, J., and T. Watteyne, "Label Switching over IEEE802.15.4e Networks. Transactions on Emerging Telecommunications Technologies", June 2013.

Authors' Addresses

Qin Wang (editor)
Univ. of Sci. and Tech. Beijing
30 Xueyuan Road
Beijing, Hebei 100083
China

Phone: +86 (10) 6233 4781
Email: wangqin@ies.ustb.edu.cn

Xavier Vilajosana
Universitat Oberta de Catalunya
156 Rambla Poblenou
Barcelona, Catalonia 08018
Spain

Phone: +34 (646) 633 681
Email: xvilajosana@uoc.edu

6TiSCH P. Thubert, Ed.
Internet-Draft Cisco
Intended status: Informational May 12, 2015
Expires: November 13, 2015

An Architecture for IPv6 over the TSCH mode of IEEE 802.15.4
draft-ietf-6tisch-architecture-08

Abstract

This document is the first volume of the 6TiSCH architecture of an IPv6 Multi-Link subnet that is composed of a high speed powered backbone and a number of IEEE802.15.4 TSCH low-power wireless networks attached and synchronized by Backbone Routers. The architecture defines mechanisms to establish and maintain routing and scheduling in a centralized, distributed, or mixed fashion.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The emergence of wireless technology has enabled a variety of new devices to get interconnected, at a very low marginal cost per device, at any distance ranging from Near Field to interplanetary, and in circumstances where wiring may not be practical, for instance on fast-moving or rotating devices.

At the same time, a new breed of Time Sensitive Networks is being developed to enable traffic that is highly sensitive to jitter, quite sensitive to latency, and with a high degree of operational criticality so that loss should be minimized at all times. Such traffic is not limited to professional Audio/ Video networks, but is also found in command and control operations such as industrial automation and vehicular sensors and actuators. At IEEE802.1, the Audio/Video Task Group [IEEE802.1TSNTG] Time Sensitive Networking (TSN) to address Deterministic Ethernet. The Medium access Control (MAC) of IEEE802.15.4 [IEEE802154] has evolved with the new IEEE802.15.4e TimeSlotted Channel Hopping (TSCH) [I-D.ietf-6tisch-tsch] mode for deterministic industrial-type applications. TSCH was introduced with the IEEE802.15.4e [IEEE802154e] amendment and will be wrapped up in the next revision of the IEEE802.15.4 standard. For all practical purpose, this document is expected to be insensitive to the future versions of the IEEE802.15.4 standard, which is thus referenced undated.

Though at a different time scale, both TSN and TSCH standards provide Deterministic capabilities to the point that a packet that pertains to a certain flow crosses the network from node to node following a very precise schedule, as a train that leaves intermediate stations at precise times along its path. With TSCH, time is formatted into timeSlots, and an individual cell is allocated to unicast or broadcast communication at the MAC level. The time-slotted operation reduces collisions, saves energy, and enables to more closely engineer the network for deterministic properties. The channel hopping aspect is a simple and efficient technique to combat multipath fading and external interference (for example by Wi-Fi emitters).

This document is the first volume of an architecture for an IPv6 Multi-Link subnet that is composed of a high speed powered backbone and a number of IEEE802.15.4 TSCH wireless networks attached and synchronized by backbone routers. Route Computation may be achieved in a centralized fashion by a Path Computation Element (PCE) [PCE], in a distributed fashion using the Routing Protocol for Low Power and Lossy Networks (RPL) [RFC6550], or in a mixed mode. The Backbone Routers may perform proxy IPv6 Neighbor Discovery (ND) [RFC4861] operations over the backbone on behalf of the wireless devices (also called motes), so they can share a same IPv6 subnet and appear to be connected to the same backbone as classical devices. The Backbone Routers may alternatively redistribute the registration in a routing protocol such as OSPF [RFC5340] or BGP [RFC2545], or inject them in a mobility protocol such as MIPv6 [RFC6275], NEMO [RFC3963], or LISP [RFC6830].

The 6TiSCH architecture defines four ways a schedule can be managed and TimeSlots can be allocated: Static Scheduling, neighbor-to-neighbor Scheduling, remote monitoring and scheduling management, and Hop-by-hop scheduling. In the case of remote monitoring and scheduling management, TimeSlots and other device resources are managed by an abstract Network Management Entity (NME), which may cooperate with the PCE in order to minimize the interaction with and the load on the constrained device.

The 6TiSCH architecture supports three different forwarding models, G-MPLS Track Forwarding, which switches a frame received at a particular TimeSlot into another TimeStot at Layer-2, 6LoWPAN Fragment Forwarding, which allows to forward individual 6loWPAN fragments along the route set by the first fragment, and classical IPv6 Forwarding, where the node selects a feasible successor at Layer-3 on a per packet basis, based on its routing table.

2. Terminology

Readers are expected to be familiar with all the terms and concepts that are discussed in "Neighbor Discovery for IP version 6" [RFC4861], "IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals" [RFC4919], Neighbor Discovery Optimization for Low-power and Lossy Networks [RFC6775] where the 6LoWPAN Router (6LR) and the 6LoWPAN Border Router (6LBR) are introduced, and "Multi-link Subnet Support in IPv6" [I-D.ietf-ipv6-multilink-subnets].

Readers may benefit from reading the "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks" [RFC6550] specification; "Multi-Link Subnet Issues" [RFC4903]; "Mobility Support in IPv6" [RFC6275]; "Neighbor Discovery Proxies (ND Proxy)" [RFC4389]; "IPv6 Stateless Address Autoconfiguration" [RFC4862]; "FCFS SAVI: First-Come, First-Served Source Address Validation Improvement for Locally Assigned IPv6 Addresses" [RFC6620]; and "Optimistic Duplicate Address

Detection" [RFC4429] prior to this specification for a clear understanding of the art in ND-proxying and binding.

The draft uses terminology defined or referenced in [I-D.ietf-6tisch-terminology], [I-D.chakrabarti-nordmark-6man-efficient-nd], [I-D.ietf-roll-rpl-industrial-applicability], [RFC4080], and [RFC5191].

The draft also conforms to the terms and models described in [RFC3444] and [RFC5889] and uses the vocabulary and the concepts defined in [RFC4291] for the IPv6 Architecture.

3. Applications and Goals

Some aspects of this architecture derive from existing industrial standards for Process Control such as ISA100.11a [ISA100.11a]and WirelessHART [WirelessHART], by its focus on Deterministic Networking, in particular with the use of the IEEE802.15.4 TSCH MAC and a centralized PCE. This approach leverages the TSCH MAC benefits for high reliability against interference, low-power consumption on deterministic traffic, and its Traffic Engineering capabilities. In such applications, Deterministic Networking applies mainly to control loops and movement detection, but it can also be used for supervisory control flows and management.

An incremental set of industrial requirements is addressed with the addition of an autonomic and distributed routing operation based on RPL. These use-cases include plant setup and decommissioning, as well as monitoring of lots of lesser importance measurements such as corrosion and events. RPL also enables mobile use cases such as mobile workers and cranes, as discussed in [I-D.ietf-roll-rpl-industrial-applicability].

A Backbone Router is included in order to scale the factory plant subnet to address large deployments, with proxy ND and time synchronization over a high speed backbone.

The architecture also applies to building automation that leverage RPL’s storing mode to address multipath over a large number of hops, in-vehicle command and control that can be as demanding as industrial applications, commercial automation and asset Tracking with mobile scenarios, home automation and domotics which become more reliable and thus provide a better user experience, and resource management (energy, water, etc.).

4. Overview

The scope of the present work is a subnet that, in its basic configuration, is made of a TSCH [I-D.ietf-6tisch-tsch] MAC Low Power Lossy Network (LLN).

```
               ---+-------- ............ ------------
                  |      External Network       |
                  |                          +-----+
               +-----+                       | NME |
               |     | LLN Border            |     |
               |     | router                +-----+
               +-----+
             o    o   o
      o     o   o     o
         o   o LLN   o    o     o
            o   o   o       o
                    o
```

Figure 1: Basic Configuration of a 6TiSCH Network

Security aspects of the join process by which a device obtains access to the network are discussed in Section 10. With TSCH, devices are time-synchronized at the MAC level. The use of a particular RPL Instance for time synchronization is discussed in Section 7.3. With this mechanism, the time synchronization starts at the RPL root and follows the RPL DODAGs with no timing loop.

The LLN devices communicate over IPv6 [RFC2460] using the 6LoWPAN Header Compression (6LoWPAN HC) [RFC6282]. From the perspective of Layer-3, a single LLN interface (typically an IEEE802.15.4-compliant radio) may be seen as a collection of Links with different capabilities for unicast or multicast services. An IPv6 subnet spans over multiple links, effectively forming a Multi-Link subnet. Within that subnet, neighbor devices are discovered with 6LoWPAN Neighbor Discovery [RFC6775] (6LoWPAN ND). RPL [RFC6550] enables routing within the LLN, in the so called Route Over fashion, either in storing (stateful) or non-storing (stateless, with routing headers) mode.

RPL forms Destination Oriented Directed Acyclic Graphs (DODAGs) within Instances of the protocol, each Instance being associated with an Objective Function (OF) to form a routing topology. A particular LLN device, the LLN Border Router (LBR), acts as RPL root, 6LoWPAN HC terminator, and Border Router for the LLN to the outside. The LBR is usually powered. More on RPL Instances can be found in section 3.1 of RPL [RFC6550], in particular "3.1.2. RPL Identifiers" and "3.1.3. Instances, DODAGs, and DODAG Versions".

This architecture expects that a 6LoWPAN node can connect as a leaf to a RPL network, where the leaf support is the minimal functionality to connect as a host to a RPL network without the need to participate to the full routing protocol. The architecture also expects that a 6LoWPAN node that is not aware at all of the RPL protocol may also connect as a host. The derived requirements are listed in [I-D.thubert-6lo-rfc6775-update-reqs].

An extended configuration of the subnet comprises multiple LLNs. The LLNs are interconnected and synchronized over a backbone, that can be wired or wireless. The backbone can be a classical IPv6 network, with Neighbor Discovery operating as defined in [RFC4861] and [RFC4862]. This architecture requires new work to standardize the the registration of 6LoWPAN nodes to the Backbone Routers.

In the extended configuration, a Backbone Router (6BBR) acts as an Energy Aware Default Router (NEAR) as defined in [I-D.chakrabarti-nordmark-6man-efficient-nd]. The 6BBR performs ND proxy operations between the registered devices and the classical ND devices that are located over the backbone. 6TiSCH 6BBRs synchronize with one another over the backbone, so as to ensure that the multiple LLNs that form the IPv6 subnet stay tightly synchronized.

```
                  ---+-------- ............ ------------
                     |      External Network       |
                     |                          +-----+
                     |             +-----+      | NME |
                  +-----+          |  +-----+   |     |
                  |     | Router   |  | PCE |   +-----+
                  |     |          +--|     |
                  +-----+             +-----+
                     |                   |
                     | Subnet Backbone   |
               +--------------------+------------------+
               |                    |                  |
            +-----+             +-----+             +-----+
            |     | Backbone    |     | Backbone    |     | Backbone
       o    |     | router      |     | router      |     | router
            +-----+             +-----+             +-----+
       o                  o                   o                 o   o
           o    o   o         o   o  o   o         o  o   o    o
      o             o        o  LLN      o      o         o      o
         o   o    o       o      o o     o  o   o    o    o     o
```

Figure 2: Extended Configuration of a 6TiSCH Network

In order to serve nodes that are multiple hops away, an integrated RPL root and 6LBR may be collocated with the 6BBR, or attached to the

6BBR in which case they would perform the registration on behalf of the remote LLN nodes - they proxy the efficient ND registration over the LLN in order for the 6BBR to perform proxy ND operations over the backbone.

If the Backbone is Deterministic (such as defined by the Time Sensitive Networking WG at IEEE), then the Backbone Router ensures that the end-to-end deterministic behavior is maintained between the LLN and the backbone. The DetNet Architecture [I-D.finn-detnet-architecture] studies Layer-3 aspects of Deterministic Networks, and covers networks that span multiple Layer-2 domains.

5. Scope

5.1. Components

In order to control the complexity and the size of the 6TiSCH work, the architecture and the associated IETF work are staged in volumes. This document covers the first stage of the work, as specified by the WG charter. If the work continues as expected, further volumes will complete this piece and provide the full coverage of IPv6 over TSCH.

The main architectural blocks are represented below to help detail what is covered and what is not yet covered from the global 6TiSCH architecture by this initial volume:

```
      +-----+-----+
      | PCEP|TEAS/|
      | PCE |CCAMP|
+-----+-----+-----+-----+-------+-----+
|    (COMI)       |PANA |6LoWPAN| RPL |
| CoAP  / DTLS    |     |   ND  |     |
+-----+-----+-----+-----+-------+-----+
|       UDP       |          ICMP     |
+-----+-----+-----+-----+-------+-----+-----+
|                 IPv6                      |
+-------------------------------------------+
|  6LoWPAN adaptation and compression (HC)  |
+-------------------------------------------+
|                   6top                    |
+-------------------------------------------+
|            IEEE802.15.4    TSCH           |
+-------------------------------------------+
```

Figure 3: Envisioned 6TiSCH protocol stack

RPL is the routing protocol of choice for LLNs. So far, there was no identified need to define a 6TiSCH specific Objective Function. The Minimal 6TiSCH Configuration [I-D.ietf-6tisch-minimal] describes the operation of RPL over a static schedule used in a slotted aloha fashion, whereby all active slots may be used for emission or reception of both unicast and multicast frames.

The architecture of the operation of RPL over a dynamic schedule is deferred to a subsequent volume of the architecture.

6TiSCH has adopted the general direction of CoAP Management Interface (COMI) [I-D.vanderstok-core-comi] for the management of devices. This is leveraged for instance for the implementation of the generic data model for the 6top sublayer management interface [I-D.ietf-6tisch-6top-interface]. The proposed implementation is based on CoAP and CBOR, and specified in 6TiSCH Resource Management and Interaction using CoAP [I-D.ietf-6tisch-coap].

The work on centralized track computation is deferred to a subsequent volume of the architecture. The Path Computation Element (PCE) is certainly the core component of that architecture. Around the PCE, a protocol such as an extension to a TEAS [TEAS] protocol (maybe running over CoAP as illustrated) will be required to expose the device capabilities and the network peers to the PCE, and a protocol such as a lightweight PCEP or an adaptation of CCAMP [CCAMP] G-MPLS formats and procedures will be used to publish the tracks, computed by the PCE, to the devices (maybe in a fashion similar to RSVP-TE).

The selection of an authentication, an authorization and a Transport layer security protocols are out of scope for this volume.

The Datagram Transport Layer Security (DTLS) [RFC6347] is represented as an example of a protocol that could be used to protect CoAP datagrams, and work at [DICE] may optimize the protocol for constrained devices.

Similarly, the Protocol for Carrying Authentication for Network access (PANA) [RFC5191] is represented as an example of a protocol that could be leveraged to secure the join process, as a Layer-3 alternate to IEEE802.1x/EAP. Work resulting from [ACE] could be considered as well. Regardless, the security model must ensure that, prior to a join process, packets from a untrusted device are controlled in volume and in reachability. An overview of the security aspects of the join process can be found in Section 10. Related contributions are presented in Appendix A.

The 6TiSCH Operation sublayer (6top) [I-D.wang-6tisch-6top-sublayer] is an Logical Link Control (LLC) or a portion thereof that provides the abstraction of an IP link over a TSCH MAC. The work on the operations of that layer, in particular related to dynamic scheduling, is only introduced here, and should be detailed further in a subsequent volume of the architecture.

5.2. Dependencies

At the time of this writing, the components and protocols that are required to implement this stage of architecture are not fully available from the IETF. In particular, the requirements on an evolution of 6LoWPAN Neighbor Discovery that are needed to implement the Backbone Router as covered by this stage of the architecture are detailed in [I-D.thubert-6lo-rfc6775-update-reqs].

The 6TiSCH Architecture applies the concepts of Deterministic Networking on a Layer-3 network. The 6TiSCH Architecture should inherit from DetNet [I-D.finn-detnet-architecture] work and thus depends on it. In turn, DetNet is expected to integrate and maintain consistency with the work that has taken place and is continuing at IEEE802.1TSN and AVnu.

The current charter positions 6TiSCH on IEEE802.15.4 only. Though most of the design should be portable on other link types, 6TiSCH has a strong dependency on IEEE802.15.4 and its evolution. A new version of the IEEE802.15.4 standard is expected in 2015. That version should integrate TSCH as well as other amendments and fixes into the main specification. The impact on this Architecture should be minimal to non-existent, but deeper work such as 6top and security may be impacted. A 6TiSCH Interest Group was formed at IEEE to maintain the synchronization and help foster work at the IEEE should 6TiSCH demand it.

ISA100 [ISA100] Common Network Management (CNM) is another external work of interest for 6TiSCH. The group, referred to as ISA100.20, defines a Common Network Management framework that should enable the management of resources that are controlled by heterogeneous protocols such as ISA100.11a [ISA100.11a], WirelessHART [WirelessHART], and 6TiSCH. Interestingly, the establishment of 6TiSCH Deterministic paths, called tracks, are also in scope, and ISA100.20 is working on requirements for DetNet.

6. 6LoWPAN (and RPL)

The architecture expects that a 6LoWPAN node that is not aware at all of the RPL protocol may still connect as a host. It suggests to extend 6LoWPAN ND [RFC6775] to carry the sequence number that is needed by RPL to track the movements of the device, and optionally some abstract information about the RPL instance (topology) that the device will be reachable over.

In this design, the root of the RPL network is integrated with the 6LoWPAN ND 6LBR, but it is logically separated from the Backbone Router (6BBR) that is used to connect the RPL topology to the backbone. This way, the root has all information from 6LoWPAN ND and RPL about the LLN devices attached to it.

This architecture also expects that the root of the RPL network (proxy-)registers the LLN devices on their behalf to the 6BBR, for whatever operation the 6BBR performs on the backbone, such as ND proxy, or redistribution in a routing protocol. It suggests to use an extension of the mixed mode of Efficient ND [I-D.chakrabarti-nordmark-6man-efficient-nd] for the registration as described in [I-D.thubert-6lowpan-backbone-router].

It results that, as illustrated in Figure 4, the periodic signaling would start at the leaf node with 6LoWPAN ND, then would be carried over RPL to the RPL root, and then with Efficient-ND to the 6BBR. Efficient ND being an adaptation of 6LoWPAN ND, it makes sense to keep those two homogeneous in the way they use the source and the target addresses in the Neighbor Solicitation (NS) messages for registration, as well as in the options that they use for that process.

```
    6LoWPAN Node        6LR             6LBR            6BBR
     (RPL leaf)       (router)         (root)
         |               |               |               |
         |  6LoWPAN ND   |6LoWPAN ND+RPL | Efficient ND  | IPv6 ND
         |   LLN link    |Route-Over mesh|  IPv6 link    | Backbone
         |               |               |               |
         |  NS(ARO)      |               |               |
         |-------------->|               |               |
         | 6LoWPAN ND    | DAR (then DAO)|               |
         |               |-------------->|               |
         |               |               |  NS(ARO)      |
         |               |               |-------------->|
         |               |               |               | DAD
         |               |               |               |------>
         |               |               |               |
         |               |               |  NA(ARO)      |
         |               |               |<--------------|
         |               | DAC           |               |
         |               |<--------------|               |
         |  NA(ARO)      |               |               |
         |<--------------|               |               |
```

Figure 4: (Re-)Registration Flow over Multi-Link Subnet

As the network builds up, a node should start as a leaf to join the RPL network, and may later turn into both a RPL-capable router and a 6LR, so as to accept leaf nodes to recursively join the network.

6.1. RPL Leaf Support in 6LoWPAN ND

RPL needs a set of information in order to advertise a leaf node through a DAO message and establish reachability.

At the bare minimum the leaf device must provide a sequence number that matches the RPL specification in section 7. Section 4.1 of [I-D.chakrabarti-nordmark-6man-efficient-nd], on the Address Registration Option (ARO), already incorporates that addition with a new field in the option called the Transaction ID.

If for some reason the node is aware of RPL topologies, then providing the RPL InstanceID for the instances to which the node wishes to participate would be a welcome addition. In the absence of such information, the RPL router must infer the proper instanceID from external rules and policies.

On the backbone, the InstanceID is expected to be mapped onto a an overlay that matches the instanceID, for instance a VLANID.

6.2. registration Failures Due to Movement

Registration to the 6LBR through DAR/DAC messages [RFC6775] may percolate slowly through an LLN mesh, and it might happen that in the meantime, the 6LoWPAN node moves and registers somewhere else. Both RPL and 6LoWPAN ND lack the capability to indicate that the same node is registered elsewhere, so as to invalidate states down the deprecated path.

In its current expression and functionality, 6LoWPAN ND considers that the registration is used for the purpose of DAD only as opposed to that of achieving reachability, and as long as the same node registers the IPv6 address, the protocol is functional. In order to act as a RPL leaf registration protocol and achieve reachability, the device must use the same TID for all its concurrent registrations, and registrations with a past TID should be declined. The state for an obsolete registration in the 6LR, as well as the RPL routers on the way, should be invalidated. This can only be achieved with the addition of a new Status in the DAC message, and a new error/clean-up flow in RPL.

6.3. Proxy registration

The 6BBR provides the capability to defend an address that is owned by a 6LoWPAN Node, and attract packets to that address, whether it is done by proxying ND over a MultiLink Subnet, redistributing the address in a routing protocol or advertising it through an alternate proxy registration such as the Locator/ID Separation Protocol [RFC6830] (LISP) or Mobility Support in IPv6 [RFC6275] (MIPv6). In a LLN, it makes sense to piggyback the request to proxy/defend an address with its registration.

6.4. Target Registration

In their current incarnations, both 6LoWPAN ND and Efficient ND expect that the address being registered is the source of the NS(ARO) message and thus impose that a Source Link-Layer Address (SLLA) option be present in the message. In a mesh scenario where the 6LBR is physically separated from the 6LoWPAN Node, the 6LBR does not own the address being registered. This suggests that [I-D.chakrabarti-nordmark-6man-efficient-nd] should evolve to register the Target of the NS message as opposed to the Source Address. From another perspective, it may happen, in the use case of a Star topology, that the 6LR, 6LBR and 6BBR are effectively collapsed and should support 6LoWPAN ND clients. The convergence of efficient ND and 6LoWPAN ND into a single protocol is thus highly desirable.

In any case, as long as the DAD process is not complete for the address used as source of the packet, it is against the current practice to advertise the SLLA, since this may corrupt the ND cache of the destination node, as discussed in the Optimistic DAD specification [RFC4429] with regards to the TENTATIVE state.

This may look like a chicken and an egg problem, but in fact 6LoWPAN ND acknowledges that the Link-Local Address that is based on an EUI-64 address of a LLN node may be autoconfigured without the need for DAD. It results that a node could use that Address as source, with an SLLA option in the message if required, to register any other addresses, either Global or Unique-Local Addresses, which would be indicated in the Target.

The suggested change is to register the target of the NS message, and use Target Link-Layer Address (TLLA) in the NS as opposed to the SLLA in order to install a Neighbor Cache Entry. This would apply to both Efficient ND and 6LoWPAN ND in a very same manner, with the caveat that depending on the nature of the link between the 6LBR and the 6BBR, the 6LBR may resort to classical ND or DHCPv6 to obtain the address that it uses to source the NS registration messages, whether for itself or on behalf of LLN nodes.

6.5. RPL root vs. 6LBR

6LoWPAN ND is unclear on how the 6LBR is discovered, and how the liveliness of the 6LBR is asserted over time. On the other hand, the discovery and liveliness of the RPL root are obtained through the RPL protocol.

When 6LoWPAN ND is coupled with RPL, the 6LBR and RPL root functionalities are co-located in order that the address of the 6LBR be indicated by RPL DIO messages and to associate the unique ID from the DAR/DAC exchange with the state that is maintained by RPL. The DAR/DAC exchange becomes a preamble to the DAO messages that are used from then on to reconfirm the registration, thus eliminating a duplication of functionality between DAO and DAR messages.

6.6. Securing the Registration

A typical attack against IPv6 ND is address spoofing, whereby a rogue node claims the IPv6 Address of another node in and hijacks its traffic. The threats against IPv6 ND as described in SEcure Neighbor Discovery (SEND) [RFC3971] are applicable to 6LoPWAN ND as well, but the solution can not work as the route over network does not permit direct peer to peer communication.

Additionally SEND requires considerably enlarged ND messages to carry cryptographic material, and requires that each protected address is generated cryptographically, which implies the computation of a different key for each Cryptographically Generated Address (CGA). SEND as defined in [RFC3971] is thus largely unsuitable for application in a LLN.

With 6LoWPAN ND, as illustrated in Figure 4, it is possible to leverage the registration state in the 6LBR, which may store additional security information for later proof of ownership. If this information proves the ownership independently of the address itself, then a single proof may be used to protect multiple addresses.

Once an Address is registered, the 6LBR maintains a state for that Address and is in position to bind securely the first registration with the Node that placed it, whether the Address is CGA or not. It should thus be possible to protect the ownership of all the addresses of a 6LoWPAN Node with a single key, and there should not be a need to carry the cryptographic material more than once to the 6LBR.

The energy constraint is usually a foremost factor, and attention should be paid to minimize the burden on the CPU. Hardware-assisted support of variants of the Counter with CBC-MAC [RFC3610] (CCM) authenticated encryption block cipher mode such as CCM* are common in LowPower ship-set implementations, and 6LoWPAN ND security mechanism should be capable to reuse them when applicable.

Finally, the code footprint in the device being also an issue, the capability to reuse not only hardware-assist mechanisms but also software across layers has to be considered. For instance, if code has to be present for upper-layer operations, e.g AES-CCM Cipher Suites for Transport Layer Security (TLS) [RFC6655], then the capability to reuse that code should be considered.

7. TSCH and 6top

7.1. 6top

6top is a logical link control sitting between the IP layer and the TSCH MAC layer, which provides the link abstraction that is required for IP operations. The 6top operations are specified in [I-D.wang-6tisch-6top-sublayer]. In particular, 6top provides a management interface that enables an external management entity to schedule cells and slotFrames, and allows the addition of complementary functionality, for instance to support a dynamic schedule management based on observed resource usage as discussed in Section 8.1.2.

The 6top data model and management interfaces are further discussed in Section 8.1.3.

7.1.1. Hard Cells

The architecture defines "soft" cells and "hard" cells. "Hard" cells are owned and managed by an separate scheduling entity (e.g. a PCE) that specifies the slotOffset/channelOffset of the cells to be added/moved/deleted, in which case 6top can only act as instructed, and may not move hard cells in the TSCH schedule on its own.

7.1.2. Soft Cells

6top contains a monitoring process which monitors the performance of cells, and can move a cell in the TSCH schedule when it performs poorly. This is only applicable to cells which are marked as "soft". To reserve a soft cell, the higher layer does not indicate the exact slotOffset/channelOffset of the cell to add, but rather the resulting bandwidth and QoS requirements. When the monitoring process triggers a cell reallocation, the two neighbor devices communicating over this cell negotiate its new position in the TSCH schedule.

7.2. 6top and RPL Objective Function operations

An implementation of a RPL [RFC6550] Objective Function (OF), such as the RPL Objective Function Zero (OF0) [RFC6552] that is used in the Minimal 6TiSCH Configuration [I-D.ietf-6tisch-minimal] to support RPL over a static schedule, may leverage, for its internal computation, the information maintained by 6top.

Most OFs require metrics about reachability, such as the ETX. 6top creates and maintains an abstract neighbor table, and this state may be leveraged to feed an OF and/or store OF information as well. In particular, 6top creates and maintains an abstract neighbor table. A neighbor table entry contains a set of statistics with respect to that specific neighbor including the time when the last packet has been received from that neighbor, a set of cell quality metrics (e.g. RSSI or LQI), the number of packets sent to the neighbor or the number of packets received from it. This information can be obtained through 6top management APIs as detailed in the 6top sublayer specification [I-D.wang-6tisch-6top-sublayer] and used for instance to compute a Rank Increment that will determine the selection of the preferred parent.

6top provides statistics about the underlying layer so the OF can be tuned to the nature of the TSCH MAC layer. 6top also enables the RPL OF to influence the MAC behaviour, for instance by configuring the periodicity of IEEE802.15.4 Extended Beacons (EB’s). By augmenting the EB periodicity, it is possible to change the network dynamics so as to improve the support of devices that may change their point of attachment in the 6TiSCH network.

Some RPL control messages, such as the DODAG Information Object (DIO) are ICMPv6 messages that are broadcast to all neighbor nodes. With 6TiSCH, the broadcast channel requirement is addressed by 6top by configuring TSCH to provide a broadcast channel, as opposed to, for instance, piggybacking the DIO messages in Enhance Beacons. Consideration was given towards finding a way to embed the Route Advertisements and the RPL DIO messages (both of which are multicast) into the IEEE802.15.4 Enhanced Beacons. It was determined that this produced undue timer coupling among layers, that the resulting packet size was potentially too large, and required it is not yet clear that there is any need for Enhanced Beacons in a production network.

7.3. Network Synchronization

Nodes in a TSCH network must be time synchronized. A node keeps synchronized to its time source neighbor through a combination of frame-based and acknowledgment-based synchronization. In order to maximize battery life and network throughput, it is advisable that RPL ICMP discovery and maintenance traffic (governed by the trickle timer) be somehow coordinated with the transmission of time synchronization packets (especially with enhanced beacons). This could be achieved through an interaction of the 6top sublayer and the RPL objective Function, or could be controlled by a management entity.

Time distribution requires a loop-less structure. Nodes taken in a synchronization loop will rapidly desynchronize from the network and become isolated. It is expected that a RPL DAG with a dedicated global Instance is deployed for the purpose of time synchronization. That Instance is referred to as the Time Synchronization Global Instance (TSGI). The TSGI can be operated in either of the 3 modes that are detailed in section 3.1.3 of RPL [RFC6550], "Instances, DODAGs, and DODAG Versions". Multiple uncoordinated DODAGs with independent roots may be used if all the roots share a common time source such as the Global Positioning System (GPS). In the absence of a common time source, the TSGI should form a single DODAG with a virtual root. A backbone network is then used to synchronize and coordinate RPL operations between the backbone routers that act as sinks for the LLN. Optionally, RPL’s periodic operations may be used to transport the network synchronization. This may mean that 6top would need to trigger (override) the trickle timer if no other traffic has occurred for such a time that nodes may get out of synchronization.

A node that has not joined the TSGI advertises a MAC level Join Priority of 0xFF to notify its neighbors that is not capable of serving as time parent. A node that has joined the TSGI advertises a MAC level Join Priority set to its DAGRank() in that Instance, where DAGRank() is the operation specified in section 3.5.1 of [RFC6550], "Rank Comparison".

A root is configured or obtains by some external means the knowledge of the RPLInstanceID for the TSGI. The root advertises its DagRank in the TSGI, that must be less than 0xFF, as its Join Priority (JP) in its IEEE802.15.4 Extended Beacons (EB). We'll note that the JP is now specified between 0 and 0x3F leaving 2 bits in the octet unused in the IEEE802.15.4e specification. After consultation with IEEE authors, it was asserted that 6TiSCH can make a full use of the octet to carry an integer value up to 0xFF.

A node that reads a Join Priority of less than 0xFF should join the neighbor with the lesser Join Priority and use it as time parent. If the node is configured to serve as time parent, then the node should join the TSGI, obtain a Rank in that Instance and start advertising its own DagRank in the TSGI as its Join Priority in its EBs.

7.4. SlotFrames and Priorities

6TiSCH enables in essence the capability to use IPv6 over a MAC layer that enables to schedule some of the transmissions. In order to ensure that the medium is free of contending packets when time arrives for a scheduled transmission, a window of time is defined around the scheduled transmission time where the medium must be free of contending energy.

One simple way to obtain such a window is to format time and frequencies in cells of transmission of equal duration. This is the method that is adopted in IEEE802.15.4 TSCH as well as the Long Term Evolution (LTE) of cellular networks.

In order to describe that formatting of time and frequencies, the 6TiSCH architecture defines a global concept that is called a Channel Distribution and Usage (CDU) matrix; a CDU matrix is a matrix of cells with an height equal to the number of available channels (indexed by ChannelOffsets) and a width (in timeSlots) that is the period of the network scheduling operation (indexed by slotOffsets) for that CDU matrix. The size of a cell is a timeSlot duration, and values of 10 to 15 milliseconds are typical in 802.15.4 TSCH to accommodate for the transmission of a frame and an ack, including the security validation on the receive side which may take up to a few milliseconds on some device architecture.

A CDU matrix iterates over and over with a pseudo-random rotation from an epoch time. In a given network, there might be multiple CDU matrices that operate with different width, so they have different durations and represent different periodic operations. It is recommended that all CDU matrices in a 6TiSCH domain operate with the same cell duration and are aligned, so as to reduce the chances of interferences from slotted-aloha operations. The knowledge of the CDU matrices is shared between all the nodes and used in particular to define slotFrames.

A slotFrame is a MAC-level abstraction that is common to all nodes and contains a series of timeSlots of equal length and precedence. It is characterized by a slotFrame_ID, and a slotFrame_size. A slotFrame aligns to a CDU matrix for its parameters, such as number and duration of timeSlots.

Multiple slotFrames can coexist in a node schedule, i.e., a node can have multiple activities scheduled in different slotFrames, based on the precedence of the 6TiSCH topologies. The slotFrames may be aligned to different CDU matrices and thus have different width. There is typically one slotFrame for scheduled traffic that has the highest precedence and one or more slotFrame(s) for RPL traffic. The timeSlots in the slotFrame are indexed by the SlotOffset; the first cell is at SlotOffset 0.

When a packet is received from a higher layer for transmission, 6top inserts that packet in the outgoing queue which matches the packet best (Differentiated Services [RFC2474] can therefore be used). At each scheduled transmit slot, 6top looks for the frame in all the outgoing queues that best matches the cells. If a frame is found, it is given to the TSCH MAC for transmission.

7.5. Distributing the reservation of cells

6TiSCH expects a high degree of scalability together with a distributed routing functionality based on RPL. To achieve this goal, the spectrum must be allocated in a way that allows for spatial reuse between zones that will not interfere with one another. In a large and spatially distributed network, a 6TiSCH node is often in a good position to determine usage of spectrum in its vicinity.

Use cases for distributed routing are often associated with a statistical distribution of best-effort traffic with variable needs for bandwidth on each individual link. With 6TiSCH, the link abstraction is implemented as a bundle of cells; the size of a bundle is optimal when both the energy wasted idle listening and the packet drops due to congestion loss are minimized. This can be maintained if the number of cells in a bundle is adapted dynamically, and with enough reactivity, to match the variations of best-effort traffic. In turn, the agility to fulfill the needs for additional cells improves when the number of interactions with other devices and the protocol latencies are minimized.

6TiSCH limits that interaction to RPL parents that will only negotiate with other RPL parents, and performs that negotiation by groups of cells as opposed to individual cells. The 6TiSCH architecture allows RPL parents to adjust dynamically, and independently from the PCE, the amount of bandwidth that is used to communicate between themselves and their children, in both directions; to that effect, an allocation mechanism enables a RPL parent to obtain the exclusive use of a portion of a CDU matrix within its interference domain. Note that a PCE is expected to have precedence in the allocation, so that a RPL parent would only be able to obtain portions that are not in-use by the PCE.

The 6TiSCH architecture introduces the concept of chunks [I-D.ietf-6tisch-terminology]) to operate such spectrum distribution for a whole group of cells at a time. The CDU matrix is formatted into a set of chunks, each of them identified uniquely by a chunk-ID. The knowledge of this formatting is shared between all the nodes in a 6TiSCH network. 6TiSCH also defines the process of chunk ownership appropriation whereby a RPL parent discovers a chunk that is not used in its interference domain (e.g lack of energy detected in reference cells in that chunk); then claims the chunk, and then defends it in case another RPL parent would attempt to appropriate it while it is in use. The chunk is the basic unit of ownership that is used in that process.

```
                 +-----+-----+-----+-----+-----+-----+-----+     +-----+
 chan.Off. 0     |chnkA|chnkP|chnk7|chnkO|chnk2|chnkK|chnk1| ... |chnkZ|
                 +-----+-----+-----+-----+-----+-----+-----+     +-----+
 chan.Off. 1     |chnkB|chnkQ|chnkA|chnkP|chnk3|chnkL|chnk2| ... |chnk1|
                 +-----+-----+-----+-----+-----+-----+-----+     +-----+
                   ...
                 +-----+-----+-----+-----+-----+-----+-----+     +-----+
 chan.Off. 15    |chnkO|chnk6|chnkN|chnk1|chnkJ|chnkZ|chnkI| ... |chnkG|
                 +-----+-----+-----+-----+-----+-----+-----+     +-----+
                    0     1     2     3     4     5     6           M
```

Figure 5: CDU matrix Partitioning in Chunks

As a result of the process of chunk ownership appropriation, the RPL parent has exclusive authority to decide which cell in the appropriated chunk can be used by which node in its interference domain. In other words, it is implicitly delegated the right to manage the portion of the CDU matrix that is represented by the chunk. The RPL parent may thus orchestrate which transmissions occur in any of the cells in the chunk, by allocating cells from the chunk to any form of communication (unicast, multicast) in any direction between itself and its children. Initially, those cells are added to the heap of free cells, then dynamically placed into existing bundles, in new bundles, or allocated opportunistically for one transmission.

The appropriation of a chunk can also be requested explicitly by the PCE to any node. In that case, the node still may need to perform the appropriation process to validate that no other node has claimed that chunk already. After a successful appropriation, the PCE owns the cells in that chunk, and may use them as hard cells to set up tracks.

8. Communication Paradigms and Interaction Models

[I-D.ietf-6tisch-terminology] defines the terms of Communication Paradigms and Interaction Models, which can be placed in parallel to the Information Models and Data Models that are defined in [RFC3444].

A Communication Paradigms would be an abstract view of a protocol exchange, and would come with an Information Model for the information that is being exchanged. In contrast, an Interaction Models would be more refined and could point on standard operation such as a Representational state transfer (REST) "GET" operation and would match a Data Model for the data that is provided over the protocol exchange.

section 2.1.3 of [I-D.ietf-roll-rpl-industrial-applicability] and next sections discuss application-layer paradigms, such as Source-sink (SS) that is a Multipeer to Multipeer (MP2MP) model primarily used for alarms and alerts, Publish-subscribe (PS, or pub/sub) that is typically used for sensor data, as well as Peer-to-peer (P2P) and Peer-to-multipeer (P2MP) communications. Additional considerations on Duocast and its N-cast generalization are also provided. Those paradigms are frequently used in industrial automation, which is a major use case for IEEE802.15.4 TSCH wireless networks with [ISA100.11a] and [WirelessHART], that provides a wireless access to [HART] applications and devices.

This specification focuses on Communication Paradigms and Interaction Models for packet forwarding and TSCH resources (cells) management. Management mechanisms for the TSCH schedule at Link-layer (one-hop), Network-layer (multithop along a track), and Application-layer (remote control) are discussed in Section 8.1. Link-layer frame forwarding interactions are discussed in Section 8.2, and Network-layer Packet routing is addressed in Section 8.3.

8.1. Schedule Management Mechanisms

6TiSCH uses 4 paradigms to manage the TSCH schedule of the LLN nodes: Static Scheduling, neighbor-to-neighbor Scheduling, remote monitoring and scheduling management, and Hop-by-hop scheduling. Multiple mechanisms are defined that implement the associated Interaction Models, and can be combined and used in the same LLN. Which mechanism(s) to use depends on application requirements.

8.1.1. Static Scheduling

In the simplest instantiation of a 6TiSCH network, a common fixed schedule may be shared by all nodes in the network. Cells are shared, and nodes contend for slot access in a slotted aloha manner.

A static TSCH schedule can be used to bootstrap a network, as an initial phase during implementation, or as a fall-back mechanism in case of network malfunction. This schedule can be preconfigured or learnt by a node when joining the network. Regardless, the schedule remains unchanged after the node has joined a network. The Routing Protocol for LLNs (RPL) is used on the resulting network. This "minimal" scheduling mechanism that implements this paradigm is detailed in [I-D.ietf-6tisch-minimal].

8.1.2. Neighbor-to-neighbor Scheduling

In the simplest instantiation of a 6TiSCH network described in Section 8.1.1, nodes may expect a packet at any cell in the schedule and will waste energy idle listening. In a more complex instantiation of a 6TiSCH network, a matching portion of the schedule is established between peers to reflect the observed amount of transmissions between those nodes. The aggregation of the cells between a node and a peer forms a bundle that the 6top layer uses to implement the abstraction of a link for IP. The bandwidth on that link is proportional to the number of cells in the bundle.

If the size of a bundle is configured to fit an average amount of bandwidth, peak traffic is dropped. If the size is configured to allow for peak emissions, energy is be wasted idle listening.

In the most efficient instantiation of a 6TiSCH network, the size of the bundles that implement the links may be changed dynamically in order to adapt to the need of end-to-end flows routed by RPL. An optional On-The-Fly (OTF) component may be used to monitor bandwidth usage and perform requests for dynamic allocation by the 6top sublayer. The OTF component is not part of the 6top sublayer. It may be collocated on the same device or may be partially or fully offloaded to an external system.

The 6top sublayer [I-D.wang-6tisch-6top-sublayer] defines a protocol for neighbor nodes to reserve soft cells to one another. Because this reservation is done without global knowledge of the schedule of nodes in the LLN, scheduling collisions are possible. 6top defines a monitoring process which continuously tracks the packet delivery ratio of soft cells. It uses these statistics to trigger the reallocation of a soft cell in the schedule, using a negotiation protocol between the neighbors nodes communicating over that cell.

Monitoring and relocation is done in the 6top layer. For the upper layer, the connection between two neighbor nodes appears as an number of cells. Depending on traffic requirements, the upper layer can request 6top to add or delete a number of cells scheduled to a particular neighbor, without being responsible for choosing the exact slotOffset/channelOffset of those cells.

8.1.3. remote Monitoring and Schedule Management

The 6top interface document [I-D.ietf-6tisch-6top-interface] specifies the generic data model that can be used to monitor and manage resources of the 6top sublayer. Abstract methods are suggested for use by a management entity in the device. The data model also enables remote control operations on the 6top sublayer.

The capability to interact with the node 6top sublayer from multiple hops away can be leveraged for monitoring, scheduling, or a combination of thereof. The architecture supports variations on the deployment model, and focuses on the flows rather than whether there is a proxy or a translation operation en-route.

[I-D.ietf-6tisch-coap] defines an mapping of the 6top set of commands, which is described in [I-D.ietf-6tisch-6top-interface], to CoAP resources. This allows an entity to interact with the 6top layer of a node that is multiple hops away in a RESTful fashion.

[I-D.ietf-6tisch-coap] defines a basic set CoAP resources and associated RESTful access methods (GET/PUT/POST/DELETE). The payload (body) of the CoAP messages is encoded using the CBOR format. The draft also defines the concept of "profiles" to allow for future or specific extensions, as well as a mechanism for a CoAP client to discover the profiles installed on a node.

The entity issuing the CoAP requests can be a central scheduling entity (e.g. a PCE), a node multiple hops away with the authority to modify the TSCH schedule (e.g. the head of a local cluster), or a external device monitoring the overall state of the network (e.g. NME). It is also possible that a mapping entity on the backbone transforms a non-CoAP protocol such as PCEP into the RESTful interfaces that the 6TiSCH devices support.

8.1.4. Hop-by-hop Scheduling

A node can reserve a track to a destination node multiple hops away by installing soft cells at each intermediate node. This forms a track of soft cells. It is the responsibility of the 6top sublayer of each node on the track to monitor these soft cells and trigger relocation when needed.

This hop-by-hop reservation mechanism is expected to be similar in essence to [RFC3209] and/or [RFC4080]/[RFC5974]. The protocol for a node to trigger hop-by-hop scheduling is not yet defined.

8.2. Forwarding Models

By forwarding, this specification means the per-packet operation that allows to deliver a packet to a next hop or an upper layer in this node. Forwarding is based on pre-existing state that was installed as a result of a routing computation Section 8.3. 6TiSCH supports three different forwarding model, G-MPLS Track Forwarding (TF), 6LoWPAN Fragment Forwarding (FF) and IPv6 Forwarding (6F).

8.2.1. Track Forwarding

A Track is a unidirectional path between a source and a destination. In a Track cell, the normal operation of IEEE802.15.4 Automatic Repeat-reQuest (ARQ) usually happens, though the acknowledgment may be omitted in some cases, for instance if there is no scheduled cell for a retry.

Track Forwarding is the simplest and fastest. A bundle of cells set to receive (RX-cells) is uniquely paired to a bundle of cells that are set to transmit (TX-cells), representing a layer-2 forwarding state that can be used regardless of the network layer protocol. This model can effectively be seen as a Generalized Multi-protocol Label Switching (G-MPLS) operation in that the information used to switch a frame is not an explicit label, but rather related to other properties of the way the packet was received, a particular cell in the case of 6TiSCH. As a result, as long as the TSCH MAC (and Layer-2 security) accepts a frame, that frame can be switched regardless of the protocol, whether this is an IPv6 packet, a 6LoWPAN fragment, or a frame from an alternate protocol such as WirelessHART or ISA100.11a.

A data frame that is forwarded along a Track normally has a destination MAC address that is set to broadcast - or a multicast address depending on MAC support. This way, the MAC layer in the intermediate nodes accepts the incoming frame and 6top switches it without incurring a change in the MAC header. In the case of IEEE802.15.4, this means effectively broadcast, so that along the Track the short address for the destination of the frame is set to 0xFFFF.

A Track is thus formed end-to-end as a succession of paired bundles, a receive bundle from the previous hop and a transmit bundle to the next hop along the Track, and a cell in such a bundle belongs to at most one Track. For a given iteration of the device schedule, the effective channel of the cell is obtained by adding a pseudo-random number to the channelOffset of the cell, which results in a rotation of the frequency that used for transmission. The bundles may be computed so as to accommodate both variable rates and retransmissions, so they might not be fully used at a given iteration of the schedule. The 6TiSCH architecture provides additional means to avoid waste of cells as well as overflows in the transmit bundle, as follows:

In one hand, a TX-cell that is not needed for the current iteration may be reused opportunistically on a per-hop basis for routed packets. When all of the frame that were received for a given Track are effectively transmitted, any available TX-cell for that Track can be reused for upper layer traffic for which the next-hop router matches the next hop along the Track. In that case, the cell that is being used is effectively a TX-cell from the Track, but the short address for the destination is that of the next-hop router. It results that a frame that is received in a RX-cell of a Track with a destination MAC address set to this node as opposed to broadcast must be extracted from the Track and delivered to the upper layer (a frame with an unrecognized MAC address is dropped at the lower MAC layer and thus is not received at the 6top sublayer).

On the other hand, it might happen that there are not enough TX-cells in the transmit bundle to accommodate the Track traffic, for instance if more retransmissions are needed than provisioned. In that case, the frame can be placed for transmission in the bundle that is used for layer-3 traffic towards the next hop along the track as long as it can be routed by the upper layer, that is, typically, if the frame transports an IPv6 packet. The MAC address should be set to the next-hop MAC address to avoid confusion. It results that a frame that is received over a layer-3 bundle may be in fact associated to a Track. In a classical IP link such as an Ethernet, off-track traffic is typically in excess over reservation to be routed along the non-reserved path based on its QoS setting. But with 6TiSCH, since the use of the layer-3 bundle may be due to transmission failures, it makes sense for the receiver to recognize a frame that should be re-tracked, and to place it back on the appropriate bundle if possible. A frame should be re-tracked if the Per-Hop-Behavior group indicated in the Differentiated Services Field in the IPv6 header is set to Deterministic Forwarding, as discussed in Section 8.3.1. A frame is re-tracked by scheduling it for transmission over the transmit bundle associated to the Track, with the destination MAC address set to broadcast.

There are 2 modes for a Track, transport mode and tunnel mode.

8.2.1.1. Transport Mode

In transport mode, the Protocol Data Unit (PDU) is associated with flow-dependant meta-data that refers uniquely to the Track, so the 6top sublayer can place the frame in the appropriate cell without ambiguity. In the case of IPv6 traffic, this flow identification is transported in the Flow Label of the IPv6 header. Associated with the source IPv6 address, the Flow Label forms a globally unique identifier for that particular Track that is validated at egress before restoring the destination MAC address (DMAC) and punting to the upper layer.

```
                       |                                    ^
   +--------------+    |                                    |
   |     IPv6     |    |                                    |
   +--------------+    |                                    |
   |  6LoWPAN HC  |    |                                    |
   +--------------+  ingress                              egress
   |     6top     |   sets     +----+          +----+     restores
   +--------------+  dmac to   |    |          |    |     dmac to
   |   TSCH MAC   |   brdcst   |    |          |    |      self
   +--------------+    |       |    |          |    |       |
   |   LLN PHY    |    +-------+    +--...-----+    +-------+
   +--------------+
```

Track Forwarding, Transport Mode

8.2.1.2. Tunnel Mode

In tunnel mode, the frames originate from an arbitrary protocol over a compatible MAC that may or may not be synchronized with the 6TiSCH network. An example of this would be a router with a dual radio that is capable of receiving and sending WirelessHART or ISA100.11a frames with the second radio, by presenting itself as an access Point or a Backbone Router, respectively.

In that mode, some entity (e.g. PCE) can coordinate with a WirelessHART Network Manager or an ISA100.11a System Manager to specify the flows that are to be transported transparently over the Track.

```
      +--------------+
      |     IPv6     |
      +--------------+
      |  6LoWPAN HC  |
      +--------------+             set             restore
      |     6top     |            +dmac+          +dmac+
      +--------------+          to|brdcst       to|nexthop
      |   TSCH MAC   |            |    |          |    |
      +--------------+            |    |          |    |
      |   LLN PHY    |    +-------+    +--...-----+    +-------+
      +--------------+    |   ingress                 egress   |
                          |                                    |
      +--------------+    |                                    |
      |   LLN PHY    |    |                                    |
      +--------------+    |                                    |
      |   TSCH MAC   |    |                                    |
      +--------------+    | dmac =                             | dmac =
      |ISA100/WiHART |    | nexthop                            v nexthop
      +--------------+
```

Figure 6: Track Forwarding, Tunnel Mode

In that case, the flow information that identifies the Track at the ingress 6TiSCH router is derived from the RX-cell. The dmac is set to this node but the flow information indicates that the frame must be tunneled over a particular Track so the frame is not passed to the upper layer. Instead, the dmac is forced to broadcast and the frame is passed to the 6top sublayer for switching.

At the egress 6TiSCH router, the reverse operation occurs. Based on metadata associated to the Track, the frame is passed to the appropriate link layer with the destination MAC restored.

8.2.1.3. Tunnel Metadata

Metadata coming with the Track configuration is expected to provide the destination MAC address of the egress endpoint as well as the tunnel mode and specific data depending on the mode, for instance a service access point for frame delivery at egress. If the tunnel egress point does not have a MAC address that matches the configuration, the Track installation fails.

In transport mode, if the final layer-3 destination is the tunnel termination, then it is possible that the IPv6 address of the destination is compressed at the 6LoWPAN sublayer based on the MAC address. It is thus mandatory at the ingress point to validate that the MAC address that was used at the 6LoWPAN sublayer for compression matches that of the tunnel egress point. For that reason, the node that injects a packet on a Track checks that the destination is effectively that of the tunnel egress point before it overwrites it to broadcast. The 6top sublayer at the tunnel egress point reverts that operation to the MAC address obtained from the tunnel metadata.

8.2.2. Fragment Forwarding

Considering that 6LoWPAN packets can be as large as 1280 bytes (the IPv6 MTU), and that the non-storing mode of RPL implies Source Routing that requires space for routing headers, and that a IEEE802.15.4 frame with security may carry in the order of 80 bytes of effective payload, an IPv6 packet might be fragmented into more than 16 fragments at the 6LoWPAN sublayer.

This level of fragmentation is much higher than that traditionally experienced over the Internet with IPv4 fragments, where fragmentation is already known as harmful.

In the case to a multihop route within a 6TiSCH network, Hop-by-Hop recomposition occurs at each hop in order to reform the packet and route it. This creates additional latency and forces intermediate nodes to store a portion of a packet for an undetermined time, thus impacting critical resources such as memory and battery.

[I-D.thubert-roll-forwarding-frags] describes a mechanism whereby the datagram tag in the 6LoWPAN Fragment is used as a label for switching at the 6LoWPAN sublayer. The draft allows for a degree of flow control based on an Explicit Congestion Notification, as well as end-to-end individual fragment recovery.

```
                          |                                    ^
      +--------------+    |                                    |
      |     IPv6     |    |       +----+          +----+       |
      +--------------+    |       |    |          |    |       |
      |  6LoWPAN HC  |    |       learn           learn        |
      +--------------+    |       |    |          |    |       |
      |     6top     |    |       |    |          |    |       |
      +--------------+    |       |    |          |    |       |
      |   TSCH MAC   |    |       |    |          |    |       |
      +--------------+    |       |    |          |    |       |
      |   LLN PHY    |    +-------+    +--...-----+    +-------+
      +--------------+
```

Figure 7: Forwarding First Fragment

In that model, the first fragment is routed based on the IPv6 header that is present in that fragment. The 6LoWPAN sublayer learns the next hop selection, generates a new datagram tag for transmission to the next hop, and stores that information indexed by the incoming MAC address and datagram tag. The next fragments are then switched based on that stored state.

```
                          |                                    ^
      +--------------+    |                                    |
      |     IPv6     |    |                                    |
      +--------------+    |                                    |
      |  6LoWPAN HC  |    |       replay          replay       |
      +--------------+    |       |    |          |    |       |
      |     6top     |    |       |    |          |    |       |
      +--------------+    |       |    |          |    |       |
      |   TSCH MAC   |    |       |    |          |    |       |
      +--------------+    |       |    |          |    |       |
      |   LLN PHY    |    +-------+    +--...-----+    +-------+
      +--------------+
```

Figure 8: Forwarding Next Fragment

A bitmap and an ECN echo in the end-to-end acknowledgment enable the source to resend the missing fragments selectively. The first fragment may be resent to carve a new path in case of a path failure. The ECN echo set indicates that the number of outstanding fragments should be reduced.

8.2.3. IPv6 Forwarding

As the packets are routed at Layer-3, traditional QoS and RED operations are expected to prioritize flows; the application of

Differentiated Services is further discussed in [I-D.svshah-tsvwg-lln-diffserv-recommendations].

```
                      |                                    ^
      +--------------+    |                                    |
      |     IPv6     |    |       +-QoS+          +-QoS+       |
      +--------------+    |       |    |          |    |       |
      |  6LoWPAN HC  |    |       |    |          |    |       |
      +--------------+    |       |    |          |    |       |
      |     6top     |    |       |    |          |    |       |
      +--------------+    |       |    |          |    |       |
      |   TSCH MAC   |    |       |    |          |    |       |
      +--------------+    |       |    |          |    |       |
      |   LLN PHY    |    +-------+    +--...-----+    +-------+
      +--------------+
```

Figure 9: IP Forwarding

8.3. Centralized vs. Distributed Routing

6TiSCH supports a mixed model of centralized routes and distributed routes. Centralized routes can for example be computed by a entity such as a PCE. Distributed routes are computed by RPL.

Both methods may inject routes in the Routing Tables of the 6TiSCH routers. In either case, each route is associated with a 6TiSCH topology that can be a RPL Instance topology or a track. The 6TiSCH topology is indexed by a Instance ID, in a format that reuses the RPLInstanceID as defined in RPL [RFC6550].

Both RPL and PCE rely on shared sources such as policies to define Global and Local RPLInstanceIDs that can be used by either method. It is possible for centralized and distributed routing to share a same topology. Generally they will operate in different slotFrames, and centralized routes will be used for scheduled traffic and will have precedence over distributed routes in case of conflict between the slotFrames.

8.3.1. Packet Marking and Handling

All packets inside a 6TiSCH domain must carry the Instance ID that identifies the 6TiSCH topology that is to be used for routing and forwarding that packet. The location of that information must be the same for all packets forwarded inside the domain.

For packets that are routed by a PCE along a Track, the tuple formed by the IPv6 source address and a local RPLInstanceID in the packet identify uniquely the Track and associated transmit bundle.

Additionally, an IP packet that is sent along a Track uses the Differentiated Services Per-Hop-Behavior Group called Deterministic Forwarding, as described in [I-D.svshah-tsvwg-deterministic-forwarding].

For packets that are routed by RPL, that information is the RPLInstanceID which is carried in the RPL Packet Information, as discussed in section 11.2 of [RFC6550], "Loop Avoidance and Detection".

The RPL Packet Information (RPI) is carried in IPv6 packets as a RPL option in the IPv6 Hop-By-Hop Header [RFC6553].

6Lo is currently considering a Next Header Compression (NHC) for the RPI (RPI-NHC). The RPI-NHC is specified in [I-D.thubert-6lo-rpl-nhc], and is the compressed equivalent to the whole HbH header with the RPL option.

An alternative form of compression that integrates the compression on IP-in-IP encapsulation and the Routing Header type 3 [RFC6554] with that of the RPI in a new 6LoWPAN dispatch/header type is concurrently being evaluated as [I-D.thubert-6lo-routing-dispatch].

Either way, the method and format used for encoding the RPLInstanceID is generalized to all 6TiSCH topological Instances, which include both RPL Instances and Tracks.

9. IANA Considerations

This specification does not require IANA action.

10. Security Considerations

This architecture operates on IEEE802.15.4 and expects link-layer security to be enabled at all times between connected devices, except for the very first step of the device join process, where a joining device may need some initial, unsecured exchanges so as to obtain its initial key material. Work has already started at the 6TiSCH Security Design Team and an overview of the current state of that work is presented in Section 10.1.

Future work on 6TiSCH security and will examine in deeper detail how to secure transactions end-to-end, and to maintain the security posture of a device over its lifetime. The result of that work will be described in a subsequent volume of this architecture.

10.1. Join Process Highlights

The architecture specifies three logical elements to describe the join process:

Joining Node (JN): Node that wishes to become part of the network;

Join Coordination Entity (JCE) : A Join Coordination Entity (JCE) that arbitrates network access and hands out network parameters (such as keying material);

Join Assistant (JA), a one-hop (radio) neighbor of the joining node that acts as proxy network node and may provide connectivity with the JCE.

The join protocol consists of three major activities:

Device Authentication: The JN and the JA mutually authenticate each other and establish a shared key, so as to ensure on-going authenticated communications. This may involve a server as a third party.

Authorization: The JA decides on whether/how to authorize a JN (if denied, this may result in loss of bandwidth). Conversely, the JN decides on whether/how to authorize the network (if denied, it will not join the network). Authorization decisions may involve other nodes in the network.

Configuration/Parameterization: The JA distributes configuration information to the JN, such as scheduling information, IP address assignment information, and network policies. This may originate from other network devices, for which the JA may act as proxy. This step may also include distribution of information from the JN to the JA and other nodes in the network and, more generally, synchronization of information between these entities.

The device joining process is depicted in Figure 10, where it is assumed that devices have access to certificates and where entities have access to the root CA keys of their communicating parties (initial set-up requirement). Under these assumptions, the authentication step of the device joining process does not require online involvement of a third party. Mutual authentication is performed between the JN and the JA using their certificates, which also results in a shared key between these two entities.

The JA assists the JN in mutual authentication with a remote server node (primarily via provision of a communication path with the server), which also results in a shared (end-to-end) key between those two entities. The server node may be a JCE that arbitrages the network authorization of the JN (where the JA will deny bandwidth if authorization is not successful); it may distribute network-specific configuration parameters (including network-wide keys) to the JN. In its turn, the JN may distribute and synchronize information (including, e.g., network statistics) to the server node and, if so desired, also to the JA. The actual decision of the JN to become part of the network may depend on authorization of the network itself.

The server functionality is a role which may be implemented with one (centralized) or multiple devices (distributed). In either case, mutual authentication is established with each physical server entity with which a role is implemented.

Note that in the above description, the JA does not solely act as a relay node, thereby allowing it to first filter traffic to be relayed based on cryptographic authentication criteria - this provides first-level access control and mitigates certain types of denial-of-service attacks on the network at large.

Depending on more detailed insight in cost/benefit trade-offs, this process might be complemented by a more "relaxed" mechanism, where the JA acts as a relay node only. The final architecture will provide mechanisms to also cover cases where the initial set-up requirements are not met or where some other out-of-sync behavior occurs; it will also suggest some optimizations in case JCE-related information is already available with the JA (via caching of information).

When a device rejoins the network in the same authorization domain, the authorization step could be omitted if the server distributes the authorization state for the device to the JA when the device initially joined the network. However, this generally still requires the exchange of updated configuration information, e.g., related to time schedules and bandwidth allocation.

```
   {joining node}     {neighbor}               {server, etc.}   Example:
   +---------+        +---------+                 +---------+
   | Joining |        |  Join   |              +--|    CA   |certificate
   |  Node   |        |Assistant|              |  +---------+   issuance
   +---------+        +---------+              |  +---------+
      |                    |                   +--|Authoriz.| membership
      |<----Beaconing------|                   |  +---------+ test (JCE)
      |                    |                   |  +---------+
      |<--Authentication-->|                   +--| Routing | IP address
      |                    |<--Authorization-->|  +---------  assignment
      |<-------------------|                   |  +---------+
      |                    |                   +--| Gateway | backbone,
      |------------------->|                   |  +---------+    cloud
      |                    |<--Configuration-->|  +---------+
      |<-------------------|                   +--|Bandwidth|  PCE
                                                  +---------+  schedule
       .                    .                   .
       .                    .                   .
```

Figure 10: Network joining, with only authorization by third party

11. Acknowledgments

11.1. Contributors

The co-authors of this document are listed below:

Robert Assimiti for his breakthrough work on RPL over TSCH and initial text and guidance.

Kris Pister for creating it all and his continuing guidance through the elaboration of this design.

Michael Richardson for his leadership role in the Security Design Team and his contribution throughout this document.

Rene Struik for the security section and his contribution to the Security Design Team.

Xavier Vilajosana who lead the design of the minimal support with RPL and contributed deeply to the 6top design and the G-MPLS operation of track switching.

Qin Wang who lead the design of the 6top sublayer and contributed related text that was moved and/or adapted in this document.

Thomas Watteyne for his contribution to the whole design, in particular on TSCH and security.

11.2. Special Thanks

Special thanks to Tero Kivinen, Jonathan Simon, Giuseppe Piro, Subir Das and Yoshihiro Ohba for their deep contribution to the initial security work, and to Diego Dujovne for starting and leading the On-the-Fly effort.

Special thanks also to Pat Kinney for his support in maintaining the connection active and the design in line with work happening at IEEE802.15.4.

Also special thanks to Ted Lemon who was the INT Area A-D while this specification was developed for his great support and help throughout.

11.3. And Do not Forget

This specification is the result of multiple interactions, in particular during the 6TiSCH (bi)Weekly Interim call, relayed through the 6TiSCH mailing list at the IETF.

The authors wish to thank: Alaeddine Weslati, Chonggang Wang, Georgios Exarchakos, Zhuo Chen, Alfredo Grieco, Bert Greevenbosch, Cedric Adjih, Deji Chen, Martin Turon, Dominique Barthel, Elvis Vogli, Geraldine Texier, Malisa Vucinic, Guillaume Gaillard, Herman Storey, Kazushi Muraoka, Ken Bannister, Kuor Hsin Chang, Laurent Toutain, Maik Seewald, Maria Rita Palattella, Michael Behringer, Nancy Cam Winget, Nicola Accettura, Nicolas Montavont, Oleg Hahm, Patrick Wetterwald, Paul Duffy, Peter van der Stock, Rahul Sen, Pieter de Mil, Pouria Zand, Rouhollah Nabati, Rafa Marin-Lopez, Raghuram Sudhaakar, Sedat Gormus, Shitanshu Shah, Steve Simlo, Tengfei Chang, Tina Tsou, Tom Phinney, Xavier Lagrange, Ines Robles and Samita Chakrabarti for their participation and various contributions.

12. References

12.1. Normative References

[I-D.ietf-6tisch-terminology]
Palattella, M., Thubert, P., Watteyne, T., and Q. Wang, "Terminology in IPv6 over the TSCH mode of IEEE 802.15.4e", draft-ietf-6tisch-terminology-04 (work in progress), March 2015.

[I-D.ietf-6tisch-tsch]
Watteyne, T., Palattella, M., and L. Grieco, "Using IEEE802.15.4e TSCH in an IoT context: Overview, Problem Statement and Goals", draft-ietf-6tisch-tsch-06 (work in progress), March 2015.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.

[RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, September 2007.

[RFC6282] Hui, J. and P. Thubert, "Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks", RFC 6282, September 2011.

[RFC6550] Winter, T., Thubert, P., Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, JP., and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks", RFC 6550, March 2012.

[RFC6552] Thubert, P., "Objective Function Zero for the Routing Protocol for Low-Power and Lossy Networks (RPL)", RFC 6552, March 2012.

[RFC6553] Hui, J. and JP. Vasseur, "The Routing Protocol for Low-Power and Lossy Networks (RPL) Option for Carrying RPL Information in Data-Plane Datagrams", RFC 6553, March 2012.

[RFC6554] Hui, J., Vasseur, JP., Culler, D., and V. Manral, "An IPv6 Routing Header for Source Routes with the Routing Protocol for Low-Power and Lossy Networks (RPL)", RFC 6554, March 2012.

[RFC6775] Shelby, Z., Chakrabarti, S., Nordmark, E., and C. Bormann, "Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)", RFC 6775, November 2012.

12.2. Informative References

[I-D.chakrabarti-nordmark-6man-efficient-nd]
Chakrabarti, S., Nordmark, E., Thubert, P., and M. Wasserman, "IPv6 Neighbor Discovery Optimizations for Wired and Wireless Networks", draft-chakrabarti-nordmark-6man-efficient-nd-07 (work in progress), February 2015.

[I-D.dujovne-6tisch-on-the-fly]
Dujovne, D., Grieco, L., Palattella, M., and N. Accettura, "6TiSCH On-the-Fly Scheduling", draft-dujovne-6tisch-on-the-fly-05 (work in progress), March 2015.

[I-D.finn-detnet-architecture]
Finn, N., Thubert, P., and M. Teener, "Deterministic Networking Architecture", draft-finn-detnet-architecture-01 (work in progress), March 2015.

[I-D.ietf-6tisch-6top-interface]
Wang, Q., Vilajosana, X., and T. Watteyne, "6TiSCH Operation Sublayer (6top) Interface", draft-ietf-6tisch-6top-interface-03 (work in progress), March 2015.

[I-D.ietf-6tisch-coap]
Sudhaakar, R. and P. Zand, "6TiSCH Resource Management and Interaction using CoAP", draft-ietf-6tisch-coap-03 (work in progress), March 2015.

[I-D.ietf-6tisch-minimal]
Vilajosana, X. and K. Pister, "Minimal 6TiSCH Configuration", draft-ietf-6tisch-minimal-06 (work in progress), March 2015.

[I-D.ietf-ipv6-multilink-subnets]
Thaler, D. and C. Huitema, "Multi-link Subnet Support in IPv6", draft-ietf-ipv6-multilink-subnets-00 (work in progress), July 2002.

[I-D.ietf-roll-rpl-industrial-applicability]
Phinney, T., Thubert, P., and R. Assimiti, "RPL applicability in industrial networks", draft-ietf-roll-rpl-industrial-applicability-02 (work in progress), October 2013.

[I-D.richardson-6tisch-security-architecture]
Richardson, M., "security architecture for 6top: requirements and structure", draft-richardson-6tisch-security-architecture-02 (work in progress), April 2014.

[I-D.struik-6tisch-security-architecture-elements]
Struik, R., Ohba, Y., and S. Das, "6TiSCH Security Architectural Elements, Desired Protocol Properties, and Framework", draft-struik-6tisch-security-architecture-elements-01 (work in progress), October 2014.

[I-D.svshah-tsvwg-deterministic-forwarding]
Shah, S. and P. Thubert, "Deterministic Forwarding PHB", draft-svshah-tsvwg-deterministic-forwarding-03 (work in progress), March 2015.

[I-D.svshah-tsvwg-lln-diffserv-recommendations]
Shah, S. and P. Thubert, "Differentiated Service Class Recommendations for LLN Traffic", draft-svshah-tsvwg-lln-diffserv-recommendations-04 (work in progress), February 2015.

[I-D.thubert-6lo-rfc6775-update-reqs]
Thubert, P. and P. Stok, "Requirements for an update to 6LoWPAN ND", draft-thubert-6lo-rfc6775-update-reqs-06 (work in progress), January 2015.

[I-D.thubert-6lo-routing-dispatch]
Thubert, P., Bormann, C., Toutain, L., and R. Cragie, "A Routing Header Dispatch for 6LoWPAN", draft-thubert-6lo-routing-dispatch-03 (work in progress), January 2015.

[I-D.thubert-6lo-rpl-nhc]
Thubert, P. and C. Bormann, "A compression mechanism for the RPL option", draft-thubert-6lo-rpl-nhc-02 (work in progress), October 2014.

[I-D.thubert-6lowpan-backbone-router]
Thubert, P., "6LoWPAN Backbone Router", draft-thubert-6lowpan-backbone-router-03 (work in progress), February 2013.

[I-D.thubert-roll-forwarding-frags]
Thubert, P. and J. Hui, "LLN Fragment Forwarding and Recovery", draft-thubert-roll-forwarding-frags-02 (work in progress), September 2013.

[I-D.vanderstok-core-comi]
Stok, P., Greevenbosch, B., Bierman, A., Schoenwaelder, J., and A. Sehgal, "CoAP Management Interface", draft-vanderstok-core-comi-06 (work in progress), February 2015.

[I-D.wang-6tisch-6top-sublayer]
Wang, Q., Vilajosana, X., and T. Watteyne, "6TiSCH Operation Sublayer (6top)", draft-wang-6tisch-6top-sublayer-01 (work in progress), July 2014.

[RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, December 1998.

[RFC2545] Marques, P. and F. Dupont, "Use of BGP-4 Multiprotocol Extensions for IPv6 Inter-Domain Routing", RFC 2545, March 1999.

[RFC3209] Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V., and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels", RFC 3209, December 2001.

[RFC3444] Pras, A. and J. Schoenwaelder, "On the Difference between Information Models and Data Models", RFC 3444, January 2003.

[RFC3610] Whiting, D., Housley, R., and N. Ferguson, "Counter with CBC-MAC (CCM)", RFC 3610, September 2003.

[RFC3963] Devarapalli, V., Wakikawa, R., Petrescu, A., and P. Thubert, "Network Mobility (NEMO) Basic Support Protocol", RFC 3963, January 2005.

[RFC3971] Arkko, J., Kempf, J., Zill, B., and P. Nikander, "SEcure Neighbor Discovery (SEND)", RFC 3971, March 2005.

[RFC4080] Hancock, R., Karagiannis, G., Loughney, J., and S. Van den Bosch, "Next Steps in Signaling (NSIS): Framework", RFC 4080, June 2005.

[RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, February 2006.

[RFC4389] Thaler, D., Talwar, M., and C. Patel, "Neighbor Discovery Proxies (ND Proxy)", RFC 4389, April 2006.

[RFC4429] Moore, N., "Optimistic Duplicate Address Detection (DAD) for IPv6", RFC 4429, April 2006.

[RFC4903] Thaler, D., "Multi-Link Subnet Issues", RFC 4903, June 2007.

[RFC4919] Kushalnagar, N., Montenegro, G., and C. Schumacher, "IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals", RFC 4919, August 2007.

[RFC5191] Forsberg, D., Ohba, Y., Patil, B., Tschofenig, H., and A. Yegin, "Protocol for Carrying Authentication for Network Access (PANA)", RFC 5191, May 2008.

[RFC5340] Coltun, R., Ferguson, D., Moy, J., and A. Lindem, "OSPF for IPv6", RFC 5340, July 2008.

[RFC5889] Baccelli, E. and M. Townsley, "IP Addressing Model in Ad Hoc Networks", RFC 5889, September 2010.

[RFC5974] Manner, J., Karagiannis, G., and A. McDonald, "NSIS Signaling Layer Protocol (NSLP) for Quality-of-Service Signaling", RFC 5974, October 2010.

[RFC6275] Perkins, C., Johnson, D., and J. Arkko, "Mobility Support in IPv6", RFC 6275, July 2011.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, January 2012.

[RFC6620] Nordmark, E., Bagnulo, M., and E. Levy-Abegnoli, "FCFS SAVI: First-Come, First-Served Source Address Validation Improvement for Locally Assigned IPv6 Addresses", RFC 6620, May 2012.

[RFC6655] McGrew, D. and D. Bailey, "AES-CCM Cipher Suites for Transport Layer Security (TLS)", RFC 6655, July 2012.

[RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The Locator/ID Separation Protocol (LISP)", RFC 6830, January 2013.

12.3. Other Informative References

[ACE] IETF, "Authentication and Authorization for Constrained Environments", <https://datatracker.ietf.org/doc/charter-ietf-ace/>.

[CCAMP] IETF, "Common Control and Measurement Plane", <https://datatracker.ietf.org/doc/charter-ietf-ccamp/>.

[DICE] IETF, "DTLS In Constrained Environments", <https://datatracker.ietf.org/doc/charter-ietf-dice/>.

[HART] www.hartcomm.org, "Highway Addressable remote Transducer, a group of specifications for industrial process and control devices administered by the HART Foundation".

[IEEE802.1TSNTG]
IEEE Standards Association, "IEEE 802.1 Time-Sensitive Networks Task Group", March 2013, <http://www.ieee802.org/1/pages/avbridges.html>.

[IEEE802154]
IEEE standard for Information Technology, "IEEE std. 802.15.4, Part. 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks".

[IEEE802154e]
IEEE standard for Information Technology, "IEEE standard for Information Technology, IEEE std. 802.15.4, Part. 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks, June 2011 as amended by IEEE std. 802.15.4e, Part. 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) Amendment 1: MAC sublayer", April 2012.

[ISA100] ISA/ANSI, "ISA100, Wireless Systems for Automation", <https://www.isa.org/isa100/>.

[ISA100.11a]
ISA/ANSI, "Wireless Systems for Industrial Automation: Process Control and Related Applications - ISA100.11a-2011 - IEC 62734", 2011, <http://www.isa.org/Community/SP100WirelessSystemsforAutomation>.

[PCE] IETF, "Path Computation Element", <https://datatracker.ietf.org/doc/charter-ietf-pce/>.

[TEAS] IETF, "Traffic Engineering Architecture and Signaling", <https://datatracker.ietf.org/doc/charter-ietf-teas/>.

[WirelessHART]
www.hartcomm.org, "Industrial Communication Networks - Wireless Communication Network and Communication Profiles - WirelessHART - IEC 62591", 2010.

Appendix A. Personal submissions relevant to the next volumes

This volume only covers a portion of the total work that is needed to cover the full 6TiSCH architecture. Missing portions include Deterministic Networking with Track Forwarding, Dynamic Scheduling, and Security.

[I-D.richardson-6tisch-security-architecture] elaborates on the potential use of 802.1AR certificates, and some options for the join process are presented in more details.

[I-D.struik-6tisch-security-architecture-elements] describes 6TiSCH security architectural elements with high level requirements and the security framework that are relevant for the design of the 6TiSCH security solution.

[I-D.dujovne-6tisch-on-the-fly] discusses the use of the 6top sublayer [I-D.wang-6tisch-6top-sublayer] to adapt dynamically the number of cells between a RPL parent and a child to the needs of the actual traffic.

Author’s Address

Pascal Thubert (editor)
Cisco Systems, Inc
Building D
45 Allee des Ormes - BP1200
MOUGINS - Sophia Antipolis 06254
FRANCE

Phone: +33 497 23 26 34
Email: pthubert@cisco.com

6TiSCH   J. Munoz, Ed.
Internet-Draft   Telecom Bretagne
Intended status: Informational   G. Gaillard
Expires: January 7, 2016   D. Barthel
                                                             Orange Labs
                                                            July 6, 2015

Example Packets for the Minimal 6TiSCH Configuration
draft-munoz-6tisch-minimal-examples-00

Abstract

This draft contains example packets exchanged by nodes implementing draft-ietf-6tisch-minimal. All packets are presented both in raw binary and fully parsed contents. This document can be used as a reference when implementing draft-ietf-6tisch-minimal.

Status of This Memo

Copyright Notice

Table of Contents

1. Tools Used

All results presented in this document are collected by running the OpenWSN firmware [OpenWSN] in simulation mode and capturing the packets exchanged using a version of Wireshark with an updated IEEE802.15.4e TSCH/6TiSCH dissector.

These are the version of the source code used:

1. Wireshark dissector: https://github.com/openwsn-berkeley/dissectors/releases/tag/6tisch_0.3

2. OpenWSN firmware: https://github.com/openwsn-berkeley/openwsn-fw/commit/ad2af054b245eeccc76746f42806414a96076ca0

3. OpenWSN software: https://github.com/openwsn-berkeley/openwsn-sw/commit/64134308ab708463ec2c3707caf0207e3c46f20c

2. Network Topology

```
   Network prefix: bbbb::/64
   MAC address:    14-15-92-cc-00-00-00-0x

           PDR=100%       PDR=100%
    +-----+        +-----+        +-----+
    | x=1 |--------| x=2 |--------| x=3 |
    +-----+        +-----+        +-----+
    DAGroot
```

3. Packet Examples

3.1. Known Errors in These examples

Looks for "FIXME" in the examples below.

3.2. Enhanced Beacon

Enhanced Beacon sent by 1

```
== Dissected packet ==

IEEE 802.15.4 Beacon, Dst: Broadcast, Src: 14:15:92:cc:00:00:00:01
Header Termination 1 IE Beacon
    Frame Control Field: 0xea00, Frame Type: Beacon, Information
    Elements present, Destination Addressing Mode: Short/16-bit,
    Source Addressing Mode: Long/64-bit
        .... .... .... .000 = Frame Type: Beacon (0x0000)
        .... .... .... 0... = Security Enabled: False
        .... .... ...0 .... = Frame Pending: False
        .... .... ..0. .... = Acknowledge Request: False
        .... .... .0.. .... = Intra-PAN: False
        .... ...0 .... .... = Sequence Number Suppression: False
        .... ..1. .... .... = Information Elements present: True
        .... 10.. .... .... = Destination Addressing Mode:
                              Short/16-bit (0x0002)
        ..10 .... .... .... = Frame Version: 2
        11.. .... .... .... = Source Addressing Mode:
                              Long/64-bit (0x0003)
    Sequence Number: 67
    Destination PAN: 0xcafe
    Destination: 0xffff
    Extended Source: 14:15:92:cc:00:00:00:01 (14:15:92:cc:00:00:00:01)
    Header Information Elements: Header Termination 1 IE (0x3f00)
        Information Element Length: 0x00
        .011 1111 0... .... = Information Element ID:
                              Header Termination 1 IE (0x007e)
```

```
        0... .... .... .... = Payload IE: TRUE - Header IE: FALSE: False
    Payload Information Elements: MLME IE (0x881a) TSCH Synchronization
    IE TSCH Timeslot IE Channel Hopping IE TSCH Slotframe and Link IE
        Information Element Length: 0x001a
        .000 1... .... .... = Information Element ID: MLME IE (0x0001)
        1... .... .... .... = Payload IE: TRUE - Header IE: FALSE: True
        Short MLME Information Elements: TSCH Synchronization
                                         IE (0x1a06)
            MLME Short Information Element Length: 0x0006
            .001 1010 .... .... = MLME Short Information Element ID:
                                  TSCH Synchronization IE (0x001a)
            0... .... .... .... = MLME Short Information Element type.
                                  Short: FALSE - Long: TRUE: False
        Data (6 bytes)

0000  4c 7a 01 00 00 00                                 Lz....
            Data: 4c7a01000000
            [Length: 6]
        Short MLME Information Elements: TSCH Timeslot IE (0x1c01)
            MLME Short Information Element Length: 0x0001
            .001 1100 .... .... = MLME Short Information Element ID:
                                  TSCH Timeslot IE (0x001c)
            0... .... .... .... = MLME Short Information Element type.
                                  Short: FALSE - Long: TRUE: False
        Data (1 byte)

0000  00                                                .
            Data: 00
            [Length: 1]
        Long MLME Information Elements: Channel Hopping IE (0xc801)
            MLME Long Information Element Length: 0x0001
            .100 1... .... .... = MLME Long Information Element ID:
                                  Channel Hopping IE (0x0009)
            1... .... .... .... = MLME Short Information Element type.
                                  Short: FALSE - Long: TRUE: True
        Data (1 byte)

0000  00                                                .
            Data: 00
            [Length: 1]
        Short MLME Information Elements: TSCH Slotframe and
                                         Link IE (0x1b0a)
            MLME Short Information Element Length: 0x000a
            .001 1011 .... .... = MLME Short Information Element ID:
                                  TSCH Slotframe and Link IE (0x001b)
            0... .... .... .... = MLME Short Information Element type.
                                  Short: FALSE - Long: TRUE: False
        Data (10 bytes)
```

```
0000  01 01 0b 00 01 00 00 00 00 0f                     ..........
            Data: 01010b0001000000000f
            [Length: 10]
    FCS: 0xfae3 (Correct)

== Raw Bytes ==

0000   00 ea 43 fe ca ff ff 01 00 00 00 cc 92 15 14 00  ..C.............
0010   3f 1a 88 06 1a 4c 7a 01 00 00 00 01 1c 00 01 c8  ?....Lz.........
0020   00 0a 1b 01 01 0b 00 01 00 00 00 00 0f e3 fa     ...............
```

   Enhanced Beacon sent by 2

```
== Dissected packet ==

IEEE 802.15.4 Beacon, Dst: Broadcast, Src: 14:15:92:cc:00:00:00:02
Header Termination 1 IE Beacon
    Frame Control Field: 0xea00, Frame Type: Beacon, Information
    Elements present, Destination Addressing Mode: Short/16-bit,
    Source Addressing Mode: Long/64-bit
        .... .... .... .000 = Frame Type: Beacon (0x0000)
        .... .... .... 0... = Security Enabled: False
        .... .... ...0 .... = Frame Pending: False
        .... .... ..0. .... = Acknowledge Request: False
        .... .... .0.. .... = Intra-PAN: False
        .... ...0 .... .... = Sequence Number Suppression: False
        .... ..1. .... .... = Information Elements present: True
        .... 10.. .... .... = Destination Addressing Mode:
                              Short/16-bit (0x0002)
        ..10 .... .... .... = Frame Version: 2
        11.. .... .... .... = Source Addressing Mode:
                              Long/64-bit (0x0003)
    Sequence Number: 229
    Destination PAN: 0xcafe
    Destination: 0xffff
    Extended Source: 14:15:92:cc:00:00:00:02 (14:15:92:cc:00:00:00:02)
    Header Information Elements: Header Termination 1 IE (0x3f00)
        Information Element Length: 0x00
        .011 1111 0... .... = Information Element ID:
                              Header Termination 1 IE (0x007e)
        0... .... .... .... = Payload IE: TRUE - Header IE: FALSE: False
    Payload Information Elements: MLME IE (0x881a) TSCH Synchronization
    IE TSCH Timeslot IE Channel Hopping IE TSCH Slotframe and Link IE
        Information Element Length: 0x001a
        .000 1... .... .... = Information Element ID: MLME IE (0x0001)
        1... .... .... .... = Payload IE: TRUE - Header IE: FALSE: True
        Short MLME Information Elements: TSCH Synchronization IE
                                         (0x1a06)
```

```
            MLME Short Information Element Length: 0x0006
            .001 1010 .... .... = MLME Short Information Element ID:
                                  TSCH Synchronization IE (0x001a)
            0... .... .... .... = MLME Short Information Element type.
                                  Short: FALSE - Long: TRUE: False
        Data (6 bytes)

0000  1a 7c 01 00 00 02                                 .|....
            Data: 1a7c01000002
            [Length: 6]
        Short MLME Information Elements: TSCH Timeslot IE (0x1c01)
            MLME Short Information Element Length: 0x0001
            .001 1100 .... .... = MLME Short Information Element ID:
                                  TSCH Timeslot IE (0x001c)
            0... .... .... .... = MLME Short Information Element type.
                                  Short: FALSE - Long: TRUE: False
        Data (1 byte)

0000  00                                                .
            Data: 00
            [Length: 1]
        Long MLME Information Elements: Channel Hopping IE (0xc801)
            MLME Long Information Element Length: 0x0001
            .100 1... .... .... = MLME Long Information Element ID:
                                  Channel Hopping IE (0x0009)
            1... .... .... .... = MLME Short Information Element type.
                                  Short: FALSE - Long: TRUE: True
        Data (1 byte)

0000  00                                                .
            Data: 00
            [Length: 1]
        Short MLME Information Elements: TSCH Slotframe and
                                         Link IE (0x1b0a)
            MLME Short Information Element Length: 0x000a
            .001 1011 .... .... = MLME Short Information Element ID:
                                  TSCH Slotframe and Link IE (0x001b)
            0... .... .... .... = MLME Short Information Element type.
                                  Short: FALSE - Long: TRUE: False
        Data (10 bytes)

0000  01 01 0b 00 01 00 00 00 00 0f                     ..........
            Data: 01010b0001000000000f
            [Length: 10]
    FCS: 0x89c4 (Correct)

== Raw Bytes ==
```

```
0000   00 ea e5 fe ca ff ff 02 00 00 00 cc 92 15 14 00  ................
0010   3f 1a 88 06 1a 1a 7c 01 00 00 02 01 1c 00 01 c8  ?.....|.........
0020   00 0a 1b 01 01 0b 00 01 00 00 00 00 0f c4 89     ...............
```

Enhanced Beacon sent by 3

```
== Dissected packet ==

IEEE 802.15.4 Beacon, Dst: Broadcast, Src: 14:15:92:cc:00:00:00:03
Header Termination 1 IE Beacon
    Frame Control Field: 0xea00, Frame Type: Beacon, Information
    Elements present, Destination Addressing Mode: Short/16-bit,
    Source Addressing Mode: Long/64-bit
        .... .... .... .000 = Frame Type: Beacon (0x0000)
        .... .... .... 0... = Security Enabled: False
        .... .... ...0 .... = Frame Pending: False
        .... .... ..0. .... = Acknowledge Request: False
        .... .... .0.. .... = Intra-PAN: False
        .... ...0 .... .... = Sequence Number Suppression: False
        .... ..1. .... .... = Information Elements present: True
        .... 10.. .... .... = Destination Addressing Mode:
                              Short/16-bit (0x0002)
        ..10 .... .... .... = Frame Version: 2
        11.. .... .... .... = Source Addressing Mode:
                              Long/64-bit (0x0003)
    Sequence Number: 105
    Destination PAN: 0xcafe
    Destination: 0xffff
    Extended Source: 14:15:92:cc:00:00:00:03 (14:15:92:cc:00:00:00:03)
    Header Information Elements: Header Termination 1 IE (0x3f00)
        Information Element Length: 0x00
        .011 1111 0... .... = Information Element ID:
                              Header Termination 1 IE (0x007e)
        0... .... .... .... = Payload IE: TRUE - Header IE: FALSE: False
    Payload Information Elements: MLME IE (0x881a)
    TSCH Synchronization IE TSCH Timeslot IE
    Channel Hopping IE TSCH Slotframe and Link IE
        Information Element Length: 0x001a
        .000 1... .... .... = Information Element ID: MLME IE (0x0001)
        1... .... .... .... = Payload IE: TRUE - Header IE: FALSE: True
        Short MLME Information Elements:
        TSCH Synchronization IE (0x1a06)
            MLME Short Information Element Length: 0x0006
            .001 1010 .... .... = MLME Short Information Element ID:
                                  TSCH Synchronization IE (0x001a)
            0... .... .... .... = MLME Short Information Element type.
                                  Short: FALSE - Long: TRUE: False
        Data (6 bytes)
```

```
0000  5e 7f 01 00 00 04                                ^.....
            Data: 5e7f01000004
            [Length: 6]
        Short MLME Information Elements: TSCH Timeslot IE (0x1c01)
            MLME Short Information Element Length: 0x0001
            .001 1100 .... .... = MLME Short Information Element ID:
                                  TSCH Timeslot IE (0x001c)
            0... .... .... .... = MLME Short Information Element type.
                                  Short: FALSE - Long: TRUE: False
        Data (1 byte)

0000  00                                               .
            Data: 00
            [Length: 1]
        Long MLME Information Elements: Channel Hopping IE (0xc801)
            MLME Long Information Element Length: 0x0001
            .100 1... .... .... = MLME Long Information Element ID:
                                  Channel Hopping IE (0x0009)
            1... .... .... .... = MLME Short Information Element type.
                                  Short: FALSE - Long: TRUE: True
        Data (1 byte)

0000  00                                               .
            Data: 00
            [Length: 1]
        Short MLME Information Elements: TSCH Slotframe and
                                         Link IE (0x1b0a)
            MLME Short Information Element Length: 0x000a
            .001 1011 .... .... = MLME Short Information Element ID:
                                  TSCH Slotframe and Link IE (0x001b)
            0... .... .... .... = MLME Short Information Element type.
                                  Short: FALSE - Long: TRUE: False
        Data (10 bytes)

0000  01 01 0b 00 01 00 00 00 00 0f                    ..........
            Data: 01010b0001000000000f
            [Length: 10]
    FCS: 0x47f0 (Correct)

== Raw Bytes ==

0000   00 ea 69 fe ca ff ff 03 00 00 00 cc 92 15 14 00  ..i.............
0010   3f 1a 88 06 1a 5e 7f 01 00 00 04 01 1c 00 01 c8  ?....^..........
0020   00 0a 1b 01 01 0b 00 01 00 00 00 00 0f f0 47     ..............G
```

3.3. RPL DIO

RPL DIO sent by 1

```
== Dissected packet ==

IEEE 802.15.4 Data, Dst: Broadcast, Src: 14:15:92:cc:00:00:00:01
    Frame Control Field: 0xe801, Frame Type: Data,
    Destination Addressing Mode: Short/16-bit,
    Source Addressing Mode: Long/64-bit
        .... .... .... .001 = Frame Type: Data (0x0001)
        .... .... .... 0... = Security Enabled: False
        .... .... ...0 .... = Frame Pending: False
        .... .... ..0. .... = Acknowledge Request: False
        .... .... .0.. .... = Intra-PAN: False
        .... ...0 .... .... = Sequence Number Suppression: False
        .... ..0. .... .... = Information Elements present: False
        .... 10.. .... .... = Destination Addressing Mode:
                              Short/16-bit (0x0002)
        ..10 .... .... .... = Frame Version: 2
        11.. .... .... .... = Source Addressing Mode:
                              Long/64-bit (0x0003)
    Sequence Number: 157
    Destination PAN: 0xcafe
    Destination: 0xffff
    Extended Source: 14:15:92:cc:00:00:00:01 (14:15:92:cc:00:00:00:01)
    FCS: 0x529a (Correct)
6LoWPAN
    IPHC Header
        011. .... = Pattern: IP header compression (0x03)
        ...1 1... .... .... = Traffic class and flow label: Version,
                              traffic class, and flow label
                              compressed (0x0003)
        .... .0.. .... .... = Next header: Inline
        .... ..00 .... .... = Hop limit: Inline (0x0000)
        .... .... 0... .... = Context identifier extension: False
        .... .... .0.. .... = Source address compression: Stateless
        .... .... ..11 .... = Source address mode: Compressed (0x0003)
        .... .... .... 1... = Multicast address compression: True
        .... .... .... .0.. = Destination address compression: Stateless
        .... .... .... ..11 = Destination address mode:
                              8-bits inline (0x0003)
        [Source context: fe80:: (fe80::)]
        [Destination context: fe80:: (fe80::)]
    Next header: ICMPv6 (0x3a)
    Hop limit: 64
    Source: fe80::1615:92cc:0:1 (fe80::1615:92cc:0:1)
    Destination: ff02::1a (ff02::1a)
```

```
Internet Protocol Version 6, Src: fe80::1615:92cc:0:1
(fe80::1615:92cc:0:1), Dst: ff02::1a (ff02::1a)
    0110 .... = Version: 6
        [0110 .... = This field makes the filter
                     "ip.version == 6" possible: 6]
    .... 0000 0000 .... .... .... .... .... = Traffic class: 0x00000000
        .... 0000 00.. .... .... .... .... .... = Differentiated
                                                  Services Field:
                                                  Default (0x00000000)
        .... .... ..0. .... .... .... .... .... = ECN-Capable Transport
                                                  (ECT): Not set
        .... .... ...0 .... .... .... .... .... = ECN-CE: Not set
    .... .... .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
    Payload length: 28
    Next header: ICMPv6 (58)
    Hop limit: 64
    Source: fe80::1615:92cc:0:1 (fe80::1615:92cc:0:1)
    Destination: ff02::1a (ff02::1a)
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
Internet Control Message Protocol v6
    Type: RPL Control (155)
    Code: 1 (DODAG Information Object)
    Checksum: 0x171b [incorrect, should be 0xd255]
        [Expert Info (Warn/Checksum): ICMPv6 Checksum Incorrect,
                                      should be 0xd255]
            [ICMPv6 Checksum Incorrect, should be 0xd255]
            [Severity level: Warn]
            [Group: Checksum]
    RPLInstanceID: 0
    Version: 0
    Rank: 256
    Flags: 0x88
        1... .... = Grounded (G): True
        .0.. .... = Zero: False
        ..00 1... = Mode of Operation (MOP): Non-Storing Mode of
                                             Operation (0x01)
        .... .000 = DODAG Preference: 0
    Destination Advertisement Trigger Sequence Number (DTSN): 51
    Flags: 0x00
    Reserved: 00
    DODAGID: bbbb::1415:92cc:0:1 (bbbb::1415:92cc:0:1)

== Raw Bytes ==

0000   01 e8 9d fe ca ff ff 01 00 00 00 cc 92 15 14 78  ...............x
0010   3b 3a 40 1a 9b 01 17 1b 00 00 01 00 88 33 00 00  ;:@..........3..
0020   bb bb 00 00 00 00 00 00 14 15 92 cc 00 00 00 01  ................
```

```
0030   9a 52                                               .R
```

RPL DIO sent by 2

== Dissected packet ==

```
IEEE 802.15.4 Data, Dst: Broadcast, Src: 14:15:92:cc:00:00:00:02
    Frame Control Field: 0xe801, Frame Type: Data,
    Destination Addressing Mode: Short/16-bit,
    Source Addressing Mode: Long/64-bit
        .... .... .... .001 = Frame Type: Data (0x0001)
        .... .... .... 0... = Security Enabled: False
        .... .... ...0 .... = Frame Pending: False
        .... .... ..0. .... = Acknowledge Request: False
        .... .... .0.. .... = Intra-PAN: False
        .... ...0 .... .... = Sequence Number Suppression: False
        .... ..0. .... .... = Information Elements present: False
        .... 10.. .... .... = Destination Addressing Mode:
                              Short/16-bit (0x0002)
        ..10 .... .... .... = Frame Version: 2
        11.. .... .... .... = Source Addressing Mode:
                              Long/64-bit (0x0003)
    Sequence Number: 235
    Destination PAN: 0xcafe
    Destination: 0xffff
    Extended Source: 14:15:92:cc:00:00:00:02 (14:15:92:cc:00:00:00:02)
    FCS: 0xf442 (Correct)
6LoWPAN
    IPHC Header
        011. .... = Pattern: IP header compression (0x03)
        ...1 1... .... .... = Traffic class and flow label: Version,
                              traffic class, and flow label
                              compressed (0x0003)
        .... .0.. .... .... = Next header: Inline
        .... ..00 .... .... = Hop limit: Inline (0x0000)
        .... .... 0... .... = Context identifier extension: False
        .... .... .0.. .... = Source address compression: Stateless
        .... .... ..11 .... = Source address mode: Compressed (0x0003)
        .... .... .... 1... = Multicast address compression: True
        .... .... .... .0.. = Destination address compression: Stateless
        .... .... .... ..11 = Destination address mode:
                              8-bits inline (0x0003)
        [Source context: fe80:: (fe80::)]
        [Destination context: fe80:: (fe80::)]
    Next header: ICMPv6 (0x3a)
    Hop limit: 64
    Source: fe80::1615:92cc:0:2 (fe80::1615:92cc:0:2)
    Destination: ff02::1a (ff02::1a)
```

```
Internet Protocol Version 6, Src: fe80::1615:92cc:0:2
(fe80::1615:92cc:0:2), Dst: ff02::1a (ff02::1a)
    0110 .... = Version: 6
        [0110 .... = This field makes the filter
                    "ip.version == 6" possible: 6]
    .... 0000 0000 .... .... .... .... .... = Traffic class: 0x00000000
        .... 0000 00.. .... .... .... .... .... = Differentiated
                                                  Services Field:
                                                  Default (0x00000000)
        .... .... ..0. .... .... .... .... .... = ECN-Capable Transport
                                                  (ECT): Not set
        .... .... ...0 .... .... .... .... .... = ECN-CE: Not set
    .... .... .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
    Payload length: 28
    Next header: ICMPv6 (58)
    Hop limit: 64
    Source: fe80::1615:92cc:0:2 (fe80::1615:92cc:0:2)
    Destination: ff02::1a (ff02::1a)
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
Internet Control Message Protocol v6
    Type: RPL Control (155)
    Code: 1 (DODAG Information Object)
    Checksum: 0x14e7 [incorrect, should be 0xd021]
        [Expert Info (Warn/Checksum): ICMPv6 Checksum Incorrect,
                                      should be 0xd021]
            [ICMPv6 Checksum Incorrect, should be 0xd021]
            [Severity level: Warn]
            [Group: Checksum]
    RPLInstanceID: 0
    Version: 0
    Rank: 819
    Flags: 0x88
        1... .... = Grounded (G): True
        .0.. .... = Zero: False
        ..00 1... = Mode of Operation (MOP): Non-Storing Mode of
                                             Operation (0x01)
        .... .000 = DODAG Preference: 0
    Destination Advertisement Trigger Sequence Number (DTSN): 51
    Flags: 0x00
    Reserved: 00
    DODAGID: bbbb::1415:92cc:0:1 (bbbb::1415:92cc:0:1)

== Raw Bytes ==

0000   01 e8 eb fe ca ff ff 02 00 00 00 cc 92 15 14 78  ...............x
0010   3b 3a 40 1a 9b 01 14 e7 00 00 03 33 88 33 00 00  ;:@........3.3..
0020   bb bb 00 00 00 00 00 00 14 15 92 cc 00 00 00 01  ................
```

```
0030   42 f4                                          B.

   RPL DIO sent by 3

== Dissected packet ==

IEEE 802.15.4 Data, Dst: Broadcast, Src: 14:15:92:cc:00:00:00:03
    Frame Control Field: 0xe801, Frame Type: Data,
    Destination Addressing Mode: Short/16-bit,
    Source Addressing Mode: Long/64-bit
        .... .... .... .001 = Frame Type: Data (0x0001)
        .... .... .... 0... = Security Enabled: False
        .... .... ...0 .... = Frame Pending: False
        .... .... ..0. .... = Acknowledge Request: False
        .... .... .0.. .... = Intra-PAN: False
        .... ...0 .... .... = Sequence Number Suppression: False
        .... ..0. .... .... = Information Elements present: False
        .... 10.. .... .... = Destination Addressing Mode:
                              Short/16-bit (0x0002)
        ..10 .... .... .... = Frame Version: 2
        11.. .... .... .... = Source Addressing Mode:
                              Long/64-bit (0x0003)
    Sequence Number: 231
    Destination PAN: 0xcafe
    Destination: 0xffff
    Extended Source: 14:15:92:cc:00:00:00:03 (14:15:92:cc:00:00:00:03)
    FCS: 0x3dc7 (Correct)
6LoWPAN
    IPHC Header
        011. .... = Pattern: IP header compression (0x03)
        ...1 1... .... .... = Traffic class and flow label: Version,
                              traffic class, and flow label
                              compressed (0x0003)
        .... .0.. .... .... = Next header: Inline
        .... ..00 .... .... = Hop limit: Inline (0x0000)
        .... .... 0... .... = Context identifier extension: False
        .... .... .0.. .... = Source address compression: Stateless
        .... .... ..11 .... = Source address mode: Compressed (0x0003)
        .... .... .... 1... = Multicast address compression: True
        .... .... .... .0.. = Destination address compression: Stateless
        .... .... .... ..11 = Destination address mode:
                              8-bits inline (0x0003)
        [Source context: fe80:: (fe80::)]
        [Destination context: fe80:: (fe80::)]
    Next header: ICMPv6 (0x3a)
    Hop limit: 64
    Source: fe80::1615:92cc:0:3 (fe80::1615:92cc:0:3)
    Destination: ff02::1a (ff02::1a)
```

```
Internet Protocol Version 6, Src: fe80::1615:92cc:0:3
(fe80::1615:92cc:0:3), Dst: ff02::1a (ff02::1a)
    0110 .... = Version: 6
        [0110 .... = This field makes the filter
                     "ip.version == 6" possible: 6]
    .... 0000 0000 .... .... .... .... .... = Traffic class: 0x00000000
        .... 0000 00.. .... .... .... .... .... = Differentiated
                                                  Services Field:
                                                  Default (0x00000000)
        .... .... ..0. .... .... .... .... .... = ECN-Capable Transport
                                                  (ECT): Not set
        .... .... ...0 .... .... .... .... .... = ECN-CE: Not set
    .... .... .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
    Payload length: 28
    Next header: ICMPv6 (58)
    Hop limit: 64
    Source: fe80::1615:92cc:0:3 (fe80::1615:92cc:0:3)
    Destination: ff02::1a (ff02::1a)
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
Internet Control Message Protocol v6
    Type: RPL Control (155)
    Code: 1 (DODAG Information Object)
    Checksum: 0x1234 [incorrect, should be 0xcd6e]
        [Expert Info (Warn/Checksum): ICMPv6 Checksum Incorrect,
                                      should be 0xcd6e]
            [ICMPv6 Checksum Incorrect, should be 0xcd6e]
            [Severity level: Warn]
            [Group: Checksum]
    RPLInstanceID: 0
    Version: 0
    Rank: 1509
    Flags: 0x88
        1... .... = Grounded (G): True
        .0.. .... = Zero: False
        ..00 1... = Mode of Operation (MOP): Non-Storing Mode
                    of Operation (0x01)
        .... .000 = DODAG Preference: 0
    Destination Advertisement Trigger Sequence Number (DTSN): 51
    Flags: 0x00
    Reserved: 00
    DODAGID: bbbb::1415:92cc:0:1 (bbbb::1415:92cc:0:1)

== Raw Bytes ==

0000   01 e8 e7 fe ca ff ff 03 00 00 00 cc 92 15 14 78  ...............x
0010   3b 3a 40 1a 9b 01 12 34 00 00 05 e5 88 33 00 00  ;:@....4.....3..
0020   bb bb 00 00 00 00 00 00 14 15 92 cc 00 00 00 01  ................
```

```
0030   c7 3d                                            .=
```

3.4. RPL DAO

3.4.1. RPL DAO from 2

[RPL DAO from 2] 2->1

```
== Dissected packet ==

IEEE 802.15.4 Data, Dst: 14:15:92:cc:00:00:00:01,
Src: 14:15:92:cc:00:00:00:02
    Frame Control Field: 0xec21, Frame Type: Data, Acknowledge Request,
    Destination Addressing Mode: Long/64-bit,
    Source Addressing Mode: Long/64-bit
        .... .... .... .001 = Frame Type: Data (0x0001)
        .... .... .... 0... = Security Enabled: False
        .... .... ...0 .... = Frame Pending: False
        .... .... ..1. .... = Acknowledge Request: True
        .... .... .0.. .... = Intra-PAN: False
        .... ...0 .... .... = Sequence Number Suppression: False
        .... ..0. .... .... = Information Elements present: False
        .... 11.. .... .... = Destination Addressing Mode:
                              Long/64-bit (0x0003)
        ..10 .... .... .... = Frame Version: 2
        11.. .... .... .... = Source Addressing Mode:
                              Long/64-bit (0x0003)
    Sequence Number: 226
    Destination PAN: 0xcafe
    Destination: 14:15:92:cc:00:00:00:01 (14:15:92:cc:00:00:00:01)
    Extended Source: 14:15:92:cc:00:00:00:02 (14:15:92:cc:00:00:00:02)
    FCS: 0xaf57 (Correct)
6LoWPAN
    IPHC Header
        011. .... = Pattern: IP header compression (0x03)
        ...1 1... .... .... = Traffic class and flow label: Version,
                              traffic class, and flow label
                              compressed (0x0003)
        .... .1.. .... .... = Next header: Compressed
        .... ..00 .... .... = Hop limit: Inline (0x0000)
        .... .... 0... .... = Context identifier extension: False
        .... .... .0.. .... = Source address compression: Stateless
        .... .... ..11 .... = Source address mode: Compressed (0x0003)
        .... .... .... 0... = Multicast address compression: False
        .... .... .... .0.. = Destination address compression: Stateless
        .... .... .... ..11 = Destination address mode:
                              Compressed (0x0003)
        [Source context: fe80:: (fe80::)]
```

```
        [Destination context: fe80:: (fe80::)]
    Hop limit: 64
    Source: fe80::1615:92cc:0:2 (fe80::1615:92cc:0:2)
    Destination: fe80::1615:92cc:0:1 (fe80::1615:92cc:0:1)
    IPv6 extension header
        1110 .... = Pattern: IPv6 extension header (0x0e)
        .... 000. = Header ID: IPv6 hop-by-hop options (0x00)
        .... ...1 = Next header: Compressed
        Header length: 6
        Data (6 bytes)

0000  63 04 00 00 2e 03                                 c.....
            Data: 630400002e03
            [Length: 6]
    IPv6 extension header
        1110 .... = Pattern: IPv6 extension header (0x0e)
        .... 111. = Header ID: IPv6 header (0x07)
        .... ...0 = Next header: Inline
    IPHC Header
        011. .... = Pattern: IP header compression (0x03)
        ...1 1... .... .... = Traffic class and flow label: Version,
                              traffic class, and flow label
                              compressed (0x0003)
        .... .0.. .... .... = Next header: Inline
        .... ..10 .... .... = Hop limit: 64 (0x0002)
        .... .... 0... .... = Context identifier extension: False
        .... .... .0.. .... = Source address compression: Stateless
        .... .... ..01 .... = Source address mode:
                              64-bits inline (0x0001)
        .... .... .... 0... = Multicast address compression: False
        .... .... .... .0.. = Destination address compression: Stateless
        .... .... .... ..11 = Destination address mode:
                              Compressed (0x0003)
        [Source context: fe80:: (fe80::)]
        [Destination context: fe80:: (fe80::)]
    Next header: ICMPv6 (0x3a)
    Source: fe80::1415:92cc:0:2 (fe80::1415:92cc:0:2)
    Destination: fe80::1615:92cc:0:1 (fe80::1615:92cc:0:1)
Internet Protocol Version 6, Src: fe80::1615:92cc:0:2
(fe80::1615:92cc:0:2), Dst: fe80::1615:92cc:0:1 (fe80::1615:92cc:0:1)
    0110 .... = Version: 6
        [0110 .... = This field makes the filter
                     "ip.version == 6" possible: 6]
    .... 0000 0000 .... .... .... .... .... = Traffic class: 0x00000000
        .... 0000 00.. .... .... .... .... .... = Differentiated
                                                  Services Field:
                                                  Default (0x00000000)
        .... .... ..0. .... .... .... .... .... = ECN-Capable Transport
```

```
                                              (ECT): Not set
        .... .... ...0 .... .... .... .... .... = ECN-CE: Not set
    .... .... .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
    Payload length: 114
    Next header: IPv6 hop-by-hop option (0)
    Hop limit: 64
    Source: fe80::1615:92cc:0:2 (fe80::1615:92cc:0:2)
    Destination: fe80::1615:92cc:0:1 (fe80::1615:92cc:0:1)
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
    Hop-by-Hop Option
        Next header: IPv6 (41)
        Length: 0 (8 bytes)
        IPv6 Option (RPL Option)
            Type: RPL Option (99)
            Length: 4
            Flag: 0x00
                0... .... = Down: False
                .0.. .... = Rank Error: False
                ..0. .... = Forwarding Error: False
                ...0 0000 = Reserved: 0x00
            RPLInstanceID: 0x00
            Sender Rank: 0x2e03
Internet Protocol Version 6, Src: fe80::1415:92cc:0:2
(fe80::1415:92cc:0:2), Dst: fe80::1615:92cc:0:1 (fe80::1615:92cc:0:1)
    0110 .... = Version: 6
        [0110 .... = This field makes the filter
                     "ip.version == 6" possible: 6]
    .... 0000 0000 .... .... .... .... .... = Traffic class: 0x00000000
        .... 0000 00.. .... .... .... .... .... = Differentiated
                                              Services Field:
                                              Default (0x00000000)
        .... .... ..0. .... .... .... .... .... = ECN-Capable Transport
                                              (ECT): Not set
        .... .... ...0 .... .... .... .... .... = ECN-CE: Not set
    .... .... .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
    Payload length: 66
    Next header: ICMPv6 (58)
    Hop limit: 64
    Source: fe80::1415:92cc:0:2 (fe80::1415:92cc:0:2)
    Destination: fe80::1615:92cc:0:1 (fe80::1615:92cc:0:1)
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
Internet Control Message Protocol v6
    Type: RPL Control (155)
    Code: 2 (Destination Advertisement Object)
    Checksum: 0x11d6 [incorrect, should be 0x8a4b]
        [Expert Info (Warn/Checksum): ICMPv6 Checksum Incorrect,
```

```
                                      should be 0x8a4b]
            [ICMPv6 Checksum Incorrect, should be 0x8a4b]
            [Severity level: Warn]
            [Group: Checksum]
    RPLInstanceID: 0
    Flags: 0x40
        0... .... = DAO-ACK Request (K): False
        .1.. .... = DODAGID Present (D): True
        ..00 0000 = Reserved: 0
    Reserved: 00
    DAO Sequence: 0
    DODAGID: bbbb::1415:92cc:0:1 (bbbb::1415:92cc:0:1)
    ICMPv6 RPL Option (RPL Target bbbb::1415:92cc:0:3/128)
        Type: RPL Target (5)
        Length: 18
        Reserved
        Target Length: 128
        Target: bbbb::1415:92cc:0:3 (bbbb::1415:92cc:0:3)
    ICMPv6 RPL Option (Transit Information bbbb::1415:92cc:0:1)
        Type: Transit Information (6)
        Length: 20
        Flags: 0x00
            0... .... = External: Not set
            .000 0000 = Reserved: 0
        Path Control: 0
        Path Sequence: 89
        Path Lifetime: 170
        Parent Address: bbbb::1415:92cc:0:1 (bbbb::1415:92cc:0:1)
```

== Raw Bytes ==

```
0000   21 ec e2 fe ca 01 00 00 00 cc 92 15 14 02 00 00  !...............
0010   00 cc 92 15 14 7c 33 40 e1 06 63 04 00 00 2e 03  .....|3@..c.....
0020   ee 7a 13 3a 14 15 92 cc 00 00 00 02 9b 02 11 d6  .z.:............
0030   00 40 00 00 bb bb 00 00 00 00 00 00 14 15 92 cc  .@..............
0040   00 00 00 01 05 12 00 80 bb bb 00 00 00 00 00 00  ................
0050   14 15 92 cc 00 00 00 03 06 14 00 00 59 aa bb bb  ............Y...
0060   00 00 00 00 00 00 14 15 92 cc 00 00 00 01 57     ..............W.
```

3.4.2. RPL DAO from 3

[RPL DAO from 3] 3->2

== Dissected packet ==

```
IEEE 802.15.4 Data, Dst: 14:15:92:cc:00:00:00:02,
Src: 14:15:92:cc:00:00:00:03
    Frame Control Field: 0xec21, Frame Type: Data, Acknowledge Request,
```

```
    Destination Addressing Mode: Long/64-bit,
    Source Addressing Mode: Long/64-bit
        .... .... .... .001 = Frame Type: Data (0x0001)
        .... .... .... 0... = Security Enabled: False
        .... .... ...0 .... = Frame Pending: False
        .... .... ..1. .... = Acknowledge Request: True
        .... .... .0.. .... = Intra-PAN: False
        .... ...0 .... .... = Sequence Number Suppression: False
        .... ..0. .... .... = Information Elements present: False
        .... 11.. .... .... = Destination Addressing Mode:
                              Long/64-bit (0x0003)
        ..10 .... .... .... = Frame Version: 2
        11.. .... .... .... = Source Addressing Mode:
                              Long/64-bit (0x0003)
    Sequence Number: 92
    Destination PAN: 0xcafe
    Destination: 14:15:92:cc:00:00:00:02 (14:15:92:cc:00:00:00:02)
    Extended Source: 14:15:92:cc:00:00:00:03 (14:15:92:cc:00:00:00:03)
    FCS: 0x6a88 (Correct)
6LoWPAN
    IPHC Header
        011. .... = Pattern: IP header compression (0x03)
        ...1 1... .... .... = Traffic class and flow label: Version,
                              traffic class, and flow label
                              compressed (0x0003)
        .... .1.. .... .... = Next header: Compressed
        .... ..00 .... .... = Hop limit: Inline (0x0000)
        .... .... 0... .... = Context identifier extension: False
        .... .... .0.. .... = Source address compression: Stateless
        .... .... ..01 .... = Source address mode:
                              64-bits inline (0x0001)
        .... .... .... 0... = Multicast address compression: False
        .... .... .... .0.. = Destination address compression: Stateless
        .... .... .... ..01 = Destination address mode:
                              64-bits inline (0x0001)
        [Source context: fe80:: (fe80::)]
        [Destination context: fe80:: (fe80::)]
    Hop limit: 64
    Source: fe80::1415:92cc:0:3 (fe80::1415:92cc:0:3)
    Destination: fe80::1415:92cc:0:1 (fe80::1415:92cc:0:1)
    IPv6 extension header
        1110 .... = Pattern: IPv6 extension header (0x0e)
        .... 000. = Header ID: IPv6 hop-by-hop options (0x00)
        .... ...1 = Next header: Compressed
        Header length: 6
        Data (6 bytes)

0000  63 04 00 00 de 05                                 c.....
```

```
            Data: 63040000de05
            [Length: 6]
    IPv6 extension header
        1110 .... = Pattern: IPv6 extension header (0x0e)
        .... 111. = Header ID: IPv6 header (0x07)
        .... ...0 = Next header: Inline
    IPHC Header
        011. .... = Pattern: IP header compression (0x03)
        ...1 1... .... .... = Traffic class and flow label: Version,
                              traffic class, and flow label
                              compressed (0x0003)
        .... .0.. .... .... = Next header: Inline
        .... ..10 .... .... = Hop limit: 64 (0x0002)
        .... .... 0... .... = Context identifier extension: False
        .... .... .0.. .... = Source address compression: Stateless
        .... .... ..01 .... = Source address mode:
                              64-bits inline (0x0001)
        .... .... .... 0... = Multicast address compression: False
        .... .... .... .0.. = Destination address compression: Stateless
        .... .... .... ..11 = Destination address mode:
                              Compressed (0x0003)
        [Source context: fe80:: (fe80::)]
        [Destination context: fe80:: (fe80::)]
    Next header: ICMPv6 (0x3a)
    Source: fe80::1415:92cc:0:3 (fe80::1415:92cc:0:3)
    Destination: fe80::1415:92cc:0:1 (fe80::1415:92cc:0:1)
Internet Protocol Version 6, Src: fe80::1415:92cc:0:3
(fe80::1415:92cc:0:3), Dst: fe80::1415:92cc:0:1 (fe80::1415:92cc:0:1)
    0110 .... = Version: 6
        [0110 .... = This field makes the filter
                     "ip.version == 6" possible: 6]
    .... 0000 0000 .... .... .... .... .... = Traffic class: 0x00000000
        .... 0000 00.. .... .... .... .... .... = Differentiated
                                                  Services Field:
                                                  Default (0x00000000)
        .... .... ..0. .... .... .... .... .... = ECN-Capable Transport
                                                  (ECT): Not set
        .... .... ...0 .... .... .... .... .... = ECN-CE: Not set
    .... .... .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
    Payload length: 94
    Next header: IPv6 hop-by-hop option (0)
    Hop limit: 64
    Source: fe80::1415:92cc:0:3 (fe80::1415:92cc:0:3)
    Destination: fe80::1415:92cc:0:1 (fe80::1415:92cc:0:1)
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
    Hop-by-Hop Option
        Next header: IPv6 (41)
```

```
        Length: 0 (8 bytes)
        IPv6 Option (RPL Option)
            Type: RPL Option (99)
            Length: 4
            Flag: 0x00
                0... .... = Down: False
                .0.. .... = Rank Error: False
                ..0. .... = Forwarding Error: False
                ...0 0000 = Reserved: 0x00
            RPLInstanceID: 0x00
            Sender Rank: 0xde05
Internet Protocol Version 6, Src: fe80::1415:92cc:0:3
(fe80::1415:92cc:0:3), Dst: fe80::1415:92cc:0:1 (fe80::1415:92cc:0:1)
    0110 .... = Version: 6
        [0110 .... = This field makes the filter
                     "ip.version == 6" possible: 6]
    .... 0000 0000 .... .... .... .... .... = Traffic class: 0x00000000
        .... 0000 00.. .... .... .... .... .... = Differentiated
                                                  Services Field:
                                                  Default (0x00000000)
        .... .... ..0. .... .... .... .... .... = ECN-Capable Transport
                                                  (ECT): Not set
        .... .... ...0 .... .... .... .... .... = ECN-CE: Not set
    .... .... .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
    Payload length: 46
    Next header: ICMPv6 (58)
    Hop limit: 64
    Source: fe80::1415:92cc:0:3 (fe80::1415:92cc:0:3)
    Destination: fe80::1415:92cc:0:1 (fe80::1415:92cc:0:1)
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
Internet Control Message Protocol v6
    Type: RPL Control (155)
    Code: 2 (Destination Advertisement Object)
    Checksum: 0x791a [incorrect, should be 0xf38f]
        [Expert Info (Warn/Checksum): ICMPv6 Checksum Incorrect,
                                      should be 0xf38f]
            [ICMPv6 Checksum Incorrect, should be 0xf38f]
            [Severity level: Warn]
            [Group: Checksum]
    RPLInstanceID: 0
    Flags: 0x40
        0... .... = DAO-ACK Request (K): False
        .1.. .... = DODAGID Present (D): True
        ..00 0000 = Reserved: 0
    Reserved: 00
    DAO Sequence: 0
    DODAGID: bbbb::1415:92cc:0:1 (bbbb::1415:92cc:0:1)
```

```
    ICMPv6 RPL Option (Transit Information bbbb::1415:92cc:0:2)
        Type: Transit Information (6)
        Length: 20
        Flags: 0x00
            0... .... = External: Not set
            .000 0000 = Reserved: 0
        Path Control: 0
        Path Sequence: 90
        Path Lifetime: 170
        Parent Address: bbbb::1415:92cc:0:2 (bbbb::1415:92cc:0:2)
```

== Raw Bytes ==

```
0000   21 ec 5c fe ca 02 00 00 00 cc 92 15 14 03 00 00  !.\.............
0010   00 cc 92 15 14 7c 11 40 14 15 92 cc 00 00 00 03  .....|.@........
0020   14 15 92 cc 00 00 00 01 e1 06 63 04 00 00 de 05  ..........c.....
0030   ee 7a 13 3a 14 15 92 cc 00 00 00 03 9b 02 79 1a  .z.:..........y.
0040   00 40 00 00 bb bb 00 00 00 00 00 14 15 92 cc     .@..............
0050   00 00 00 01 06 14 00 00 5a aa bb bb 00 00 00 00  ........Z.......
0060   00 00 14 15 92 cc 00 00 00 02 88 6a              ...........j
```

[RPL DAO from 3] 2->1

== Dissected packet ==

```
IEEE 802.15.4 Data, Dst: 14:15:92:cc:00:00:00:01,
Src: 14:15:92:cc:00:00:00:02
    Frame Control Field: 0xec21, Frame Type: Data, Acknowledge Request,
    Destination Addressing Mode: Long/64-bit,
    Source Addressing Mode: Long/64-bit
        .... .... .... .001 = Frame Type: Data (0x0001)
        .... .... .... 0... = Security Enabled: False
        .... .... ...0 .... = Frame Pending: False
        .... .... ..1. .... = Acknowledge Request: True
        .... .... .0.. .... = Intra-PAN: False
        .... ...0 .... .... = Sequence Number Suppression: False
        .... ..0. .... .... = Information Elements present: False
        .... 11.. .... .... = Destination Addressing Mode:
                              Long/64-bit (0x0003)
        ..10 .... .... .... = Frame Version: 2
        11.. .... .... .... = Source Addressing Mode:
                              Long/64-bit (0x0003)
    Sequence Number: 222
    Destination PAN: 0xcafe
    Destination: 14:15:92:cc:00:00:00:01 (14:15:92:cc:00:00:00:01)
    Extended Source: 14:15:92:cc:00:00:00:02 (14:15:92:cc:00:00:00:02)
    FCS: 0xd991 (Correct)
6LoWPAN
```

```
    IPHC Header
        011. .... = Pattern: IP header compression (0x03)
        ...1 1... .... .... = Traffic class and flow label: Version,
                              traffic class, and flow label
                              compressed (0x0003)
        .... .1.. .... .... = Next header: Compressed
        .... ..00 .... .... = Hop limit: Inline (0x0000)
        .... .... 0... .... = Context identifier extension: False
        .... .... .0.. .... = Source address compression: Stateless
        .... .... ..01 .... = Source address mode:
                              64-bits inline (0x0001)
        .... .... .... 0... = Multicast address compression: False
        .... .... .... .0.. = Destination address compression: Stateless
        .... .... .... ..11 = Destination address mode:
                              Compressed (0x0003)
        [Source context: fe80:: (fe80::)]
        [Destination context: fe80:: (fe80::)]
    Hop limit: 63
    Source: fe80::1415:92cc:0:3 (fe80::1415:92cc:0:3)
    Destination: fe80::1615:92cc:0:1 (fe80::1615:92cc:0:1)
    IPv6 extension header
        1110 .... = Pattern: IPv6 extension header (0x0e)
        .... 000. = Header ID: IPv6 hop-by-hop options (0x00)
        .... ...1 = Next header: Compressed
        Header length: 6
        Data (6 bytes)

0000  63 04 00 00 2b 03                                 c...+.
            Data: 630400002b03
            [Length: 6]
    IPv6 extension header
        1110 .... = Pattern: IPv6 extension header (0x0e)
        .... 111. = Header ID: IPv6 header (0x07)
        .... ...0 = Next header: Inline
    IPHC Header
        011. .... = Pattern: IP header compression (0x03)
        ...1 1... .... .... = Traffic class and flow label: Version,
                              traffic class, and flow label
                              compressed (0x0003)
        .... .0.. .... .... = Next header: Inline
        .... ..10 .... .... = Hop limit: 64 (0x0002)
        .... .... 0... .... = Context identifier extension: False
        .... .... .0.. .... = Source address compression: Stateless
        .... .... ..01 .... = Source address mode:
                              64-bits inline (0x0001)
        .... .... .... 0... = Multicast address compression: False
        .... .... .... .0.. = Destination address compression: Stateless
        .... .... .... ..11 = Destination address mode:
```

```
                              Compressed (0x0003)
        [Source context: fe80:: (fe80::)]
        [Destination context: fe80:: (fe80::)]
    Next header: ICMPv6 (0x3a)
    Source: fe80::1415:92cc:0:3 (fe80::1415:92cc:0:3)
    Destination: fe80::1615:92cc:0:1 (fe80::1615:92cc:0:1)
Internet Protocol Version 6, Src: fe80::1415:92cc:0:3
(fe80::1415:92cc:0:3), Dst: fe80::1615:92cc:0:1 (fe80::1615:92cc:0:1)
    0110 .... = Version: 6
        [0110 .... = This field makes the filter
                    "ip.version == 6" possible: 6]
    .... 0000 0000 .... .... .... .... .... = Traffic class: 0x00000000
        .... 0000 00.. .... .... .... .... .... = Differentiated
                                                  Services Field:
                                                  Default (0x00000000)
        .... .... ..0. .... .... .... .... .... = ECN-Capable Transport
                                                  (ECT): Not set
        .... .... ...0 .... .... .... .... .... = ECN-CE: Not set
    .... .... .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
    Payload length: 94
    Next header: IPv6 hop-by-hop option (0)
    Hop limit: 63
    Source: fe80::1415:92cc:0:3 (fe80::1415:92cc:0:3)
    Destination: fe80::1615:92cc:0:1 (fe80::1615:92cc:0:1)
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
    Hop-by-Hop Option
        Next header: IPv6 (41)
        Length: 0 (8 bytes)
        IPv6 Option (RPL Option)
            Type: RPL Option (99)
            Length: 4
            Flag: 0x00
                0... .... = Down: False
                .0.. .... = Rank Error: False
                ..0. .... = Forwarding Error: False
                ...0 0000 = Reserved: 0x00
            RPLInstanceID: 0x00
            Sender Rank: 0x2b03
Internet Protocol Version 6, Src: fe80::1415:92cc:0:3
(fe80::1415:92cc:0:3), Dst: fe80::1615:92cc:0:1 (fe80::1615:92cc:0:1)
    0110 .... = Version: 6
        [0110 .... = This field makes the filter
                    "ip.version == 6" possible: 6]
    .... 0000 0000 .... .... .... .... .... = Traffic class: 0x00000000
        .... 0000 00.. .... .... .... .... .... = Differentiated
                                                  Services Field:
                                                  Default (0x00000000)
```

```
        .... .... ..0. .... .... .... .... .... = ECN-Capable Transport
        (ECT): Not set
        .... .... ...0 .... .... .... .... .... = ECN-CE: Not set
    .... .... .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
    Payload length: 46
    Next header: ICMPv6 (58)
    Hop limit: 64
    Source: fe80::1415:92cc:0:3 (fe80::1415:92cc:0:3)
    Destination: fe80::1615:92cc:0:1 (fe80::1615:92cc:0:1)
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
Internet Control Message Protocol v6
    Type: RPL Control (155)
    Code: 2 (Destination Advertisement Object)
    Checksum: 0x791a [incorrect, should be 0xf18f]
        [Expert Info (Warn/Checksum): ICMPv6 Checksum Incorrect,
                                      should be 0xf18f]
            [ICMPv6 Checksum Incorrect, should be 0xf18f]
            [Severity level: Warn]
            [Group: Checksum]
    RPLInstanceID: 0
    Flags: 0x40
        0... .... = DAO-ACK Request (K): False
        .1.. .... = DODAGID Present (D): True
        ..00 0000 = Reserved: 0
    Reserved: 00
    DAO Sequence: 0
    DODAGID: bbbb::1415:92cc:0:1 (bbbb::1415:92cc:0:1)
    ICMPv6 RPL Option (Transit Information bbbb::1415:92cc:0:2)
        Type: Transit Information (6)
        Length: 20
        Flags: 0x00
            0... .... = External: Not set
            .000 0000 = Reserved: 0
        Path Control: 0
        Path Sequence: 90
        Path Lifetime: 170
        Parent Address: bbbb::1415:92cc:0:2 (bbbb::1415:92cc:0:2)

== Raw Bytes ==

0000   21 ec de fe ca 01 00 00 00 cc 92 15 14 02 00 00  !...............
0010   00 cc 92 15 14 7c 13 3f 14 15 92 cc 00 00 00 03  .....|.?........
0020   e1 06 63 04 00 00 2b 03 ee 7a 13 3a 14 15 92 cc  ..c...+..z.:....
0030   00 00 00 03 9b 02 79 1a 00 40 00 00 bb bb 00 00  ......y..@......
0040   00 00 00 00 14 15 92 cc 00 00 00 01 06 14 00 00  ................
0050   5a aa bb bb 00 00 00 00 00 00 14 15 92 cc 00 00  Z...............
0060   00 02 91 d9                                      ....
```

3.5. ACK

ACK

```
== Dissected packet ==

IEEE 802.15.4 Ack, Sequence Number: 92, Dst: 14:15:92:cc:00:00:00:03,
Src: 14:15:92:cc:00:00:00:02 Time Correction IE Ack
    Frame Control Field: 0xee02, Frame Type: Ack, Information
    Elements present, Destination Addressing Mode: Long/64-bit,
    Source Addressing Mode: Long/64-bit
        .... .... .... .010 = Frame Type: Ack (0x0002)
        .... .... .... 0... = Security Enabled: False
        .... .... ...0 .... = Frame Pending: False
        .... .... ..0. .... = Acknowledge Request: False
        .... .... .0.. .... = Intra-PAN: False
        .... ...0 .... .... = Sequence Number Suppression: False
        .... ..1. .... .... = Information Elements present: True
        .... 11.. .... .... = Destination Addressing Mode:
                              Long/64-bit (0x0003)
        ..10 .... .... .... = Frame Version: 2
        11.. .... .... .... = Source Addressing Mode:
                              Long/64-bit (0x0003)
    Sequence Number: 92
    Destination PAN: 0xcafe
    Destination: 14:15:92:cc:00:00:00:03 (14:15:92:cc:00:00:00:03)
    Extended Source: 14:15:92:cc:00:00:00:02 (14:15:92:cc:00:00:00:02)
    Header Information Elements: Time Correction IE (0x0f02)
        Information Element Length: 0x02
        .000 1111 0... .... = Information Element ID:
                              Time Correction IE (0x001e)
        0... .... .... .... = Payload IE: TRUE - Header IE: FALSE: False
    Data (2 bytes)

0000  00 00                                              ..
        Data: 0000
        [Length: 2]
    FCS: 0x27fc (Correct)

== Raw Bytes ==

0000   02 ee 5c fe ca 03 00 00 00 cc 92 15 14 02 00 00  ..\.............
0010   00 cc 92 15 14 02 0f 00 00 fc 27                 ..........'
```

3.6. ICMPv6 echo request/reply

3.6.1. ping 2

[ping 2] ICMPv6 echo request 1->2

```
== Dissected packet ==

IEEE 802.15.4 Data, Dst: 14:15:92:cc:00:00:00:02,
Src: 14:15:92:cc:00:00:00:01
    Frame Control Field: 0xec21, Frame Type: Data, Acknowledge Request,
    Destination Addressing Mode: Long/64-bit,
    Source Addressing Mode: Long/64-bit
        .... .... .... .001 = Frame Type: Data (0x0001)
        .... .... .... 0... = Security Enabled: False
        .... .... ...0 .... = Frame Pending: False
        .... .... ..1. .... = Acknowledge Request: True
        .... .... .0.. .... = Intra-PAN: False
        .... ...0 .... .... = Sequence Number Suppression: False
        .... ..0. .... .... = Information Elements present: False
        .... 11.. .... .... = Destination Addressing Mode:
                              Long/64-bit (0x0003)
        ..10 .... .... .... = Frame Version: 2
        11.. .... .... .... = Source Addressing Mode:
                              Long/64-bit (0x0003)
    Sequence Number: 76
    Destination PAN: 0xcafe
    Destination: 14:15:92:cc:00:00:00:02 (14:15:92:cc:00:00:00:02)
    Extended Source: 14:15:92:cc:00:00:00:01 (14:15:92:cc:00:00:00:01)
    FCS: 0x70b6 (Correct)
6LoWPAN
    IPHC Header
        011. .... = Pattern: IP header compression (0x03)
        ...1 1... .... .... = Traffic class and flow label: Version,
                              traffic class, and flow label
                              compressed (0x0003)
        .... .1.. .... .... = Next header: Compressed
        .... ..00 .... .... = Hop limit: Inline (0x0000)
        .... .... 0... .... = Context identifier extension: False
        .... .... .0.. .... = Source address compression: Stateless
        .... .... ..00 .... = Source address mode: Inline (0x0000)
        .... .... .... 0... = Multicast address compression: False
        .... .... .... .0.. = Destination address compression: Stateless
        .... .... .... ..00 = Destination address mode: Inline (0x0000)
    Hop limit: 128
    Source: bbbb::1 (bbbb::1)
    Destination: bbbb::1415:92cc:0:2 (bbbb::1415:92cc:0:2)
    IPv6 extension header
```

```
        1110 .... = Pattern: IPv6 extension header (0x0e)
        .... 111. = Header ID: IPv6 header (0x07)
        .... ...0 = Next header: Inline
    IPHC Header
        011. .... = Pattern: IP header compression (0x03)
        ...1 1... .... .... = Traffic class and flow label: Version,
                              traffic class, and flow label
                              compressed (0x0003)
        .... .0.. .... .... = Next header: Inline
        .... ..00 .... .... = Hop limit: Inline (0x0000)
        .... .... 0... .... = Context identifier extension: False
        .... .... .0.. .... = Source address compression: Stateless
        .... .... ..11 .... = Source address mode:
                              Compressed (0x0003)
        .... .... .... 0... = Multicast address compression: False
        .... .... .... .0.. = Destination address compression: Stateless
        .... .... .... ..11 = Destination address mode:
                              Compressed (0x0003)
        [Source context: fe80:: (fe80::)]
        [Destination context: fe80:: (fe80::)]
    Next header: ICMPv6 (0x3a)
    Hop limit: 128
    Source: fe80::1 (fe80::1)
    Destination: fe80::1415:92cc:0:2 (fe80::1415:92cc:0:2)
Internet Protocol Version 6, Src: bbbb::1 (bbbb::1),
Dst: bbbb::1415:92cc:0:2 (bbbb::1415:92cc:0:2)
    0110 .... = Version: 6
        [0110 .... = This field makes the filter
                     "ip.version == 6" possible: 6]
    .... 0000 0000 .... .... .... .... .... = Traffic class: 0x00000000
        .... 0000 00.. .... .... .... .... .... = Differentiated
                                                  Services Field:
                                                  Default (0x00000000)
        .... .... ..0. .... .... .... .... .... = ECN-Capable Transport
                                                  (ECT): Not set
        .... .... ...0 .... .... .... .... .... = ECN-CE: Not set
    .... .... .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
    Payload length: 80
    Next header: IPv6 (41)
    Hop limit: 128
    Source: bbbb::1 (bbbb::1)
    Destination: bbbb::1415:92cc:0:2 (bbbb::1415:92cc:0:2)
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
Internet Protocol Version 6, Src: fe80::1 (fe80::1),
Dst: fe80::1415:92cc:0:2 (fe80::1415:92cc:0:2)
    0110 .... = Version: 6
        [0110 .... = This field makes the filter
```

```
                     "ip.version == 6" possible: 6]
    .... 0000 0000 .... .... .... .... .... = Traffic class: 0x00000000
        .... 0000 00.. .... .... .... .... .... = Differentiated
                                                  Services Field:
                                                  Default (0x00000000)
        .... .... ..0. .... .... .... .... .... = ECN-Capable Transport
                                                  (ECT): Not set
        .... .... ...0 .... .... .... .... .... = ECN-CE: Not set
    .... .... .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
    Payload length: 40
    Next header: ICMPv6 (58)
    Hop limit: 128
    Source: fe80::1 (fe80::1)
    Destination: fe80::1415:92cc:0:2 (fe80::1415:92cc:0:2)
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
Internet Control Message Protocol v6
    Type: Echo (ping) request (128)
    Code: 0
    Checksum: 0xb68c [incorrect, should be 0x3102]
        [Expert Info (Warn/Checksum): ICMPv6 Checksum Incorrect,
                                      should be 0x3102]
            [ICMPv6 Checksum Incorrect, should be 0x3102]
            [Severity level: Warn]
            [Group: Checksum]
    Identifier: 0x0001
    Sequence: 16
    [Response In: 292]
    Data (32 bytes)

0000  61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70   abcdefghijklmnop
0010  71 72 73 74 75 76 77 61 62 63 64 65 66 67 68 69   qrstuvwabcdefghi
        Data: 6162636465666768696a6b6c6d6e6f7071727374757677761...
        [Length: 32]

== Raw Bytes ==

0000   21 ec 4c fe ca 02 00 00 00 cc 92 15 14 01 00 00  !.L.............
0010   00 cc 92 15 14 7c 00 80 bb bb 00 00 00 00 00 00  .....|..........
0020   00 00 00 00 00 00 00 01 bb bb 00 00 00 00 00 00  ................
0030   14 15 92 cc 00 00 00 02 ee 78 33 3a 80 80 00 b6  .........x3:....
0040   8c 00 01 00 10 61 62 63 64 65 66 67 68 69 6a 6b  .....abcdefghijk
0050   6c 6d 6e 6f 70 71 72 73 74 75 76 77 61 62 63 64  lmnopqrstuvwabcd
0060   65 66 67 68 69 b6 70                             efghi.p

   [ping 2] ICMPv6 echo reply 2->1

== Dissected packet ==
```

```
IEEE 802.15.4 Data, Dst: 14:15:92:cc:00:00:00:01,
Src: 14:15:92:cc:00:00:00:02
    Frame Control Field: 0xec21, Frame Type: Data, Acknowledge Request,
    Destination Addressing Mode: Long/64-bit,
    Source Addressing Mode: Long/64-bit
        .... .... .... .001 = Frame Type: Data (0x0001)
        .... .... .... 0... = Security Enabled: False
        .... .... ...0 .... = Frame Pending: False
        .... .... ..1. .... = Acknowledge Request: True
        .... .... .0.. .... = Intra-PAN: False
        .... ...0 .... .... = Sequence Number Suppression: False
        .... ..0. .... .... = Information Elements present: False
        .... 11.. .... .... = Destination Addressing Mode:
                              Long/64-bit (0x0003)
        ..10 .... .... .... = Frame Version: 2
        11.. .... .... .... = Source Addressing Mode:
                              Long/64-bit (0x0003)
    Sequence Number: 33
    Destination PAN: 0xcafe
    Destination: 14:15:92:cc:00:00:00:01 (14:15:92:cc:00:00:00:01)
    Extended Source: 14:15:92:cc:00:00:00:02 (14:15:92:cc:00:00:00:02)
    FCS: 0x77e8 (Correct)
6LoWPAN
    IPHC Header
        011. .... = Pattern: IP header compression (0x03)
        ...1 1... .... .... = Traffic class and flow label: Version,
                              traffic class, and flow label
                              compressed (0x0003)
        .... .1.. .... .... = Next header: Compressed
        .... ..00 .... .... = Hop limit: Inline (0x0000)
        .... .... 0... .... = Context identifier extension: False
        .... .... .0.. .... = Source address compression: Stateless
        .... .... ..01 .... = Source address mode:
                              64-bits inline (0x0001)
        .... .... .... 0... = Multicast address compression: False
        .... .... .... .0.. = Destination address compression: Stateless
        .... .... .... ..01 = Destination address mode:
                              64-bits inline (0x0001)
        [Source context: fe80:: (fe80::)]
        [Destination context: fe80:: (fe80::)]
    Hop limit: 64
    Source: fe80::1415:92cc:0:2 (fe80::1415:92cc:0:2)
    Destination: fe80::1 (fe80::1)
    IPv6 extension header
        1110 .... = Pattern: IPv6 extension header (0x0e)
        .... 000. = Header ID: IPv6 hop-by-hop options (0x00)
        .... ...1 = Next header: Compressed
        Header length: 6
```

```
        Data (6 bytes)

0000  63 04 00 00 32 03                                  c...2.
            Data: 630400003203
            [Length: 6]
    IPv6 extension header
        1110 .... = Pattern: IPv6 extension header (0x0e)
        .... 111. = Header ID: IPv6 header (0x07)
        .... ...0 = Next header: Inline
    IPHC Header
        011. .... = Pattern: IP header compression (0x03)
        ...1 1... .... .... = Traffic class and flow label: Version,
                              traffic class,
                              and flow label compressed (0x0003)
        .... .0.. .... .... = Next header: Inline
        .... ..10 .... .... = Hop limit: 64 (0x0002)
        .... .... 0... .... = Context identifier extension: False
        .... .... .0.. .... = Source address compression: Stateless
        .... .... ..01 .... = Source address mode:
                              64-bits inline (0x0001)
        .... .... .... 0... = Multicast address compression: False
        .... .... .... .0.. = Destination address compression: Stateless
        .... .... .... ..11 = Destination address mode:
                              Compressed (0x0003)
        [Source context: fe80:: (fe80::)]
        [Destination context: fe80:: (fe80::)]
    Next header: ICMPv6 (0x3a)
    Source: fe80::1415:92cc:0:2 (fe80::1415:92cc:0:2)
    Destination: fe80::1 (fe80::1)
Internet Protocol Version 6,
Src: fe80::1415:92cc:0:2 (fe80::1415:92cc:0:2), Dst: fe80::1 (fe80::1)
    0110 .... = Version: 6
        [0110 .... = This field makes the filter
                     "ip.version == 6" possible: 6]
    .... 0000 0000 .... .... .... .... .... = Traffic class: 0x00000000
        .... 0000 00.. .... .... .... .... .... = Differentiated
                                                  Services Field:
                                                  Default (0x00000000)
        .... .... ..0. .... .... .... .... .... = ECN-Capable Transport
                                                  (ECT): Not set
        .... .... ...0 .... .... .... .... .... = ECN-CE: Not set
    .... .... .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
    Payload length: 88
    Next header: IPv6 hop-by-hop option (0)
    Hop limit: 64
    Source: fe80::1415:92cc:0:2 (fe80::1415:92cc:0:2)
    Destination: fe80::1 (fe80::1)
    [Source GeoIP: Unknown]
```

```
    [Destination GeoIP: Unknown]
    Hop-by-Hop Option
        Next header: IPv6 (41)
        Length: 0 (8 bytes)
        IPv6 Option (RPL Option)
            Type: RPL Option (99)
            Length: 4
            Flag: 0x00
                0... .... = Down: False
                .0.. .... = Rank Error: False
                ..0. .... = Forwarding Error: False
                ...0 0000 = Reserved: 0x00
            RPLInstanceID: 0x00
            Sender Rank: 0x3203
Internet Protocol Version 6,
Src: fe80::1415:92cc:0:2 (fe80::1415:92cc:0:2), Dst: fe80::1 (fe80::1)
    0110 .... = Version: 6
        [0110 .... = This field makes the filter
                     "ip.version == 6" possible: 6]
    .... 0000 0000 .... .... .... .... .... = Traffic class: 0x00000000
        .... 0000 00.. .... .... .... .... .... = Differentiated
                                                  Services Field:
                                                  Default (0x00000000)
        .... .... ..0. .... .... .... .... .... = ECN-Capable Transport
                                                  (ECT): Not set
        .... .... ...0 .... .... .... .... .... = ECN-CE: Not set
    .... .... .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
    Payload length: 40
    Next header: ICMPv6 (58)
    Hop limit: 64
    Source: fe80::1415:92cc:0:2 (fe80::1415:92cc:0:2)
    Destination: fe80::1 (fe80::1)
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
Internet Control Message Protocol v6
    Type: Echo (ping) reply (129)
    Code: 0
    Checksum: 0xb58c [incorrect, should be 0x3002]
        [Expert Info (Warn/Checksum): ICMPv6 Checksum Incorrect,
                                      should be 0x3002]
            [ICMPv6 Checksum Incorrect, should be 0x3002]
            [Severity level: Warn]
            [Group: Checksum]
    Identifier: 0x0001
    Sequence: 16
    [Response To: 289]
    [Response Time: 24.840 ms]
    Data (32 bytes)
```

```
0000  61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70   abcdefghijklmnop
0010  71 72 73 74 75 76 77 61 62 63 64 65 66 67 68 69   qrstuvwabcdefghi
        Data: 6162636465666768696a6b6c6d6e6f7071727374757677761...
        [Length: 32]
```

== Raw Bytes ==

```
0000   21 ec 21 fe ca 01 00 00 00 cc 92 15 14 02 00 00  !.!.............
0010   00 cc 92 15 14 7c 11 40 14 15 92 cc 00 00 00 02  .....|.@........
0020   00 00 00 00 00 00 00 01 e1 06 63 04 00 00 32 03  ..........c...2.
0030   ee 7a 13 3a 14 15 92 cc 00 00 00 02 81 00 b5 8c  .z.:............
0040   00 01 00 10 61 62 63 64 65 66 67 68 69 6a 6b 6c  ....abcdefghijkl
0050   6d 6e 6f 70 71 72 73 74 75 76 77 61 62 63 64 65  mnopqrstuvwabcde
0060   66 67 68 69 e8 77                                fghi.w
```

3.6.2. ping 3

[ping 3] ICMPv6 echo request 1->2

== Dissected packet ==

```
IEEE 802.15.4 Data,
Dst: 14:15:92:cc:00:00:00:02, Src: 14:15:92:cc:00:00:00:01
    Frame Control Field: 0xec21, Frame Type: Data, Acknowledge Request,
    Destination Addressing Mode: Long/64-bit,
    Source Addressing Mode: Long/64-bit
        .... .... .... .001 = Frame Type: Data (0x0001)
        .... .... .... 0... = Security Enabled: False
        .... .... ...0 .... = Frame Pending: False
        .... .... ..1. .... = Acknowledge Request: True
        .... .... .0.. .... = Intra-PAN: False
        .... ...0 .... .... = Sequence Number Suppression: False
        .... ..0. .... .... = Information Elements present: False
        .... 11.. .... .... = Destination Addressing Mode:
                              Long/64-bit (0x0003)
        ..10 .... .... .... = Frame Version: 2
        11.. .... .... .... = Source Addressing Mode:
                              Long/64-bit (0x0003)
    Sequence Number: 222
    Destination PAN: 0xcafe
    Destination: 14:15:92:cc:00:00:00:02 (14:15:92:cc:00:00:00:02)
    Extended Source: 14:15:92:cc:00:00:00:01 (14:15:92:cc:00:00:00:01)
    FCS: 0xd5d8 (Correct)
6LoWPAN
    IPHC Header
        011. .... = Pattern: IP header compression (0x03)
        ...1 1... .... .... = Traffic class and flow label: Version,
                              traffic class,
```

```
                              and flow label compressed (0x0003)
        .... .1.. .... .... = Next header: Compressed
        .... ..00 .... .... = Hop limit: Inline (0x0000)
        .... .... 0... .... = Context identifier extension: False
        .... .... .0.. .... = Source address compression: Stateless
        .... .... ..00 .... = Source address mode: Inline (0x0000)
        .... .... .... 0... = Multicast address compression: False
        .... .... .... .0.. = Destination address compression: Stateless
        .... .... .... ..00 = Destination address mode: Inline (0x0000)
    Hop limit: 128
    Source: bbbb::1 (bbbb::1)
    Destination: bbbb::1415:92cc:0:2 (bbbb::1415:92cc:0:2)
    IPv6 extension header
        1110 .... = Pattern: IPv6 extension header (0x0e)
        .... 001. = Header ID: IPv6 routing (0x01)
        .... ...1 = Next header: Compressed
        Header length: 14
        Data (14 bytes)

0000  03 01 88 00 00 00 14 15 92 cc 00 00 00 03         ..............
            Data: 030188000000141592cc00000003
            [Length: 14]
    IPv6 extension header
        1110 .... = Pattern: IPv6 extension header (0x0e)
        .... 111. = Header ID: IPv6 header (0x07)
        .... ...0 = Next header: Inline
    IPHC Header
        011. .... = Pattern: IP header compression (0x03)
        ...1 1... .... .... = Traffic class and flow label: Version,
                              traffic class,
                              and flow label compressed (0x0003)
        .... .0.. .... .... = Next header: Inline
        .... ..00 .... .... = Hop limit: Inline (0x0000)
        .... .... 0... .... = Context identifier extension: False
        .... .... .0.. .... = Source address compression: Stateless
        .... .... ..11 .... = Source address mode: Compressed (0x0003)
        .... .... .... 0... = Multicast address compression: False
        .... .... .... .0.. = Destination address compression: Stateless
        .... .... .... ..11 = Destination address mode:
                              Compressed (0x0003)
        [Source context: fe80:: (fe80::)]
        [Destination context: fe80:: (fe80::)]
    Next header: ICMPv6 (0x3a)
    Hop limit: 128
    Source: fe80::1 (fe80::1)
    Destination: fe80::1415:92cc:0:2 (fe80::1415:92cc:0:2)
Internet Protocol Version 6,
Src: bbbb::1 (bbbb::1), Dst: bbbb::1415:92cc:0:2 (bbbb::1415:92cc:0:2)
```

```
    0110 .... = Version: 6
        [0110 .... = This field makes the filter
                     "ip.version == 6" possible: 6]
    .... 0000 0000 .... .... .... .... .... = Traffic class: 0x00000000
        .... 0000 00.. .... .... .... .... .... = Differentiated
                                                  Services Field:
                                                  Default (0x00000000)
        .... .... ..0. .... .... .... .... .... = ECN-Capable Transport
                                                  (ECT): Not set
        .... .... ...0 .... .... .... .... .... = ECN-CE: Not set
    .... .... .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
    Payload length: 96
    Next header: IPv6 routing (43)
    Hop limit: 128
    Source: bbbb::1 (bbbb::1)
    Destination: bbbb::1415:92cc:0:2 (bbbb::1415:92cc:0:2)
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
    Routing Header, Type : RPL (3)
        Next Header: IPv6 (41)
        Length: 1 (16 bytes)
        Type: RPL (3)
        Segments Left: 1
        1000 .... .... .... .... .... .... .... = Compressed Internal
                                                  Octets (CmprI): 8
        .... 1000 .... .... .... .... .... .... = Compressed Final
                                                  Octets (CmprE): 8
        .... .... 0000 .... .... .... .... .... = Padding Bytes: 0
        .... .... .... 0000 0000 0000 0000 0000 = Reserved: 0
        [Total Segments: 1]
        Address: 141592cc00000003
        [Full Address: bbbb::1415:92cc:0:3 (bbbb::1415:92cc:0:3)]
Internet Protocol Version 6,
Src: fe80::1 (fe80::1), Dst: fe80::1415:92cc:0:2 (fe80::1415:92cc:0:2)
    0110 .... = Version: 6
        [0110 .... = This field makes the filter
                     "ip.version == 6" possible: 6]
    .... 0000 0000 .... .... .... .... .... = Traffic class: 0x00000000
        .... 0000 00.. .... .... .... .... .... = Differentiated
                                                  Services Field:
                                                  Default (0x00000000)
        .... .... ..0. .... .... .... .... .... = ECN-Capable Transport
                                                  (ECT): Not set
        .... .... ...0 .... .... .... .... .... = ECN-CE: Not set
    .... .... .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
    Payload length: 40
    Next header: ICMPv6 (58)
    Hop limit: 128
```

```
    Source: fe80::1 (fe80::1)
    Destination: fe80::1415:92cc:0:2 (fe80::1415:92cc:0:2)
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
Internet Control Message Protocol v6
    Type: Echo (ping) request (128)
    Code: 0
    Checksum: 0xb681 [incorrect, should be 0x30f8]
        [Expert Info (Warn/Checksum): ICMPv6 Checksum Incorrect,
        should be 0x30f8]
            [ICMPv6 Checksum Incorrect, should be 0x30f8]
            [Severity level: Warn]
            [Group: Checksum]
    Identifier: 0x0001
    Sequence: 26
    [No response seen]
        [Expert Info (Warn/Sequence):
        No response seen to ICMPv6 request in frame 790]
            [No response seen to ICMPv6 request in frame 790]
            [Severity level: Warn]
            [Group: Sequence]
    Data (32 bytes)

0000  61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70   abcdefghijklmnop
0010  71 72 73 74 75 76 77 61 62 63 64 65 66 67 68 69   qrstuvwabcdefghi
        Data: 6162636465666768696a6b6c6d6e6f7071727374757677761...
        [Length: 32]

== Raw Bytes ==

0000   21 ec de fe ca 02 00 00 00 cc 92 15 14 01 00 00  !...............
0010   00 cc 92 15 14 7c 00 80 bb bb 00 00 00 00 00 00  .....|..........
0020   00 00 00 00 00 00 00 01 bb bb 00 00 00 00 00 00  ................
0030   14 15 92 cc 00 00 00 02 e3 0e 03 01 88 00 00 00  ................
0040   14 15 92 cc 00 00 00 03 ee 78 33 3a 80 80 00 b6  .........x3:....
0050   81 00 01 00 1a 61 62 63 64 65 66 67 68 69 6a 6b  .....abcdefghijk
0060   6c 6d 6e 6f 70 71 72 73 74 75 76 77 61 62 63 64  lmnopqrstuvwabcd
0070   65 66 67 68 69 d8 d5                             efghi..

   [ping 3] ICMPv6 echo request 2->3

== Dissected packet ==

IEEE 802.15.4 Data,
Dst: 14:15:92:cc:00:00:00:03, Src: 14:15:92:cc:00:00:00:02
    Frame Control Field: 0xec21, Frame Type: Data, Acknowledge Request,
    Destination Addressing Mode: Long/64-bit,
    Source Addressing Mode: Long/64-bit
```

```
        .... .... .... .001 = Frame Type: Data (0x0001)
        .... .... .... 0... = Security Enabled: False
        .... .... ...0 .... = Frame Pending: False
        .... .... ..1. .... = Acknowledge Request: True
        .... .... .0.. .... = Intra-PAN: False
        .... ...0 .... .... = Sequence Number Suppression: False
        .... ..0. .... .... = Information Elements present: False
        .... 11.. .... .... = Destination Addressing Mode:
                              Long/64-bit (0x0003)
        ..10 .... .... .... = Frame Version: 2
        11.. .... .... .... = Source Addressing Mode:
                              Long/64-bit (0x0003)
    Sequence Number: 115
    Destination PAN: 0xcafe
    Destination: 14:15:92:cc:00:00:00:03 (14:15:92:cc:00:00:00:03)
    Extended Source: 14:15:92:cc:00:00:00:02 (14:15:92:cc:00:00:00:02)
    FCS: 0x469e (Correct)
6LoWPAN
    IPHC Header
        011. .... = Pattern: IP header compression (0x03)
        ...1 1... .... .... = Traffic class and flow label: Version,
                              traffic class,
                              and flow label compressed (0x0003)
        .... .1.. .... .... = Next header: Compressed
        .... ..00 .... .... = Hop limit: Inline (0x0000)
        .... .... 0... .... = Context identifier extension: False
        .... .... .0.. .... = Source address compression: Stateless
        .... .... ..01 .... = Source address mode:
                              64-bits inline (0x0001)
        .... .... .... 0... = Multicast address compression: False
        .... .... .... .0.. = Destination address compression: Stateless
        .... .... .... ..11 = Destination address mode:
                              Compressed (0x0003)
        [Source context: fe80:: (fe80::)]
        [Destination context: fe80:: (fe80::)]
    Hop limit: 127
    Source: fe80::1 (fe80::1)
    Destination: fe80::1615:92cc:0:3 (fe80::1615:92cc:0:3)
    IPv6 extension header
        1110 .... = Pattern: IPv6 extension header (0x0e)
        .... 001. = Header ID: IPv6 routing (0x01)
        .... ...1 = Next header: Compressed
        Header length: 14
        Data (14 bytes)

0000  03 00 88 00 00 00 14 15 92 cc 00 00 00 03         ..............
            Data: 0300880000000141592cc00000003
            [Length: 14]
```

```
    IPv6 extension header
        1110 .... = Pattern: IPv6 extension header (0x0e)
        .... 111. = Header ID: IPv6 header (0x07)
        .... ...0 = Next header: Inline
    IPHC Header
        011. .... = Pattern: IP header compression (0x03)
        ...1 1... .... .... = Traffic class and flow label: Version,
                              traffic class,
                              and flow label compressed (0x0003)
        .... .0.. .... .... = Next header: Inline
        .... ..00 .... .... = Hop limit: Inline (0x0000)
        .... .... 0... .... = Context identifier extension: False
        .... .... .0.. .... = Source address compression: Stateless
        .... .... ..11 .... = Source address mode: Compressed (0x0003)
        .... .... .... 0... = Multicast address compression: False
        .... .... .... .0.. = Destination address compression: Stateless
        .... .... .... ..11 = Destination address mode:
                              Compressed (0x0003)
        [Source context: fe80:: (fe80::)]
        [Destination context: fe80:: (fe80::)]
    Next header: ICMPv6 (0x3a)
    Hop limit: 128
    Source: fe80::1 (fe80::1)
    Destination: fe80::1615:92cc:0:3 (fe80::1615:92cc:0:3)
Internet Protocol Version 6,
Src: fe80::1 (fe80::1), Dst: fe80::1615:92cc:0:3 (fe80::1615:92cc:0:3)
    0110 .... = Version: 6
        [0110 .... = This field makes the filter
                     "ip.version == 6" possible: 6]
    .... 0000 0000 .... .... .... .... .... = Traffic class: 0x00000000
        .... 0000 00.. .... .... .... .... .... = Differentiated
                                                  Services Field:
                                                  Default (0x00000000)
        .... .... ..0. .... .... .... .... .... = ECN-Capable Transport
                                                  (ECT): Not set
        .... .... ...0 .... .... .... .... .... = ECN-CE: Not set
    .... .... .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
    Payload length: 96
    Next header: IPv6 routing (43)
    Hop limit: 127
    Source: fe80::1 (fe80::1)
    Destination: fe80::1615:92cc:0:3 (fe80::1615:92cc:0:3)
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
    Routing Header, Type : RPL (3)
        Next Header: IPv6 (41)
        Length: 1 (16 bytes)
        Type: RPL (3)
```

```
        Segments Left: 0
        1000 .... .... .... .... .... .... .... = Compressed Internal
                                                  Octets (CmprI): 8
        .... 1000 .... .... .... .... .... .... = Compressed Final
                                                  Octets (CmprE): 8
        .... .... 0000 .... .... .... .... .... = Padding Bytes: 0
        .... .... .... 0000 0000 0000 0000 0000 = Reserved: 0
        [Total Segments: 1]
        Address: 141592cc00000003
        [Full Address: fe80::1415:92cc:0:3 (fe80::1415:92cc:0:3)]
Internet Protocol Version 6,
Src: fe80::1 (fe80::1), Dst: fe80::1615:92cc:0:3 (fe80::1615:92cc:0:3)
    0110 .... = Version: 6
        [0110 .... = This field makes the filter
                     "ip.version == 6" possible: 6]
    .... 0000 0000 .... .... .... .... .... = Traffic class: 0x00000000
        .... 0000 00.. .... .... .... .... .... = Differentiated
                                                  Services Field:
                                                  Default (0x00000000)
        .... .... ..0. .... .... .... .... .... = ECN-Capable Transport
                                                  (ECT): Not set
        .... .... ...0 .... .... .... .... .... = ECN-CE: Not set
    .... .... .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
    Payload length: 40
    Next header: ICMPv6 (58)
    Hop limit: 128
    Source: fe80::1 (fe80::1)
    Destination: fe80::1615:92cc:0:3 (fe80::1615:92cc:0:3)
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
Internet Control Message Protocol v6
    Type: Echo (ping) request (128)
    Code: 0
    Checksum: 0xb681 [incorrect, should be 0x2ef7]
        [Expert Info (Warn/Checksum): ICMPv6 Checksum Incorrect,
        should be 0x2ef7]
            [ICMPv6 Checksum Incorrect, should be 0x2ef7]
            [Severity level: Warn]
            [Group: Checksum]
    Identifier: 0x0001
    Sequence: 26
    [No response seen]
        [Expert Info (Warn/Sequence):
        No response seen to ICMPv6 request in frame 795]
            [No response seen to ICMPv6 request in frame 795]
            [Severity level: Warn]
            [Group: Sequence]
    Data (32 bytes)
```

```
0000  61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70   abcdefghijklmnop
0010  71 72 73 74 75 76 77 61 62 63 64 65 66 67 68 69   qrstuvwabcdefghi
        Data: 6162636465666768696a6b6c6d6e6f7071727374757677761...
        [Length: 32]

== Raw Bytes ==

0000   21 ec 73 fe ca 03 00 00 00 cc 92 15 14 02 00 00  !.s.............
0010   00 cc 92 15 14 7c 13 7f 00 00 00 00 00 00 00 01  .....|..........
0020   e3 0e 03 00 88 00 00 00 14 15 92 cc 00 00 00 03  ................
0030   ee 78 33 3a 80 80 00 b6 81 00 01 00 1a 61 62 63  .x3:.........abc
0040   64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73  defghijklmnopqrs
0050   74 75 76 77 61 62 63 64 65 66 67 68 69 9e 46     tuvwabcdefghi.F

   [ping 3] ICMPv6 echo reply 3->2

== Dissected packet ==

IEEE 802.15.4 Data,
Dst: 14:15:92:cc:00:00:00:02, Src: 14:15:92:cc:00:00:00:03
    Frame Control Field: 0xec21, Frame Type: Data, Acknowledge Request,
    Destination Addressing Mode: Long/64-bit,
    Source Addressing Mode: Long/64-bit
        .... .... .... .001 = Frame Type: Data (0x0001)
        .... .... .... 0... = Security Enabled: False
        .... .... ...0 .... = Frame Pending: False
        .... .... ..1. .... = Acknowledge Request: True
        .... .... .0.. .... = Intra-PAN: False
        .... ...0 .... .... = Sequence Number Suppression: False
        .... ..0. .... .... = Information Elements present: False
        .... 11.. .... .... = Destination Addressing Mode:
                              Long/64-bit (0x0003)
        ..10 .... .... .... = Frame Version: 2
        11.. .... .... .... = Source Addressing Mode:
                              Long/64-bit (0x0003)
    Sequence Number: 177
    Destination PAN: 0xcafe
    Destination: 14:15:92:cc:00:00:00:02 (14:15:92:cc:00:00:00:02)
    Extended Source: 14:15:92:cc:00:00:00:03 (14:15:92:cc:00:00:00:03)
    FCS: 0x183c (Correct)
6LoWPAN
    IPHC Header
        011. .... = Pattern: IP header compression (0x03)
        ...1 1... .... .... = Traffic class and flow label: Version,
                              traffic class,
                              and flow label compressed (0x0003)
        .... .1.. .... .... = Next header: Compressed
        .... ..00 .... .... = Hop limit: Inline (0x0000)
```

```
        .... .... 0... .... = Context identifier extension: False
        .... .... .0.. .... = Source address compression: Stateless
        .... .... ..01 .... = Source address mode:
                              64-bits inline (0x0001)
        .... .... .... 0... = Multicast address compression: False
        .... .... .... .0.. = Destination address compression: Stateless
        .... .... .... ..01 = Destination address mode:
                              64-bits inline (0x0001)
        [Source context: fe80:: (fe80::)]
        [Destination context: fe80:: (fe80::)]
    Hop limit: 64
    Source: fe80::1415:92cc:0:3 (fe80::1415:92cc:0:3)
    Destination: fe80::1 (fe80::1)
    IPv6 extension header
        1110 .... = Pattern: IPv6 extension header (0x0e)
        .... 000. = Header ID: IPv6 hop-by-hop options (0x00)
        .... ...1 = Next header: Compressed
        Header length: 6
        Data (6 bytes)

0000  63 04 00 00 ad 05                                 c.....
            Data: 63040000ad05
            [Length: 6]
    IPv6 extension header
        1110 .... = Pattern: IPv6 extension header (0x0e)
        .... 111. = Header ID: IPv6 header (0x07)
        .... ...0 = Next header: Inline
    IPHC Header
        011. .... = Pattern: IP header compression (0x03)
        ...1 1... .... .... = Traffic class and flow label: Version,
                              traffic class,
                              and flow label compressed (0x0003)
        .... .0.. .... .... = Next header: Inline
        .... ..10 .... .... = Hop limit: 64 (0x0002)
        .... .... 0... .... = Context identifier extension: False
        .... .... .0.. .... = Source address compression: Stateless
        .... .... ..01 .... = Source address mode:
                              64-bits inline (0x0001)
        .... .... .... 0... = Multicast address compression: False
        .... .... .... .0.. = Destination address compression: Stateless
        .... .... .... ..11 = Destination address mode:
                              Compressed (0x0003)
        [Source context: fe80:: (fe80::)]
        [Destination context: fe80:: (fe80::)]
    Next header: ICMPv6 (0x3a)
    Source: fe80::1415:92cc:0:3 (fe80::1415:92cc:0:3)
    Destination: fe80::1 (fe80::1)
Internet Protocol Version 6,
```

```
Src: fe80::1415:92cc:0:3 (fe80::1415:92cc:0:3), Dst: fe80::1 (fe80::1)
    0110 .... = Version: 6
        [0110 .... = This field makes the filter
                     "ip.version == 6" possible: 6]
    .... 0000 0000 .... .... .... .... .... = Traffic class: 0x00000000
        .... 0000 00.. .... .... .... .... .... = Differentiated
                                                  Services Field:
                                                  Default (0x00000000)
        .... .... ..0. .... .... .... .... .... = ECN-Capable Transport
                                                  (ECT): Not set
        .... .... ...0 .... .... .... .... .... = ECN-CE: Not set
    .... .... .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
    Payload length: 88
    Next header: IPv6 hop-by-hop option (0)
    Hop limit: 64
    Source: fe80::1415:92cc:0:3 (fe80::1415:92cc:0:3)
    Destination: fe80::1 (fe80::1)
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
    Hop-by-Hop Option
        Next header: IPv6 (41)
        Length: 0 (8 bytes)
        IPv6 Option (RPL Option)
            Type: RPL Option (99)
            Length: 4
            Flag: 0x00
                0... .... = Down: False
                .0.. .... = Rank Error: False
                ..0. .... = Forwarding Error: False
                ...0 0000 = Reserved: 0x00
            RPLInstanceID: 0x00
            Sender Rank: 0xad05
Internet Protocol Version 6,
Src: fe80::1415:92cc:0:3 (fe80::1415:92cc:0:3), Dst: fe80::1 (fe80::1)
    0110 .... = Version: 6
        [0110 .... = This field makes the filter
                     "ip.version == 6" possible: 6]
    .... 0000 0000 .... .... .... .... .... = Traffic class: 0x00000000
        .... 0000 00.. .... .... .... .... .... = Differentiated
                                                  Services Field:
                                                  Default (0x00000000)
        .... .... ..0. .... .... .... .... .... = ECN-Capable Transport
                                                  (ECT): Not set
        .... .... ...0 .... .... .... .... .... = ECN-CE: Not set
    .... .... .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
    Payload length: 40
    Next header: ICMPv6 (58)
    Hop limit: 64
```

```
    Source: fe80::1415:92cc:0:3 (fe80::1415:92cc:0:3)
    Destination: fe80::1 (fe80::1)
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
Internet Control Message Protocol v6
    Type: Echo (ping) reply (129)
    Code: 0
    Checksum: 0xb581 [incorrect, should be 0x2ff7]
        [Expert Info (Warn/Checksum): ICMPv6 Checksum Incorrect,
                                      should be 0x2ff7]
            [ICMPv6 Checksum Incorrect, should be 0x2ff7]
            [Severity level: Warn]
            [Group: Checksum]
    Identifier: 0x0001
    Sequence: 26
    Data (32 bytes)

0000  61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70   abcdefghijklmnop
0010  71 72 73 74 75 76 77 61 62 63 64 65 66 67 68 69   qrstuvwabcdefghi
        Data: 6162636465666768696a6b6c6d6e6f7071727374757677761...
        [Length: 32]

== Raw Bytes ==

0000   21 ec b1 fe ca 02 00 00 00 cc 92 15 14 03 00 00  !...............
0010   00 cc 92 15 14 7c 11 40 14 15 92 cc 00 00 00 03  .....|.@........
0020   00 00 00 00 00 00 00 01 e1 06 63 04 00 00 ad 05  ..........c.....
0030   ee 7a 13 3a 14 15 92 cc 00 00 00 03 81 00 b5 81  .z.:............
0040   00 01 00 1a 61 62 63 64 65 66 67 68 69 6a 6b 6c  ....abcdefghijkl
0050   6d 6e 6f 70 71 72 73 74 75 76 77 61 62 63 64 65  mnopqrstuvwabcde
0060   66 67 68 69 3c 18                                fghi.

   [ping 3] ICMPv6 echo reply 2->1

== Dissected packet ==

IEEE 802.15.4 Data, Dst: 14:15:92:cc:00:00:00:01,
Src: 14:15:92:cc:00:00:00:02
    Frame Control Field: 0xec21, Frame Type: Data, Acknowledge Request,
    Destination Addressing Mode: Long/64-bit, Source Addressing Mode:
    Long/64-bit
        .... .... .... .001 = Frame Type: Data (0x0001)
        .... .... .... 0... = Security Enabled: False
        .... .... ...0 .... = Frame Pending: False
        .... .... ..1. .... = Acknowledge Request: True
        .... .... .0.. .... = Intra-PAN: False
        .... ...0 .... .... = Sequence Number Suppression: False
        .... ..0. .... .... = Information Elements present: False
```

```
        .... 11.. .... .... = Destination Addressing Mode:
                              Long/64-bit (0x0003)
        ..10 .... .... .... = Frame Version: 2
        11.. .... .... .... = Source Addressing Mode:
                              Long/64-bit (0x0003)
    Sequence Number: 118
    Destination PAN: 0xcafe
    Destination: 14:15:92:cc:00:00:00:01 (14:15:92:cc:00:00:00:01)
    Extended Source: 14:15:92:cc:00:00:00:02 (14:15:92:cc:00:00:00:02)
    FCS: 0x9f5a (Correct)
6LoWPAN
    IPHC Header
        011. .... = Pattern: IP header compression (0x03)
        ...1 1... .... .... = Traffic class and flow label: Version,
                              traffic class,
                              and flow label compressed (0x0003)
        .... .1.. .... .... = Next header: Compressed
        .... ..00 .... .... = Hop limit: Inline (0x0000)
        .... .... 0... .... = Context identifier extension: False
        .... .... .0.. .... = Source address compression: Stateless
        .... .... ..01 .... = Source address mode:
                              64-bits inline (0x0001)
        .... .... .... 0... = Multicast address compression: False
        .... .... .... .0.. = Destination address compression: Stateless
        .... .... .... ..01 = Destination address mode:
                              64-bits inline (0x0001)
        [Source context: fe80:: (fe80::)]
        [Destination context: fe80:: (fe80::)]
    Hop limit: 63
    Source: fe80::1415:92cc:0:3 (fe80::1415:92cc:0:3)
    Destination: fe80::1 (fe80::1)
    IPv6 extension header
        1110 .... = Pattern: IPv6 extension header (0x0e)
        .... 000. = Header ID: IPv6 hop-by-hop options (0x00)
        .... ...1 = Next header: Compressed
        Header length: 6
        Data (6 bytes)

0000  63 04 00 00 36 03                                 c...6.
            Data: 630400003603
            [Length: 6]
    IPv6 extension header
        1110 .... = Pattern: IPv6 extension header (0x0e)
        .... 111. = Header ID: IPv6 header (0x07)
        .... ...0 = Next header: Inline
    IPHC Header
        011. .... = Pattern: IP header compression (0x03)
        ...1 1... .... .... = Traffic class and flow label: Version,
```

```
                              traffic class,
                              and flow label compressed (0x0003)
        .... .0.. .... .... = Next header: Inline
        .... ..10 .... .... = Hop limit: 64 (0x0002)
        .... .... 0... .... = Context identifier extension: False
        .... .... .0.. .... = Source address compression: Stateless
        .... .... ..01 .... = Source address mode:
                              64-bits inline (0x0001)
        .... .... .... 0... = Multicast address compression: False
        .... .... .... .0.. = Destination address compression: Stateless
        .... .... .... ..11 = Destination address mode:
                              Compressed (0x0003)
        [Source context: fe80:: (fe80::)]
        [Destination context: fe80:: (fe80::)]
    Next header: ICMPv6 (0x3a)
    Source: fe80::1415:92cc:0:3 (fe80::1415:92cc:0:3)
    Destination: fe80::1 (fe80::1)
Internet Protocol Version 6,
Src: fe80::1415:92cc:0:3 (fe80::1415:92cc:0:3), Dst: fe80::1 (fe80::1)
    0110 .... = Version: 6
        [0110 .... = This field makes the filter
                     "ip.version == 6" possible: 6]
    .... 0000 0000 .... .... .... .... .... = Traffic class: 0x00000000
        .... 0000 00.. .... .... .... .... .... = Differentiated
                                                  Services Field:
                                                  Default (0x00000000)
        .... .... ..0. .... .... .... .... .... = ECN-Capable Transport
                                                  (ECT): Not set
        .... .... ...0 .... .... .... .... .... = ECN-CE: Not set
    .... .... .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
    Payload length: 88
    Next header: IPv6 hop-by-hop option (0)
    Hop limit: 63
    Source: fe80::1415:92cc:0:3 (fe80::1415:92cc:0:3)
    Destination: fe80::1 (fe80::1)
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
    Hop-by-Hop Option
        Next header: IPv6 (41)
        Length: 0 (8 bytes)
        IPv6 Option (RPL Option)
            Type: RPL Option (99)
            Length: 4
            Flag: 0x00
                0... .... = Down: False
                .0.. .... = Rank Error: False
                ..0. .... = Forwarding Error: False
                ...0 0000 = Reserved: 0x00
```

```
            RPLInstanceID: 0x00
            Sender Rank: 0x3603
Internet Protocol Version 6,
Src: fe80::1415:92cc:0:3 (fe80::1415:92cc:0:3), Dst: fe80::1 (fe80::1)
    0110 .... = Version: 6
        [0110 .... = This field makes the filter
                     "ip.version == 6" possible: 6]
    .... 0000 0000 .... .... .... .... .... = Traffic class: 0x00000000
        .... 0000 00.. .... .... .... .... .... = Differentiated
                                                  Services Field:
                                                  Default (0x00000000)
        .... .... ..0. .... .... .... .... .... = ECN-Capable Transport
                                                  (ECT): Not set
        .... .... ...0 .... .... .... .... .... = ECN-CE: Not set
    .... .... .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
    Payload length: 40
    Next header: ICMPv6 (58)
    Hop limit: 64
    Source: fe80::1415:92cc:0:3 (fe80::1415:92cc:0:3)
    Destination: fe80::1 (fe80::1)
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
Internet Control Message Protocol v6
    Type: Echo (ping) reply (129)
    Code: 0
    Checksum: 0xb581 [incorrect, should be 0x2ff7]
        [Expert Info (Warn/Checksum): ICMPv6 Checksum Incorrect,
                                      should be 0x2ff7]
            [ICMPv6 Checksum Incorrect, should be 0x2ff7]
            [Severity level: Warn]
            [Group: Checksum]
    Identifier: 0x0001
    Sequence: 26
    Data (32 bytes)

0000  61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70   abcdefghijklmnop
0010  71 72 73 74 75 76 77 61 62 63 64 65 66 67 68 69   qrstuvwabcdefghi
        Data: 6162636465666768696a6b6c6d6e6f7071727374757677761...
        [Length: 32]

== Raw Bytes ==

0000   21 ec 76 fe ca 01 00 00 00 cc 92 15 14 02 00 00  !.v.............
0010   00 cc 92 15 14 7c 11 3f 14 15 92 cc 00 00 00 03  .....|.?........
0020   00 00 00 00 00 00 00 01 e1 06 63 04 00 00 36 03  ..........c...6.
0030   ee 7a 13 3a 14 15 92 cc 00 00 00 03 81 00 b5 81  .z.:............
0040   00 01 00 1a 61 62 63 64 65 66 67 68 69 6a 6b 6c  ....abcdefghijkl
0050   6d 6e 6f 70 71 72 73 74 75 76 77 61 62 63 64 65  mnopqrstuvwabcde
```

```
0060   66 67 68 69 5a 9f                                fghiZ.
```

4. IANA Considerations

This memo includes no request to IANA.

5. Security Considerations

This memo only presents example packets exchanged. It does not define any protocol; there are hence no security considerations in this document.

6. Acknowledgments

The authors would like to thank the OpenWSN community, the 6TiSCH working group and the participants at the 6TiSCH plugtests for there feedback which has helped shape this document.

7. References

7.1. Normative References

[I-D.ietf-6tisch-minimal]
Vilajosana, X. and K. Pister, "Minimal 6TiSCH Configuration", draft-ietf-6tisch-minimal-10 (work in progress), June 2015.

7.2. External Informative References

[OpenWSN] Watteyne, T., Vilajosana, X., Kerkez, B., Chraim, F., Weekly, K., Wang, Q., Glaser, S., and K. Pister, "OpenWSN: a Standards-Based Low-Power Wireless Development Environment", Transactions on Emerging Telecommunications Technologies , August 2012.

Authors’ Addresses

Jonathan Munoz (editor)
Telecom Bretagne
2, rue de la Chataigneraie
Cesson Sevigne 35576
France

Email: jonathan.munoz@telecom-bretagne.eu

Guillaume Gaillard
Orange Labs
28 Chemin du Vieux Chene
Meylan 38240
France

Email: guillaume.gaillard@orange.com

Dominique Barthel
Orange Labs
28 Chemin du Vieux Chene
Meylan 38240
France

Email: dominique.barthel@orange.com

6TiSCH P. Thubert, Ed.
Internet-Draft Cisco
Intended status: Informational June 11, 2015
Expires: December 13, 2015

6TiSCH requirements for DetNet
draft-thubert-6tisch-4detnet-01

Abstract

This document builds on the 6TiSCH architecture that defines, among others, mechanisms to establish and maintain deterministic routing and scheduling in a centralized fashion. The document details dependencies on DetNet and PCE controller to express topologies and capabilities, as well as abstract state that the controller must be able to program into the network devices to enable deterministic forwarding operations.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The emergence of wireless technology has enabled a variety of new devices to get interconnected, at a very low marginal cost per device, at any distance ranging from Near Field to interplanetary, and in circumstances where wiring may not be practical, for instance on fast-moving or rotating devices.

At the same time, a new breed of Time Sensitive Networks is being developed to enable traffic that is highly sensitive to jitter, quite sensitive to latency, and with a high degree of operational criticality so that loss should be minimized at all times. Such traffic is not limited to professional Audio/ Video networks, but is also found in command and control operations such as industrial automation and vehicular sensors and actuators.

At IEEE802.1, the Audio/Video Task Group [IEEE802.1TSNTG] Time Sensitive Networking (TSN) to address Deterministic Ethernet. The Medium access Control (MAC) of IEEE802.15.4 [IEEE802154] has evolved with the new TimeSlotted Channel Hopping (TSCH) [I-D.ietf-6tisch-tsch] mode for deterministic industrial-type applications. TSCH was introduced with the IEEE802.15.4e [IEEE802154e] amendment and will be wrapped up in the next revision of the IEEE802.15.4 standard. For all practical purpose, this document is expected to be insensitive to the future versions of the IEEE802.15.4 standard, which is thus referenced undated.

Though at a different time scale, both TSN and TSCH standards provide Deterministic capabilities to the point that a packet that pertains to a certain flow crosses the network from node to node following a very precise schedule, as a train that leaves intermediate stations at precise times along its path. With TSCH, time is formatted into timeSlots, and an individual cell is allocated to unicast or broadcast communication at the MAC level. The time-slotted operation reduces collisions, saves energy, and enables to more closely engineer the network for deterministic properties. The channel hopping aspect is a simple and efficient technique to combat multi-path fading and co-channel interferences (for example by Wi-Fi emitters).

The 6TiSCH Architecture [I-D.ietf-6tisch-architecture] defines a remote monitoring and scheduling management of a TSCH network by a Path Computation Element (PCE), which cooperates with an abstract Network Management Entity (NME) to manage timeSlots and device resources in a manner that minimizes the interaction with and the load placed on the constrained devices.

This Architecture applies the concepts of Deterministic Networking on a TSCH network to enable the switching of timeSlots in a G-MPLS manner. This document details the dependencies that 6TiSCH has on PCE [PCE] and DetNet [I-D.finn-detnet-architecture] to provide the necessary capabilities that may be specific to such networks. In turn, DetNet is expected to integrate and maintain consistency with the work that has taken place and is continuing at IEEE802.1TSN and AVnu.

2. Terminology

Readers are expected to be familiar with all the terms and concepts that are discussed in "Multi-link Subnet Support in IPv6" [I-D.ietf-ipv6-multilink-subnets].

The draft uses terminology defined or referenced in [I-D.ietf-6tisch-terminology] and [I-D.ietf-roll-rpl-industrial-applicability].

The draft also conforms to the terms and models described in [RFC3444] and uses the vocabulary and the concepts defined in [RFC4291] for the IPv6 Architecture.

3. 6TiSCH Overview

The scope of the present work is a subnet that, in its basic configuration, is made of a TSCH [I-D.ietf-6tisch-tsch] MAC Low Power Lossy Network (LLN).

```
               ---+-------- ............ ------------
                  |      External Network       |
                  |                          +-----+
               +-----+                       | NME |
               |     | LLN Border            |     |
               |     | router                +-----+
               +-----+
             o    o   o
      o     o   o     o
         o   o LLN   o    o     o
            o   o   o       o
                    o
```

Figure 1: Basic Configuration of a 6TiSCH Network

In the extended configuration, a Backbone Router (6BBR) federates multiple 6TiSCH in a single subnet over a backbone. 6TiSCH 6BBRs synchronize with one another over the backbone, so as to ensure that the multiple LLNs that form the IPv6 subnet stay tightly synchronized.

```
                  ---+-------- ............ -----------
                     |      External Network       |
                     |                          +-----+
                     |             +-----+      | NME |
                  +-----+          |  +-----+   |     |
                  |     | Router   |  | PCE |   +-----+
                  |     |          +--|     |
                  +-----+             +-----+
                     |                   |
                     | Subnet Backbone   |
               +--------------------+------------------+
               |                    |                  |
            +-----+             +-----+             +-----+
            |     | Backbone    |     | Backbone    |     | Backbone
       o    |     | router      |     | router      |     | router
            +-----+             +-----+             +-----+
       o                  o                   o                 o   o
           o    o   o         o   o  o   o         o  o   o    o
      o             o        o  LLN      o      o         o      o
         o   o    o      o      o o     o  o   o    o    o     o
```

Figure 2: Extended Configuration of a 6TiSCH Network

If the Backbone is Deterministic, then the Backbone Router ensures that the end-to-end deterministic behavior is maintained between the LLN and the backbone. This SHOULD be done in conformance to the DetNet Architecture [I-D.finn-detnet-architecture] which studies Layer-3 aspects of Deterministic Networks, and covers networks that span multiple Layer-2 domains. One particular requirement is that the PCE MUST be able to compute a deterministic path and to end across the TSCH network and an IEEE802.1 TSN Ethernet backbone, and DetNet MUST enable end-to-end deterministic forwarding.

6TiSCH defines the concept of a Track, which is a complex form of a uni-directional Circuit ([I-D.ietf-6tisch-terminology]). As opposed to a simple circuit that is a sequence of nodes and links, a Track is shaped as a directed acyclic graph towards a destination to support multi-path forwarding and route around failures. A Track may also branch off and rejoin, for the purpose of the so-called Packet Replication and Elimination (PRE), over non congruent branches. PRE may be used to complement layer-2 Automatic Repeat reQuest (ARQ) to meet industrial expectations in Packet Delivery Ratio (PDR), in particular when the Track extends beyond the 6TiSCH network.

```
                    +-----+
                    | IoT |
                    | G/W |
                    +-----+
                       ^  <---- Elimination
                      | |
       Track branch   | |
              +-------+ +--------+ Subnet Backbone
              |                  |
           +--|--+            +--|--+
           |  |  | Backbone   |  |  | Backbone
      o    |  |  | router     |  |  | router
           +--/--+            +--|--+
      o     /    o     o---o----/       o
          o    o---o--/   o      o   o  o   o
     o     \  /     o               o   LLN    o
        o   v  <---- Replication
            o
```

Figure 3: End-to-End deterministic Track

In the example above, a Track is laid out from a field device in a 6TiSCH network to an IoT gateway that is located on a IEEE802.1 TSN backbone.

The Replication function in the field device sends a copy of each packet over two different branches, and the PCE schedules each hop of both branches so that the two copies arrive in due time at the gateway. In case of a loss on one branch, hopefully the other copy of the packet still makes it in due time. If two copies make it to the IoT gateway, the Elimination function in the gateway ignores the extra packet and presents only one copy to upper layers.

At each 6TiSCH hop along the Track, the PCE may schedule more than one timeSlot for a packet, so as to support Layer-2 retries (ARQ). It is also possible that the field device only uses the second branch if sending over the first branch fails.

In current deployments, a TSCH Track does not necessarily support PRE but is systematically multi-path. This means that a Track is scheduled so as to ensure that each hop has at least two forwarding solutions, and the forwarding decision is to try the preferred one and use the other in case of Layer-2 transmission failure as detected by ARQ.

3.1. TSCH and 6top

6top is a logical link control sitting between the IP layer and the TSCH MAC layer, which provides the link abstraction that is required for IP operations. The 6top operations are specified in [I-D.wang-6tisch-6top-sublayer].

The 6top data model and management interfaces are further discussed in [I-D.ietf-6tisch-6top-interface] and [I-D.ietf-6tisch-coap].

The architecture defines "soft" cells and "hard" cells. "Hard" cells are owned and managed by an separate scheduling entity (e.g. a PCE) that specifies the slotOffset/channelOffset of the cells to be added/moved/deleted, in which case 6top can only act as instructed, and may not move hard cells in the TSCH schedule on its own.

3.2. SlotFrames and Priorities

A slotFrame is the base object that the PCE needs to manipulate to program a schedule into an LLN node. Elaboration on that concept can be fond in section "SlotFrames and Priorities" of [I-D.ietf-6tisch-architecture]

IEEE802.15.4 TSCH avoids contention on the medium by formatting time and frequencies in cells of transmission of equal duration. In order to describe that formatting of time and frequencies, the 6TiSCH architecture defines a global concept that is called a Channel Distribution and Usage (CDU) matrix; a CDU matrix is a matrix of cells with an height equal to the number of available channels (indexed by ChannelOffsets) and a width (in timeSlots) that is the period of the network scheduling operation (indexed by slotOffsets) for that CDU matrix. The size of a cell is a timeSlot duration, and values of 10 to 15 milliseconds are typical in 802.15.4 TSCH to accommodate for the transmission of a frame and an ack, including the security validation on the receive side which may take up to a few milliseconds on some device architecture.

The frequency used by a cell in the matrix rotates in a pseudo-random fashion, from an initial position at an epoch time, as the matrix iterates over and over.

A CDU matrix is computed by the PCE, but unallocated timeSlots may be used opportunistically by the nodes for classical best effort IP traffic. The PCE has precedence in the allocation in case of a conflict.

In a given network, there might be multiple CDU matrices that operate with different width, so they have different durations and represent different periodic operations. It is recommended that all CDU matrices in a 6TiSCH domain operate with the same cell duration and are aligned, so as to reduce the chances of interferences from slotted-aloha operations. The PCE MUST compute the CDU matrices and shared that knowledge with all the nodes. The matrices are used in particular to define slotFrames.

A slotFrame is a MAC-level abstraction that is common to all nodes and contains a series of timeSlots of equal length and precedence. It is characterized by a slotFrame_ID, and a slotFrame_size. A slotFrame aligns to a CDU matrix for its parameters, such as number and duration of timeSlots.

Multiple slotFrames can coexist in a node schedule, i.e., a node can have multiple activities scheduled in different slotFrames, based on the precedence of the 6TiSCH topologies. The slotFrames may be aligned to different CDU matrices and thus have different width. There is typically one slotFrame for scheduled traffic that has the highest precedence and one or more slotFrame(s) for RPL traffic. The timeSlots in the slotFrame are indexed by the SlotOffset; the first cell is at SlotOffset 0.

The 6TiSCH architecture introduces the concept of chunks ([I-D.ietf-6tisch-terminology]) to operate such spectrum distribution for a whole group of cells at a time. The CDU matrix is formatted into a set of chunks, each of them identified uniquely by a chunk-ID. The PCE MUST compute the partitioning of CDU matrices into chunks and shared that knowledge with all the nodes in a 6TiSCH network.

```
             +-----+-----+-----+-----+-----+-----+-----+     +-----+
chan.Off. 0  |chnkA|chnkP|chnk7|chnkO|chnk2|chnkK|chnk1| ... |chnkZ|
             +-----+-----+-----+-----+-----+-----+-----+     +-----+
chan.Off. 1  |chnkB|chnkQ|chnkA|chnkP|chnk3|chnkL|chnk2| ... |chnk1|
             +-----+-----+-----+-----+-----+-----+-----+     +-----+
               ...
             +-----+-----+-----+-----+-----+-----+-----+     +-----+
chan.Off. 15 |chnkO|chnk6|chnkN|chnk1|chnkJ|chnkZ|chnkI| ... |chnkG|
             +-----+-----+-----+-----+-----+-----+-----+     +-----+
                0     1     2     3     4     5     6           M
```

Figure 4: CDU matrix Partitioning in Chunks

The appropriation of a chunk can be requested explicitly by the PCE to any node. After a successful appropriation, the PCE owns the cells in that chunk, and may use them as hard cells to set up Tracks.

3.3. Schedule Management by a PCE

6TiSCH supports a mixed model of centralized routes and distributed routes. Centralized routes can for example be computed by a entity such as a PCE. Distributed routes are computed by RPL.

Both methods may inject routes in the Routing Tables of the 6TiSCH routers. In either case, each route is associated with a 6TiSCH topology that can be a RPL Instance topology or a track. The 6TiSCH topology is indexed by a Instance ID, in a format that reuses the RPLInstanceID as defined in RPL [RFC6550].

Both RPL and PCE rely on shared sources such as policies to define Global and Local RPLInstanceIDs that can be used by either method. It is possible for centralized and distributed routing to share a same topology. Generally they will operate in different slotFrames, and centralized routes will be used for scheduled traffic and will have precedence over distributed routes in case of conflict between the slotFrames.

Section "Schedule Management Mechanisms" of the 6TiSCH architecture describes 4 paradigms to manage the TSCH schedule of the LLN nodes: Static Scheduling, neighbor-to-neighbor Scheduling, remote monitoring and scheduling management, and Hop-by-hop scheduling. The Track operation for DetNet corresponds to a remote monitoring and scheduling management by a PCE.

The 6top interface document [I-D.ietf-6tisch-6top-interface] specifies the generic data model that can be used to monitor and manage resources of the 6top sublayer. Abstract methods are suggested for use by a management entity in the device. The data model also enables remote control operations on the 6top sublayer.

[I-D.ietf-6tisch-coap] defines an mapping of the 6top set of commands, which is described in [I-D.ietf-6tisch-6top-interface], to CoAP resources. This allows an entity to interact with the 6top layer of a node that is multiple hops away in a RESTful fashion.

[I-D.ietf-6tisch-coap] also defines a basic set CoAP resources and associated RESTful access methods (GET/PUT/POST/DELETE). The payload (body) of the CoAP messages is encoded using the CBOR format. The PCE commands are expected to be issued directly as CoAP requests or to be mapped back and forth into CoAP by a gateway function at the edge of the 6TiSCH network. For instance, it is possible that a mapping entity on the backbone transforms a non-CoAP protocol such as PCEP into the RESTful interfaces that the 6TiSCH devices support. This architecture will be refined to comply with DetNet [I-D.finn-detnet-architecture] when the work is formalized.

3.4. Track Forwarding

By forwarding, this specification means the per-packet operation that allows to deliver a packet to a next hop or an upper layer in this node. Forwarding is based on pre-existing state that was installed as a result of the routing computation of a Track by a PCE. The 6TiSCH architecture supports three different forwarding model, G-MPLS Track Forwarding (TF), 6LoWPAN Fragment Forwarding (FF) and IPv6 Forwarding (6F) which is the classical IP operation. The DetNet case relates to the Track Forwarding operation under the control of a PCE.

A Track is a unidirectional path between a source and a destination. In a Track cell, the normal operation of IEEE802.15.4 Automatic Repeat-reQuest (ARQ) usually happens, though the acknowledgment may be omitted in some cases, for instance if there is no scheduled cell for a retry.

Track Forwarding is the simplest and fastest. A bundle of cells set to receive (RX-cells) is uniquely paired to a bundle of cells that are set to transmit (TX-cells), representing a layer-2 forwarding state that can be used regardless of the network layer protocol. This model can effectively be seen as a Generalized Multi-protocol Label Switching (G-MPLS) operation in that the information used to switch a frame is not an explicit label, but rather related to other properties of the way the packet was received, a particular cell in the case of 6TiSCH. As a result, as long as the TSCH MAC (and Layer-2 security) accepts a frame, that frame can be switched regardless of the protocol, whether this is an IPv6 packet, a 6LoWPAN fragment, or a frame from an alternate protocol such as WirelessHART or ISA100.11a.

A data frame that is forwarded along a Track normally has a destination MAC address that is set to broadcast - or a multicast address depending on MAC support. This way, the MAC layer in the intermediate nodes accepts the incoming frame and 6top switches it without incurring a change in the MAC header. In the case of IEEE802.15.4, this means effectively broadcast, so that along the Track the short address for the destination of the frame is set to 0xFFFF.

A Track is thus formed end-to-end as a succession of paired bundles, a receive bundle from the previous hop and a transmit bundle to the next hop along the Track, and a cell in such a bundle belongs to at most one Track. For a given iteration of the device schedule, the effective channel of the cell is obtained by adding a pseudo-random number to the channelOffset of the cell, which results in a rotation of the frequency that used for transmission. The bundles may be computed so as to accommodate both variable rates and retransmissions, so they might not be fully used at a given iteration of the schedule. The 6TiSCH architecture provides additional means to avoid waste of cells as well as overflows in the transmit bundle, as follows:

In one hand, a TX-cell that is not needed for the current iteration may be reused opportunistically on a per-hop basis for routed packets. When all of the frame that were received for a given Track are effectively transmitted, any available TX-cell for that Track can be reused for upper layer traffic for which the next-hop router matches the next hop along the Track. In that case, the cell that is being used is effectively a TX-cell from the Track, but the short address for the destination is that of the next-hop router. It results that a frame that is received in a RX-cell of a Track with a destination MAC address set to this node as opposed to broadcast must be extracted from the Track and delivered to the upper layer (a frame with an unrecognized MAC address is dropped at the lower MAC layer and thus is not received at the 6top sublayer).

On the other hand, it might happen that there are not enough TX-cells in the transmit bundle to accommodate the Track traffic, for instance if more retransmissions are needed than provisioned. In that case, the frame can be placed for transmission in the bundle that is used for layer-3 traffic towards the next hop along the track as long as it can be routed by the upper layer, that is, typically, if the frame transports an IPv6 packet. The MAC address should be set to the next-hop MAC address to avoid confusion. It results that a frame that is received over a layer-3 bundle may be in fact associated to a Track. In a classical IP link such as an Ethernet, off-track traffic is typically in excess over reservation to be routed along the non-reserved path based on its QoS setting. But with 6TiSCH, since the use of the layer-3 bundle may be due to transmission failures, it makes sense for the receiver to recognize a frame that should be re-tracked, and to place it back on the appropriate bundle if possible. A frame should be re-tracked if the Per-Hop-Behavior group indicated in the Differentiated Services Field in the IPv6 header is set to Deterministic Forwarding, as discussed in Section 4.1. A frame is re-tracked by scheduling it for transmission over the transmit bundle associated to the Track, with the destination MAC address set to broadcast.

There are 2 modes for a Track, transport mode and tunnel mode.

3.4.1. Transport Mode

In transport mode, the Protocol Data Unit (PDU) is associated with flow-dependant meta-data that refers uniquely to the Track, so the 6top sublayer can place the frame in the appropriate cell without ambiguity. In the case of IPv6 traffic, this flow identification is transported in the Flow Label of the IPv6 header. Associated with the source IPv6 address, the Flow Label forms a globally unique identifier for that particular Track that is validated at egress before restoring the destination MAC address (DMAC) and punting to the upper layer.

```
                          |                                    ^
      +--------------+    |                                    |
      |     IPv6     |    |                                    |
      +--------------+    |                                    |
      |  6LoWPAN HC  |    |                                    |
      +--------------+  ingress                              egress
      |     6top     |   sets     +----+          +----+     restores
      +--------------+  dmac to   |    |          |    |     dmac to
      |   TSCH MAC   |   brdcst   |    |          |    |      self
      +--------------+    |       |    |          |    |       |
      |   LLN PHY    |    +-------+    +--...-----+    +-------+
      +--------------+
```

Track Forwarding, Transport Mode

3.4.2. Tunnel Mode

In tunnel mode, the frames originate from an arbitrary protocol over a compatible MAC that may or may not be synchronized with the 6TiSCH network. An example of this would be a router with a dual radio that is capable of receiving and sending WirelessHART or ISA100.11a frames with the second radio, by presenting itself as an access Point or a Backbone Router, respectively.

In that mode, some entity (e.g. PCE) can coordinate with a WirelessHART Network Manager or an ISA100.11a System Manager to specify the flows that are to be transported transparently over the Track.

```
      +--------------+
      |     IPv6     |
      +--------------+
      |  6LoWPAN HC  |
      +--------------+          set            restore
      |     6top     |         +dmac+          +dmac+
      +--------------+       to|brdcst       to|nexthop
      |   TSCH MAC   |         |     |         |     |
      +--------------+         |     |         |     |
      |   LLN PHY    |    +-------+    +--...-----+    +-------+
      +--------------+    |   ingress              egress   |
                          |                                 |
      +--------------+    |                                 |
      |   LLN PHY    |    |                                 |
      +--------------+    |                                 |
      |   TSCH MAC   |    |                                 |
      +--------------+    | dmac =                          | dmac =
      |ISA100/WiHART |    | nexthop                         v nexthop
      +--------------+
```

Figure 5: Track Forwarding, Tunnel Mode

In that case, the flow information that identifies the Track at the ingress 6TiSCH router is derived from the RX-cell. The dmac is set to this node but the flow information indicates that the frame must be tunneled over a particular Track so the frame is not passed to the upper layer. Instead, the dmac is forced to broadcast and the frame is passed to the 6top sublayer for switching.

At the egress 6TiSCH router, the reverse operation occurs. Based on metadata associated to the Track, the frame is passed to the appropriate link layer with the destination MAC restored.

3.4.3. Tunnel Metadata

Metadata coming with the Track configuration is expected to provide the destination MAC address of the egress endpoint as well as the tunnel mode and specific data depending on the mode, for instance a service access point for frame delivery at egress. If the tunnel egress point does not have a MAC address that matches the configuration, the Track installation fails.

In transport mode, if the final layer-3 destination is the tunnel termination, then it is possible that the IPv6 address of the destination is compressed at the 6LoWPAN sublayer based on the MAC address. It is thus mandatory at the ingress point to validate that the MAC address that was used at the 6LoWPAN sublayer for compression matches that of the tunnel egress point. For that reason, the node that injects a packet on a Track checks that the destination is effectively that of the tunnel egress point before it overwrites it to broadcast. The 6top sublayer at the tunnel egress point reverts that operation to the MAC address obtained from the tunnel metadata.

4. Operations of Interest for DetNet and PCE

In a classical system, the 6TiSCH device does not place the request for bandwidth between self and another device in the network. Rather, an Operation Control System invoked through an Human/Machine Interface (HMI) indicates the Traffic Specification, in particular in terms of latency and reliability, and the end nodes. With this, the PCE must compute a Track between the end nodes and provision the network with per-flow state that describes the per-hop operation for a given packet, the corresponding timeSlots, and the flow identification that enables to recognize when a certain packet belongs to a certain Track, sort out duplicates, etc...

For a static configuration that serves a certain purpose for a long period of time, it is expected that a node will be provisioned in one shot with a full schedule, which incorporates the aggregation of its behavior for multiple Tracks. 6TiSCH expects that the programing of the schedule will be done over COAP as discussed in 6TiSCH Resource Management and Interaction using CoAP [I-D.ietf-6tisch-coap].

But an Hybrid mode may be required as well whereby a single Track is added, modified, or removed, for instance if it appears that a Track does not perform as expected for, say, PDR. For that case, the expectation is that a protocol that flows along a Track (to be), in a fashion similar to classical Traffic Engineering (TE) [CCAMP], may be used to update the state in the devices. 6TiSCH provides means for a device to negotiate a timeSlot with a neighbor, but in general that flow was not designed and no protocol was selected and it is expected that DetNet will determine the appropriate end-to-end protocols to be used in that case.

```
                      Stream Management Entity

                      Operational System and HMI

     -+-+-+-+-+-+-+ Northbound -+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-

               PCE         PCE              PCE              PCE

     -+-+-+-+-+-+-+ Southbound -+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-

             --- 6TiSCH------6TiSCH------6TiSCH------6TiSCH--
    6TiSCH /     Device      Device      Device      Device   \
    Device-                                                    - 6TiSCH
           \     6TiSCH      6TiSCH      6TiSCH      6TiSCH   /  Device
             ----Device------Device------Device------Device--
```

Figure 6

4.1. Packet Marking and Handling

Section "Packet Marking and Handling" of [I-D.ietf-6tisch-architecture] describes the packet tagging and marking that is expected in 6TiSCH networks.

4.1.1. Tagging Packets for Flow Identification

For packets that are routed by a PCE along a Track, the tuple formed by the IPv6 source address and a local RPLInstanceID is tagged in the packets to identify uniquely the Track and associated transmit bundle of timeSlots.

It results that the tagging that is used for a DetNet flow outside the 6TiSCH LLN MUST be swapped into 6TiSCH formats and back as the packet enters and then leaves the 6TiSCH network.

Note: The method and format used for encoding the RPLInstanceID at 6lo is generalized to all 6TiSCH topological Instances, which includes Tracks.

4.1.2. Replication, Retries and Elimination

6TiSCH expects elimination and replication of packets along a complex Track, but has no position about how the sequence numbers would be tagged in the packet.

As it goes, 6TiSCH expects that timeSlots corresponding to copies of a same packet along a Track are correlated by configuration, and does not need to process the sequence numbers.

The semantics of the configuration MUST enable correlated timeSlots to be grouped for transmit (and respectively receive) with a 'OR' relations, and then a 'AND' relation MUST be configurable between groups. The semantics is that if the transmit (and respectively receive) operation succeeded in one timeSlot in a 'OR' group, then all the other timeSLots in the group are ignored. Now, if there are at least two groups, the 'AND' relation between the groups indicates that one operation must succeed in each of the groups.

On the transmit side, timeSlots provisioned for retries along a same branch of a Track are placed a same 'OR' group. The 'OR' relation indicates that if a transmission is acknowledged, then further transmissions SHOULD NOT be attempted for timeSlots in that group. There are as many 'OR' groups as there are branches of the Track departing from this node. Different 'OR' groups are programmed for the purpose of replication, each group corresponding to one branch of the Track. The 'AND' relation between the groups indicates that transmission over any of branches MUST be attempted regardless of whether a transmission succeeded in another branch. It is also possible to place cells to different next-hop routers in a same 'OR' group. This allows to route along multi-path tracks, trying one next-hop and then another only if sending to the first fails.

On the receive side, all timeSlots are programmed in a same 'OR' group. Retries of a same copy as well as converging branches for elimination are converged, meaning that the first successful reception is enough and that all the other timeSlots can be ignored.

4.1.3. Differentiated Services Per-Hop-Behavior

Additionally, an IP packet that is sent along a Track uses the Differentiated Services Per-Hop-Behavior Group called Deterministic Forwarding, as described in [I-D.svshah-tsvwg-deterministic-forwarding].

4.2. Topology and capabilities

6TiSCH nodes are usually IoT devices, characterized by very limited amount of memory, just enough buffers to store one or a few IPv6 packets, and limited bandwidth between peers. It results that a node will maintain only a small number of peering information, and will not be able to store many packets waiting to be forwarded. Peers can be identified through MAC or IPv6 addresses, but a Cryptographically Generated Address [RFC3972] (CGA) may also be used.

Neighbors can be discovered over the radio using mechanism such as beacons, but, though the neighbor information is available in the 6TiSCH interface data model, 6TiSCH does not describe a protocol to pro-actively push the neighborhood information to a PCE. This protocol should be described and should operate over CoAP. The protocol should be able to carry multiple metrics, in particular the same metrics as used for RPL operations [RFC6551]

The energy that the device consumes in sleep, transmit and receive modes can be evaluated and reported. So can the amount of energy that is stored in the device and the power that it can be scavenged from the environment. The PCE SHOULD be able to compute Tracks that will implement policies on how the energy is consumed, for instance balance between nodes, ensure that the spent energy does not exceeded the scavenged energy over a period of time, etc...

5. IANA Considerations

This specification does not require IANA action.

6. Security Considerations

On top of the classical protection of control signaling that can be expected to support DetNet, it must be noted that 6TiSCH networks operate on limited resources that can be depleted rapidly if an attacker manages to operate a DoS attack on the system, for instance by placing a rogue device in the network, or by obtaining management control and to setup extra paths.

7. Acknowledgments

This specification derives from the 6TiSCH architecture, which is the result of multiple interactions, in particular during the 6TiSCH (bi)Weekly Interim call, relayed through the 6TiSCH mailing list at the IETF.

The authors wish to thank: Kris Pister, Thomas Watteyne, Xavier Vilajosana, Qin Wang, Tom Phinney, Robert Assimiti, Michael Richardson, Zhuo Chen, Malisa Vucinic, Alfredo Grieco, Martin Turon, Dominique Barthel, Elvis Vogli, Guillaume Gaillard, Herman Storey, Maria Rita Palattella, Nicola Accettura, Patrick Wetterwald, Pouria Zand, Raghuram Sudhaakar, and Shitanshu Shah for their participation and various contributions.

8. References

8.1. Normative References

[I-D.ietf-6tisch-6top-interface]
Wang, Q., Vilajosana, X., and T. Watteyne, "6TiSCH Operation Sublayer (6top) Interface", draft-ietf-6tisch-6top-interface-03 (work in progress), March 2015.

[I-D.ietf-6tisch-architecture]
Thubert, P., "An Architecture for IPv6 over the TSCH mode of IEEE 802.15.4", draft-ietf-6tisch-architecture-08 (work in progress), May 2015.

[I-D.ietf-6tisch-coap]
Sudhaakar, R. and P. Zand, "6TiSCH Resource Management and Interaction using CoAP", draft-ietf-6tisch-coap-03 (work in progress), March 2015.

[I-D.ietf-6tisch-terminology]
Palattella, M., Thubert, P., Watteyne, T., and Q. Wang, "Terminology in IPv6 over the TSCH mode of IEEE 802.15.4e", draft-ietf-6tisch-terminology-04 (work in progress), March 2015.

[I-D.ietf-6tisch-tsch]
Watteyne, T., Palattella, M., and L. Grieco, "Using IEEE802.15.4e TSCH in an IoT context: Overview, Problem Statement and Goals", draft-ietf-6tisch-tsch-06 (work in progress), March 2015.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

8.2. Informative References

[I-D.finn-detnet-architecture]
Finn, N., Thubert, P., and M. Teener, "Deterministic Networking Architecture", draft-finn-detnet-architecture-01 (work in progress), March 2015.

[I-D.ietf-ipv6-multilink-subnets]
Thaler, D. and C. Huitema, "Multi-link Subnet Support in IPv6", draft-ietf-ipv6-multilink-subnets-00 (work in progress), July 2002.

[I-D.ietf-roll-rpl-industrial-applicability]
Phinney, T., Thubert, P., and R. Assimiti, "RPL applicability in industrial networks", draft-ietf-roll-rpl-industrial-applicability-02 (work in progress), October 2013.

[I-D.svshah-tsvwg-deterministic-forwarding]
Shah, S. and P. Thubert, "Deterministic Forwarding PHB", draft-svshah-tsvwg-deterministic-forwarding-03 (work in progress), March 2015.

[I-D.thubert-6lowpan-backbone-router]
Thubert, P., "6LoWPAN Backbone Router", draft-thubert-6lowpan-backbone-router-03 (work in progress), February 2013.

[I-D.wang-6tisch-6top-sublayer]
Wang, Q., Vilajosana, X., and T. Watteyne, "6TiSCH Operation Sublayer (6top)", draft-wang-6tisch-6top-sublayer-01 (work in progress), July 2014.

[RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, December 1998.

[RFC3209] Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V., and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels", RFC 3209, December 2001.

[RFC3444] Pras, A. and J. Schoenwaelder, "On the Difference between Information Models and Data Models", RFC 3444, January 2003.

[RFC3972] Aura, T., "Cryptographically Generated Addresses (CGA)", RFC 3972, March 2005.

[RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, February 2006.

[RFC4903] Thaler, D., "Multi-Link Subnet Issues", RFC 4903, June 2007.

[RFC4919] Kushalnagar, N., Montenegro, G., and C. Schumacher, "IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals", RFC 4919, August 2007.

[RFC6282] Hui, J. and P. Thubert, "Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks", RFC 6282, September 2011.

[RFC6550] Winter, T., Thubert, P., Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, JP., and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks", RFC 6550, March 2012.

[RFC6551] Vasseur, JP., Kim, M., Pister, K., Dejean, N., and D. Barthel, "Routing Metrics Used for Path Calculation in Low-Power and Lossy Networks", RFC 6551, March 2012.

[RFC6775] Shelby, Z., Chakrabarti, S., Nordmark, E., and C. Bormann, "Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)", RFC 6775, November 2012.

8.3. Other Informative References

[ACE] IETF, "Authentication and Authorization for Constrained Environments", <https://datatracker.ietf.org/doc/charter-ietf-ace/>.

[CCAMP] IETF, "Common Control and Measurement Plane", <https://datatracker.ietf.org/doc/charter-ietf-ccamp/>.

[DICE] IETF, "DTLS In Constrained Environments", <https://datatracker.ietf.org/doc/charter-ietf-dice/>.

[HART] www.hartcomm.org, "Highway Addressable remote Transducer, a group of specifications for industrial process and control devices administered by the HART Foundation".

[IEEE802.1TSNTG] IEEE Standards Association, "IEEE 802.1 Time-Sensitive Networks Task Group", March 2013, <http://www.ieee802.org/1/pages/avbridges.html>.

[IEEE802154] IEEE standard for Information Technology, "IEEE std. 802.15.4, Part. 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks".

[IEEE802154e]
IEEE standard for Information Technology, "IEEE standard for Information Technology, IEEE std. 802.15.4, Part. 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks, June 2011 as amended by IEEE std. 802.15.4e, Part. 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) Amendment 1: MAC sublayer", April 2012.

[ISA100] ISA/ANSI, "ISA100, Wireless Systems for Automation", <https://www.isa.org/isa100/>.

[ISA100.11a]
ISA/ANSI, "Wireless Systems for Industrial Automation: Process Control and Related Applications - ISA100.11a-2011 - IEC 62734", 2011, <http://www.isa.org/Community/SP100WirelessSystemsforAutomation>.

[PCE] IETF, "Path Computation Element", <https://datatracker.ietf.org/doc/charter-ietf-pce/>.

[TEAS] IETF, "Traffic Engineering Architecture and Signaling", <https://datatracker.ietf.org/doc/charter-ietf-teas/>.

[WirelessHART]
www.hartcomm.org, "Industrial Communication Networks - Wireless Communication Network and Communication Profiles - WirelessHART - IEC 62591", 2010.

Author’s Address

Pascal Thubert (editor)
Cisco Systems, Inc
Building D
45 Allee des Ormes - BP1200
MOUGINS - Sophia Antipolis 06254
FRANCE

Phone: +33 497 23 26 34
Email: pthubert@cisco.com

core P. van der Stok
Internet-Draft consultant
Intended status: Standards Track A. Bierman
Expires: January 6, 2016 YumaWorks
J. Schoenwaelder
Jacobs University
A. Sehgal
consultant
July 5, 2015

CoAP Management Interface
draft-vanderstok-core-comi-07

Abstract

This document describes a network management interface for constrained devices, called CoMI. CoMI is an adaptation of the RESTCONF protocol for use in constrained devices and networks. It is designed to reduce the message sizes, server code size, and application development complexity. The Constrained Application Protocol (CoAP) is used to access management data resources specified in YANG, or SMIv2 converted to YANG. The payload of the CoMI message is encoded in Concise Binary Object Representation (CBOR).

Note

Discussion and suggestions for improvement are requested, and should be sent to core@ietf.org.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The Constrained Application Protocol (CoAP) [RFC7252] is designed for Machine to Machine (M2M) applications such as smart energy and building control. Constrained devices need to be managed in an automatic fashion to handle the large quantities of devices that are expected in future installations. The messages between devices need to be as small and infrequent as possible. The implementation complexity and runtime resources need to be as small as possible.

The draft [I-D.ietf-netconf-restconf] describes a REST-like interface called RESTCONF, which uses HTTP methods to access structured data defined in YANG [RFC6020]. RESTCONF allows access to data resources contained in NETCONF [RFC6241] data-stores. RESTCONF messages can be encoded in XML [XML] or JSON [RFC7159]. The GET method is used to retrieve data resources and the POST, PUT, PATCH, and DELETE methods are used to create, replace, merge, and delete data resources.

A large amount of Management Information Base (MIB) [RFC3418] specifications already exists for monitoring purposes. This data can be accessed in RESTCONF if the server converts the SMIv2 modules to YANG, using the mapping rules defined in [RFC6643].

The CoRE Management Interface (CoMI) is intended to work on standardized data-sets in a stateless client-server fashion. The RESTCONF protocol is adapted and optimized for use in constrained environments, using CoAP instead of HTTP. Standardized data sets promote interoperability between small devices and applications from different manufacturers. Stateless communication is encouraged to keep communications simple and the amount of state information small in line with the design objectives of 6lowpan [RFC4944] [RFC6775], RPL [RFC6650], and CoAP [RFC7252].

RESTCONF uses the HTTP methods HEAD, and OPTIONS, which are not available in CoAP. HTTP uses TCP which is not recommended for CoAP. The transport protocols available to CoAP are much better suited for constrained networks.

CoMI is low resource oriented, uses CoAP, and only supports the methods GET, PUT, PATCH, POST and DELETE. The payload of CoMI is encoded in CBOR [RFC7049] which is automatically generated from JSON [RFC7159]. CBOR has a binary format and hence has more coding efficiency than JSON. To promote small packets, CoMI uses an additional "data-identifier string-to-number conversion" to minimise CBOR payloads and URI length. It is assumed that the managed device is the most constrained entity. The client might be more capable, however this is not necessarily the case.

Currently, small managed devices need to support at least two protocols: CoAP and SNMP [RFC3411]. When the MIB can be accessed with the CoAP protocol, the SNMP protocol can be replaced with the CoAP protocol. Although the SNMP server size is not huge (see Appendix A), the code for the security aspects of SMIv3 [RFC3414] is not negligible. Using CoAP to access secured management objects reduces the code complexity of the stack in the constrained device, and harmonizes applications development.

The objective of CoMI is to provide a CoAP based Function Set that reads and sets values of managed objects in devices to (1) initialize parameter values at start-up, (2) acquire statistics during operation, and (3) maintain nodes by adjusting parameter values during operation.

The end goal of CoMI is to provide information exchange over the CoAP transport protocol in a uniform manner as a first step to the full management functionality as specified in [I-D.ersue-constrained-mgmt].

1.1. Design considerations

CoMI supports discovery of resources, accompanied by reading, writing and notification of resource values. As such it is close to the device management of the Open Mobile Alliance described in [OMA]. A comparison between CoMI and LWM2M management can be found in Appendix C. CoMI supports MIB modules which have been translated from SMIv2 to YANG, using [RFC6643]. This mapping is read-only so writable SMIv2 objects need to be converted to YANG using an implementation-specific mapping.

CoMI uses a simple URI to access the management object resources. Complexity introduced by instance selection, or multiple object specification is expressed with uri-query attributes. The choice for uri-query attributes makes the URI structure less context dependent.

The YANG data model contains a lot of information that can be exploited by automation tools and need not be transported in the request messages, ultimately leading to reduced message sizes.

1.2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

Readers of this specification should be familiar with all the terms and concepts discussed in [RFC3410], [RFC3416], and [RFC2578].

The following terms are defined in the NETCONF protocol [RFC6241]: client, configuration data, data-store, and server.

The following terms are defined in the YANG data modelling language [RFC6020]: container, data node, key, key leaf, leaf, leaf-list, and list.

The following terms are defined in RESTCONF protocol [I-D.ietf-netconf-restconf]: data resource, data-store resource, edit operation, query parameter, target resource, and unified data-store.

The following terms are defined in this document:

YANG hash: CoMI object identifier, which is a 30-bit numeric hash of the YANG object identifier string for the object. When a YANG hash value is printed in a request target URI, error-path or other string, then the lowercase hexadecimal representation is used. Leading zeros are used so the value uses 8 hex characters.

Data-node instance: An instance of a data-node specified in a YANG module present in the server. The instance is stored in the memory of the server.

Notification-node instance: An instance of a schema node of type notification, specified in a YANG module present in the server. The instance is generated in the server at the occurrence of the corresponding event and appended to the default stream.

The following list contains the abbreviations used in this document.

XXXX: TODO, and others to follow.

1.2.1. Tree Diagrams

A simplified graphical representation of the data model is used in this document. The meaning of the symbols in these diagrams is as follows:

Brackets "[" and "]" enclose list keys.

Abbreviations before data node names: "rw" means configuration data (read-write) and "ro" state data (read-only).

Symbols after data node names: "?" means an optional node, "!" means a presence container, and "*" denotes a list and leaf-list.

Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").

Ellipsis ("...") stands for contents of subtrees that are not shown.

2. CoMI Architecture

This section describes the CoMI architecture to use CoAP for the reading and modifying of instrumentation variables used for the management of the instrumented node.

```
Client
+--------------------------------------------------------------+
| +----------------+    +----------------+                     |
| |    SMIv2       | >  |      YANG      |    >     COAP       |
| |specification(2)|    |specification(1)|      Request(3)     |
| +----------------+    +----------------+[         *          |
+-----------------------------*-----------[---------*----------+
                              *           [         *
                              *           [    +-----------+
                      mapping *   security[    |  Network  |
                              *      (8)  [    | packet(4) |
                              *           [    +-----------+
Server                        *           [         *
+-----------------------------*-----------[---------*----------+
|                             *           [         *          |
|                             *                 Retrieval,     |
|                             *               Modification(5)  |
|                            \*/                     *          |
| +-------------------------------------------------*--------+ |
| |                    +--------------+       +------------+ | |
| |                    |configuration |       |Operational | | |
| |                    |     (6b)     |       |  state(6a) | | |
| |                    +--------------+       +------------+ | |
| |                    variable store (6)           *        | |
| +-------------------------------------------------*--------+ |
|                                                   *          |
|                                                Variable      |
|                                            Instrumentation(7)|
+--------------------------------------------------------------+
```

Figure 1: Abstract CoMI architecture

Figure 1 is a high level representation of the main elements of the CoAP management architecture. A client sends requests as payload in packets over the network to a managed constrained node.

Objectives are:

- o Equip a constrained node with a management server that provides information about the operational characteristics of the code running in the constrained node.

- o The server provides this information in a variable store that contains values describing the performance characteristics and the code parameter values.

- o The client receives the performance characteristics on a regular basis or on request.

- o The client sets the parameter values in the server at bootstrap and intermittently when operational conditions change.

- o The constrained network requires the payload to be as small as possible, and the constrained server memory requirements should be as small as possible.

For interoperability it is required that in addition to using the Internet Protocol for data transport:

- o The names, type, and semantics of the instrumentation variables are standardized.

- o The instrumentation variables are described in a standard language.

- o The signature of the CoAP request in the server is standardized.

- o The format of the packet payload is standardized.

- o The notification from server to client is standardized.

The different numbered components of Figure 1 are discussed according to component number.

(1) YANG specification: contains a set of named and versioned modules. A module specifies a hierarchy of named and typed resources. A resource is uniquely identified by a sequence of its name and the names of the enveloping resources following the hierarchy order. The YANG specification serves as input to the writers of application and instrumentation code and the humans analysing the returned values (arrow from YANG specification to Variable store). The specification can be used to check the correctness of the CoAP request and do the CBOR encoding.

(2) SMIv2 specification: A named module specifies a set of variables and "conceptual tables". Named variables have simple types. Conceptual tables are composed of typed named columns. The variable name and module name identify the variable uniquely. There is an algorithm to translate SMIv2 specifications to YANG specifications.

(3) CoAP request: The CoAP request needs a Universal Resource Identifier (URI) and the payload of the packet to send a request. The URI is composed of the schema, server, path and query and

looks like coap://entry.example.com/<path>?<query>. Fragments are not supported. Allowed operations are PUT, PATCH, GET, DELETE, and POST. New variables can be created with POST when they exist in the YANG specification. The Observe option can be used to return variable values regularly or on event occurrence (notification).

(3.1) CoAP <path>: The path identifies the variable in the form "/mg/<hash-value>".

(3.2) CoAP <query>: The query parameter is used to specify additional (optional) aspects like the module name, list instance, and others. The idea is to keep the path simple and put variations on variable specification in the query.

(3.3) CoAP discovery: Discovery of the variables is done with standard CoAP resource discovery using /.well-known/core with ?rt=/core.mg.

(4) Network packet: The payload contains the CBOR encoding of JSON objects. This object corresponds to the converted RESTCONF message payload.

(5) Retrieval, modification: The server needs to parse the CBOR encoded message and identify the corresponding instances in the Variable store. In addition, this component includes the code for CoAP Observe and block options.

(6) Variable store: The store is composed of two parts: Operational state and Configuration data-store (see Section 2.1). CoMI does not differentiate between variable store types. The Variable store contains data-node instances. Values are stored in the appropriate instances, and or values are returned from the instances into the payload of the packet.

(7) Variable instrumentation: This code depends on implementation of drivers and other node specific aspects. The Variable instrumentation code stores the values of the parameters into the appropriate places in the operational code. The variable instrumentation code reads current execution values from the operational code and stores them in the appropriate instances.

(8) Security: The server MUST prevent unauthorized users from reading or writing any data resources. CoMI relies on DTLS [RFC6347] which is specified to secure CoAP communication.

2.1. RESTCONF/YANG Architecture

CoMI adapts the RESTCONF architecture so data exchange and implementation requirements are optimized for constrained devices.

The RESTCONF protocol uses a unified data-store to edit conceptual data structures supported by the server. The details of transaction preparation and non-volatile storage of the data are hidden from the RESTCONF client. CoMI also uses a unified data-store, to allow stateless editing of configuration variables and the notification of operational variables.

The child schema nodes of the unified data-store include all the top-level YANG data nodes in all the YANG modules supported by the server. The YANG data structures represent a hierarchy of data resources. The client discovers the list of YANG modules, and important conformance information such as the module revision dates, YANG features supported, and YANG deviations required. The individual data nodes are discovered indirectly by parsing the YANG modules supported by the server.

The YANG data definition statements contain a lot of information that can help automation tools, developers, and operators use the data model correctly and efficiently. The YANG definitions and server YANG module capability advertisements provide an "API contract" that allow a client to determine the detailed server management capabilities very quickly. CoMI allows access to the same data resources as a RESTCONF server, except the messages are optimized to reduce identifier and payload size.

RESTCONF uses a simple algorithmic mapping from YANG to URI syntax to identify the target resource of a retrieval or edit operation. A client can construct operations or scripts using a predictable syntax, based on the YANG data definitions. The target resource URI can reference a data resource instance, or the data-store itself (to retrieve the entire data-store or create a top-level data resource instance). CoMI uses a compression algorithm to reduce the size of the data-node instance identifier (see Section 2.2.

2.2. Compression of data-node instance identifier

The RESTCONF protocol uses the full path of the desired data resource in the target resource URI. The JSON encoding will include the module name string to specify the YANG module. If a representation of the target resource is included in the request or response message in RESTCONF messages, then the data definition name string is used to identify each node in the message. The module namespace (or name) may also be present in these identifiers.

In order to greatly reduce the size of identifiers used in CoMI, numeric object identifiers are used instead of these strings. The specific encoding of the object identifiers is not hard-wired in the protocol.

YANG Hash is the default encoding for object identifiers. This encoding in considered to be "unstructured" since the particular values for each object are determined by a hash algorithm. It is possible for 2 different objects to generate the same hash value. If this occurs, then the client and server will both need to rehash the colliding object identifiers to new unused hash values.

In order to eliminate the need for rehashing, CoMI allows for alternate "structured" object identifier encoding formats. Structured object identifier MUST be managed such that no object ID collisions are possible, and therefore no rehash procedures are needed. Structured object identifiers can also be selected to minimize the size of a subset of the object identifiers (e.g., the most requested objects).

In Section 4.5 the discovery of the object ID compression scheme is described.

3. CoAP Interface

In CoAP a group of links can constitute a Function Set. The format of the links is specified in [I-D.ietf-core-interfaces]. This note specifies a Management Function Set. CoMI end-points that implement the CoMI management protocol support at least one discoverable management resource of resource type (rt): core.mg, with path: /mg, where mg is short-hand for management. The name /mg is recommended but not compulsory (see Section 4.5).

The path prefix /mg has resources accessible with the following five paths:

/mg: YANG-based data with path "/mg" and using CBOR content encoding format. This path represents a data-store resource which contains YANG data resources as its descendant nodes. All identifiers referring to YANG data nodes within this path are encoded as YANG hash values (see Section 5.5).

/mg/mod.uri: URI identifying the location of the server module information, with path "/mg/mod.uri" and CBOR content format. This YANG data is encoded with plain identifier strings, not YANG hash values.

/mg/mod.set: String identifying the module set ID in use by the server, which is defined as the 'module-set-id' leaf in the ietf-yang-library module. This resource MUST change to a new value when the set of YANG modules in use by the server changes.

/mg/num.typ: String identifying the object ID numbering scheme used by the CoMI server. The only value defined in this document is 'yanghash' to indicate that the YANG Hash numbering scheme defined in this document is used. It is possible for other object numbering schemes to be defined outside the scope of this document.

/mg/srv.typ: String identifying the CoMI server type. The value 'ro' indicates that the server is a read-only server and no editing operations are supported. A read-only server is not required to provide YANG deviation statements for any writable YANG data nodes. The value 'rw' indicates that the server is a read-write server and editing operations are supported. A read-write server is required to provide YANG deviation statements for any writable YANG data nodes that are not fully implemented.

/mg/yh.uri: URI indicating the location of the server YANG hash information if any objects needed to be re-hashed by the server. It has the path "/mg/yh.uri" and is encoded in CBOR format. The "ietf-yang-hash" module of Section 5.3 is used to define the syntax and semantics of this data structure. This YANG data is encoded with plain identifier strings, not YANG hash values. The server will only have this resource if there are any objects that needed to be re-hashed due to a hash collision.

/mg/stream: String identifying the default stream resource to which YANG notification instances are appended. Notification support is optional, so this resource will not exist if the server does not support any notifications.

The mapping of YANG data node instances to CoMI resources is as follows: A YANG module describes a set of data trees composed of YANG data nodes. Every root of a data tree in a YANG module loaded in the CoMI server represents a resource of the server. All data root descendants represent sub-resources.

The resource identifiers of the instances of the YANG specifications are YANG hash values, as described in Section 5.1. When multiple instances of a list node exist, the instance selection is described in Section 4.1.3.4

The profile of the management function set, with IF=core.mg, is shown in the table below, following the guidelines of [I-D.ietf-core-interfaces]:

name	path	rt	Data Type
Management	/mg	core.mg	n/a
Data	/mg	core.mg.data	application/cbor
Module Set URI	/mg/mod.uri	core.mg.moduri	application/cbor
Module Set ID	/mg/mod.set	core.mg.modset	application/cbor
Numbering Type	/mg/num.typ	core.mg.num-type	application/cbor
Server Type	/mg/srv.typ	core.mg.srv-type	application/cbor
YANG Hash Info	/mg/yh.uri	core.mg.yang-hash	application/cbor
Events	/mg/stream	core.mg.stream	application/cbor

4. MG Function Set

The MG Function Set provides a CoAP interface to perform a subset of the functions provided by RESTCONF.

A subset of the operations defined in RESTCONF are used in CoMI:

Operation	Description
GET	Retrieve the data-store resource or a data resource
POST	Create a data resource
PUT	Create or replace a data resource
PATCH	Replace a data resource partially
DELETE	Delete a data resource

4.1. Data Retrieval

4.1.1. GET

One or more instances of data resources are retrieved by the client with the GET method. The RESTCONF GET operation is supported in CoMI. The same constraints apply as defined in section 3.3 of [I-D.ietf-netconf-restconf]. The operation is mapped to the GET method defined in section 5.8.1 of [RFC7252].

It is possible that the size of the payload is too large to fit in a single message. In the case that management data is bigger than the maximum supported payload size, the Block mechanism from [I-D.ietf-core-block] is used, as explained in more detail in Section 4.4.

There are two query parameters for the GET method. A CoMI server MUST implement the keys parameter and MAY implement the select parameter to allow common data retrieval filtering functionality.

Query Parameter	Description
keys	Request to select instances of a YANG definition
select	Request selected sub-trees from the target resource

The "keys" parameter is used to specify a specific instance of the resource. When keys is not specified, all instances are returned. When no or one instance of the resource exists, the keys parameter is not needed.

4.1.2. Mapping of the 'select' Parameter

RESTCONF uses the 'select' parameter to specify an expression which can represent a subset of all data nodes within the target resource [I-D.ietf-netconf-restconf]. This parameter is useful for filtering sub-trees and retrieving only a subset that a managing application is interested in.

However, filtering is a resource intensive task and not all constrained devices can be expected to have enough computing resources such that they will be able to successfully filter and return a subset of a sub-tree. This is especially likely to be true with Class 0 devices that have significantly lesser RAM than 10 KiB

[RFC7228]. Since CoMI is targeted at constrained devices and networks, only a limited subset of the 'select' parameter is used here.

Unlike the RESTCONF 'select' parameter, CoMI does not use object names in "XPath" or "path-expr" format to identify the subset that needs to be filtered. Parsing XML is resource intensive for constrained devices [management] and using object names can lead to large message sizes. Instead, CoMI utilizes the YANG hashes described in Section 5 to identify the sub-trees that should be filtered from a target resource. Using these hashes ensures that a constrained node can identify the target sub-tree without expending many resources and that the messages generated are also efficiently encoded.

The implementation of the 'select' parameter is already optional for constrained devices, however, even when implemented it is expected to be a best effort feature, rather than a service that nodes must provide. This implies that if a node receives the 'select' parameter specifying a set of sub-trees that should be returned, it will only return those that it is able to.

4.1.3. Retrieval Examples

In all examples the path is expressed in readable names and as a hash value of the name (where the hash value in the payload is expressed as a hexadecimal number, and the hash value in the URL as a base64 number). The examples in this section use a JSON payload with one or more entries describing the pair (identifier, value). CoMI transports the CBOR format to transport the equivalent contents. The CBOR syntax of the payloads is specified in Section 5.

4.1.3.1. Single instance retrieval

A request to read the values of instances of a management object or the leaf of an object is sent with a confirmable CoAP GET message. A single object is specified in the URI path prefixed with /mg.

Using for example the clock container from [RFC7317], a request is sent to retrieve the value of clock/current-datetime specified in module system-state. The answer to the request returns a (identifier, value) pair.

```
REQ: GET example.com/mg/system-state/clock/current-datetime

RES: 2.05 Content (Content-Format: application/cbor)
{
    "current-datetime" : "2014-10-26T12:16:31Z"
}
```

The YANG hash value for 'current-datetime' is calculated by constructing the schema node identifier for the object:

/sys:system-state/sys:clock/sys:current-datetime

The 30 bit murmur3 hash value is calculated on this string (0x15370408 and VNwQI). The request using this hash value is shown below:

```
REQ: GET example.com/mg/VNwQI

RES: 2.05 Content (Content-Format: application/cbor)
{
    0x15370408 : "2014-10-26T12:16:31Z"
}
```

The specified object can be an entire object. Accordingly, the returned payload is composed of all the leaves associated with the object. Each leaf is returned as a (YANG hash, value) pair. For example, the GET of the clock object, sent by the client, results in the following returned payload sent by the managed entity:

```
REQ: GET example.com/mg/system-state/clock
   (Content-Format: application/cbor)

RES: 2.05 Content (Content-Format: application/cbor)
{
      "clock/current-datetime" : "2014-10-26T12:16:51Z",
      "clock/boot-datetime" : "2014-10-21T03:00:00Z"
}
```

The YANG hash values for 'clock', 'current-datetime', and 'boot-datetime' are calculated by constructing the schema node identifier for the objects, and then calculating the 30 bit murmur3 hash values (shown in parenthesis):

/sys:system-state/sys:clock (0x2eb2fa3b and usvo7)
/sys:system-state/sys:clock/sys:current-datetime (0x15370408)
/sys:system-state/sys:clock/sys:boot-datetime (0x1fa25361)

The request using the hash values is shown below:

```
REQ: GET example.com/mg/usvo7
   (Content-Format: application/cbor)

RES: 2.05 Content (Content-Format: application/cbor)
{
      0x15370408 : "2014-10-26T12:16:51Z",
      0x1fa25361 : "2014-10-21T03:00:00Z"
}
```

4.1.3.2. Multiple instance retrieval

A "list" node can have multiple instances. Accordingly, the returned payload is composed of all the instances associated with the list node. Each instance is returned as a (identifier, value) pair. The "keys" query parameter is used to identify a specific list instance by specifying a given index value (see Section 4.1.3.4).

For example, the GET of the /interfaces/interface/ipv6/neighbor instance identified with interface index "eth0" [RFC7223], sent by the client, results in the following returned payload sent by the managed entity:

```
REQ: GET example.com/mg/interfaces/interface/ipv6/neighbor?keys=eth0
   (Content-Format: application/cbor)

RES: 2.05 Content (Content-Format: application/cbor)
{
   "neighbor":[
     {
        "ip" : "fe80::200:f8ff:fe21:67cf",
        "link-layer-address" : "00:00::10:01:23:45"
     },
     {
        "ip" : "fe80::200:f8ff:fe21:6708",
        "link-layer-address" : "00:00::10:54:32:10"
     },
     {
        "ip" : "fe80::200:f8ff:fe21:88ee",
        "link-layer-address" : "00:00::10:98:76:54"
     }
   ]
}
```

The YANG hash values for 'neighbor', 'ip', and 'link-layer-address' are calculated by constructing the schema node identifier for the objects, and then calculating the 30 bit murmur3 hash values (shown in parenthesis):

```
/if:interfaces/if:interface/ip:ipv6/ip:neighbor (0x2354bc49 and jVLxJ)
/if:interfaces/if:interface/ip:ipv6/ip:neighbor/ip:ip
    (0x20b8907e and guJB_)
/if:interfaces/if:interface/ip:ipv6/ip:neighbor/ip:link-layer-address
   (0x16f47fd8)
```

The request using the hash values is shown below:

```
REQ: GET example.com/mg/jVLxJ?keys=eth0
   (Content-Format: application/cbor)

RES: 2.05 Content (Content-Format: application/cbor)
{
   0x2354bc49 : [
     {
        0x20b8907e : "fe80::200:f8ff:fe21:67cf",
        0x16f47fd8 : "00:00::10:01:23:45"
     },
     {
        0x20b8907e : "fe80::200:f8ff:fe21:6708",
        0x16f47fd8 : "00:00::10:54:32:10"
     },
     {
        0x20b8907e : "fe80::200:f8ff:fe21:88ee",
        0x16f47fd8 : "00:00::10:98:76:54"
     }
   ]
}
```

4.1.3.3. Access to MIB Data

The YANG translation of the SMI specifying the ipNetToMediaTable [RFC4293] yields:

```
   container IP-MIB {
     container ipNetToPhysicalTable {
       list ipNetToPhysicalEntry {
          key "ipNetToPhysicalIfIndex
               ipNetToPhysicalNetAddressType
               ipNetToPhysicalNetAddress";
          leaf ipNetToMediaIfIndex {
             type: int32;
          }
          leaf ipNetToPhysicalIfIndex {
            type if-mib:InterfaceIndex;
          }
          leaf ipNetToPhysicalNetAddressType {
            type inet-address:InetAddressType;
          }
          leaf ipNetToPhysicalNetAddress {
            type inet-address:InetAddress;
          }
          leaf ipNetToPhysicalPhysAddress {
            type yang:phys-address {
               length "0..65535";
            }
          }
          leaf ipNetToPhysicalLastUpdated {
            type yang:timestamp;
          }
          leaf ipNetToPhysicalType {
            type enumeration { ... }
          }
          leaf ipNetToPhysicalState {
            type enumeration { ... }
          }
          leaf ipNetToPhysicalRowStatus {
            type snmpv2-tc:RowStatus;
          }
       }
    }
```

The following example shows an "ipNetToPhysicalTable" with 2 instances, using JSON encoding:

```
{
  "IP-MIB/ipNetToPhysicalTable/ipNetToPhysicalEntry" : [
        {
          "ipNetToPhysicalIfIndex" : 1,
          "ipNetToPhysicalNetAddressType" : "ipv4",
          "ipNetToPhysicalNetAddress" : "10.0.0.51",
          "ipNetToPhysicalPhysAddress" : "00:00:10:01:23:45",
          "ipNetToPhysicalLastUpdated" : "2333943",
          "ipNetToPhysicalType" : "static",
          "ipNetToPhysicalState" : "reachable",
          "ipNetToPhysicalRowStatus" : "active"
        },
        {
          "ipNetToPhysicalIfIndex" : 1,
          "ipNetToPhysicalNetAddressType" : "ipv4",
          "ipNetToPhysicalNetAddress" : "9.2.3.4",
          "ipNetToPhysicalPhysAddress" : "00:00:10:54:32:10",
          "ipNetToPhysicalLastUpdated" : "2329836",
          "ipNetToPhysicalType" : "dynamic",
          "ipNetToPhysicalState" : "unknown",
          "ipNetToPhysicalRowStatus" : "active"
        }
      ]
    }
  }
}
```


The YANG hash values for ’ipNetToPhysicalEntry’ and its child nodes are calculated by constructing the schema node identifier for the objects, and then calculating the 30 bit murmur3 hash values (shown in parenthesis):

```
/ip-mib:IP-MIB/ip-mib:ipNetToPhysicalTable (0x30b7bc3f and wt7w_)
/ip-mib:IP-MIB/ip-mib:ipNetToPhysicalTable/ip-mib:ipNetToPhysicalEntry
   (0x1067f289 and QZ_KJ)
/ip-mib:IP-MIB/ip-mib:ipNetToPhysicalTable/ip-mib:ipNetToPhysicalEntry/
   ip-mib:ipNetToPhysicalIfIndex (0x00d38564)
/ip-mib:IP-MIB/ip-mib:ipNetToPhysicalTable/ip-mib:ipNetToPhysicalEntry/
   ip-mib:ipNetToPhysicalNetAddressType (0x2745e222)
/ip-mib:IP-MIB/ip-mib:ipNetToPhysicalTable/ip-mib:ipNetToPhysicalEntry/
   ip-mib:ipNetToPhysicalNetAddress (0x387804eb)
/ip-mib:IP-MIB/ip-mib:ipNetToPhysicalTable/ip-mib:ipNetToPhysicalEntry/
   ip-mib:ipNetToPhysicalPhysAddress (0x1a51514a)
/ip-mib:IP-MIB/ip-mib:ipNetToPhysicalTable/ip-mib:ipNetToPhysicalEntry/
   ip-mib:ipNetToPhysicalLastUpdated (0x03f95578)
/ip-mib:IP-MIB/ip-mib:ipNetToPhysicalTable/ip-mib:ipNetToPhysicalEntry/
   ip-mib:ipNetToPhysicalType (0x24ade115)
/ip-mib:IP-MIB/ip-mib:ipNetToPhysicalTable/ip-mib:ipNetToPhysicalEntry/
   ip-mib:ipNetToPhysicalState (0x09e640ef)
/ip-mib:IP-MIB/ip-mib:ipNetToPhysicalTable/ip-mib:ipNetToPhysicalEntry/
   ip-mib:ipNetToPhysicalRowStatus (0x3b5c1ab6)
```

The following example shows a request for the entire ipNetToPhysicalTable. Since all the instances are requested, no "keys" query parameter is needed.

```
REQ: GET example.com/mg/wt7w_

RES: 2.05 Content (Content-Format: application/cbor)
{
      0x1067f289 : [
        {
          0x00d38564 : 1,
          0x2745e222 : "ipv4",
          0x387804eb : "10.0.0.51",
          0x1a51514a : "00:00:10:01:23:45",
          0x03f95578 : "2333943",
          0x24ade115 : "static",
          0x09e640ef : "reachable",
          0x3b5c1ab6 : "active"
        },
        {
          0x00d38564 : 1,
          0x2745e222 : "ipv4",
          0x387804eb : "9.2.3.4",
          0x1a51514a : "00:00:10:54:32:10",
          0x03f95578 : "2329836",
          0x24ade115 : "dynamic",
          0x09e640ef : "unknown",
          0x3b5c1ab6 : "active"
        }
      ]
}
```

4.1.3.4. The ’keys’ Query Parameter

There is a mandatory query parameter that MUST be supported by servers called "keys". This parameter is used to specify the key values for an instance of an object identified by a YANG hash value. Any key leaf values of the instance are passed in order. The first key leaf in the top-most list is the first key encoded in the ’keys’ parameter.

The key leafs from top to bottom and left to right are encoded as a comma-delimited list. If a key leaf value is missing then all values for that key leaf are returned.

Example: In this example exactly 1 instance is requested from the ipNetToPhysicalEntry (from a previous example).

```
REQ: GET example.com/mg/QZ_KJ?keys=1,ipv4,10.0.0.51

RES: 2.05 Content (Content-Format: application/cbor)
{
   0x1067f289 : [
      {
        0x00d38564 : 1,
        0x2745e222 : "ipv4",
        0x387804eb : "10.0.0.51",
        0x1a51514a : "00:00:10:01:23:45",
        0x03f95578 : "2333943",
        0x24ade115 : "static",
        0x09e640ef : "reachable",
        0x3b5c1ab6 : "active"
      }
   ]
}
```

An example illustrates the syntax of keys query parameter. In this example the following YANG module is used:

```
  module foo-mod {
    namespace foo-mod-ns;
    prefix foo;

    list A {
      key "key1 key2";
      leaf key1 { type string; }
      leaf key2 { type int32; }
      list B {
        key "key3";
        leaf key3 { type string; }
        leaf col1 { type uint32; }
      }
    }
  }
```

The path identifier for the leaf "col1" is the following string:

```
   /foo:A/foo:B/foo:col1
```

The YANG hash for this identifier string has values: 0xa9abdcca and pq9zK).

The following string represents the RESTCONF target resource URI expression for the "col1" leaf for the key values "top", 17, and "group1":

```
/restconf/data/foo-mod:A="top",17/B="group1"/col1
```

The following string represents the CoMI target resource identifier for the same instance of the "col1" leaf:

```
/mg/pq9zK?keys="top",17,"group1"
```

4.1.3.5. The 'select' Query Parameter

The select parameter is used along with the GET method to provide a sub-tree filter mechanism. A list of YANG hashes that should be filtered is provided along with a list of keys identifying the instances that should be returned. When the keys parameter is used together with the select, the key values are added in brackets without using the "keys=" text.

The following example shows an "ipNetToPhysicalTable" (from a previous example) with 4 instances, using JSON encoding:

```
{
  "IP-MIB/ipNetToPhysicalTable/ipNetToPhysicalEntry" : [
        {
          "ipNetToPhysicalIfIndex" : 1,
          "ipNetToPhysicalNetAddressType" : "ipv4",
          "ipNetToPhysicalNetAddress" : "10.0.0.51",
          "ipNetToPhysicalPhysAddress" : "00:00:10:01:23:45",
          "ipNetToPhysicalLastUpdated" : "2333943",
          "ipNetToPhysicalType" : "static",
          "ipNetToPhysicalState" : "reachable",
          "ipNetToPhysicalRowStatus" : "active"
        },
        {
          "ipNetToPhysicalIfIndex" : 1,
          "ipNetToPhysicalNetAddressType" : "ipv4",
          "ipNetToPhysicalNetAddress" : "9.2.3.4",
          "ipNetToPhysicalPhysAddress" : "00:00:10:54:32:10",
          "ipNetToPhysicalLastUpdated" : "2329836",
          "ipNetToPhysicalType" : "dynamic",
          "ipNetToPhysicalState" : "unknown",
          "ipNetToPhysicalRowStatus" : "active"
        },
        {
          "ipNetToPhysicalIfIndex" : 2,
          "ipNetToPhysicalNetAddressType" : "ipv4",
          "ipNetToPhysicalNetAddress" : "10.24.2.53",
          "ipNetToPhysicalPhysAddress" : "00:00:10:28:19:CA",
          "ipNetToPhysicalLastUpdated" : "2124368",
          "ipNetToPhysicalType" : "static",
          "ipNetToPhysicalState" : "unknown",
          "ipNetToPhysicalRowStatus" : "active"
        },
        {
          "ipNetToPhysicalIfIndex" : 3,
          "ipNetToPhysicalNetAddressType" : "ipv4",
          "ipNetToPhysicalNetAddress" : "192.168.2.12",
          "ipNetToPhysicalPhysAddress" : "00:00:10:29:11:32",
          "ipNetToPhysicalLastUpdated" : "1925384",
          "ipNetToPhysicalType" : "dynamic",
          "ipNetToPhysicalState" : "reachable",
          "ipNetToPhysicalRowStatus" : "active"
        }
      ]
    }
  }
}
```

Data may be retrieved using the select query parameter in the following way:

```
REQ: GET example.com/mg/?select=wt7w_(ipv4,reachable)

RES: 2.05 Content (Content-Format: application/cbor)
{
   0x1067f289 : [
      {
        0x00d38564 : 1,
        0x2745e222 : "ipv4",
        0x387804eb : "10.0.0.51",
        0x1a51514a : "00:00:10:01:23:45",
        0x03f95578 : "2333943",
        0x24ade115 : "static",
        0x09e640ef : "reachable",
        0x3b5c1ab6 : "active"
        },
        {
          0x00d38564 : 3,
          0x2745e222 : "ipv4",
          0x387804eb : "192.168.2.12",
          0x1a51514a : "00:00:10:29:11:32",
          0x03f95578 : "1925384",
          0x24ade115 : "dynamic",
          0x09e640ef : "reachable",
          0x3b5c1ab6 : "active"
        }
   ]
}
```

In this example exactly 2 instances are returned as response from the ipNetToPhysicalTable because both those instances match the provided keys.

Supposing there were multiple YANG hashes with their own sets of keys that were to be filtered, the select query parameter can be used to retrieve results from these in one go as well. The following string represents the CoMI target resource identifier when multiple YANG hashes, with their own sets of keys are queried:

```
/mg/?select=hash1(hash1-key1,hash1-key2,...),hash2(hash2-key1)...
```

4.2. Data Editing

CoMI allows data-store contents to be created, modified and deleted using CoAP methods.

Data-editing is an optional feature. The server will indicate its editing capability with the "/core.rg.srv-type resource type. If the value is 'rw' then the server supports editing operations. If the value is 'ro' then the server does not support editing operations.

4.2.1. Data Ordering

A CoMI server is not required to support entry insertion of lists and leaf-lists that are ordered by the user (i.e., YANG statement "ordered-by user"). The 'insert' and 'point' query parameters from RESTCONF are not used in CoMI.

A CoMI server SHOULD preserve the relative order of all user-ordered list and leaf-list entries that are received in a single edit request. These YANG data node types are encoded as arrays so messages will preserve their order.

4.2.2. POST

Data resource instances are created with the POST method. The RESTCONF POST operation is supported in CoMI, however it is only allowed for creation of data resources. The same constraints apply as defined in section 3.4.1 of [I-D.ietf-netconf-restconf]. The operation is mapped to the POST method defined in section 5.8.2 of [RFC7252].

There are no query parameters for the POST method.

4.2.3. PUT

Data resource instances are created or replaced with the PUT method. The PUT operation is supported in CoMI. A request to set the values of instances of an object/leaf is sent with a confirmable CoAP PUT message. The Response is piggybacked to the CoAP ACK message corresponding with the Request. The same constraints apply as defined in section 3.5 of [I-D.ietf-netconf-restconf]. The operation is mapped to the PUT method defined in section 5.8.3 of [RFC7252].

There are no query parameters for the PUT method.

4.2.4. PATCH

Data resource instances are partially replaced with the PATCH method [I-D.vanderstok-core-patch]. The PATCH operation is supported in CoMI. A request to set the values of instances of a subset of the values of the resource is sent with a confirmable CoAP PATCH message. The Response is piggybacked to the CoAP ACK message corresponding with the Request. The same constraints apply as defined in section

3.5 of [I-D.ietf-netconf-restconf]. The operation is mapped to the PATCH method defined in [I-D.vanderstok-core-patch].

There are no query parameters for the PATCH method.

4.2.5. DELETE

Data resource instances are deleted with the DELETE method. The RESTCONF DELETE operation is supported in CoMI. The same constraints apply as defined in section 3.7 of [I-D.ietf-netconf-restconf]. The operation is mapped to the DELETE method defined in section 5.8.4 of [RFC7252].

There are no optional query parameters for the PUT method.

4.2.6. Editing Multiple Resources

Editing multiple data resources at once can allow a client to use fewer messages to make a configuration change. It also allows multiple edits to all be applied or none applied, which is not possible if the data resources are edited one at a time.

It is easy to add multiple entries at once. The "PATCH" method can be used to simply patch the parent node(s) of the data resources to be added. If multiple top-level data resources need to be added, then the data-store itself ('/mg') can be patched.

If other operations need to be performed, or multiple operations need to be performed at once, then the YANG Patch [I-D.ietf-netconf-yang-patch] media type can be used with the PATCH method. A YANG patch is an ordered list of edits on the target resource, which can be a specific data node instance, or the data-store itself. The resource type used by YANG Patch is 'application/yang.patch'. A status message is returned in the response, using resource type 'application/yang.patch.status'.

The following YANG tree diagram describes the YANG Patch structure, Each 'edit' list entry has its own operation, sub-resource target, and new value (if needed).

```
+--rw yang-patch
    +--rw patch-id?   string
    +--rw comment?    string
    +--rw edit* [edit-id]
       +--rw edit-id      string
       +--rw operation    enumeration
       +--rw target       target-resource-offset
       +--rw point?       target-resource-offset
       +--rw where?       enumeration
       +--rw value
```

The YANG Hash values for the YANG Patch request objects are calculated as follows:

```
0b346308: /ypatch:yang-patch
29988080: /ypatch:yang-patch/ypatch:patch-id
0c258737: /ypatch:yang-patch/ypatch:comment
316beed6: /ypatch:yang-patch/ypatch:edit
2f51f9f7: /ypatch:yang-patch/ypatch:edit/ypatch:edit-id
28f4669e: /ypatch:yang-patch/ypatch:edit/ypatch:operation
2cb909c9: /ypatch:yang-patch/ypatch:edit/ypatch:target
387d0cd8: /ypatch:yang-patch/ypatch:edit/ypatch:point
21899571: /ypatch:yang-patch/ypatch:edit/ypatch:where
1d86d302: /ypatch:yang-patch/ypatch:edit/ypatch:value
```

Refer to [I-D.ietf-netconf-yang-patch] for more details on the YANG Patch request and response contents.

4.3. Notify functions

Notification by the server to a selection of clients when an event occurs in the server is an essential function for the management of servers. CoMI allows events specified in YANG [RFC5277] to be notified to a selection of requesting clients. There is one, so-called "default", stream in a CoMI server. The /mg/stream resource identifies the default stream. When a CoMI server generates an internal event, it is appended to the default stream, and the contents of a notification instance is ready to be sent to all CoMI clients which observe the default stream resource.

Reception of generated notification instances is enabled with the CoAP Observe [I-D.ietf-core-observe] function. The client subscribes to the notifications by sending a GET request with an "Observe" option, specifying the /mg/stream resource.

Every time an event is generated, the default stream is cleared, and the generated notification instance is appended to the stream. After appending the instance, the contents of the instance is sent to all observing clients.

Suppose the server generates the event specified with:

```
module example-port {
  ...
  prefix ep;
  ...
  notification example-port-fault {
    description
      "Event generated if a hardware fault on a
       line card port is detected";
    leaf port-name {
      type string;
      description "Port name";
    }
    leaf port-fault {
      type string;
      description "Error condition detected";
    }
  }
}

}
```

The YANG Hash values for this notification are assigned as follows:

```
1eed4674: /ep:example-port-fault
0cec9c71: /ep:example-port-fault/ep:port-name
228d3fa1: /ep:example-port-fault/ep:fault

}
```

By executing a GET on the /mg/stream resource the client receives the following response:

```
REQ: GET example.com/mg/stream
    (observe option register)

RES: 2.05 Content (Content-Format: application/cbor)
{
   "example-port-fault" : {
      "port-name" : "0/4/21",
      "port-fault" : "Open pin 2"
   }
}

TODO: fix YANG Hash/CBOR encoding example

RES: 2.05 Content (Content-Format: application/cbor)
{
   1eed4674 : {
      cec9c71 : "0/4/21",
      228d3fa1 : "Open pin 2"
   }
}
```

In the example, the request returns a success response with the contents of the last generated event. Consecutively the server will regularly notify the client when a new event is generated.

To check that the client is still alive, the server MUST send confirmable notifications once in a while. When the client does not confirm the notification from the server, the server will remove the client from the list of observers [I-D.ietf-core-observe].

In the registration request, the client MAY include a "Response-To-Uri-Host" and optionally "Response-To-Uri-Port" option as defined in [I-D.becker-core-coap-sms-gprs]. In this case, the observations SHOULD be sent to the address and port indicated in these options. This can be useful when the client wants the managed device to send the trap information to a multicast address.

4.4. Use of Block

The CoAP protocol provides reliability by acknowledging the UDP datagrams. However, when large pieces of text need to be transported the datagrams get fragmented, thus creating constraints on the resources in the client, server and intermediate routers. The block option [I-D.ietf-core-block] allows the transport of the total payload in individual blocks of which the size can be adapted to the underlying fragment sizes such as: (UDP datagram size ˜64KiB, IPv6 MTU of 1280, IEEE 802.15.4 payload of 60-80 bytes). Each block is individually acknowledged to guarantee reliability.

The block size is specified as exponents of the power 2. The SZX exponent value can have 7 values ranging from 0 to 6 with associated block sizes given by 2**(SZX+4); for example SZX=0 specifies block size 16, and SZX=3 specifies block size 128.

The block number of the block to transmit can be specified. There are two block options: Block1 option for the request payload transported with PUT, POST or PATCH, and the block2 option for the response payload with GET. Block1 and block2 can be combined. Examples showing the use of block option in conjunction with observer options are provided in [I-D.ietf-core-block].

Notice that the Block mechanism splits the data at fixed positions, such that individual data fields may become fragmented. Therefore, assembly of multiple blocks may be required to process the complete data field.

4.5. Resource Discovery

The presence and location of (path to) the management data are discovered by sending a GET request to "/.well-known/core" including a resource type (RT) parameter with the value "core.mg" [RFC6690]. Upon success, the return payload will contain the root resource of the management data. It is up to the implementation to choose its root resource, but it is recommended that the value "/mg" is used, where possible. The example below shows the discovery of the presence and location of management data.

```
REQ: GET /.well-known/core?rt=core.mg

RES: 2.05 Content </mg>; rt="core.mg"
```

Management objects MAY be discovered with the standard CoAP resource discovery. The implementation can add the hash values of the object identifiers to /.well-known/core with rt="core.mg.data". The available objects identified by the hash values can be discovered by sending a GET request to "/.well-known/core" including a resource type (RT) parameter with the value "core.mg.data". Upon success, the return payload will contain the registered hash values and their location. The example below shows the discovery of the presence and location of management data.

```
REQ: GET /.well-known/core?rt=core.mg.data

RES: 2.05 Content </mg/BaAiN>; rt="core.mg.data",
</mg/CF_fA>; rt="core.mg.data"
```

Lists of hash values may become prohibitively long. It is discouraged to provide long lists of objects on discovery. Therefore, it is recommended that details about management objects are discovered following the RESTCONF protocol. The YANG module information is stored in the "ietf-yang-library" module [I-D.ietf-netconf-restconf]. The resource "/mg/mod.uri" is used to retrieve the location of the YANG module library.

Since many constrained servers within a deployment are likely to be similar, the module list can be stored locally on each server, or remotely on a different server.

```
Local in example.com server:

REQ: GET example.com/mg/mod.uri

RES: 2.05 Content (Content-Format: application/cbor)
{
  "mod.uri" : "example.com/mg/modules"
}
```

```
Remote in example-remote-server:

REQ: GET example.com/mg/mod.uri

RES: 2.05 Content (Content-Format: application/cbor)
{
  "moduri" : "example-remote-server.com/mg/group17/modules"
}
```

Within the YANG module library all information about the module is stored such as: module identifier, identifier hierarchy, grouping, features and revision numbers.

The hash identifier is obtained as specified in Section 5.1. When a collision occurred in the name space of the target server, a rehash is executed as explained in Section 5.2.

4.6. Error Return Codes

The RESTCONF return status codes defined in section 6 of the RESTCONF draft are used in CoMI error responses, except they are converted to CoAP error codes.

TODO: complete RESTCONF to CoAP error code mappings

TODO: assign an error cpde for a rehash-error.

RESTCONF Status Line	CoAP Status Code
100 Continue	none?
200 OK	2.05
201 Created	2.01
202 Accepted	none?
204 No Content	?
304 Not Modified	2.03
400 Bad Request	4.00
403 Forbidden	4.03
404 Not Found	4.04
405 Method Not Allowed	4.05
409 Conflict	none?
412 Precondition Failed	4.12
413 Request Entity Too Large	4.13
414 Request-URI Too Large	4.00
415 Unsupported Media Type	4.15
500 Internal Server Error	5.00
501 Not Implemented	5.01
503 Service Unavailable	5.03

5. Mapping YANG to CoMI payload

A mapping for the encoding of YANG data in CBOR is necessary for the efficient transport of management data in the CoAP payload. Since object names may be rather long and may occur repeatedly, CoMI allows for association of a given object path identifier string value with an integer, called a "YANG hash".

5.1. YANG Hash Generation

The association between string value and string number is done through a hash algorithm. The 30 least significant bits of the "murmur3" 32-bit hash algorithm are used. This hash algorithm is described online at http://en.wikipedia.org/wiki/MurmurHash. Implementation are available online, including at https://code.google.com/p/smhasher/wiki/MurmurHash. When converting 4 input bytes to a 32-bit integer in the hash algorithm, the Little-Endian convention MUST be used.

The hash is generated for the string representing the object path identifier. A canonical representation of the path identifier is used.

Prefix values are used on every node.

The prefix values defined in the YANG module containing the data object are used for the path expression. For external modules, this is the value of the 'prefix' sub-statement in the 'import' statement for each external module.

Path expressions for objects which augment data nodes in external modules are calculated in the augmenting module, using the prefix values in the augmenting module.

Choice and case node names are not included in the path expression. Only 'container', 'list', 'leaf', 'leaf-list', and 'anyxml' nodes are listed in the path expression.

The "murmur3_32" hash function is executed for the entire path string. The value '42' is used as the seed for the hash function. The YANG hash is subsequently calculated by taking the 30 least significant bits.

The resulting 30-bit number is used by the server, unless the value is already being used for a different object by the server. In this case, the re-hash procedure in the following section is executed.

5.2. Re-Hash Error Procedure

A hash collision occurs if two different path identifier strings have the same hash value. If the server has over 30,000 objects in its YANG modules, then the probability of a collision is 10% or higher. If a hash collision occurs on the server, then the object that is causing the conflict has to be altered, such that the new hash value does not conflict with any value already in use by the server.

In most cases, the hash function is expected to produce unique values for all the objects supported by a constrained device. Given a known set of YANG modules, both server and client can calculate the YANG hashes independently, and offline.

Even though collisions are expected to happen rather rarely, they need to be considered. Collisions can be detected before deployment, if the vendor knows which modules are supported by the server, and hence all YANG hashes can be calculated. Collisions are only an issue when they occur at the same server. The client needs to discover any re-hash mappings on a per server basis.

If the server needs to re-hash any object identifiers, then it MUST create a "rehash-map" entry for all its rehashed objects, as described in the following YANG module.

5.3. ietf-yang-hash YANG Module

The "ietf-yang-hash" YANG module is used by the server to report any objects that have been mapped to produce a new hash value that does not conflict with any other YANG hash values used by the server.

YANG tree diagram for "ietf-yang-hash" module:

```
  +--ro yang-hash
      +--ro rehash* [hash]
         +--ro hash      uint32
         +--ro object*
            +--ro module     string
            +--ro newhash    uint32
            +--ro pathlen?   uint32
            +--ro path?      string
```

```
<CODE BEGINS> file "ietf-yang-hash@2015-06-06.yang"

module ietf-yang-hash {
  namespace "urn:ietf:params:xml:ns:yang:ietf-yang-hash";
  prefix "yh";

  organization
    "IETF CORE (Constrained RESTful Environments) Working Group";

  contact
    "WG Web:   <http://tools.ietf.org/wg/core/>
     WG List:  <mailto:core@ietf.org>
```

```
        WG Chair: Carsten Bormann
                  <mailto:cabo@tzi.org>

        WG Chair: Andrew McGregor
                  <mailto:andrewmcgr@google.com>

        Editor:   Peter van der Stok
                  <mailto:consultancy@vanderstok.org>

        Editor:   Andy Bierman
                  <mailto:andy@yumaworks.com>

        Editor:   Juergen Schoenwaelder
                  <mailto:j.schoenwaelder@jacobs-university.de>

        Editor:   Anuj Sehgal
                  <mailto:s.anuj@jacobs-university.de>";

     description
       "This module contains re-hash information for the CoMI protocol.

        Copyright (c) 2015 IETF Trust and the persons identified as
        authors of the code.  All rights reserved.

        Redistribution and use in source and binary forms, with or
        without modification, is permitted pursuant to, and subject
        to the license terms contained in, the Simplified BSD License
        set forth in Section 4.c of the IETF Trust's Legal Provisions
        Relating to IETF Documents
        (http://trustee.ietf.org/license-info).

        This version of this YANG module is part of RFC XXXX; see
        the RFC itself for full legal notices.";

     // RFC Ed.: replace XXXX with actual RFC number and remove this
     // note.

     // RFC Ed.: remove this note
     // Note: extracted from draft-vanderstok-core-comi-07.txt

     // RFC Ed.: update the date below with the date of RFC publication
     // and remove this note.
     revision 2015-06-06 {
       description
         "Initial revision.";
       reference
         "RFC XXXX: CoMI Protocol.";
     }
```

```
     container yang-hash {
       config false;
       description
         "Contains information on the YANG Hash values used by
          the server.";

       list rehash {
         key hash;
         description
           "Each entry describes an re-hash mapping in use by
            the server.";

         leaf hash {
           type uint32;
           description
             "The hash value that has a collision.  This hash value
              cannot be used on the server.  The rehashed
              value for each affected object must be used instead.";
         }

         list object {
           min-elements 2;

           description
             "Each entry identifies one of the objects involved in the
              hash collision and contains the rehash information for
              that object.";

           leaf module {
             type string;
             mandatory true;
             description
               "The module name for this object.";
           }

           leaf newhash {
             type uint32;
             mandatory true;
             description
               "The new hash value for this object.";
           }

           leaf pathlen {
             type uint32;
             description
               "The length of the path expression of the object with
                this hash value.  This object MUST be included
                for any objects in the rehash entry with the
```

```
                 same 'module' value.";
            }

            leaf path {
              type string;
              description
                "The path expression of the object with
                 this hash value. This object MUST be included
                 for any objects in the rehash entry with the
                 same 'module' and 'pathlen' values.";
            }

          }
        }
      }
   }

   <CODE ENDS>
```

5.4. YANG Re-Hash Examples

In this example there are three YANG modules, "foo", "bar", and "bar1".

```
module foo {
  namespace "http://example.com/ns/foo";
  prefix "f";
  revision 2015-06-07;

  container A {
    list B {
      key name;
      leaf name { type string; }
      leaf col1 { type int32; }
      leaf counter1 { type uint32; }
    }
  }
}

module bar {
  namespace "http://example.com/ns/bar";
  prefix "b";
  revision 2015-06-07;

  leaf bar { type string; }
}

module bar1 {
  namespace "http://example.com/ns/bar1";
  prefix "b1";
  import foo { prefix f; }
  revision 2015-06-07;

  augment /f:A/f:B {
    leaf bar1 { type string; }
  }
}
```

This set of 3 YANG modules containing a total of 7 objects produces the following object list. Note that actual hash values are not shown, since these modules do not actually cause the YANG Hash clashes described in the examples.

```
      Object    Path                Hash

   foo:

      container /f:A                 h1
      list      /f:A/f:B             h2
      leaf      /f:A/f:B/f:name      h3
      leaf      /f:A/f:B/f:col1      h4
      leaf      /f:A/f:B/f:counter1 h5

   bar:

      leaf      /b:bar               h6

   bar1:

     leaf      /f:A/f:B/b1:bar1     h7
```

5.4.1. Multiple Modules

In this example, assume that the following 3 objects produce the same hash value, so 'h3', 'h6', and 'h7' have the same value (e.g. '1234'):

The client might retrieve the container "/f:A" which could cause its sub-nodes to be returned. Instead, the server will return a message with the resource type "core.mg.", representing the "yang-hash" data structure.

```
REQ: GET example.com/mg/h1

RES: 4.00 "Bad Request" (Content-Format: application/cbor)
{
   "ietf-yang-hash:yang-hash" : {
     "rehash" : [
        {
          "hash" : 1234,
           "object" : [
             {
               "module" : "foo",
               "newhash" : 5678
             },
             {
               "module" : "bar",
               "newhash" : 3579
             },
             {
               "module" : "bar1",
               "newhash" : 8182
             }
           ]
        }
     ]
  }
}
```

5.4.2. Same Module

In this example, assume that the following 4 objects produce the same hash value, so ’h3’, ’h5’, ’h6’, and ’h7’ all have the same value (e.g. ’1234’):

The client might retrieve the list "/f:A/f:B" which would cause its sub-nodes to be returned. Instead, the server will return a message with the resource type "core.mg.yanh-hash", representing the "yang-hash" data structure. Note that the "pathlen" field is not needed for the ’h6’ and ’h7’ objects.

REQ: GET example.com/mg/h2?keys="entry1"

RES: 4.00 "Bad Request" (Content-Format: application/cbor)


```
{
   "ietf-yang-hash:yang-hash" : {
     "rehash" : [
        {
          "hash" : 1234,
           "object" : [
             {
               "module" : "foo",
               "newhash" : 5678,
               "pathlen" : 15
             },
             {
               "module" : "foo",
               "newhash" : 7863,
               "pathlen" : 19
             },
             {
               "module" : "bar",
               "newhash" : 3579
             },
             {
               "module" : "bar1",
               "newhash" : 8182
             }
           ]
        }
     ]
  }
}
```


5.4.3. Same Module and Same Path Length

In this example, assume that the following 5 objects produce the same hash value, so 'h3', 'h4', 'h5', 'h6', and 'h7' all have the same value (e.g. '1234'):

The client might retrieve the list "/f:A/f:B" which would cause its sub-nodes to be returned. Instead, the server will return a message with the resource type "core.mg.yang-hash", representing the "yang-hash" data structure. The "path" leaf is included 2 entries because the "module" and "pathlen" values are the same for the objects.

```
REQ: GET example.com/mg/h2?keys="entry2"

RES: 4.00 "Bad Request" (Content-Format: application/cbor)
{
   "ietf-yang-hash:yang-hash" : {
     "rehash" : [
        {
          "hash" : 1234,
           "object" : [
             {
               "module" : "foo",
               "newhash" : 5678,
               "pathlen" : 15,
               "path" : "/f:A/f:B/f:name"
             },
             {
               "module" : "foo",
               "newhash" : 7863,
               "pathlen" : 15,
               "path" : "/f:A/f:B/f:col1"
             },
             {
               "module" : "foo",
               "newhash" : 9172,
               "pathlen" : 19
             },
             {
               "module" : "bar",
               "newhash" : 3579
             },
             {
               "module" : "bar1",
               "newhash" : 8182
             }
           ]
        }
     ]
  }
}
```

5.5. YANG Hash in URL

When a URL contains a YANG hash, it is encoded using base64url "URL and Filename safe" encoding as specified in [RFC4648].

The hash H is represented as a 30-bit integer, divided into five 6-bit integers as follows:

```
B1 = (H & 0x3f000000) >> 24
B2 = (H & 0xfc0000) >> 18
B3 = (H & 0x03f000) >> 12
B4 = (H & 0x000fc0) >> 6
B5 =  H & 0x00003f
```

Subsequently, each 6-bit integer Bx is translated into a character Cx using Table 2 from [RFC4648], and a string is formed by concatenating the characters in the order C1, C2, C3, C4, C5.

For example, the YANG hash 0x29abdcca is encoded as "pq9zK".

6. Mapping YANG to CBOR

6.1. High level encoding

When encoding YANG variables in CBOR, the CBOR encodings entry is a map. The key is the YANG hash of entry variable, whereas the value contains its value.

For encoding of the variable values, a CBOR datatype is used. Section 6.2 provides the mapping between YANG datatypes and CBOR datatypes.

6.2. Conversion from YANG datatypes to CBOR datatypes

Table 1 defines the mapping between YANG datatypes and CBOR datatypes.

Elements of types not in this table, and of which the type cannot be inferred from a type in this table, are ignored in the CBOR encoding by default. Examples include the "description" and "key" elements. However, conversion rules for some elements to CBOR MAY be defined elsewhere.

YANG type	CBOR type	Specification
int8, int16, int32, int64, uint16, uint32, uint64, decimal64	unsigned int (major type 0) or negative int (mayor type 1)	The CBOR integer type depends on the sign of the actual value.
boolean	either "true" (major type 7,	

	simple value 21) or "false" (major type 7, simple value 20)	
string	text string (major type 3)	
enumeration	unsigned int (major type 0)	
bits	array of text strings	Each text string contains the name of a bit value that is set.
binary	byte string (major type 2)	
empty	null (major type 7, simple value 22)	TBD: This MAY not be applicable to true MIBs, as SNMP may not support empty variables...
union		Similar to the JSON transcription from [I-D.ietf-netmod-yang-json], the elements in a union MUST be determined using the procedure specified in section 9.12 of [RFC6020].
leaf-list	array (major type 4)	The array is encapsulated in the map associated with the YANG variable.
list	array (major type 4) of maps (major type 5)	Each array element contains a map of associated YANG hash - value pairs.
container	map (major type 5)	The map contains YANG hash - value pairs corresponding to the elements in the container.
smiv2:oid	array of integers	Each integer contains an element of the OID, the first integer in the array corresponds to the most left element in the OID.

Table 1: Conversion of YANG datatypes to CBOR

7. Error Handling

In case a request is received which cannot be processed properly, the managed entity MUST return an error message. This error message MUST contain a CoAP 4.xx or 5.xx response code, and SHOULD include additional information in the payload.

Such an error message payload is encoded in CBOR, using the following structure:

TODO: Adapt RESTCONF <errors> data structure for use in CoMI. Need to select the most important fields like <error-path>.

```
errorMsg     : ErrorMsg;

*ErrorMsg {
  errorCode  : uint;
  ?errorText : tstr;
}
```

The variable "errorCode" has one of the values from the table below, and the OPTIONAL "errorText" field contains a human readable explanation of the error.

CoMI Error Code	CoAP Error Code	Description
0	4.00	General error
1	4.00	Malformed CBOR data
2	4.00	Incorrect CBOR datatype
3	4.00	Unknown MIB variable
4	4.00	Unknown conversion table
5	4.05	Attempt to write read-only variable
0..2	5.01	Access exceptions
0..18	5.00	SMI error status

The CoAP error code 5.01 is associated with the exceptions defined in [RFC3416] and CoAP error code 5.00 is associated with the error-status defined in [RFC3416].

8. Security Considerations

For secure network management, it is important to restrict access to MIB variables only to authorised parties. This requires integrity protection of both requests and responses, and depending on the application encryption.

CoMI re-uses the security mechanisms already available to CoAP as much as possible. This includes DTLS [RFC6347] for protected access to resources, as well suitable authentication and authorisation mechanisms.

Among the security decisions that need to be made are selecting security modes and encryption mechanisms (see [RFC7252]). This requires a trade-off, as the NoKey mode gives no protection at all, but is easy to implement, whereas the X.509 mode is quite secure, but may be too complex for constrained devices.

In addition, mechanisms for authentication and authorisation may need to be selected.

CoMI avoids defining new security mechanisms as much as possible. However some adaptations may still be required, to cater for CoMI’s specific requirements.

9. IANA Considerations

’rt="core.mg.data"’ needs registration with IANA.

’rt="core.mg.moduri"’ needs registration with IANA.

’rt="core.mg.modset"’ needs registration with IANA.

’rt="core.mg.yang-hash"’ needs registration with IANA.

’rt="core.mg.yang-stream"’ needs registration with IANA.

Content types to be registered:

- application/comi+cbor

10. Acknowledgements

We are very grateful to Bert Greevenbosch who was one of the original authors of the CoMI specification and specified CBOR encoding and use of hashes. Mehmet Ersue and Bert Wijnen explained the encoding aspects of PDUs transported under SNMP. Carsten Bormann has given feedback on the use of CBOR. The draft has benefited from comments (alphabetical order) by Dee Denteneer, Esko Dijk, Michael van Hartskamp, Zach Shelby, Michel Veillette, Michael Verschoor, and Thomas Watteyne. The CBOR encoding borrows extensively from Ladislav Lhotka’s description on conversion from YANG to JSON.

This material is based upon work supported by Philips Research, Huawei, and The Space & Terrestrial Communications Directorate (S&TCD); the latter under Contract No. W15P7T-13-C-A616. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of Philips Research, Huawei, or The Space & Terrestrial Communications Directorate (S&TCD).

Juergen Schoenwaelder and Anuj Sehgal were partly funded by Flamingo, a Network of Excellence project (ICT-318488) supported by the European Commission under its Seventh Framework Programme.

11. Changelog

Changes from version 00 to version 01

- Focus on MIB only
- Introduced CBOR, JSON, removed BER
- defined mappings from SMI to xx
- Introduced the concept of addressable table rows

Changes from version 01 to version 02

- Focus on CBOR, used JSON for examples, removed XML and EXI
- added uri-query attributes mod and con to specify modules and contexts
- Definition of CBOR string conversion tables for data reduction
- use of Block for multiple fragments
- Error returns generalized

o SMI - YANG - CBOR conversion

Changes from version 02 to version 03

o Added security considerations

Changes from version 03 to version 04

o Added design considerations section

o Extended comparison of management protocols in introduction

o Added automatic generation of CBOR tables

o Moved lowpan table to Appendix

Changes from version 04 to version 05

o Merged SNMP access with RESTCONF access to management objects in small devices

o Added CoMI architecture section

o Added RESTCONf NETMOD description

o Rewrote section 5 with YANG examples

o Added server and payload size appendix

o Removed Appendix C for now. It will be replaced with a YANG example.

Changes from version 04 to version 05

o Extended examples with hash representation

o Added keys query parameter text

o Added select query parameter text

o Better separation between specification and instance

o Section on discovery updated

o Text on rehashing introduced

o Elaborated SMI MIB example

- o Yang libary use described

- o use of BigEndian/LittleEndian in Hash generation specified

Changes from version 05 to version 06

- o Hash values in payload as hexadecimal and in URL in base64 numbers

- o Streamlined CoMI architecture text

- o Added select query parameter text

- o Data editing optional

- o Text on Notify added

- o Text on rehashing improved with example

Changes from version 06 to version 07

- o reduced payload size by removing JSON hierachy

- o changed rehash handling to support small clients

- o added LWM2M comparison

- o Notification handling as specified in YANG

- o Added Patch function

- o Rehashing completely reviewed

- o Discover type of YANG name encoding

- o Added new resource types

- o Read-only servers introduced

- o Multiple updates explained

12. References

12.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, October 2006.

[RFC5277] Chisholm, S. and H. Trevino, "NETCONF Event Notifications", RFC 5277, July 2008.

[RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

[RFC7049] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", RFC 7049, October 2013.

[RFC7159] Bray, T., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, March 2014.

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, June 2014.

[I-D.becker-core-coap-sms-gprs] Becker, M., Li, K., Kuladinithi, K., and T. Poetsch, "Transport of CoAP over SMS", draft-becker-core-coap-sms-gprs-05 (work in progress), August 2014.

[I-D.ietf-core-block] Bormann, C. and Z. Shelby, "Block-wise transfers in CoAP", draft-ietf-core-block-17 (work in progress), March 2015.

[I-D.ietf-core-observe] Hartke, K., "Observing Resources in CoAP", draft-ietf-core-observe-16 (work in progress), December 2014.

[I-D.ietf-netmod-yang-json] Lhotka, L., "JSON Encoding of Data Modeled with YANG", draft-ietf-netmod-yang-json-04 (work in progress), June 2015.

[I-D.ietf-netconf-restconf] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", draft-ietf-netconf-restconf-06 (work in progress), June 2015.

[I-D.vanderstok-core-patch] Stok, P. and A. Sehgal, "Patch Method for Constrained Application Protocol (CoAP)", draft-vanderstok-core-patch-00 (work in progress), March 2015.

12.2. Informative References

[RFC2578] McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., "Structure of Management Information Version 2 (SMIv2)", STD 58, RFC 2578, April 1999.

[RFC3410] Case, J., Mundy, R., Partain, D., and B. Stewart, "Introduction and Applicability Statements for Internet-Standard Management Framework", RFC 3410, December 2002.

[RFC3411] Harrington, D., Presuhn, R., and B. Wijnen, "An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks", STD 62, RFC 3411, December 2002.

[RFC3414] Blumenthal, U. and B. Wijnen, "User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)", STD 62, RFC 3414, December 2002.

[RFC3416] Presuhn, R., "Version 2 of the Protocol Operations for the Simple Network Management Protocol (SNMP)", STD 62, RFC 3416, December 2002.

[RFC3418] Presuhn, R., "Management Information Base (MIB) for the Simple Network Management Protocol (SNMP)", STD 62, RFC 3418, December 2002.

[RFC4293] Routhier, S., "Management Information Base for the Internet Protocol (IP)", RFC 4293, April 2006.

[RFC4944] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", RFC 4944, September 2007.

[RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, January 2012.

[RFC6643] Schoenwaelder, J., "Translation of Structure of Management Information Version 2 (SMIv2) MIB Modules to YANG Modules", RFC 6643, July 2012.

[RFC6650] Falk, J. and M. Kucherawy, "Creation and Use of Email Feedback Reports: An Applicability Statement for the Abuse Reporting Format (ARF)", RFC 6650, June 2012.

[RFC6690] Shelby, Z., "Constrained RESTful Environments (CoRE) Link Format", RFC 6690, August 2012.

[RFC6775] Shelby, Z., Chakrabarti, S., Nordmark, E., and C. Bormann, "Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)", RFC 6775, November 2012.

[RFC7223] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 7223, May 2014.

[RFC7228] Bormann, C., Ersue, M., and A. Keranen, "Terminology for Constrained-Node Networks", RFC 7228, May 2014.

[RFC7317] Bierman, A. and M. Bjorklund, "A YANG Data Model for System Management", RFC 7317, August 2014.

[I-D.ietf-core-interfaces]
Shelby, Z. and M. Vial, "CoRE Interfaces", draft-ietf-core-interfaces-02 (work in progress), November 2014.

[I-D.ersue-constrained-mgmt]
Ersue, M., Romascanu, D., and J. Schoenwaelder, "Management of Networks with Constrained Devices: Problem Statement, Use Cases and Requirements", draft-ersue-constrained-mgmt-03 (work in progress), February 2013.

[I-D.ietf-lwig-coap]
Kovatsch, M., Bergmann, O., and C. Bormann, "CoAP Implementation Guidance", draft-ietf-lwig-coap-02 (work in progress), June 2015.

[XML] "Extensible Markup Language (XML)", Web http://www.w3.org/xml.

[OMA] "OMA-TS-LightweightM2M-V1_0-20131210-C", Web http://technical.openmobilealliance.org/Technical/current_releases.aspx.

[DTLS-size]
Hummen, R., Shafagh, H., Raza, S., Voigt, T., and K. Wehrle, "Delegation-based Authentication and Authorization for the IP-based Internet of Things", Web http://www.vs.inf.ethz.ch/publ/papers/mshafagh_secon14.pdf.

[dcaf] Bormann, C., Bergmann, O., and S. Gerdes, "Delegated Authenticated Authorization for Constrained Environments", Private Information .

[openwsn] Watteijne, T., "Coap size in Openwsn", Web http://builder.openwsn.org/.

[Erbium] Kovatsch, M., "Erbium Memory footprint for coap-18", Private Communication .

[management]
Schoenwalder, J. and A. Sehgal, "Management of the Internet of Things", Web http://cnds.eecs.jacobs-university.de/slides/2013-im-iot-management.pdf, 2013.

[I-D.ietf-netconf-yang-patch]
Bierman, A., Bjorklund, M., and K. Watsen, "YANG Patch Media Type", draft-ietf-netconf-yang-patch-04 (work in progress), June 2015.

Appendix A. Payload and Server sizes

This section provides information on code sizes and payload sizes for a set of management servers. Approximate code sizes are:

Code	processor	Text	Data	reference
Observe agent	erbium	800	n/a	[Erbium]
CoAP server	MSP430	1K	6	[openwsn]
SNMP server	ATmega128	9K	700	[management]
Secure SNMP	ATmega128	30K	1.5K	[management]
DTLS server	ATmega128	37K	2K	[management]
NETCONF	ATmega128	23K	627	[management]
JSON parser	CC2538	4.6K	8	[dcaf]
CBOR parser	CC2538	1.5K	2.6K	[dcaf]
DTLS server	ARM7	15K	4	[I-D.ietf-lwig-coap]
DTLS server	MSP430	15K	4	[DTLS-size]
Certificate	MSP430	23K		[DTLS-size]
Crypto	MSP430	2-8K		[DTLS-size]

Thomas says that the size of the CoAP server is rather arbitrary, as its size depends mostly on the implementation of the underlying library modules and interfaces.

Payload sizes are compared for the following request payloads, where each attribute value is null (N.B. these sizes are educated guesses, will be replaced with generated data). The identifier are assumed to be a string representation of the OID. Sizes for SysUpTime differ due to preambles of payload. "CBOR opt" stands for CBOR payload where the strings are replaced by table numbers.

Request	BERR SNMP	JSON	CBOR	CBOR opt
IPnetTOMediaTable	205	327	~327	~51
lowpanIfStatsTable		710	614	121
sysUpTime	29	13	~13	20
RESTCONF example				

Appendix B. Notational Convention for CBOR data

To express CBOR structures [RFC7049], this document uses the following conventions:

A declaration of a CBOR variable has the form:

```
name : datatype;
```

where "name" is the name of the variable, and "datatype" its CBOR datatype.

The name of the variable has no encoding in the CBOR data.

"datatype" can be a CBOR primitive such as:

tstr: A text string (major type 3)

uint: An unsigned integer (major type 0)

map(x,y): A map (major type 5), where each first element of a pair is of datatype x, and each second element of datatype y. A '.' character for either x or y means that all datatypes for that element are valid.

A datatype can also be a CBOR structure, in which case the variable's "datatype" field contains the name of the CBOR structure. Such CBOR structure is defined by a character sequence consisting of first its name, then a '{' character, then its subfields and finally a '}' character.

A CBOR structure can be encapsulated in an array, in which case its name in its definition is preceded by a '*' character. Otherwise the structure is just a grouping of fields, but without actual encoding of such grouping.

The name of an optional field is preceded by a '?' character. This means, that the field may be omitted if not required.

Appendix C. comparison with LWM2M

CoMI and LWM2M, both, provide RESTful device management services over CoAP. Differences between the designs are highlighted in this section.

Unlike CoMI, which enables the use of SMIv2 and YANG data models for device management, LWM2M defines a new object resource model. This means that data models need to be redefined in order to use LWM2M. In contrast, CoMI provides access to a large variety of SMIv2 and YANG data modules that can be used immediately.

Objects and resources within CoMI are identified with a YANG hash value, however, each object is described as a link in the CoRE Link Format by LWM2M. This approach by LWM2M can lead to larger complex URIs and more importantly payloads can grow large in size. Using a hash value to represent the objects and resources allows URIs and payloads to be smaller in size, which is important for constrained devices that may not have enough resources to process large messages.

LWM2M encodes payload data in Type-length-value (TLV), JSON or plain text formats. While the TLV encoding is binary and can result in reduced message sizes, JSON and plain text are likely to result in large message sizes when lots of resources are being monitored or configured. Furthermore, CoMI's use of CBOR gives it an advantage over the LWM2M's TLV encoding as well since this too is more efficient [citation needed].

CoMI is aligned with RESTCONF for constrained devices and uses YANG data models that have objects containing resources organized in a tree-like structure. On the other hand, LWM2M uses a very flat data model that follows the "object/instance/resouce" format, with no possibility to have subresouces. Complex data models are, as such, harder to model with LWM2M.

In situations where resources need to be modified, CoMI uses the CoAP PATCH operation when resources are modified partially. However, LWM2M uses the CoAP PUT and POST operations, even when a subset of the resource needs modifications.

Authors' Addresses

Peter van der Stok
consultant

Phone: +31-492474673 (Netherlands), +33-966015248 (France)
Email: consultancy@vanderstok.org
URI: www.vanderstok.org

Andy Bierman
YumaWorks
685 Cochran St.
Suite #160
Simi Valley, CA 93065
USA

Email: andy@yumaworks.com

Juergen Schoenwaelder
Jacobs University
Campus Ring 1
Bremen 28759
Germany

Email: j.schoenwaelder@jacobs-university.de

Anuj Sehgal
consultant
Campus Ring 1
Bremen 28759
Germany

Email: anuj@iurs.org

core P. van der Stok
Internet-Draft A. Sehgal
Intended status: Informational Consultant
Expires: January 7, 2016 July 6, 2015

Patch Method for Constrained Application Protocol (CoAP)
draft-vanderstok-core-patch-01

Abstract

The existing Constrained Application Protocol (CoAP) PUT method only allows a complete replacement of a resource. This does not permit applications to perform partial resource modifications. In case of resources with larger or complex data, or in situations where a resource continuity is required, replacing a resource is not an option. Several applications using CoAP will need to perform partial resource modifications. This proposal adds a new CoAP method, PATCH, to modify an existing CoAP resource partially.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This specification defines the new Constrained Application Protocol (CoAP) [RFC7252] method, PATCH, which is used to apply partial modifications to a resource.

PATCH is also specified for HTTP in [RFC5789]. Most of the motivation for PATCH described in [RFC5789] also applies here.

The PUT method exists to overwrite a resource with completely new contents, and cannot be used to perform partial changes. When using PUT for partial changes, proxies and caches, and even clients and servers, may get confused as to the result of the operation. PATCH was not adopted in an early design stage of CoAP, however, it has become necessary with the arrival of applications that require partial updates to resources (e.g. [I-D.vanderstok-core-comi]). Using PATCH avoids transferring all data associated with a resource in case of modifications, thereby not burdening the constrained communication medium.

This document relies on knowledge of the PATCH specification for HTTP [RFC5789]. This document provides extracts from [RFC5789] to make independent reading possible.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

1.2. Terminology and Acronyms

This document uses terminology defined in [RFC5789] and [RFC7252].

2. Patch Method

The PATCH method requests that a set of changes described in the request payload is applied to the target resource of the request. The set of changes is represented in a format identified by a media type. If the Request-URI does not point to an existing resource, the server MAY create a new resource with that URI, depending on the patch document type (whether it can logically modify a null resource) and permissions, etc. Creation of a new resource would result in a 2.01 (Created) Response Code dependent of the patch document type.

Restrictions to a PATCH can be made by including the If-Match or If-None-Match options in the request (see Section 5.10.8.1 and 5.10.8.2 of [RFC7252]). If the resource could not be created or modified, then an appropriate Error Response Code SHOULD be sent.

The difference between the PUT and PATCH requests is extensively documented in [RFC5789].

PATCH is not safe but idempotent conformant to CoAP PUT specified in [RFC7252], Section 5.8.3.

A PATCH request is idempotent to prevent bad outcomes from collisions between two PATCH requests on the same resource in a similar time frame. These collisions can be detected with the MessageId and the source end-point provided by the CoAP protocol (see section 4.5 of [RFC7252].

The server MUST apply the entire set of changes atomically and never provide a partially modified representation to a concurrently executed GET request. Given the constrained nature of the servers, most servers will only execute CoAP requests consecutively, thus preventing a concurrent partial overlapping of request modifications. In general, modifications MUST NOT be applied to the server state when an error occurs or only a partial execution is possible. The atomicity requirement holds for all directly affected resources.

A PATCH response can invalidate a cache conformant with the PUT response. Caching behaviour as function of the valid 2.xx response codes for PATCH are:

A 2.01 (Created) response invalidates any cache entry for the resource indicated by the Location-* Options; the payload is a representation of the action result.

A 2.04 (Changed) response invalidates any cache entry for the target resource; the payload is a representation of the action result.

There is no guarantee that a resource can be modified with PATCH. Servers are required to support a subset of the content formats as specified in sections 12.3 and 5.10.3 of [RFC7252]. Servers MUST ensure that a received PATCH payload is appropriate for the type of resource identified by the target resource of the request.

Clients MUST choose to use PATCH rather than PUT when the request affects partial updates of a given resource.

2.1. A Simple PATCH Example

The example is taken over from [RFC6902], which specifies a JSON notation for PATCH operations. A resource located at www.example.com/object contains a target JSON document.

```
         JSON document orginal state
              {
                 "x-coord": 256,
                 "y-coord": 45
              }

                    REQ:
                   PATCH CoAP://www.example.com/object
                    [
                  { "op":"replace","path":"x-coord","value":45}
               ]
                    RET:
                         CoAP 2.04 Changed

         JSON document final state
              {
                 "x-coord": 45,
                 "y-coord": 45
              }
```

This example illustrates use of a hypothetical PATCH on the /object/x-coord of the existing resource "object". The 2.04 (Changed) response code is conforms with the CoAP PUT method.

2.2. Response Codes

PATCH for CoAP adopts the response codes as specified in sections 5.9 and 12.1.2 of [RFC7252].

2.3. Option Numbers

PATCH for CoAP adopts the option numbers as specified in sections 5.10 and 12.2 of [RFC7252].

3. Error Handling

A PATCH request may fail under certain known conditions. These situations should be dealt with as expressed below.

Malformed PATCH payload: If a server determines that the payload provided with a PATCH request is not properly formatted, it can return a 4.00 (Bad Request) CoAP error. The definition of a malformed payload depends upon the CoAP Content-Format specified with the request.

Unsupported PATCH payload: In case a client sends payload that is inappropriate for the resource identified by the Request-URI, the server can return a 4.15 (Unsupported Content-Format) CoAP error. The server can determine if the payload is supported by checking the CoAP Content-Format specified with the request.

Unprocessable request: This situation occurs when the payload of a PATCH request is determined as valid, i.e. well-formed and supported, however, the server is unable to or incapable of processing the request. The server can return a 4.22 (Unprocessable Entity) CoAP error. More specific scenarios might include situations when:

* the server has insufficient computing resources to complete the request successfully -- 4.13 (Request Entity Too Large) CoAP Response Code,

* the resource specified in the request becomes invalid by applying the payload -- 4.06 (Not Acceptable) CoAP Response Code,

In case there are more specific errors that provide more insight into the problem, then those should be used.

Resource not found: The 4.04 (Not Found) error should be returned in case the payload of a PATCH request cannot be applied to a non-existent resource.

Request too large: If the payload of the PATCH request is larger than a CoAP server can process, then it can return the 4.13 (Request Entity Too Large) CoAP error.

Conflicting state: If the modification specified by a PATCH request cannot be applied to a resource in its current state, or causes the resource to enter an inconsistent state the server can return the 4.09 (Conflict) CoAP response. Such a situation might be encountered when a structural modification is applied to a configuration data-store, but the structures being modified do not exist or lead the device into an inconsistent state if the modifications are made.

Conflicting modification: In situations when a server detects possible conflicting modifications the server can return a 4.nr2 CoAP response code.

Concurrent modification: Resource constrained devices might need to process requests in the order they are received. In case requests are received concurrently to modify the same resource but they cannot be queued, the server can return a 4.09 (Conflict) CoAP response code.

It is possible that other error situations, not mentioned here, are encountered by a CoAP server while processing the PATCH request. In these situations other appropriate CoAP status codes can also be returned.

4. Security Considerations

This section analyses the possible threats to the CoAP PATCH protocol. It is meant to inform protocol and application developers about the security limitations of CoAP PATCH as described in this document. The security consideration of section 15 of [RFC2616], section 11 of [RFC7252], and section 5 of [RFC5789] also apply.

The security considerations for PATCH are nearly identical to the security considerations for PUT ([RFC7252]). The mechanisms used for PUT can be used for PATCH as well.

PATCH is secured following the CoAP recommendations as specified in section 9 of [RFC7252]. When more appropriate security techniques are standardized for CoAP, PATCH can also be secured by those new techniques.

5. IANA Considerations

The entry with name PATCH in the sub-registry, "CoAP Method Codes", is 0.05. the addition will follow the "IETF Review or IESG Approval" procedure as described in [RFC5226].

TODO, definition of CoAP response code 4.09 for addition to the sub-registry of CoAP response codes.

Additions to the sub-registry "CoAP Content-Formats", within the "CoRE Parameters" registry are needed for the following media type formats: "application/json-patch+json" [RFC6902], and "application/merge-patch+json" [RFC7386].

6. Acknowledgements

Klaus Hartke has pointed out some essential differences between CoAP and HTTP. We are grateful for discussions with Carsten Bormann, Kovatsch Matthias, and Thomas Watteyne.

7. Change log

When published as a RFC, this section needs to be removed.

Version 0 to version 1:

- o Changed patch motivation text.
- o Removed sub-resource concept.
- o Updated cache handling.
- o Extended example.
- o Update of error handling.

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[RFC5789] Dusseault, L. and J. Snell, "PATCH Method for HTTP", RFC 5789, March 2010.

[RFC6902] Bryan, P. and M. Nottingham, "JavaScript Object Notation (JSON) Patch", RFC 6902, April 2013.

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, June 2014.

[RFC7386] Hoffman, P. and J. Snell, "JSON Merge Patch", RFC 7386, October 2014.

8.2. Informative References

[I-D.vanderstok-core-comi] Stok, P., Greevenbosch, B., Bierman, A., Schoenwaelder, J., and A. Sehgal, "CoAP Management Interface", draft-vanderstok-core-comi-06 (work in progress), February 2015.

Authors' Addresses

Peter van der Stok
Consultant

Email: consultancy@vanderstok.org

Anuj Sehgal
Consultant

Email: anuj@iurs.org

6TiSCH Q. Wang, Ed.
Internet-Draft Univ. of Sci. and Tech. Beijing
Intended status: Informational X. Vilajosana
Expires: January 3, 2016 Universitat Oberta de Catalunya
T. Watteyne
Linear Technology
R. Sudhaakar
Cisco Systems
P. Zand
University of Twente
July 2, 2015

Transporting CoAP Messages over IEEE802.15.4e Information Elements
draft-wang-6tisch-6top-coapie-01

Abstract

This document describes the format of "CoAP IE", an IEEE802.15.4e Information Element which allows CoAP messages to be transported as part of the IEEE802.15.4e payload IE. This enables 6top-to-6top communication between neighbor nodes in a 6TiSCH network.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

1.1. Requirements Notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

1.2. Context within 6TiSCH

This document fits in the work done at the IETF 6TiSCH WG as follows:

- o [I-D.wang-6tisch-6top-sublayer] defines the operation of the 6top sublayer, which monitors and manages the communication schedule used in the [IEEE802154e] TSCH network.

- [I-D.ietf-6tisch-6top-interface] defines the interface of the 6top sublayer using the YANG data modeling language [RFC6020].

- [I-D.ietf-6tisch-coap] translates this YANG model in CoAP resources and interactions, allowing an Internet host (possibly but not necessarily constrained) to monitor and manage the 6top sublayer of a 6TiSCH device.

- This document defines a method for transporting those CoAP messages as part of the IEEE802.15.4e payload IE. It does so by defining a new IEEE802.15.4e Information Element called "CoAP IE". This allows a 6TiSCH node to monitor and manage the 6top sublayer and enables pairwise communication for signaling and control between neighbor nodes.

1.3. Motivation

The 6TiSCH architecture [I-D.ietf-6tisch-architecture] allows for both centralized and distributed monitoring and management of a 6TiSCH schedule. [I-D.ietf-6tisch-coap] defines the mechanisms necessary for the centralized case. The present document defines a mechanism enabling the communication of nodes in a 1 hop neighborhood, enabling a distributed approach.

In particular, it allows a node to monitor and manage its neighbor node’s MIB. Through the CoAP IE defined in this document, a node sends link-layer frames to its neighbor which contain, as part of the link-layer payload IE, the CoAP messages defined in [I-D.ietf-6tisch-coap]. This allows a node to interact with the 6top interface of its neighbor, in a way equivalent to an Internet host interacting with a 6TiSCH device over CoAP.

In addition, this document describe the frame formats and interaction between a node and its neighbor during softcell negotiation [I-D.wang-6tisch-6top-sublayer], through the addition of an Remote Procedure Call "RPC" element to the YANG model defined in [I-D.ietf-6tisch-6top-interface].

We call "6top-to-6top" communication the interaction between a node and its neighbor using the CoAP IE.

1.4. Status of this Document

The authors decided to present the CoAP IE as a separate document to request discussion and suggestions for improvement from the Internet community.

If the document gets support, and after suggestions for improvement have been integrated, the author propose to merge it in existing 6TiSCH I-Ds as follows:

- o Section 3 would go into [I-D.ietf-6tisch-6top-interface];
- o Section 4 would go into [I-D.ietf-6tisch-coap];
- o Section 2 and Section 5 would go into [I-D.wang-6tisch-6top-sublayer].

2. CoAP IE Format

The CoAP IE is a container for transporting CoAP messages as part of the IEEE802.15.4e payload IE, as an Information Element. It is used by both the management interface and the softcell negotiation interface for 6top-to-6top communication.

This IE is not present in [IEEE802154e]; it is defined in this document.

Format of a CoAP IE.

```
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Length      |    SubID    |T|                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               |
//                                                             //
|                   Fragmented CoAP message                     |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1

The fields in CoAP IE header are defined as follows.

- o Length = 1
- o SubID = 0x44
- o T = 0 (short type)

The content of CoAP IE is a CoAP message compliant to [RFC7252]. The CoAP message MAY use the CoAP Block option (see Section 4.2) in order to fragment large CoAP messages.

Format of CoAP IE with CoAP message.

```
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      CoAP IE header          |Ver| T |  TKL  |    Code        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Message ID                  |    Token (0-8B, assume 2B)     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Uri-path option                                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  11111111     |                                               |
+-+-+-+-+-+-+-+-+                                               |
//                                                             //
|                 CoAP message payload (variable)               |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2

The Token Length (TKL)is set to 2;

Per [RFC7252], the Uri-path field consists of the following sub-fields:

- o Option Delta: 4bits, set to 11
- o Option Length: 4bits, set to 3
- o Option value: 3 bytes

The first byte of the option value is set to "6" (for 6top), "4" (for IEEE802.15.4), or "e" (for extension). The second and third bytes refer to the resource name in the corresponding group.

3. Softcell Negotiation Interface RPC Definition

This document proposes to replace the "6top Communication Protocol" defined in [I-D.wang-6tisch-6top-sublayer] by an extension to the YANG data model defined in [I-D.ietf-6tisch-6top-interface]. This allows neighbor nodes to negotiate the allocation of soft cells using the CoAP IE.

```
   rpc softcell-negotiation {
      input {
            leaf Opcode {
               type enumeration {
                  enum RESERVATION;
                  enum REMOVE;
               }
            }
            leaf RequiredBW {
               type uint8;
            }
            leaf SlotframeID {
               type uint8;
           }
           leaf TrackID {
               type uint16;
               description
               "TrackID points to a tuple(TrackOwnerAddr,
               InstanceID)";
           }
           leaf NumofCandidate {
               type uint8;
           }
           List CandidateList {
               key "SlotOffset ChannelOffset";
               leaf SlotOffset{
                  type uint16;
               }
               leaf ChannelOffset{
                  type uint16;
               }
           }
      }
      output {
         leaf NumOfCells {
            type uint8;
         }
         List ResultedCells {
            key "SlotOffset ChannelOffset";
            leaf SlotOffset{
               type uint16;
            }
            leaf ChannelOffset{
               type uint16;
            }
         }
      }
   }
```

4. CoAP support

4.1. URI setting

Uri-Host option = target node address;

Uri-Path option = 6t/6/[6top resource name], or 6t/4/[15.4 resource name], or 6t/e/[extension resource name], where [6top resource name] refers to the data resources or RPC defined by 6top, [15.4 resource name] refers to the data resources defined by IEEE802.15.4, and [extension resource name] refers to the data resources defined by an extensions of 6top, e.g. OTF. [6top resource name] , [154 resource name] and [extension resource name] are RECOMMENDED to be at most 2 bytes long.

4.2. CoAP Block option

In [I-D.ietf-core-block], two block options (Block1 and Block2) are defined to support block-wise transfers. The format of a fragmented message in a CoAP IE is defined as follows.

Format of CoAP IE content with fragmented message.

```
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   CoAP IE header              |Ver| T |  TKL  |   Code        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Message ID                   |    Token                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Uri-path option                                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|option |option | option delta  |  NUM                  |M| SZX |
| delta |length |  extended     |                       | |     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   11111111    |                                               |
+++++++++++++++++                                               |
//                                                             //
|                 fragmented payload (64B)                      |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3

Per [I-D.ietf-core-block], the option Delta is 23 for Block1 and 27 for Block2. Related sub-fields are defined as follows.

- o Option delta: 4bits, set to 13, indicates an 8-bit unsigned integer follows the initial byte and the Option Delta minus 13.

- o Option length: 4bits, set to 2.

- o Option delta extended: 8bits, 23-13=10 and 27-13=14 for Block1 and Block2, respectively.

Per [IEEE802154], assuming the IE size constraint is 81 bytes, the related fields of the block option are defined as follows.

- o The size of the block (SZX): 3 bits, representing block size 16B/32B/64B/128B/256B/512B/1024B. Considering the IE size constrained by [IEEE802154], 16B/32B/64B block size will be used. Invalid block size values will cause the packet to be dropped quietly.

- o Whether more blocks are following (M): 1 bit;

- o The relative number of the block (NUM): 12 bits, within a sequence of blocks with the given size. NUM is 4bits or 12bits, or 20bits

4.3. Management Interface Protocol

Management and MIB handling is handled by the protocol specification defined in [I-D.ietf-6tisch-coap].

4.4. Negotiation interface protocol

The negotiation protocol is used by neighbor nodes to agree at what slotOffset/channelOffset to add/remove sotfcells. It uses a Uri-Path option to identify the target resource (i.e the negotiation interface of the neighbor).

The example below illustrates the use of this negotiation interface. It assumes the RPC softcell-negotiation is at Uri-Path "6t/6/ng".

```
   nodeA   nodeB
     |       |
     +------>| IEEE802.15.4e type: DATA
     | POST  |        CoAP Header: POST (T=CON)
     |       |           Uri-Path: "6t/6/ng"
     |       |            Payload: CBOR(
     |       |                        Opcode=RESERVATION,
     |       |                        RequiredBW,
     |       |                        SlotframeID,
     |       |                        TrackID,
     |       |                        NumOfCandidate,
     |       |                        CandidateList
     |       |                     )
     |       |
     |<------+ IEEE802.15.4e type: ACK
     |       |
     |<------+ IEEE802.15.4e type: DATA
     | 2.04  |        CoAP Header: 2.04 Changed (T=ACK, Code=2.04)
     |       |            Payload: CBOR(
     |       |                        NumOfCells,
     |       |                        ResultedCells
     |       |                     )
     |       |
     +-------> IEEE802.15.4e type: ACK
     |       |
```

Node A send a CoAP POST request, using a confirmable message. Node B sends back a IEEE802.15.4e ACK to confirm reception. This layer 2 ACK does not give any indication about the correct handling of the command, or even about whether this command is well formatted and understood. Node B parses the CoAP IE, and if correct, calls the appropriate 6top command to allocate softcells. When the allocation is done, node B sends back a CoAP Response with the appropriate return code to node A as a IEEE802.15.4e data packet. The CoAP ACK MUST be piggybacked on the Response.

4.5. Acknowledgement

For both non-fragmented CoAP message and fragmented CoAP message, an Acknowledgement message of CoAP is used. The Acknowledgement message of CoAP is inserted into a CoAP IE, which is carried in the Data Frame or Enhanced Acknowledgement frame of [IEEE802154e].

4.6. Observe

The Observe mechanism is a option for 6top-to-6top communication. The Token in the CoAP message is used to bind Observe message and its Response messages.

5. Implementation Considerations

Similar to the formatting and the parser modules used by CoAP (Layer 5), a CoAP formatting and parser modules are present in the 6top sublayer.

```
+------------------------------------+
| PCEP | CoAP |      | 6LoWPAN |     |
| PCC  | DTLS | PANA |    ND   |RPL |
+--------------------------------------------+
| TCP  |     UDP     |     ICMP     | RSVP |
+--------------------------------------------+
|                  IPv6                      |
+--------------------------------------------+
|                6LoWPAN HC                  |
+--------------------------------------------+
| +--------------+      +----------------+ |
| |  CoAP Parser |      | CoAP Formatting| |
| +--------------+      +----------------+ |
| +--------------+      +----------------+ |
| | IE Parser    |      |  IE Formatting | |
| +--------------+      +----------------+ |
+--------------------------------------------+
|          IEEE802.15.4e TSCH                |
+--------------------------------------------+
|             IEEE802.15.4                   |
+--------------------------------------------+
```

Figure 4

When the IE parser identifies a CoAP IE in the data packet, it passes the IE content (i.e. the fragmented CoAP message) to the CoAP Parser. The CoAP Parser then assembles those fragmented CoAP messages, and takes the appropriate action based on the CoAP Code, Uri-Path, and payload.

When a CoAP message is formatted, it MAY be fragmented, then passed to the IE Formatting module. The IE Formatting module puts those (possibly fragmented) CoAP message(s) into a CoAP IE and pases them to the IEEE802.15.4e TSCH layer as separate packets.

6. References

6.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

6.2. Informative References

[RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, June 2014.

[I-D.ietf-core-block]
Bormann, C. and Z. Shelby, "Block-wise transfers in CoAP", draft-ietf-core-block-17 (work in progress), March 2015.

[I-D.ietf-6tisch-tsch]
Watteyne, T., Palattella, M., and L. Grieco, "Using IEEE802.15.4e TSCH in an IoT context: Overview, Problem Statement and Goals", draft-ietf-6tisch-tsch-06 (work in progress), March 2015.

[I-D.ietf-6tisch-architecture]
Thubert, P., "An Architecture for IPv6 over the TSCH mode of IEEE 802.15.4", draft-ietf-6tisch-architecture-08 (work in progress), May 2015.

[I-D.ietf-6tisch-terminology]
Palattella, M., Thubert, P., Watteyne, T., and Q. Wang, "Terminology in IPv6 over the TSCH mode of IEEE 802.15.4e", draft-ietf-6tisch-terminology-04 (work in progress), March 2015.

[I-D.ietf-6tisch-minimal]
Vilajosana, X. and K. Pister, "Minimal 6TiSCH Configuration", draft-ietf-6tisch-minimal-10 (work in progress), June 2015.

[I-D.ietf-6tisch-6top-interface]
Wang, Q., Vilajosana, X., and T. Watteyne, "6TiSCH Operation Sublayer (6top) Interface", draft-ietf-6tisch-6top-interface-03 (work in progress), March 2015.

[I-D.ietf-6tisch-coap]
Sudhaakar, R. and P. Zand, "6TiSCH Resource Management and Interaction using CoAP", draft-ietf-6tisch-coap-03 (work in progress), March 2015.

[I-D.wang-6tisch-6top-sublayer]
Wang, Q., Vilajosana, X., and T. Watteyne, "6TiSCH Operation Sublayer (6top)", draft-wang-6tisch-6top-sublayer-01 (work in progress), July 2014.

6.3. External Informative References

[IEEE802154e]
IEEE standard for Information Technology, "IEEE std. 802.15.4e, Part. 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) Amendment 1: MAC sublayer", April 2012.

[IEEE802154]
IEEE standard for Information Technology, "IEEE std. 802.15.4, Part. 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks", June 2011.

[OpenWSN] Watteyne, T., Vilajosana, X., Kerkez, B., Chraim, F., Weekly, K., Wang, Q., Glaser, S., and K. Pister, "OpenWSN: a Standards-Based Low-Power Wireless Development Environment", Transactions on Emerging Telecommunications Technologies , August 2012.

[morell04label]
Morell, A., Vilajosana, X., Lopez-Vicario, J., and T. Watteyne, "Label Switching over IEEE802.15.4e Networks. Transactions on Emerging Telecommunications Technologies", June 2013.

Authors’ Addresses

Qin Wang (editor)
Univ. of Sci. and Tech. Beijing
30 Xueyuan Road
Beijing, Hebei 100083
China

Phone: +86 (10) 6233 4781
Email: wangqin@ies.ustb.edu.cn

Xavier Vilajosana
Universitat Oberta de Catalunya
156 Rambla Poblenou
Barcelona, Catalonia 08018
Spain

Phone: +34 (646) 633 681
Email: xvilajosana@uoc.edu

Thomas Watteyne
Linear Technology
30695 Huntwood Avenue
Hayward, CA 94544
USA

Phone: +1 (510) 400-2978
Email: twatteyne@linear.com

Raghuram S Sudhaakar
Cisco Systems, Inc
Building 24
510 McCarthy Blvd
San Jose 95135
USA

Phone: +1 408 853 0844
Email: rsudhaak@cisco.com

Pouria Zand
University of Twente
Department of Computer Science
Zilverling Building
Enschede 7522 NB
The Netherlands

Phone: +31 619040718
Email: p.zand@utwente.nl

6TiSCH   Z. Chen
Internet-Draft   C. Wang
Intended status: Informational   InterDigital Communications, LLC
Expires: January 3, 2016   July 2, 2015

Use Cases and Requirements for using Track in 6TiSCH Networks
draft-wang-6tisch-track-use-cases-01

Abstract

This document further analyzes the 6TiSCH requirements related to Track through the use of examples and use cases. The goal of this document is to trigger discussions in 6TiSCH working group so that all relevant considerations are take into account when design Track reservation schemes in 6TiSCH.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

IEEE802.15.4e [IEEE802154e] was published in 2012 as an amendment to the Medium Access Control (MAC) protocol defined by the IEEE802.15.4-2011 [IEEE802154] standard. IEEE802.15.4e will be rolled into the next revision of IEEE802.15.4, scheduled to be published in 2015. The Timeslotted Channel Hopping (TSCH) mode of IEEE802.15.4e is the object of this document. The 6TiSCH working group is chartered to enable IPv6 over the TSCH mode of the IEEE802.15.4e standard.

The requirements for 6TiSCH are well documented [I-D.ietf-6tisch-tsch]. Initially, the WG will limit its scope to distributed routing over a static schedule. In this draft, we focus and expand discussions pertaining to Track. We propose requirements and use cases for different type of Track reservation schemes.

2. Terms used in this document

The draft uses terminologies defined in [I-D.ietf-6tisch-terminology]. The following are definition of terminologies used in this draft.

Centralized Track reservation: The reservation of a track done by the central controller of the network, e.g. PCE.

Distributed Track reservation: A reservation of a track done by one or more in-network entities (typically a connection endpoint).

Track: A determined sequence of cells along a multi-hop path. It is typically the result of a reservation. The node that initializes the process for establishing a Track is the owner of the track. The latter assigns a unique identifier to the Track, called TrackID

3. Use Cases: Industrial Networks

An industry network is a good use case for a 6TiSCH network. In an industry network as shown in Figure 1, many devices are LLN devices, e.g. sensors and actuators. There are many types of applications in an industry network, such as industry process control and automation applications, e.g. an automation assembly line, and industry monitor applications, e.g. a safety monitoring application.

3.1. Industry process control and automation applications

In an industry process control and automation application as shown in Figure 1, LLN Devices are actuator and sensors in an automation assemble line. An LLN Device, for example LLN Device S, MAY periodically send signalling packets to another actuator, e.g. LLN Device D. For example, LLN Device S locate at the step 1 of the automation assemble line, whenever it finishes a task, it will send singling packets to LLN Device D located at the step 2 of the automation assemble line to trigger the next action in the automation assembly line. The delay of these packets are extremely important for the performance of the automation assembly line. As mentioned in RFC 5673 [RFC5673], tens of milliseconds of latency is typical in fast control. In many of these systems, if a packet does not arrive within the specified interval, the system enters an emergency shutdown state, often with substantial financial repercussions. Therefore, Reserving a Track between LLN device S and LLN device D can guarantee the delay of these signalling packets.

Moreover, the reliability of these signalling packets are extremely important since a packet loss may result products with defects. Therefore, a backup path may be used when the primary path is broken. Reserving multiple Tracks between LLN device S and LLN device D can also improve the reliability of these packet due to less interference. By reserving a Track, battery powered LLN Devices are able to wake up and sleep based on its TSCH schedule to save energy. In these cases, the Tracks reserved are deterministic, unless the topology of the network changes.

3.2. Industrial monitoring applications

In an industrial monitoring application, sensors such as LLN M, monitor the status of each machine or plant and send data to the Control Controller as shown in Figure 1. An LLN Device, for example LLN Device M, MAY detect a critical event, and sends a signalling emergency message to the Central Controller in the network via multiple paths. After that the LLN Device may send monitoring data to the Central Controller. The singling packets that contains an emergency message SHOULD arrive at the Central Controller with minimum delay and highest reliability. Therefore, multiple Tacks may be reserved between these sensors and the Central Controller. Moreover, a bursty traffic that contains monitoring data MAY follow the critical message. These data packets also require low latency and high reliability, thus a high bandwidth Track SHOULD be quickly set-up between these LLN Devices and the Central Controller. Therefore, the Track reservation scheme has to react faster in a more dynamic way.

```
                   ---+-------- ............ ------------
                      |      External Network       |
                      |                          +-----+
                      |               +-----+    | NME |
                   +-----+            |  +-----+ |     |
                   |     | Central    |  | PCE | +-----+
                   |     | Controller +--|     |
                   +-----+               +-----+
                      |                   |
                      | Subnet Backbone   |
                +--------------------+------------------+
                |                    |                  |
             +-----+             +-----+             +-----+
             |     | Backbone    |     | Backbone    |     | Backbone
        o    |     | router      |     | router      |     | router
             +-----+           / +-----+             +-----+
        o    /                /           o --- o                  o   o
           o    o -- o --- o    o    o    /       \  o  o    o    o
       o     \  /     o      o    o      S -- o --- D    o     o
          o   M    o      o      o o     o  o   o    o    o     o
```

Figure 1: Use Case of an Industry Network

4. Handling Tracks in 6TiSCH Networks

4.1. General Behavior of Tracks

In this section, we discuss the behavior and the benefits of Tracks. As discussed in [I-D.ietf-6tisch-architecture], Track is first a multi-hop paths from the source LLN Device to the destination LLN Device. Second, some resources of LLN Devices on the path are reserved by configuring their TSCH schedule. Therefore, an LLN Device on the Track not only knows what cells it should use to receive packets from its previous hop, but also knows what cells it should use to send packets to its next hop. There are several benefits for using Track to forward a packet from the source LLN Device to the destination LLN Device.

First, Track forwarding as described in Section 10.1 in [I-D.ietf-6tisch-architecture] is a layer-2 forwarding scheme, which introduces less process delay and overhead than layer-3 forwarding scheme. Therefore, LLN Devices can save more energy and resource, which is critical for resource constrained devices.

Second, since channel resources, i.e. cells, have been reserved for communications between LLN devices of each hop on the Track, the packets traverse along the Track as a train passes each stations along the rail track. Therefore, the throughput and delay of the traffic on a Track is guaranteed and the jitter of the traffic is small. These are extremely important features for time-sensitive applications, which require packets arrives on time.

Third, by knowing the scheduled time slots of incoming cell and outgoing cell, LLN devices on a Track could save more energy by staying in sleep state during in-active slots. This is extreme important for LLN Devices that are battery powered.

Fourth, by allocating scheduled channel frequency, both inter-Track and intra-Track interference can be reduced. This will enhance the reliability of transmissions on a Track and reduce energy consumption of LLN Devices by decreasing the number of retransmissions.

4.2. Track Reservation

Cells along a Track have to be reserved before any packet transmissions. How to efficiently allocate resources along a Track becomes a challenging problem. Generally, there are both remote Track management and hop-by-hop Track management described in [I-D.ietf-6tisch-architecture] to solve the Track reservation issue.

4.2.1. Remote Track Management

In the remote Track management scheme in section 9.3 in [I-D.ietf-6tisch-architecture], a central controller of the network, e.g. Path Computation Element (PCE) in Figure 1, can allocate hard cells of LLN Devices on a Track remotely. The network may be globally optimized by the central controller of the network.

4.2.2. Hop-by-hop Track Management

In the hop-by-hop Track management scheme in section 9.4 in [I-D.ietf-6tisch-architecture], LLN Devices can negotiate and reserve Soft Cells in their TSCH Schedule by communicating with each other. By configuring the TSCH Schedule of LLN Devices on a route, a Track can be reserved to enhance the multi-hop communications between the source and the destination. The hop-by-hop Track management schemes may be more scalable and robust than the remote Track management scheme since it does not rely on the central controller of the network.

4.3. Relationship with Detnet

Deterministic Networking (DetNet) [I-D.finn-detnet-architecture] provides a capability to carry specified unicast or multicast data streams for real-time application with extremely low data loss rates and maximum latency. Three techniques are employed by DetNet to achieve theses QoS parameters, zero congestion loss, pinned-down paths and packet replication and deletion.

As mentioned by DetNet [I-D.finn-detnet-architecture], Track in 6TiSCH network is an instance of a deterministic path. The centralized and distributed path setup solutions in Detnet CAN be used as a reference in 6TiSCH Track reservation solution. However, Track in 6TiSCH is targeted to Low-power and Lossy Networks (LLNs), techniques in Detnet must be customized for Track management in 6TiSCH considering low power consumption, TSCH MAC and constrained devices with limited buffer and computation strength. For example, Detnet proposes seamless Redundancy, Replicating packets and sending them along at least two different paths. However, Replicating packets may dramatically increase the energy consumption of the network, which may be a concern for LLN networks. Therefore, Track management should be studied in 6TiSCH WG, and the solutions can influence the design of DetNet.

5. Requirement for Track reservation schemes

The track reservation schemes are required to support both deterministic traffics such as periodical transmissions for industry process control and automation applications and dynamic traffics such as bursty transmissions for industrial monitoring applications.

5.1. Centralized Track reservation

Need a protocol for LLN devices to report their topology and TSCH schedule information to the central controller as shown in Figure 1. The central controller need the topology information to obtain a path from the source to the destination and the network can be better optimized if the central controller is aware of the TSCH schedule of all or part of LLN Devices in the network.

Need a lightweight protocol for the central controller to configure hard cells of LLN Devices using 6top interface defined in [I-D.ietf-6tisch-6top-interface]. The central controller has to configure hard cells of LLN Devices on the track remotely and LLN Devices are usually constrained devices which may not support heavyweight protocol such as RFC 5440 [RFC5440]

5.2. Distributed Track reservation

Need a fast reaction protocol to reserve a Track. LLN Devices have limited information about the topology of the network and the TSCH schedule of other LLN Devices on the path. The protocol should quickly detect a Track reservation failure. Need an efficient negotiation protocol between LLN Devices multi-hop away from each other. LLN Devices on the path have to negotiate in order to reserve a Track, which may bring extra overhead to constrained devices.

6. Conclusions

A Track can provide low latency, guaranteed throughput and high reliable for end-to-end communications. There are many use cases that can show the benefit of using a Track, such as industry networks, home networks, structure networks, health networks and vehicular networks. Moreover, different Track reservation schemes, such as centralized and distributed schemes, need to be proposed to handle a large variety of requirements.

7. Security Considerations

This draft discussed the design considerations and operations of using Track in 6TiSCH networks. It does not introduce new security threats.

8. IANA Considerations

This specification does not require IANA action.

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

9.2. Informative References

[I-D.finn-detnet-architecture]
Finn, N., Thubert, P., and M. Teener, "Deterministic Networking Architecture", draft-finn-detnet-architecture-01 (work in progress), March 2015.

[I-D.ietf-6tisch-6top-interface]
Wang, Q., Vilajosana, X., and T. Watteyne, "6TiSCH Operation Sublayer (6top) Interface", draft-ietf-6tisch-6top-interface-02 (work in progress), October 2014.

[I-D.ietf-6tisch-architecture]
Thubert, P., Watteyne, T., Struik, R., and M. Richardson, "An Architecture for IPv6 over the TSCH mode of IEEE 802.15.4e", draft-ietf-6tisch-architecture-05 (work in progress), January 2015.

[I-D.ietf-6tisch-terminology]
Palattella, M., Thubert, P., Watteyne, T., and Q. Wang, "Terminology in IPv6 over the TSCH mode of IEEE 802.15.4e", draft-ietf-6tisch-terminology-03 (work in progress), January 2015.

[I-D.ietf-6tisch-tsch]
Watteyne, T., Palattella, M., and L. Grieco, "Using IEEE802.15.4e TSCH in an IoT context: Overview, Problem Statement and Goals", draft-ietf-6tisch-tsch-05 (work in progress), January 2015.

[RFC5440] Vasseur, JP. and JL. Le Roux, "Path Computation Element (PCE) Communication Protocol (PCEP)", RFC 5440, March 2009.

[RFC5673] Pister, K., Thubert, P., Dwars, S., and T. Phinney, "Industrial Routing Requirements in Low-Power and Lossy Networks", RFC 5673, October 2009.

9.3. External Informative References

[IEEE802154]
IEEE standard for Information Technology, "IEEE std. 802.15.4, Part. 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks", June 2011.

[IEEE802154e]
IEEE standard for Information Technology, "IEEE std. 802.15.4e, Part. 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) Amendment 1: MAC sublayer", April 2012.

Authors' Addresses

Zhuo Chen
InterDigital Communications, LLC
781 Third Ave
King of Prussia, PA 19406
USA

Phone: +1 610 878 5730
Email: Zhuo.Chen@InterDigital.com

Chonggang Wang
InterDigital Communications, LLC
781 Third Ave
King of Prussia, PA 19406
USA

Phone: +1 610 878 5831
Email: Chonggang.Wang@InterDigital.com