6TiSCH
Internet-Draft
Intended status: Informational
Expires: August 18, 2014

D. Dujovne, Ed.
Universidad Diego Portales
LA. Grieco
Politecnico di Bari
MR. Palattella
University of Luxembourg
N. Accettura
Politecnico di Bari
February 14, 2014

# 6TiSCH On-the-Fly Scheduling
# draft-dujovne-6tisch-on-the-fly-02

## Abstract

This document describes the environment, problem statement, and goals of the On-The-Fly (OTF) scheduling approach for the IEEE802.15.4e TSCH MAC protocol in the context of LLNs. 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. The soft cell and Bundle reservation with OTF is distributed: through the 6top interface, neighbor nodes negotiate the cell(s) to be (re)allocated/deleted, without intervention 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 soft cells to be reserved/deleted; it is up to 6top to select the specific soft cells within the TSCH schedule.

## Status of This Memo

This Internet-Draft will expire on August 18, 2014.



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 distributed protocol in which a node negotiates the number of soft cells scheduled with its neighbors, without the intervention of a centralized entity (e.g., a PCE). In particular, this document describes the OTF allocation policies and methods used for allocating single soft cell or group of them (i.e. bundle) between two neighbors. It also proposes some potential algorithms for estimating the required bandwidth. In this document,

internal interface used between OTF and the 6top sublayer ([I-D.wang-6tisch-6top]) is defined, with a particular focus on the functionalities. Example functionalities are collecting and providing statistic information, or allocating/deleting cells and bundles. To be extensible and applicable to different scenarios, this draft is a generic framework. The exact algorithm and set of statistics used for estimating the requested bandwidth is 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 [I-D.ietf-6tisch-tsch].

## 2. Allocation policy

OTF is a distributed scheduling protocol which sends scheduling requests to the 6top sublayer. OTF thereby dynamically increases/ decreases the bandwidth allocated between neighbor nodes. 6top takes care of negociating the soft cells between neighbors. Because OTF delegates the selection of the specific cells to 6top, OTF only supports soft cell reservation. Therefore, the term "cell" is synonymous to "soft cell" in this document.

OTF is prone to schedule collision. Nodes might not be aware of the cells allocated by other pairs of neighbors nodes. A collision occurs when the same cell is allocated by different pairs in the same interference space. The 6top sublayer cannot differentiate collisions from other sources of packet loss. 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.

We call "allocation policy" the approach used by OTF for increasing or decreasing the bandwidth allocated between two nodes in order to satisfy the traffic requirements. These requirements can be expressed in terms of throughput, latency or other constraints.

OTF supports 3 different types of allocation policies: Post-allocation, Pre-allocation and Hybrid allocation.

Post-allocation policy is based on a "recovery" approach following the increased/decreased need of bandwidth. Upon reception of a bandwidth request, OTF sends soft cell allocation requests to the 6top sublayer. OTF has to estimate the number of cells to be allocated per each neighbor. OTF keeps track of such information. If afterwards, it is possible to free some cells (due for instance to the reduction of traffic exchanged between two neighbors), OTF asks 6top to de-allocate a given number of cells. Once the cells have

been deleted and 6top has notified the event to OTF, the latter can update its internal cell allocation tables.

Pre-allocation policy is based on a "provision" approach. This implies that the reservation of groups of equivalents cells (i.e., bundles), to a given couple of nodes, is done in advance. When OTF sends a bundle allocation requests to 6top, it has to indicate the desired size of the bundle and the TrackID. These are the only features that can be settled by OTF. 6top selects the soft cells belonging to the bundle. Based on the network traffic condition (e.g. queue utilization), a number of cells within the bundle is used for communication. In any case, allocated cells within a bundle are consecutive, starting from the first cell in the block. The cells which are not currently used, are still reserved for that pair of nodes, for possible future use.

OTF keeps track of the scheduled bundles, the bundle size, and the estimated number of allocated cells within a bundle. Based on this information, upon reception of a bandwidth request, OTF may ask 6top to increase or decrease the bundle size.

The post-allocation policy compared to the pre-allocation reduces the energy consumption, by allocating the exact number of soft cells when they are needed, but at the expense of increased cell allocation latency. In fact, for each requested soft cell, the 6top layer has to negotiate the request with the 6top layer of the neighbor node. Such negotiation takes some time and induces communication overhead.

The pre-allocation policy compared to the post-allocation reduces the cell allocation latency. The soft cells within the bundle are over-provisioned, and a priori scheduled. When needed, the 6top sublayer of the node can allocate them, without going through any negotiation phase with the 6top layer of the neighbor node. Thus, the pre-allocation approach provides a low-delay response after a surge in bandwidth usage. In fact, soft cells within a bundle are already scheduled and become immediately available, upon bandwidth request, without the need of a negotiation phase. The use of bundles does force the receiver module of the node to be active during the whole length of the Bundle, thus implying increased power consumption.

The hybrid allocation policy is a mix of the pre- and post-allocation policy. It tries to take advantage of both the "provision" and "recovery" approaches. Some bundles can be reserved in advance for a pair of nodes. After receiving a bandwidth request, OTF can decide if asking to 6top for (new) soft cell allocation (as per post-allocation approach, implying a negotiation phase among the neighbor nodes), or for bundle allocation/resizing (as per pre-allocation approach). In fact, tt is possible that more bandwidth is

needed, but there are still some cells within the bundle that have not been allocated yet, and that they can be used for fulfilling the request. If all the cells within the bundle have already been allocated, OTF has to send a bundle size increase request to 6top (that still translates in soft cells allocation request).

## 3. Allocation methods

Beyond the allocation policies that describe the approach used by OTF for fulfilling the node bandwidth requests, the OTF framework also includes Allocation Methods that specify how OTF actually asks the 6top sublayer to satisfy the aforementioned requests. In other words, the allocation methods represent the mechanisms that are used by the allocation policies to pre- or post- allocate extra bandwidth.

In detail, OTF includes two distinct allocation methods: soft cell and bundle allocation methods. Each Allocation Policy can use either one or both allocation methods. As specified in [I-D.wang-6tisch-6top], 6top provides a set of commands that allows to schedule/allocate/delete soft cells. The same set of commands can be used for reserving bundles.

With the soft cell allocation method, OTF asks 6top to reserve a single soft cell, for communicating with a specific neighbour node, on a given track. The 6top layer allocates and maintains such cell. It has to be noticed that if a bundle (i.e., a group of cells) was already reserved between the same couple of nodes, on the same track, then, such soft cell allocation request translates into a bundle resize request. In fact, the new allocated cell will increase the size of the already exsisting bundle. In a similar way, when OTF realizes that there is a reduction of traffic exchanged between the two neighbors, it may asks 6top to delete a soft cell, in other words, to decrease the bundle size. Instead, if no bundle with the same TrackID already existed, then the 6top soft cell create command will generate a new bundle of size 1, having the specified TrackID.

With the bundle allocation method, OTF sends bundle allocation requests to 6top sublayer, specifying the bundle size (i.e., number of soft cells forming the bundle itself) and the TrackID of the track to which the cells belong. By using the soft cells commands 6top generates the bundle. In fact, the request of scheduling N soft cell is equivalent to asking for a bundle of size N. The cells within the bundle will be allocated by 6top afterwards, according to the nodes bandwith need.

## 4. Input parameters: statistics and instant values

Short summary of a potential set of statistics and instant values that could be used as input parameters. Direct interaction with 6top.

List of parameters available from 6top: mainly statistics related to queues

Method to configure 6top to provide historical values for each requested parameter

Method to ask 6top for instant values for each requested parameter

Method for asking for a list of parameters from 6top and thus, for checking if a parameter is available or not

## 5. bundle usage management in OTF: TODO

Methods that trigger the request of increasing/decreasing the bundle, and thus, adding/deleting cells

### 5.1. Cell Reservation/Deletion

The commands to reserve/delete soft cells. Direct interaction with 6top

### 5.2. bundle Size Increase/Decrease

The commands to increase/decrease the bundle size. Direct interaction with 6top

## 6. Schedule storage on OTF: TODO

The description and access to the schedule storage on OTF

The commands to retrieve bundle usage values and statistics from OTF (based on previous values obtained by 6top?)

## 7. Bandwidth Estimation Algorithms

OTF supports different bandwidth estimation algorithms that can be used by a node in a 6TiSCH network for checking the current traffic condition and thus the actual bandwidth usage. By doing so, it is possible to adapt (increase or increase) the number of scheduled cells/bundles for a given pair of neighbors (e.g., parent node and its child), according to their needs. OTF supports several bandwidth estimation algorithms numbered from 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, it is possible, to set the specific algorithm to be used, and get information about which algorithm is actualy implemented.

The default bandwidth estimation algorithm, running over a parent node, is articulated in the following steps:

Step 1: Collect the bandwidth requests from child nodes (incoming traffic).

Step 2: Collect the node bandwidth requirement from the application (self/local traffic).

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

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

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

Step 6: Loop to Step 1.

Based on the allocation policy that is used, at Step 4 new soft cells will be scheduled, using the cell allocation method. If there are free cells in the bundle that can satisfied the current bandwidth request, such cells will be allocated. In the case of pre-allocation policy, it may happen that the number of allocated cells within the bundle is equal to the bundle size. That is, all the cells within the bundle are currently used. In this case, the node asks 6top for increasing the bundle size by using the bundle allocation method. When the hybrid allocation policy has been selected two options are available: (i) the bundle size is increased, and a number of cells larger than those currently needed, can be scheduled - according to post-allocation approach, or (ii) a number of soft cells equal exactly to those currently needed are scheduled, as per post-allocation approach.

## 8. Acknowledgements

Special thanks to Prof. Kris Pister for his valuable contribution in designing the default Bandwidth Estimation Algorithm, and to 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).

## 9. References

### 9.1. 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-01 (work in progress), February 2014.

[I-D.ietf-6tisch-architecture]
Thubert, P., Watteyne, T., and R. Assimiti, "An Architecture for IPv6 over the TSCH mode of IEEE 802.15.4e", draft-ietf-6tisch-architecture-01 (work in progress), February 2014.

[I-D.ietf-6tisch-tsch]
Watteyne, T., Palattella, M., and L. Grieco, "Using IEEE802.15.4e TSCH in an LLN context: Overview, Problem Statement and Goals", draft-ietf-6tisch-tsch-00 (work in progress), November 2013.

[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.

### 9.2. 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.

[TASA-PIMRC]
Palattella, MR., Accettura, N., Dohler, M., Grieco, LA., and G. Boggia, "Traffic Aware Scheduling Algorithm for Multi-Hop IEEE 802.15.4e Networks", IEEE PIMRC 2012, Sept. 2012, < http://www.cttc.es/resources/doc/120531-submitted-tasa-25511.pdf>.

[DeTAS] Accettura, N., Palattella, MR., Boggia, G., Grieco, LA., and M. Dohler, "Decentralized Traffic Aware Scheduling for Multi-hop Low Power Lossy Networks in the Internet of Things", IEEE WoWMoM 2013, June 2013.

## 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
Politecnico di Bari
Electrical and Electronics Department
Via Orabona 4
Bari 70125
Italy

Phone: +39 080 5963301
Email: n.accettura@poliba.it

6TiSCH P. Thubert, Ed.Internet
-Draft CiscoIntended status:
Standards Track T. WatteyneExpires: August 16, 2014
Linear Technology
RA. Assimiti
Centero

February 14, 2014 An Architecture for IPv6 over the TSCH mode of IEEE 802.15.4e draft-ietf-6tisch-architecture-01

Abstract This document presents an architecture for an IPv6 Multi-Link subnet that is composed of a high speed powered backbone and a number of IEEE802.15.4e TSCH wireless networks attached and synchronized by Backbone Routers. The TSCH schedule can be static or dynamic. 6TiSCH defines mechanisms to establish and maintain the routing and scheduling operations in a centralized, distributed, or mixed fashion. Backbone Routers perform proxy Neighbor Discovery operations over the backbone on behalf of the wireless devices, so they can share a same subnet and appear to be connected to the same backbone as classical devices

## 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

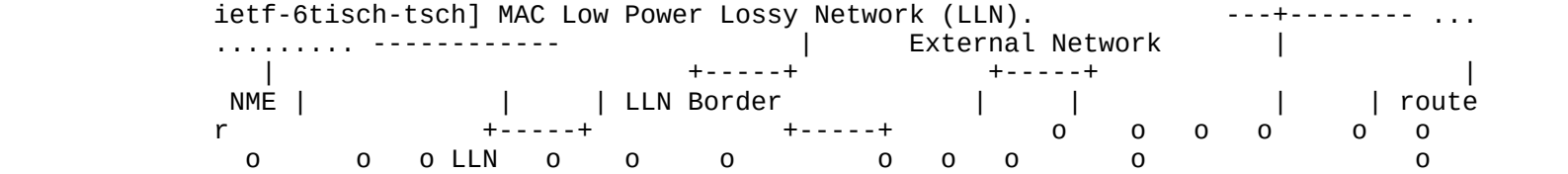
This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79. Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet- Drafts is at http://datatracker.ietf.org/drafts/current/. Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress." This Internet-Draft will expire on August 16, 2014.

 Table of Contents

 The emergence of radio technology enabled a large variety of new types of devices to be interconnected, at a very low marginal cost compared to wire, at any range from Near Field to interplanetary distances, and in circumstances where wiring would be less than practical, for instance 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 and quite sensitive to latency. Such traffic is not limited to voice and video, but also includes command and control operations such as found in industrial automation or in-vehicle sensors and actuators. At IEEE802.1, the "Audio/Video Task Group", was renamed TSN for Time Sensitive Networking to address Deterministic Ethernet. The IEEE802.15.4 Medium access Control (MAC) has evolved with IEEE802.15.4e that provides in particular the Timeslotted Channel Hopping (TSCH) mode for industrial-type applications. Though at a different time scale, both 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 timeslot 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 WiFi emitters). This document presents an architecture for an IPv6 Multi-Link subnet that is composed of a high speed powered backbone and a number of IEEE802.15.4e TSCH wireless networks attached and synchronized by backbone routers. Route Computation may be achieved in a centralized fashion by a Path Computation Element (PCE), in a distributed fashion using the Routing Protocol for Low Power and Lossy Networks (RPL), orThubert, Watteyne & AssiExpires August 16, 2014 [Page 3]

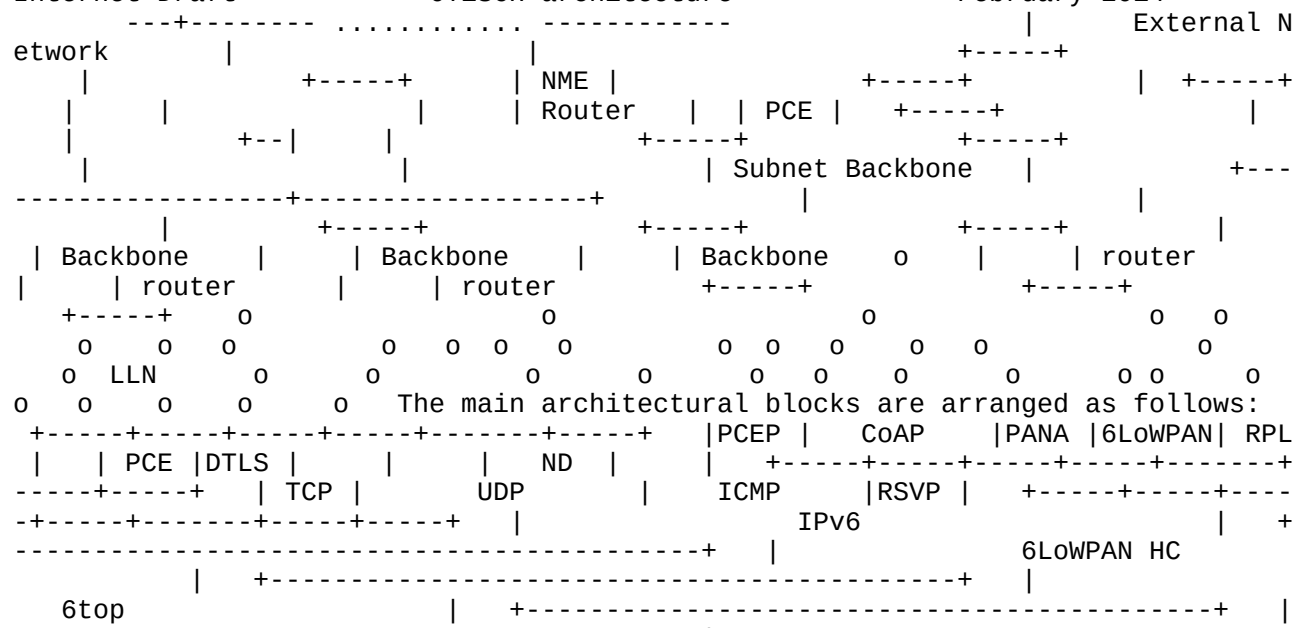
 in a mixed mode. The Backbone Routers perform proxy IPv6 neighbor Discovery (ND) operations over the backbone on behalf of the wireless devices, so they can share a same IPv6 subnet and appear to be connected to the same backbone as classical devices. Timeslots and other device resources are managed by an abstract Network Management Entity (NME) that may cooperate with the PCE in order to minimize the interaction with and the load on the constrained device.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] 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 .roll-rpl-industrial-applicability], [RFC5191] and [RFC4080]. 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 The architecture derives from existing industrial standards for Process Control by its focus on Deterministic Networking, in particular with the use of the IEEE802.15.4e TSCH MAC [IEEE802154e] and the 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. Deterministic Networking applies in particular to open and closed control loops, as well as supervisory control flows and management. An incremental set of industrial requirements are 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 asThubert, Watteyne & AssiExpires August 16, 2014 [Page 4]

 corro sion and events. RPL also enables mobile use cases such as mobile workers and cranes. A Backbone Router is included in order to scale the factory plant s ubnet to address large deployments, with proxy ND and time synchronization ove r 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 h ops, in-vehicle command and control that can be as demanding as industrial a pplications, commercial automation and asset Tracking with mobile scenarios, h ome automation and domotics which become more reliable and thus provide a bett er user experience, and resource management (energy, water, etc.).4. Overview and Scope The scope of the present work is a subnet that, in its basic conf iguration, is made of a IEEE802.15.4e Timeslotted Channel Hopping (TSCH) [I-D. ietf-6tisch-tsch] MAC Low Power Lossy Network (LLN).




The LLN devices communicate over IPv6 [RFC2460] using the 6LoWPAN Header Compr ession (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 collect ion of Links with different capabilities for unicast or multicast services. A n IPv6 subnet spans over multiple links, effectively forming a Multi-Link subn et. Within that subnet, neighbor Devices are discovered with 6LoWPAN neighbor Discovery (6LoWPAN ND) [RFC6775]. RPL [RFC6550] enables routing within the LLN, typically within the Multi-Link subnet in the so called Route Over fashi on. RPL forms Destination Oriented Directed Acyclic Graphs (DODAGs) within In stances of the protocol, each Instance being associated with an Objective Func tion (OF) to form aThubert, Watteyne & AssiExpires August 16, 2014 [Page 5]

 routing topology. A particular LLN device, the LLN Border Router (LBR), acts as RPL root, 6LoWPAN HC terminator, and LLN Border Router (LBR) to the outside. The LBR is usually powered. More on RPL Instances can be found in RPL [RFC6550], sections "3.1.2. RPL Identifiers" and "3.1.3. Instances, DODAGs, and DODAG Versions". 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]. The backbone can also support Efficiency-aware IPv6 neighbor Discovery Optimizations [I-D.chakrabarti-nordmark-6man- efficient-nd] in mixed mode as described in [I-D.thubert-6lowpan- backbone-router]. Security is often handled at layer 2 and Layer 4. Authentication during the join process can be handled by the Protocol for Carrying Authentication for Network access (PANA) [RFC5191]. The LLN devices are time-synchronized at the MAC level. The LBR that serves as time source is a RPL parent in a particular RPL instance that serves for time synchronization; this way, the time synchronization starts at the RPL root and follows the RPL DODAGs with no timing loop. In the extended configuration, the functionality of the LBR is enhanced to that of Backbone Router (BBR). A BBR is an LBR, but also an Energy Aware Default Router (NEAR) as defined in [I-D.chakrabarti-nordmark-6man-efficient-nd]. The BBR performs ND proxy operations between the registered devices and the classical ND devices that are located over the backbone. 6TiSCH BBRs synchronize with one another over the backbone, so as to ensure that the multiple LLNs that form the IPv6 subnet stay tightly synchronized. 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.Thubert, 

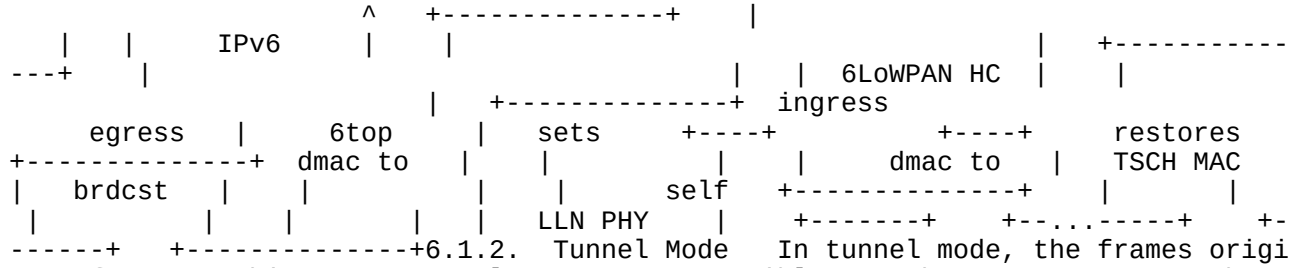



RPL is the routing protocol of choice for LLNs. (TBD RPL) whether there is a need to define a 6TiSCH OF. (tbd NME) COMAN is working on network Management for LLN. They are considering the Open Mobile Alliance (OMA) Lightweight M2M (LWM2M) Object system. This standard includes DTLS, CoAP (core plus Block and Observe patterns), SenML and CoAP Resource Directory. (tbd PCE) need to work with PCE WG to define flows to PCE, and define how to accommodate PCE routes and reservation. Will probably look a lot like GMPLS.

 (tbd PANA) There is a debate whether PANA (layer 3) or IEEE802.1x (layer 2) should be used in the join process. There is also a debate whether the node should be able to send any unprotected packet on the medium. Regardless, the security model must ensure that, prior to a join process, packets from a untrusted device must be controlled in volume and in reachability. (tbd Backbone Router) need to work with 6MAN to define ND proxy. Also need BBR sync sync between deterministic Ethernet and 6TiSCH LLNs. IEEE802.1TSN: external, maintain consistency. See also AVnu. IEEE802.15.4: external, (tbd need updates?). ISA100.20 Common Network Management: external, maintain consistency. 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.5. 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. [I-D.roll-rpl-industrial-applicability] section 2.1.3. and next discusses appplication-layer paradigms, such as Source-sink (SS) that is a Multipeer to Multipeer (MP2MP) model that is 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 IEEE802.15.4e TSCH wireless networks with [ISA100.11a] and [HART].Thubert, Watteyne & Assi 

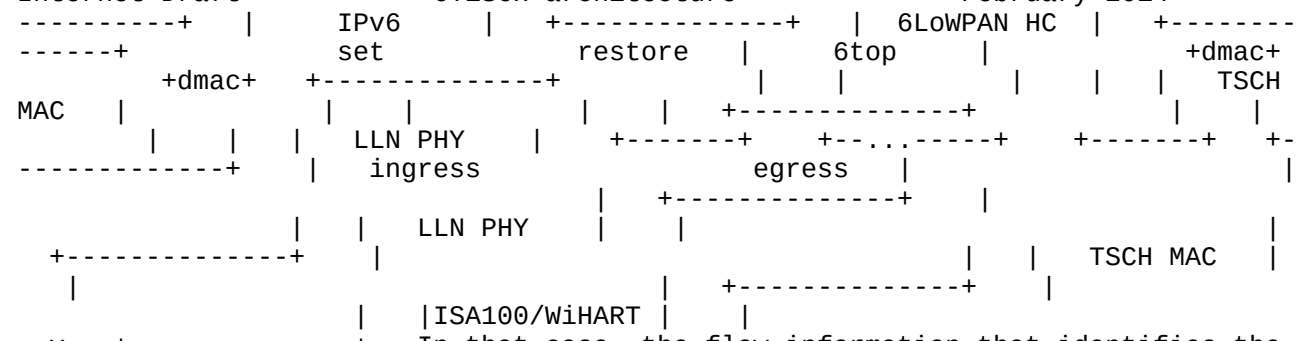
 This specification focuses on Communication Paradigms and Interaction Models for pa cket forwarding and TSCH resources (cells) management. L ink-layer and Network -layer Packet forwarding interactions are discussed in Section 6, whereas Link -layer (one-hop), Network-layer (multithop along a track), and Application-lay er (remote control) management mechanisms for the TSCH schedule are discussed in Section 8.6. Forwarding Models 6TiSCH supports three different forwardin g model, G-MPLS Track Forwarding (TF), 6LoWPAN Fragment Forwarding (FF) and IP v6 Forwarding (6F).6.1. Track Forwarding Track Forwarding is the simplest a nd fastest. A set of input cells are uniquely bound to a set of output cells, representing a forwarding state that can be used regardless of the upper laye r protocol. This model can effectively be seen as a G-MPLS operation in tha t the information used to switch is not an explicit label, but rather related to other properties of the way the packet was received, a particular cell in t he 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, whet her this is an IPv6 packet, a 6LoWPAN fragment, or a frame from an alternate protocol such as WirelessHART of ISA100.11a. A Track is defined end-to-end as a succession of timeslots. A timeslot belongs to at most one Track. For a g iven iteration of a slotframe, the timeslot is associated uniquely with a cell , which indicates the channel at which the timeslot operates for that iterat ion. A data frame that is forwarded along a Track has a destination MAC addr ess set to broadcast or a multicast address depending on MAC support. This wa y, 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 IEE E802.15.4e, this means effectively broadcast, so that along the Track the shor t address for the destination is set to 0xFFFF. Conversely, a frame that is received along a Track with a destination MAC address set to this node is extr acted from the Track stream and delivered to the upper layer. A frame with an unrecognized MAC address is ignored at the MAC layer and thus is not received at the 6top sublayer. There are 2 modes for a Track, transport mode and tun nel mode.6.1.1. Transport ModeThubert, Watteyne & AssiExpires August 16, 2014

 In transport mode, the PDU is associated flow information that refers uniquely to the Track, so the 6top sublayer can place the frame in the appropriate timeslot without ambiguity. In the case of IPv6 traffic, 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.



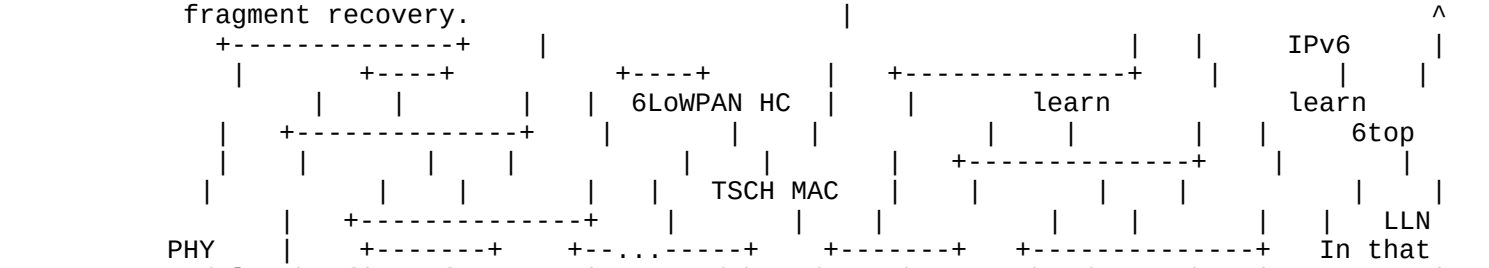

6.1.2. Tunnel Mode

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

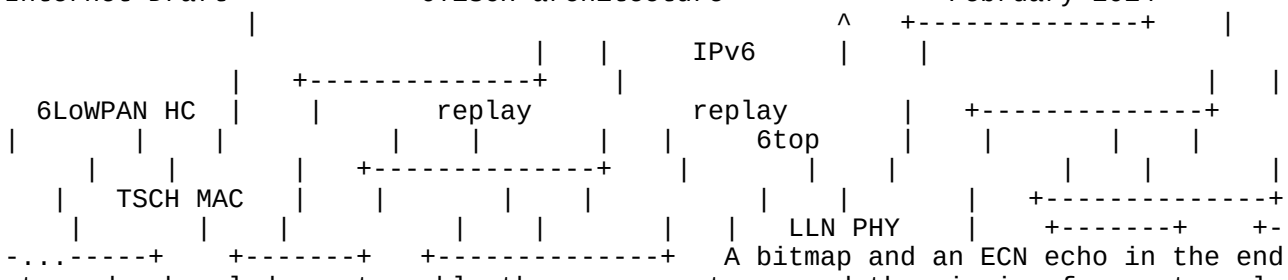



v In that case, the flow information that identifies the Track is uniquely derived from the information at the receiving end, for instance the incoming timeslots, or an ISA100.11a ContractId. At the ingress 6TiSCH router, the packet destination is recognized as self but the flow information indicates that the frame must be tunnelled over a particular 6top Track so the packet is not punted to upper layer. Instead, it is passed to the 6top sublayer for switching. The 6top sublayer in the ingress router overrides the destination MAC to broadcast and forwards. At the egress 6top 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.6.1.3. 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 revertsThubert, Watteyne & AssiExpires August 16, 2014 [Page 11]

 that operation to the MAC address obtained from the tunnel metadata.6.2. 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 base on an Explicit Congestion Notification, as well as end- to-end individual fragment recovery.




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.



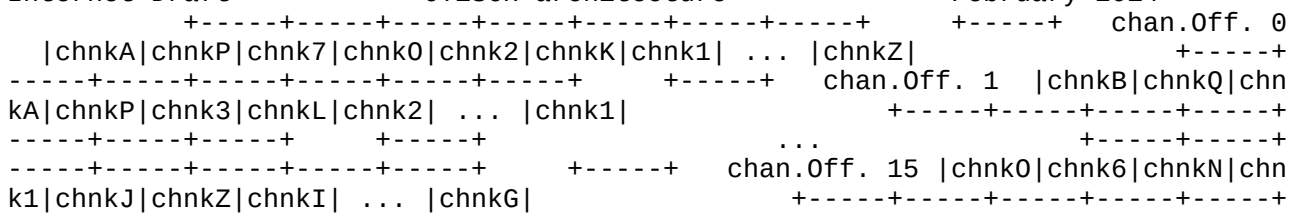

A bitmap and an ECN echo in the end-to-end acknowledgement enable the source to resend the missing fragments selectively. The first fragment may be resent to carve a new path in case of a path failure. The ECN echo set indicates that the number of outstanding fragments should be reduced.6.3. IPv6 Forwarding As the packets are routed at layer 3, traditional QoS and RED operations are expected to prioritize flows with differentiated services. A new class of service for Deterministic Forwarding is being defined to that effect in [I-D.svshah-tsvwg-lln-diffserv- recommendations].

 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 Section 8.2. The 6top data model and management interfaces are further discussed in Section 8.3. If the scheduling entity explicitly specifies the slotOffset/ channelOffset of the cells to be added/deleted, those cells are marked as "hard". 6top cannot move hard cells in the TSCH schedule. Hard cells are for example used by a central PCE. 6top contains a monitoring process which monitors the performance of cells, and can move a cell in the TSCH schedule when it performs bad. 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 motes 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 [RFC 6550] Objective Function (OF), such as the RPL Objective Function Zero (OF0) [RFC 6552] 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. 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 ASN when the last packet has been received from that neighbor, a set of cell quality metrics (RSSI, 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 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 EBs. By augmenting the EB periodicity, it is possible to change the network dynamics so as to improve the support of mobile devices.Thubert, Watteyne & AssiExpires August 16, 2014 [Page 14]

 Some RPL control messages, such as the DODAG Information Object (DIO) are broadcast to all neighbor nodes. The broadcast channel requirement is addressed by 6top by configuring TSCH to provide such a channel, as opposed to, for instance, carrying DIO messages in Enhance Beacons. In the TSCH schedule, each cell has the LinkType attribute. Setting the LinkType to ADVERTISING indicates that the cell MAY be used to send an Enhanced Beacon. When a node forms its Enhanced Beacon, the cell, with LinkType=ADVERTISING, SHOULD be included in the FrameAndLinkIE, and its LinkOption field SHOULD be set to the combination of "Receive" and "Timekeeping". The receiver of the Enhanced Beacon MAY be listening at the cell to get the Enhanced Beacon ([IEEE802154e]). 6top takes this way to establish broadcast channel, which not only allows TSCH to broadcast Enhanced Beacons, but also allows an upper layer like RPL. To support DIO and DAO broadcasts, 6top uses the payload of a Data Packet to carry the DIO or DAO. The message is inserted into the queue associated with the cells which LinkType is set to ADVERTISING. Then, taking advantage of the broadcast cell feature established with FrameAndLinkIE (as described above), the data packet with DIO or DAO in the payload can be received by neighbors, which enforces the maintenance of DODAG. A LinkOption combining "Receive" and "Timekeeping" bits indicates to the receivers of the Enhanced Beacon that the cell MUST be used as a broadcast cell. The frequency of sending Enhanced Beacons or other broadcast messages by the upper layer is determined by the timers associated with the messages. For example, the transmission of Enhance Beacons is triggered by a timer in 6top; transmission of a DIO message is triggered by the trickle timer of RPL.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 acknowledgement-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.Thubert, Watteyne & AssiExpires August 16, 2014

 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 Sync hronization Global Instance (TSGI). The TSGI can be operated in either of the 3 modes that are detailed in RPL [RFC6550] section "3.1.3. Instances, DODA Gs, 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 Po sitioning System (GPS). In the absence of a common time source, the TSGI shoul d 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 a ct as sinks for the LLN. A node that has not joined the TSGI advertises a MA C level Join Priority of 0xFF to notify its neighbors that is is not capable o f 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 [RFC6550], section "3.5.1. Rank Comparison". A ro ot is configured or obtains by some external means the knowledge of the RPLIns tanceID for the TSGI. The root advertises its DagRank in the TSGI, that MUST b e less than 0xFF, as its Join Priority (JP) in its IEEE802.15.4e Extended Beac ons (EB). We'll note that the JP is now specified between 0 and 0x3F leaving 2 bit sin the octet unused in the IEEE802.15.4e specification. After consultat ion 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 Priorit y 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, th en the node should join the TSGI, obtain a Rank in that Instance and start adv ertising its own DagRank in the TSGI as its Join Priority in its EBs.7.4. Slo tframes and Priorities 6top uses priority queues to manage concurrent data flo ws of different priorities. When a packet is received from an higher layer for transmission, the I-MUX module of 6top inserts that packet in the outgoing queue which matches the packet best (DSCP can therefore be used). At each sch eduled transmit slot, the MUX module looks for the frame in all the outgoing q ueues that best matches the cells. If a frame is found, it is given to TSCH f or transmission.7.5. Packet Marking and HandlingThubert, Watteyne & AssiExpires 

 reservation Deterministic flow allocation (hard reservation of timeslots) eg centralized RSVP? metrics? Hop-by-hop interaction with 6top. Lazy reservation (use shared slots to transport extra burst and then dynamically (de)allocate) Classical QoS (dynamic based on observation)7.6. Distributing the reservation of timeslots 6TiSCH expects a high degree of scalability together with a distributed routing functionality based on the RPL routing protocol. 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 an abstract channel usage/distribution (CUD) matrix of timeslots within its interference domain. 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 CUD 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 chunks is the basic unit of ownership that is used in that process.Thubert, Watteyne & AssiExpires August 16, 2014 [Page 17]




M 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 slotframe 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. 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. Minimal 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.Thubert, Watteyne & AssiExpires August 16, 2014 [Page 18]

 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 scheduled 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].8.2. Neighbor-to-neighbor Scheduling In the simplest instantiation of a 6TiSCH network described in Section 8.1, 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 emissions will be destroyed. 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 relocation 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.8.3. Remote Monitoring and Schedule Management

 The 6 top interface document [I-D.wang-6tisch-6top-interface] specifies the generic data model that can be used to monitor and manage resources at the 6top sublay er. Abstract methods are suggested for use by a management entity in the devi ce. The data model also enables remote control operations on the 6top sublaye r. Being able to interact with the 6top sublayer of a node multiple hops awa y can be used for monitoring, scheduling, or a combination of both. The archi tecture supports variations on the deployment model, and focuses on the flows rather than whether there is a proxy or a translational operation on the way.

[I-D.sudhaakar-6tisch-coap] defines an mapping of 6top's set of commands des cribed in [I-D.wang-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.sudhaakar-6tisch-coap] defines a basic set CoAP resou rces and associated RESTful access methods (GET/PUT/POST/DELETE). The payload (body) of the CoAP messages is encoded using the CBOR format. The draft als o 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 t he TSCH schedule (e.g. the head of a local cluster), or a external device mon itoring the overall state of the network (e.g. NME). The architecture allows f or different types of interactions between this CoAP client and a node in the network:8.4. Hop-by-hop Scheduling A node can reserve a track to a destinatio n 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 subla yer of each node on the track to monitor these soft cells and trigger reloca tion when needed. This hop-by-hop reservation mechanism is similar to [RFC2119 ] and [RFC5974]. The protocol for a node to trigger hop-by-hop scheduling i s not yet defined.9. Centralized vs. Distributed Routing 6TiSCH supports a m ixed model of centralized routes and distributed routes. Centralized routes c an for example computed by a entity such as a PCE. Distributed routes are comp uted by the RPL routing protocol.Thubert, Watteyne & AssiExpires August 16, 20 14 [Page 20]

 Both may inject routes in the Routing Tables of the 6TiSCH routers. In either case, each route is associated with a topology that is indexed by an RPLInstanceID, as defined in RPL [RFC6550]. RPL and PCE rely on shared sources to define Global and Local RPLInstanceIDs. It is possible for centralized and distributed routing to share a same topology. In this case, centralized routes have precedence over distributed routes in case of conflict. Inside the 6TiSCH domain, the flow label is used to indicate the topology that must be used for routing. The associated Routing Tables are discussed in [I-D.thubert-roll-flow-label].10. IANA Considerations This specification does not require IANA action.11. Security Considerations This specification is not found to introduce new security threat.12. Acknowledgements13. References13.1. Normative References [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997. [RFC2460] Deering, S.E. and R.M. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998. [RFC3444] Pras, A. and J. Schoenwaelder, "On the Difference between Information Models and Data Models", RFC 3444, January 2003. [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. [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.

[RFC5191] Forsberg, D., Ohba, Y., Patil, B., Tschofenig, H. and A. Yegin, "Protocol for Carrying Authentication for Network Access (PANA)", RFC 5191, May 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. [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. [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 2012.13.2. Informative References [I-D.chakrabarti-nordmark-6man-efficient-nd] Chakrabarti, S., Nordmark, E., Thubert, P. and M. Wasserman, "Wired and Wireless IPv6 Neighbor Discovery Optimizations", Internet-Draft draft-chakrabarti-nordmark-6man-efficient-nd-04, October 2013. [I-D.ietf-6tisch-minimal] Vilajosana, X. and K. Pister, "Minimal 6TiSCH Configuration", Internet-Draft draft-ietf-6tisch-minimal-00, November 2013. [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", Internet-Draft draft-ietf-6tisch-terminology-00, November 2013. [I-D.ietf-6tisch-tsch] Watteyne, T., Palattella, M. and L. Grieco, "Using IEEE802.15.4e TSCH in an LLN context: Overview, Problem Statement and Goals", Internet-Draft draft-ietf-6tisch-tsch-00, November 2013.

Internet-Draft 6TiSCH-architecture February 2014 [I-D.ietf-roll-rpl-industrial-applicability] Phinney, T., Thubert, P. and R. Assimiti, "RPL applicability in industrial networks", Internet-Draft draft-ietf-roll-rpl-industrial-applicability-02, October 2013. [I-D.ohba-6tisch-security] Chasko, S., Das, S., Lopez, R., Ohba, Y., Thubert, P. and A. Yegin, "Security Framework and Key Management Protocol Requirements for 6TiSCH", Internet-Draft draft-ohba-6tisch-security-00, October 2013. [I-D.sudhaakar-6tisch-coap] Sudhaakar, R. and P. Zand, "6TiSCH Data Model for CoAP", Internet-Draft draft-sudhaakar-6tisch-coap-00, October 2013. [I-D.svshah-tsvwg-lln-diffserv-recommendations] Shah, S. and P. Thubert, "Differentiated Service Class Recommendations for LLN Traffic", Internet-Draft draft-svshah-tsvwg-lln-diffserv-recommendations-01, August 2013. [I-D.thubert-6lowpan-backbone-router] Thubert, P., "6LoWPAN Backbone Router", Internet-Draft draft-thubert-6lowpan-backbone-router-03, February 2013. [I-D.thubert-roll-flow-label] Thubert, P., "Use of the IPv6 Flow Label within an LLN", Internet-Draft draft-thubert-roll-flow-label-02, November 2012. [I-D.thubert-roll-forwarding-frags] Thubert, P. and J. Hui, "LLN Fragment Forwarding and Recovery", Internet-Draft draft-thubert-roll-forwarding-frags-02, September 2013. [I-D.wang-6tisch-6top-interface] Wang, Q., Vilajosana, X. and T. Watteyne, "6TiSCH Operation Sublayer (6top) Interface", Internet-Draft draft-wang-6tisch-6top-interface-01, February 2014. [I-D.wang-6tisch-6top-sublayer] Wang, Q., Vilajosana, X. and T. Watteyne, "6TiSCH Operation Sublayer (6top)", Internet-Draft draft-wang-6tisch-6top-00, October 2013.13.3. External Informative References [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]Thubert, Watteyne & AssiExpires August 16, 2014 [Page 23]

IEEE Standards Association, "IEEE 802.1 Time-Sensitive Networks Task Group", March 2013, <http://www.ieee802.org/1/pages/avbridges.html>. [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. [ISA100.11a] ISA, "ISA100, Wireless Systems for Automation", May 2008, <http://www.isa.org/Community/SP100WirelessSystemsforAutomation>.
Authors' Addresses 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 Thomas Watteyne Linear Technology, Dust Networks Product Group 30695 Huntwood Avenue Hayward, CA 94544 USA Phone: +1 (510) 400-2978 Email: twatteyne@linear.com Robert Assimiti Centero 961 Indian Hills Parkway Marietta, GA 30068 USA Phone: +1 404 461 9614 Email: robert.assimiti@centerotech.com

6TiSCH
Internet-Draft
Intended status: Informational
Expires: August 17, 2014

MR. Palattella, Ed.
SnT/Univ. of Luxembourg
P. Thubert
cisco
T. Watteyne
Linear Technology / Dust Networks
Q. Wang
Univ. of Sci. and Tech. Beijing
February 13, 2014

# Terminology in IPv6 over the TSCH mode of IEEE 802.15.4e
# draft-ietf-6tisch-terminology-01

## Abstract

6TiSCH proposes an architecture for an IPv6 multilink subnet that is composed of a high speed powered backbone and a number of IEEE802.15.4e TSCH wireless networks attached and synchronized by backbone routers. This document extends existing terminology documents available for Low-power and Lossy Networks to provide additional terminology elements.

## 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

Table of Contents



## 1. Introduction

A new breed of Time Sensitive Networks is being developed to enable traffic that is highly sensitive to jitter and quite sensitive to latency. Such traffic is not limited to voice and video, but also includes command and control operations such as in industrial automation or in-vehicle sensors and actuators.

At IEEE802.1, the "Audio/Video Task Group", was renamed TSN for Time Sensitive Networking. The IEEE802.15.4 Medium Access Control (MAC) has evolved with IEEE802.15.4e which provides in particular the Time Slotted Channel Hopping (TSCH) mode for industrial-type applications. Both 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, like a train leaves intermediate stations at precise times along its path.

This document provides additional terminology elements to cover terms that are new to the context of TSCH wireless networks and other deterministic networks.

## 2. Terminology

The draft extends [I-D.ietf-roll-terminology] and use terms from RFC 6550 [RFC6550] and RFC 6552 [RFC6552], which are all included here by reference.

The draft does not reuse terms from IEEE802.15.4e such as "path" or "link" which bear a meaning that is quite different from classical IETF parlance.

This document adds the following terms:

6TiSCH: IPv6 over the Timeslotted Channel Hopping (TSCH) mode of IEEE 802.15.4e. It defines a set of IETF sublayers and protocols (in particular, for setting up a schedule with a centralized or distributed approach, managing the resource allocation), as well as the architecture to bind them together, for use in IPv6 TSCH based networks.

6F: IPv6 Forwarding. One of the three forwarding model supported by 6TiSCH. Packets are routed at layer 3, where QoS and RED operations are expected to prioritize flows with differentiated services.

6top: 6top is the adaptation layer between TSCH and upper layers like 6LoWPAN and RPL. It is defined in [I-D.draft-wang-6tsch-6top].

6top Data Convey Model: Model describing how the 6top adaptation layer feeds the data flow coming from upper layers into TSCH. It is composed by an I-MUX module, a MUX module, a set of priority queues, and a PDU (Payload Data Unit).

ASN: Absolute Slot Number, the timeslot counter, incremented by one at each timeslot. It is wide enough to not roll over in practice. See [I-D.watteyne-6tsch-tsch-lln-context].

Blacklist: Set of frequencies which should not be used for communication.

BBR: Backbone Router. In the 6TiSCH architecture, it is an LBR and also a NEAR. It performs ND proxy operations between registered devices and classical ND devices that are located over the backbone.

Broadcast cell: A scheduled cell whose neighbor MAC address is set to the broadcast address.

Bundle: A group of equivalent scheduled cells, i.e. cells identified by different [slotOffset, channelOffset], which are scheduled for a same purpose, with the same neighbor, with the same flags, and the same slotframe. The size of the bundle refers to the number of cells it contains. Given the length of the slotframe, the size of the bundle translates directly into bandwidth, either logical, or physical.

Cell: A single element in the TSCH schedule, identified by a slotOffset, a channelOffset, a slotframeHandle. A cell can be scheduled or unscheduled.

ChannelOffset: Identifies a row in the TSCH slotframe. The number of available channelOffsets is equal to the number of available frequencies. The channelOffset translates into a frequency when the communication takes place, resulting in channel hopping, as detailed in [I-D.watteyne-6tsch-tsch-lln-context].

Channel distribution/usage (CDU) matrix: : Matrix of height equal to the number of available channels (i.e, ChannelOffsets), representing the spectrum(channel) distribution among the different (RPL parent) nodes in the networks. Every single element of the matrix belongs to a specific chunk. It has to be noticed that such matrix, even though it includes all the cells grouped in chunks, belonging to different slotframes, is different from the TSCH schedule.

Chunk: A well-known list of cells, well-distributed in time and frequency, within a slotframe; a chunk represents a portion of a slotframe that is globally known by all the nodes in the network, but it can be managed separately by a single node. A node can have multiple chunks, and use them according to a specific policy. Chunks may overlap. They can be pre-programmed, or can be computed by an external entity at the network bootstrap.

Chunk ownership appropriation: The process by which an individual node obtains a chunk to manage based on peer-to-peer interaction with its neighbors.

Chunk ownership delegation: The process by which an individual node obtains a chunk to manage based on point-to-point interaction with an external entity.

Communication Paradigm: It is Associated with the Information Model [RFC3444] of the state that is exchanged, and indicates: the location of that state (e.g., centralized vs. distributed, RESTful, etc.), the numbers of parties (e.g., P2P vs. P2MP) and the relationship between parties (e.g., master/slave vs. peers) at a high level of protocol abstraction. Layer 5 client/server REST is a typical communication paradigm, but industrial protocols also use publish/subscribe which is P2MP and source/sink which is MP2MP and primarily used for alarms and alerts at the application layer. At layer 3, basic flooding, P2P synchronization and path-marking (RSVP-like) are commonly used paradigms, whereas at layer 2, master/slave polling and peer-to-peer forwarding are classical examples.

Dedicated Cell: A cell that is reserved for a given node to transmit to a specific neighbor.

Distributed cell reservation: A reservation of a cell done by one or more in-network entities (typically a connection endpoint).

Distributed track reservation: A reservation of a track done by one or more in-network entities (typically a connection endpoint).

EB: Enhanced Beacon frame used by an advertising node to announce the presence of the network. It contains information about the timeslot length, the current ASN value, the slotframes and timeslots the beaconing mote is listening on, and a 1-byte join priority (i.e., number of hops separating the node sending the EB, and the PAN coordinator).

FF: 6LoWPAN Fragment Forwarding. It is one of the three forwarding model supported by 6TiSCH. The 6LoWPAN Fragment is used as a label for switching at the 6LoWPAN sublayer, as defined in [I-D.thubert-roll-forwarding-frags].

GMPLS: Generalized Multi-Protocol Label Switching, a 2.5 layer service that is used to forward packets based on the concept of generalized labels.

Hard Cell: A scheduled cell which the 6top sublayer cannot reallocate. See [I-D.draft-wang-6tsch-6top].

Hopping Sequence: Sequence of frequencies, identified by a Hopping_Sequence_ID, used for channel hopping, when translating the channel offset value into a frequency (i.e., PHY channel). See [I-D.watteyne-6tsch-tsch-lln-context].

IE: Information Elements, a list of Type-Length-Value containers placed at the end of the MAC header, used to pass data between layers or devices. A small number of types are defined by TSCH, but a range of types is available for extensions, and thus, is exploitable by 6TiSCH. See [I-D.watteyne-6tsch-tsch-lln-context].

I-MUX module: Inverse-Multiplexer, a classifier that receives 6LoWPAN frames and places them into priority queues.

Interaction Model: It is a particular way of implementing a communication paradigm. Defined at a lower level of abstraction, it includes protocol-specific details such as a particular method (e.g., a REST GET) and a Data Model for the state to be exchanged.

KMP: Key Managment Protocol.

LBR: LLN Border Router. It is an LLN device, usually powered, that acts as a Border Router to the outside within the 6TiSCH architecture.

Link: A communication facility or medium over which nodes can communicate at the link layer, i.e., the layer immediately below IP. Thus, the IETF parlance for the term "Link" is adopted, as opposed to the incompatible IEEE802.15.4e terminology. In the context of the 6TiSCH architecture, which applies to Low Power Lossy Networks (LLNs), an IPv6 subnet is usually not congruent to a single link and techniques such as IPv6 Neighbor Discovery Proxying and Routing Over LLNs are required to achieve reachability within the multilink subnet. A link is distinct from a track. In fact, link local addresses are not expected to be used over a track for end to end communication. Finally, from the Layer 3 perspective (where the inner complexities of TSCH operations are hidden to enable classical IP routing and Forwarding), a single radio interface may be seen as a number of Links with different capabilities for unicast or multicast services.

Logical Cell: A cell that corresponds to granted bandwidth but is only lazily associated to a physical cell, based on usage.

MAC: Medium Access Control.

MUX module: Multiplexer, the entity that dequeues frames from priority queues and associates them to a cell for transmission.

NEAR: Energy Aware Default Router, as defined in [I-D.chakrabarti-nordmark-6man-efficient-nd].

NME: Network Management Entity, the entity in the network managing cells and other device resources. It may cooperate with the PCE. It interacts with LLN nodes through the backbone router.

PANA: Protocol for carrying Authentication for Network Access, as defined in [RFC5191] . It is the protocol used in the 6TiSCH architecture for handling authentication during the join process.

PCE: Path Computation Element, the entity in the network which is responsible for building and maintaining the TSCH schedule, when centralized scheduling is used.

PCE cell reservation: The reservation of a cell done by the PCE.

PCE track reservation: The reservation of a track done by the PCE.

QoS: Quality of Service.

(to) reallocate a cell: The action operated by the 6top sublayer of changing the slotOffset and/or channelOffset of a soft cell.

SA: Security Association.

(to) Schedule a cell: The action of turning an unscheduled cell into a scheduled cell.

Scheduled cell: A cell which is assigned a neighbor MAC address (broadcast address is also possible), and one or more of the following flags: TX, RX, shared, timeskeeping. A scheduled cell can be used by the IEEE802.15.4e TSCH implementation to communicate. A scheduled cell can be a hard cell or a soft cell.

Shared Cell: A cell that is used by more than one transmitter nodes at the same time and on the same channelOffset. Only cells with TX flag can be marked as "shared". A backoff algorithm is used to resolve contention.

SlotOffset: Identifies a column in the TSCH schedule, i.e., the number of timeslots since the beginning of the current iteration of the slotframe.

Slotframe: A MAC-level abstraction that is internal to the node and contains a series of timeslots of equal length and priority. It is characterized by a slotframe_ID, and a slotframe_size. Multiple slotframes can coexist in a node's schedule, i.e., a node can have multiple activities scheduled in different slotframes, based on the priority of its packets/traffic flows. The timeslots in the Slotframe are indexed by the SlotOffset; the first timeslot is at SlotOffset 0.

Soft Cell: A scheduled cell which the 6top sublayer can reallocate, as described in [I-D.draft-wang-6tsch-6top].

TF: Track Forwarding. It is the simplest and fastest forwarding model supported by 6TiSCH. It is a G-MPLS-like forwarding model. The input cell characterises the flow and indicates the output cell.

Timeslot: A basic communication unit in TSCH which allows a transmitter node to send a frame to a receiver neighbor, and that receiver neighbor to optionally send back an acknowledgment. The length of the timeslot determines the maximum size of the frame that can be exchanged.

Time Source Neighbor: A neighbor a node uses as its time reference, and to which it needs to keep its clock synchronized. A node can have one or more time source neighbors.

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.

TrackID: Unique identifier of a track, assigned by the owner of the track.

TSCH: Time Slotted Channel Hopping, a medium access mode of the [IEEE802154e] standard which uses time synchronization to

achieve ultra low-power operation and channel hopping to enable high reliability.

TSCH Schedule: A matrix of cells, each cell indexed by a slotOffset and a channelOffset. The slotframe size (the "width" of the matrix) is the number of timeslots it contains. The number of channelOffset values (the "height" of the matrix) is equal to the number of available frequencies. The TSCH schedule contains all the scheduled cells from all slotframes and is sufficient to qualify the communication in the TSCH network.

unscheduled cell: A cell which is not used by the IEEE802.15.4e TSCH implementation.

## 3. IANA Considerations

This specification does not require IANA action.

## 4. Security Considerations

This specification is not found to introduce new security threat.

## 5. Acknowledgements

Thanks to the IoT6 European Project (STREP) of the 7th Framework Program (Grant 288445).

## 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.

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

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

[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.

## 6.2. Informative References

[I-D.chakrabarti-nordmark-6man-efficient-nd]
Chakrabarti, S., Nordmark, E., Thubert, P., and M. Wasserman, "Wired and Wireless IPv6 Neighbor Discovery Optimizations", draft-chakrabarti-nordmark-6man-efficient-nd-04 (work in progress), October 2013.

[I-D.draft-sudhaakar-6tisch-coap]
Sudhaakar, R., Ed. and P. Zand, "6TiSCH Data Model for CoAP-00 (work in progress)", October 2013.

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

[I-D.ietf-roll-terminology]
Vasseur, J., "Terms used in Routing for Low power And Lossy Networks", draft-ietf-roll-terminology-13 (work in progress), October 2013.

[I-D.ohba-6tsch-security]
Chasko, S., Das, S., Lopez, R., Ohba, Y., Thubert, P., and A. Yegin, "Security Framework and Key Management Protocol Requirements for 6TSCH", draft-ohba-6tsch-security-01 (work in progress), July 2013.

[I-D.thubert-6tisch-architecture]
Thubert, P., Watteyne, T., and R. Assimiti, "An Architecture for IPv6 over the TSCH mode of IEEE 802.15.4e", draft-thubert-6tisch-architecture-01 (work in progress), October 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.vilajosana-6tisch-minimal]
Vilajosana, X. and K. Pister, "Minimal 6TiSCH Configuration", draft-vilajosana-6tisch-minimal-00 (work in progress), October 2013.

[I-D.watteyne-6tsch-tsch-lln-context]
Watteyne, T., Palattella, M., and L. Grieco, "Using IEEE802.15.4e TSCH in an LLN context: Overview, Problem Statement and Goals", draft-watteyne-6tsch-tsch-lln-context-02 (work in progress), May 2013.

## 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) Amendament 1: MAC sublayer", April 2012.

## Authors' Addresses

Maria Rita Palattella (editor)
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

Pascal Thubert
Cisco Systems, Inc
Village d'Entreprises Green Side
400, Avenue de Roumanille
Batiment T3
Biot - Sophia Antipolis 06410
FRANCE

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

Thomas Watteyne
Linear Technology / Dust Networks
30695 Huntwood Avenue
Hayward, CA 94544
USA

Phone: +1 (510) 400-2978
Email: twatteyne@linear.com

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

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

6TiSCH
INTERNET-DRAFT
Intended Status: Informational
Expires: June 17, 2014

G. Piro
(Politecnico di Bari)
G. Boggia
(Politecnico di Bari)
L. A. Grieco
(Politecnico di Bari)

December 14, 2013

# A standard compliant security framework for Low-power and Lossy Networks
draft-piro-6tisch-security-issues-01

## Abstract

The aim of this Internet Draft is to define a standard compliant security framework for Low-power and Lossy Networks, in order to enable message encryption and authentication at the MAC layer. The framework is fully compatible with both IEEE 802.15.4 and IEEE 802.15.4e standards and supports a wide range of security features to network architectures developed within the 6TiSCH Working Group. In particular, it defines different kinds of security configurations and, for each of them, proposes lightweight mechanisms for the setting-up of a secure IEEE 802.15.4 domain and the negotiation of link keys among devices.

## Status of this Memo

Table of Contents

# 1 Acronyms

In addition to the acronyms defined in [I-D.palattella-6tisch-terminology], the following acronyms are used in this document:

DH Diffie Hellman

DEX Diet EXchange

DTLS Datagram Transport Layer

HIP Host Identity Protocol

PKI Public Key Infrastructure

# 2 Introduction

The IEEE 802.15.4 standard [IEEE802154] is widely recognized as one of the most successful enabling technologies for short-range low-rate wireless communications. It covers all the details related to the Medium Access Control (MAC) and physical layers of the protocol stack and supports the possibility to protect MAC packets by means of symmetric-key cryptography techniques with several security options. However, the IEEE 802.15.4 standard does not explain how handling the initialization of a secure IEEE 802.15.4 domain, the generation and the exchange of keys, and the management of joining operations in a secure 802.15.4 network already configured in the past, thus delegating the upper layers to orchestrate, enable, configure, and negotiate security services.

The IEEE 802.15.4e [IEEE802154e] standard introduces some amendments to the IEEE 802.15.4 standard. Among its key features there is the Time-slotted Channel Hopping (TSCH), i.e., a novel MAC protocol, which better supports multi-hop communications in emerging industrial applications.

Since the IEEE 802.15.4e amendment focuses only on link-layer aspects, the 6TiSCH WG was born to define open standards in support of the adoption of IPv6 over the TSCH mode of the IEEE802.15.4e standard, thus covering all facets related to the management of network communications in complex (and eventually distributed) Low-Power and Lossy Networks (LLNs) [I-D.watteyne-6tisch-tsch] [I-D.wang-6tisch-6top].

Security aspects represent an important issue that need to be considered in a 6TiSCH network. TSCH defines mechanisms to encrypt and authenticate MAC frames but it does not define how this keying material is generated [IEEE802154]. For this reason, the 6TiSCH WG needs to (i) define the keying material and authentication mechanism needed by a new mote to join an existing network; (ii) define a mechanism to allow for the secure transfer of application data between neighbor motes; and (iii) define a mechanism to allow for the secure transfer of signaling data between motes and 6TiSCH [I-D.watteyne-6tisch-tsch].

In literature, several security strategies have been proposed for wireless sensor networks. Most of them exploits key negotiation algorithms and key management architectures summarized in [Camtepe2005], [Wang2006], and [Cayirci2007]. However, all of these works focus on a specific issue without embracing all the security features presented in [I-D.watteyne-6tisch-tsch].

Definitively, despite the high number of solutions focusing on security issues for LLNs, there is the lack of a complete and effective framework enabling security services in 6TiSCH compliant networks. Hence, the design of a one-size-fits-all solution is still an ambitious goal for researchers working in this area.

A first step in this direction has been moved by ZigBee IP specifications, i.e., a suite of high level communication protocols sitting on top of the IEEE 802.15.4 MAC [ZIGBEEIP]. ZigBee IP supports end-to-end and link-layer security and a public key infrastructure based on X.509 certificates. It imposes the adoption of the same link-key (shared among all nodes) to protect packets belonging to any kind of services (i.e., only a single security level is allowed). This makes the network highly sensible to the presence of compromised devices. Moreover, the cost needed to update the key in all devices increases with the size of the network. Finally, ZigBee does not support TSCH at the MAC layer. According to that considerations, the security architecture defined in ZigBee IP is not well suitable for 6TiSCH environments.

In the IEFT area, there are three main proposals on this topic: [I-D.garcia-core-security], [I-D.roll-security-framework], and [I-D.ohba-6tisch-security]. In [I-D.garcia-core-security] a set of security profiles in different network environments (e.g., network without security requirements, home, managed home, industrial, and advanced industrial) have been presented. For each of them, a number of security threats, as well as a list of well-known protocols and algorithms able to fixing these issues, have been also identified. In [I-D.roll-security-framework] is presented an accurate analysis on security threats at different point of the proto of a LLN, together

with the presentation of possible countermeasures to these dangers. In [I-D.ohba-6tisch-security] is presented a secure and scalable key management framework to adopt in 6TiSCH networks, as well as a set of requirements that a key management protocols should satisfy in that framework. All these IETF's proposals does not focus the attention on the design of a specific Key Management Protocol (KMP) for LLNs. Indeed, they relay on well-known approaches (i.e., like PANA [RFC5191], HIP DEX [HIPDEX], DTLS [RFC6347]) and to solutions based on the adoption of Public Key Infrastructure (PKI) to manage both node authentication and key negotiation procedures.

Nodes in a LLN have very limited storage, energy, and computational capabilities. For example, the LPC platform, which is the most powerful among those described in [Watteyne2012], have just 64 KB RAM, 512 KB program flash memory, and a 120 MHz ARM Cortex M3 micro-controller. Cryptography operations conducted for a single data packet generate, per se, a not neglectable computational overhead and energy consumption [Altolini2013]. On the other hand, the implementation and the management of complex security protocols (such as PAN, HIP DEX, and DTLS) and PKI infrastructure could not be suitable in LLNs because of the generation of a very large amount of signaling messages and the need for additional (and massive) computational loads and energy consumptions [Riaz2009]. Hence, all the available proposals cannot be easily applied in the 6TiSCH context: the design of simpler solutions, able to cover all the security aspects highlighted in [I-D.watteyne-6tisch-tsch] and able to be fully compatible with IEEE 802.15.4 and IEEE 802.15.4e specifications, is still an uncovered task.

The aim of this Internet Draft is to design a complete, simple, and standard compliant framework supporting a number of security features for the TSCH MAC layer. This work is complementary with respect to [I-D.ohba-6tisch-security] and, at the same time, it is fully integrated within the whole 6TiSCH architecture. Moreover, is has been designed in order to ensure an easy and effective implementation on real devices.

The main goals of this proposal have been summarized in the sequel:

- identify potential security configurations that could be supported by a 6TiSCH network;

- define a framework covering all the security issues (i.e., confidentiality and integrity protection, mote bootstrap, key negotiation and maintenance) presented in [I-D.watteyne-6tisch-tsch];

- propose an efficient mechanism, handled by the 6top adaptation layer, to configure a secured 6TiSCH network;

- propose a lightweight KMP to generate link keys that will be used to protect, at the MAC layer, unicast communications;

- explain the interaction between 6top and TSCH MAC layer during the configuration of the secured 6TiSCH network.

## 3 Security in IEEE 802.15.4

This section summarizes security features defined within the standard [IEEE802154], thus simplifying the comprehension of the remaining part of this Internet Draft.

The IEEE 802.15.4 standard defines eight security levels to protect MAC frames, as summarized in Fig. 1. The standard imposes the adoption of the CCM* algorithm to perform encryption and description procedures. It requires a key of 128 bit.

| Security level | Security attribute | Data Integrity | Data Confidentiality |
|---|---|---|---|
| 0 | None | No | No |
| 1 | MIC-32 | Yes | No |
| 2 | MIC-64 | Yes | No |
| 3 | MIC-128 | Yes | No |
| 4 | ENC | No | Yes |
| 5 | ENC-MIC-32 | Yes | Yes |
| 6 | ENC-MIC-64 | Yes | Yes |
| 7 | ENC-MIC-128 | Yes | Yes |

Figure 1. Security levels available for a IEEE 802.15.4 network.

At the MAC layer, encryption and decryption functionalities are implemented within the "outgoing frame security" and the "incoming frame security" procedures, respectively. They exploits a number of security attributes, summarized in what follows:

- macKeyTable: it is composed by a set of KeyDescriptor elements.

A specific KeyDescriptor element is created for each key, composed by (see Tab. 61 of the IEEE 802.15.4 standard for more details [IEEE802154]):

- The KeyIdLookupList, which is a list of KeyIdLookupDescriptor entries. A KeyIdLookupDescriptor is composed by a set of parameters (see Tab. 65 of the IEEE 802.15.4 standard for more details [IEEE802154]), i.e., KeyIdMode, KeySource, KeyIndex, DeviceAddMode, DevicePANId, and DeviceAddress, that are used to identify the key within the macKeyTable.

- The DeviceDescriptorHandleList, which contains pointers to DeviceDescriptor elements stored within the macDeviceTable. It is used to identify which devices may use the key.

- The KeyUsageList, which is a list of KeyUsageDescriptor elements. A KeyUsageDescriptor is composed by the FrameType and the CommandFrameIdentifies fields that indicate the frame type with which the considered key may be used (see Tab. 62 of the IEEE 802.15.4 standard for more details [IEEE802154]).

- The Key.

- macDeviceTable: it is composed by a set of DeviceDescriptor elements, providing some information about remote devices which the node can establish a secure communication with. A dedicated DeviceDescriptor element is associated to each remote device. It is composed by a number of fields, i.e., PANId, ShortAddress, ExtAddress, FrameCounter, and Extemp, which collect information related to a specific remote device (see Tab. 64 of the IEEE 802.15.4 standard for more details [IEEE802154]).

- macSecurityLevelTable: it is made by a set of SecurityLevelDescriptor elements, which store details about the security level required for each MAC frame type and subtype. Fields belonging to the SecurityLevelDescriptor data structure are: FrameType, ComamndFrameIdentifier, SecurityMinimum, DeviceOverrideSecurityMinimum, and AllowedSecurityLevels (see Tab. 63 of the IEEE 802.15.4 standard for more details [IEEE802154]).

- macFrameCounter: it is an integer value storing the outgoing frame counter for the considered device.

- macAutoRequestSecurityLevel: it is an integer value providing the security level used for automatic data requests.

- macAutoRequestKeyIdMode: it is an integer value indicating the key identifier mode used for automatic data requests. It is not valid if the macAutoRequestSecurityLevel attribute is set to 0x00.

- macAutoRequestKeySource: it represents a short or extended IEEE 802.15.4 MAC address, indicating the originator of the key used for automatic data requests. This attribute is not valid if the macAutoRequestKeyIdMode element is not valid or set to 0x00.

- macAutoRequestKeyIndex: it is an integer value storing the index of the key used for automatic data requests. It is not valid if the macAutoRequestKeyIdMode attribute is not valid or set to 0x00.

- macDefaultKeySource: it is the extended IEEE 802.15.4 MAC address of the originator of the default key used for key identifier mode 0x01.

During the outgoing security procedure, the high layer uses the KeyIdMode parameter to select a specific key in the macKeyTable to be used for protecting the MAC frame.

The KeyIdMode is set to 00, 01, 10, and 11 in the case the key can be derived implicitly by both sender and the receiver and its is not specified in the message, the key is determined explicitly by the KeyIndex parameter stored into the MAC header and the macDefaultKeySource, the key can be derived by considering KeyIndex and KeySource fields stored into the MAC header (with KeySource representing the short address of the device that has generated the key), and the key can be derived by considering KeyIndex and KeySource fields stored into the MAC header (with KeySource representing the IEEE extended address of the device that has generated the key), respectively.

Both IEEE 802.15.4 and IEEE 802.15.4e standards do not provide any guideline to create (and or negotiate) keys, as well as to configure the aforementioned security MAC attributes.

## 4 Definition of the secured domain

In this document, the "secured domain" concept refers to the portion of a 6TiSCH network where procedures and techniques described in this proposal must be performed in order to configure and maintain secured communications.

In a 6TiSCH network, nodes can be arranged in both peer-to-peer and star

topologies. In general, they can be organized through a Direction Oriented Directed Acyclic Graph (DODAG), which is initialized by the PAN coordinator. From the beginning, the PAN coordinator is in charge of generating initial advertising messages and handling joining procedures of neighbors. During the construction of the DODAG, a mote can become a reference node for its neighbors and, for this reason, it can handle the generation of advertisement and the management of joining procedure [PalattellaSurvey].

A cluster represents the portion of a 6TiSCH network where the distribution of radio resources and the management of TSCH frames is handled by a given mote, namely cluster head or coordinator.

A secured domain coincides with the cluster’s concept.

This means that the secured framework presented in this draft is in charge of configuring and managing security capabilities in each cluster in a 6TiSCH network. It is hence highly scalable because its complexity does not depend from the network size but only by the number of motes forming a specific cluster.

Just to simplify the description of all the security procedures and mechanisms, the rest of the document will focus the attention on a 6TiSCH network with only one cluster, which is composed by a PAN coordinator and a number of remote motes.

## 5 Security Configurations

The following network configurations are defined to support different security features within a 6TiSCH network.

- Fully Secured network: all the IEEE 802.15.4 devices forming the network are configured to fully support security services. It represents the most secured configuration: all packets, independently from the message they carry, are encrypted and authenticated. Nodes that do not support security capabilities (or that are not in posses of all the information to joining the network, such as key materials and encryption and decryption algorithms) are not allowed to join the network.

- Unsecured network: security services are not supported. Even if in possession of security capabilities, any pair of nodes is not allowed to establish a secured communication. Differently for the Fully Secured scheme, this is the lowest security level. Since the data encryption, the message integrity, and the peer authentication are not implemented, all the MAC frames are

exchanged in clear. Hence, the setup and the maintaining of the network are described by the standard and no further upgrades are required.

- Partial Secured network: only the integrity of message is supported.

- Hybrid Secured network: the network can be composed by heterogeneous nodes that could or could not support security features. As default, the network is created in an unsecured manner. All the non-unicast control messages sent by the coordinator should be transmitted in clear. In this way, in fact, it is ensured that all the devices are able to read the content of packets. A RFD node with security capabilities, that intends to exchange encrypted and/or authenticated packets with the coordinator, could negotiate a set of link key with its reference FFD.

- Flexible Secured network: as default, the network is setup with the Fully Secured configuration and all packets are encrypted and authenticated. If there is at least one node that have not security capabilities, the coordinator could decides to switch to the Hybrid Secured configuration.

The implementation of each of these network configurations is fully supported by the secured architecture devised within the IEEE 802.15.4 and the IEEE 802.15.4e standard [IEEE802154]. This means that the proposal presented in this Internet Draft does not introduces any changes to the standard but it just introduces techniques and procedures able to accomplish those aspects that have been left opened by IEEE specifications.

### 5.1 Minimum security requirements

The IEEE 802.15.4 standard imposes to specify, for each kind of MAC packet, minimum security levels that should be guaranteed. These restrictions must be detailed for each remote device.

To this end, SecurityMinimum, DeviceOverrideSecurityMinimum, and AllowedSecurityLevels parameters are stored into the DeviceDescriptor element (see Sec. 3) to define the minimum security level (i.e., one of those reported in Fig.1), the possibility to override the minimum security level (i.e., DeviceOverrideSecurityMinimum is just a boolean flag), and the list of allowed security levels in the case the minimum one could be overridden, respectively.

With reference to secure network configurations presented in Sec. 3.1, these parameters must be set as reported in Fig. 2.

| Attribute | Secured Network Configurations | | | | |
|---|---|---|---|---|---|
| | Unsecured | Fully | Partial | Hybrid | Flexible |
| SecurityMinimum | 0 | from 5 to 7 | from 1 to 4 | 0 | from 1 to 7 |
| DeviceOverride-SecurityMinimum | FALSE | FALSE | FALSE | FALSE | TRUE |
| AllowedSecuri-tyLevelsvels | 0 | from 5 to 7 | from 1 to 4 | from 0 to 7 | from 0 to 7 |

Figure 2. Setting of security attributes of the DeviceDescriptor element in each proposed secure network configuration.

The Unsecured network configuration does not support any security features. Hence, both minimum and allowable security levels are set to 0 for all the MAC frames and the possibility to override such constraints is disabled for all devices.

If the Fully Secured configuration is enabled, the minimum security level must be chosen in the range [5,7], thus allowing the possibility to support the encryption and the authentication of messages. The manufacturer must set the default value to 7; it can be updated by the network administrator. The minimum security level must not be overridden by any devices and, as a consequence, the field AllowedSecurityLevels should contain only one value, equal to the minimum security level.

If the Partial Secured configuration is enabled, the minimum security level must be chosen in the range [1,4], thus allowing the possibility to support the authentication of messages. The manufacturer must set the default value to 4; it can be updated by the network administrator. The minimum security level must not be overridden by any devices and, as a consequence, the field AllowedSecurityLevels should contain only one value, equal to the minimum security level.

If the Hybrid Secured configuration is enabled, the minimum security level must be set to 0, thus supporting the joining of devices having different security capabilities. All the security levels could be allowed and the network administrator could decide to enable only a subset of them according to the network design.

If the Flexible Secured configuration is enabled, the minimum security level must be set to 1. The joining of nodes without (or with limited)

security capabilities is permitted by setting the DeviceOverrideSecurityMinimum variable to TRUE and by including lower security levels in the list of AllowedSecurityLevels.

## 6 Initialization of a secured domain in a 6TiSCH network

The secured framework builds a secured domain (the definition of a secured domain is provided in Sec. 4) through the execution of three consecutive phases: Setting-up, Bootstrapping, and Key Negotiation (see the framework architecture in Fig. 3.

The Setting-up Phase is used to store into the device all the secrets required to initialize a secured domain. The Bootstrap Phase is used to initialize the secured domain and to compute a key that will be adopted to protect broadcast messages at the MAC layer. The Key Negotiation Phase handles the KMP and it is used to negotiate the key between a couple of nodes in a cluster.

These phases are not always mandatory for all the secured network configurations listed in Sec. 5. In particular, when both Unsecured and Partial Secured Configurations are used, it is not necessary to implement none of aforementioned phases because there is not the need to compute any key. On the contrary, the Fully Secured Configuration, and as a consequence also the Flexible Secured Configuration, requires the implementation of all the phases. For the Hybrid Secured Configuration, instead, the Bootstrap Phase is not mandatory because broadcast messages are always sent in clear.

The need to implement or not Setting-up, Bootstrap, and Key Negotiation Phases in each envisaged secured configuration is highlighted in Fig. 4.

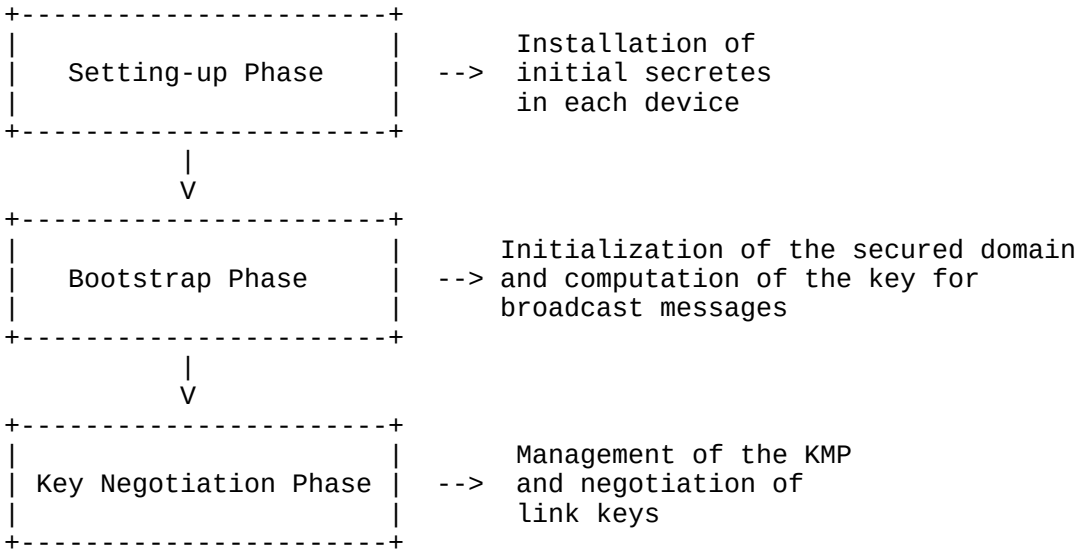



Figure 3. Summary of the proposed framework.

| Phase | Secured Network Configurations | | | | |
|---|---|---|---|---|---|
| | Unsecured | Fully | Partial | Hybrid | Flexible |
| Setting-up | NO | YES | NO | YES | YES |
| Bootstrap | NO | YES | NO | NO | YES |
| Key Negotiation | NO | YES | NO | YES | YES |

Figure 4. Implementation of Setting-up, Bootstrap, and Ken Negotiation Phases for each envisaged secured configuration.

## 6.1 Setting-up phase

The setting-up phase is used to properly configure the device that will join to a secured 6TiSCH networks. It consists in storing, within the device, parameters and initial secrets (i.e., the masterKey), which will be used by secure algorithms and procedure to setup the secure domain. They include. (i) the MasterKey, (ii) the PrimeNumbersTable, (iii) the GlobalSecurityLevelsTable, and (iv) the public key of a certification authority.

This operation may be performed by the manufacturer or by the network

administrator.

### 6.1.1 The role of the MasterKey

The MasterKey is an initial secret, which is shared among all the motes.

The MasterKey is used to generate the DefaultKey that will be exploited to to protect broadcast messages (such as the beacon frame) in a given cluster.

A security approach based on preloaded key is widely adopted, especially in the case motes need to compute a common secret without relaying on preliminary negotiation steps [Camtepe2005], [Wang2006], [Cayirci2007].

This is the case of the Fully Secured Configuration. When this configuration is enabled, in fact, all the packets, including the beacon message, must be encrypted from the beginning. This means that any mote that wants to join to the network must compute the key adopted to protect the beacon message autonomously. Otherwise it will not be able to complete the join process because it will not be able to correctly extract information from the beacon message.

The MasterKey is not directly used to encrypt and decrypt broadcast messages. Another key, i.e., the DefaultKey, is instead computed starting from the MasterKey and other time-varying parameters

From one side, the MasterKey is unique for the whole 6TiSCH network and it may potentially remain the same for the whole life of the network. From another hand, each cluster will compute its specific the DefaultKey, which may potentially have a limited life time.

By using a time-varying key, we improve the resilience of the network to known-plaintext attack [StallingsSecurityBook], through which an attacker computes a key starting from the knowledge of group of clear/encrypted messages. In fact, even if an attacker will be able to obtain, through cryptanalysis techniques, the DefaultKey, it will be able to compromise only a specific cluster of the network and for a limited amount of time.

A mote can be subjected to any kind of tamper attacks. Without any further shrewdness, an attacker that may physically access to the mote could extract the MasterKey, thus compromising the security of the whole 6TiSCH network. Hence, it is very important to ensure the protection to that tampering attacks by using specific software-based and/or hardware-based mechanisms [Walters07][Becher2006].

### 6.1.2 The role of the GlobalSecurityLevelsTable

The GlobalSecurityLevelsTable, that has been reported in Fig. 5, is used to store the minimum security level and the list of allowed security levels that must be adopted for each kind of MAC frame and for each security configuration defined in Sec. 5. The table reported in Fig. 5 does not consider ACK messages because they must not be protected (as imposed by the IEEE 802.15.4 standard [IEEE802154]).

Both the minimum security level and the list of allowed security levels must be chosen by the manufacturer or by the network administrator, according to restrictions reported in Fig. 2.

| Attribute | Frame Type | Secured Network Configurations | | | |
|---|---|---|---|---|---|
| | | Fully | Partial | Hybrid | Flexible |
| Security Minimum | Beacon | | | | |
| Security Minimum | Data | | | | |
| Security Minimum | Command MAC | | | | |
| AllowedSecurityLevels | Beacon | | | | |
| AllowedSecurityLevels | Data | | | | |
| AllowedSecurityLevels | Command MAC | | | | |

Figure 5. Structure of the GlobalSecurityLevelsTable.

### 6.1.3 The role of the PrimeNumbersTable

The PrimeNumbersTable stores a set of N prime numbers and their respective primitive roots, which are used during the Key Negotiation Phase. Its implementation is reported in Fig. 6.

These prime numbers are not the keys to protect MAC frames but are just numbers that will be exploited to generate keys according to the DH algorithm [DH].

| PrimeNumberId | Prime Number | Primitive Root |
|---|---|---|
| 1 | p1 | g1 |
| 2 | p2 | g2 |
| : | | : |
| N | pN | gN |

Figure 6. Structure of the PimeNumbersTable.

As described in Appendix A.1, the security level of the proposed approach does not depend from the size of the PrimeNumbersTable but it is only influenced by the length of each prime number.

### 6.1.4 The role of the public key of the authority

The Key Negotiation Phase exploits a KMP based on the DH algorithm. The possibility to authenticate motes through public certificates is also supported.

## 6.2 Bootstrap phase

The implementation of the Bootstrap Phase is different for both PAN coordinator and remote mote. Moreover, for a remote mote, two different procedure has been conceived for both the not-beacon-enabled and the beacon-enabled networks.

### 6.2.1 Bootstrap phase for the PAN coordinator

A 6TiSCH network is initialized by a FFD node that will become the PAN coordinator. As described in [IEEE802154], at the end of this phase the PAN coordinator has chosen both the identification number for the PAN, i.e., the PAN_ID, and the short MAC address of the IEEE 802.15.4 network, i.e., shortMACaddress.

Once such task has been finished, the 6top adaptation layer computes the DefaultKey, D_k, and configures a set of security MAC attributes through specific commands (they have been introduced in [I-D.wang-6tisch-6top]).

In particular, it executes the following operations:

a) A CONFIGURE.security command is generated by the 6top layer and sent to the MAC entity to initialize security attributes (as discussed in [I-D.wang-6tisch-6top], in Sec. 2.4.9.1). The set of parameters handled by this command are set as in the sequel:

a.1) enable = true;

a.2) macAutoRequestSecurityLevel = security level expected for the beacon message and stored within the GlobalSecurityLevelsTable;

a.3) macAutoRequestKeyIdMode = 0x03;

a.4) macAutoRequestKeySource = MAC address of the PAN coordinator;

a.5) macAutoRequestKeyIndex = 1;

a.6) macDefaultKeySource = MAC address of the PAN coordinator;

b) CONFIGURE.security.macSecurityLevelTable command is generated by the 6top layer and sent to the MAC entity to initialize macSecurityLevelTable (as discussed in [I-D.wang-6tisch-6top], in Sec. 2.4.9.3). Parameters stored into this command are taken from the GlobalSecurityLevelsTable.

c) The DefaultKey, D_k, is obtained from the MasterKey, M_k, by using a 128-bit hash function, H_128(.)

D_k= H_128(PAN_ID | shortMACaddress | M_k).

d) A new KeyIdLookupList data structure is created. A KeyIdLookupDescriptor is generated and stored into the KeyIdLookupList data structure. The KeyIdMode, the KeySource, and the KeyIndex variables of this KeyIdLookupDescriptor are set to 0x03, the MAC address of the device, and 1, respectively. Instead, DeviceAddrMode, DevicePANId, and DeviceAddress are not set due to the selected KeyIdMode (see Tab. 65 of the IEEE 802.15.4 standard for more details [IEEE802154]).

e) A KeyUsageList data structure is created. One KeyUsageDescriptor for each kind of broadcast messages is create and stored into the KeyUsageList data structure.

f) An empty DeviceDescriptorHandleList is created. No data are stored within this list because the PAN coordinator does not yet know the list of devices that may use this key.

Then, the 6top layer deliver the DefaultKey, the KeyIdLookupList, the KeyUsageList, and the DeviceDescriptorHandleList to the MAC layer by using the CONFIGURE.security.macKeyTable primitive (as discussed in [I-D.wang-6tisch-6top], in Sec. 2.4.9.2).

Triggered by the CONFIGURE.security.macKeyTable command, the MAC layer will create a KeyDescriptor associated to the DefaultKey, D_k,in which storing all the parameters received by the 6top layer, and and will store it within the macKeyTable.

The Bootstrap phase for the PAN coordinator has been summarized in Fig. 7.

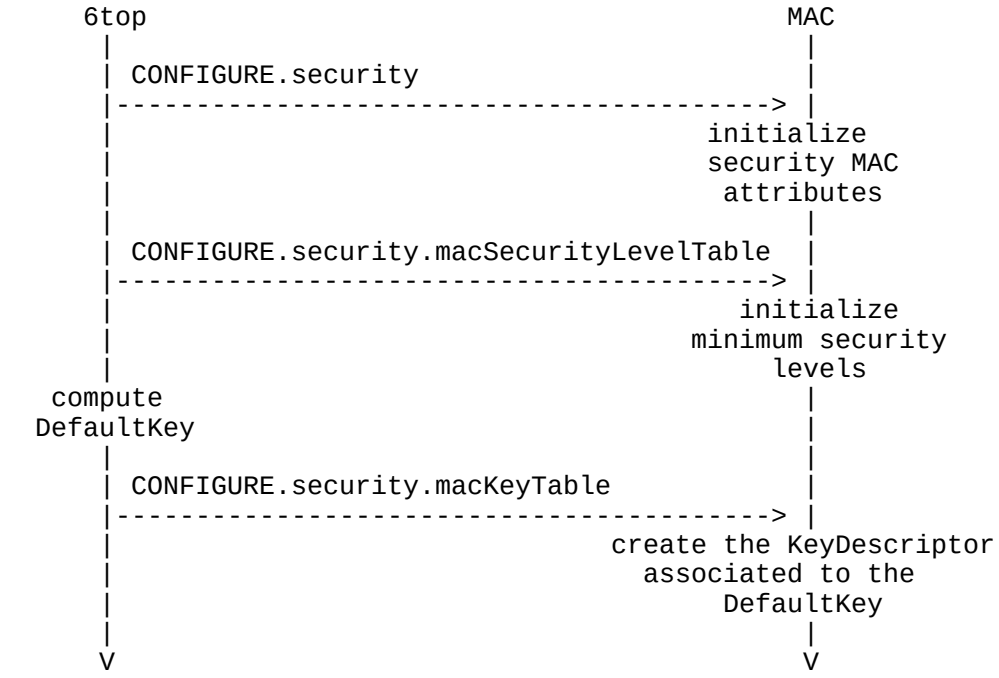



Figure 7. Bootstrap Phase for the PAN coordinator.

### 6.2.2 Bootstrap phase for a mote in a Beacon-enabled network

To join the network, a mote should associate with the coordinator. The Next Higher Layer sends to the MAC entity the MLME-ASSOCIATE.request primitive, starting the association procedure.

As for the PAN coordinator, in this phase the 6top adaptation layer should initialize security MAC attributes, compute the DefaultKey, D_k, and updates MAC security attributes accordingly.

To this end, after the reception of the beacon message, the following operations are executed:

a) The PAN_ID, the MAC address of the coordinator, and the FrameCounter are extracted from the header of the beacon message.

b) A CONFIGURE.security primitive is generated by the 6top layer and sent to the MAC entity to initialize security attributes (as discussed in [I-D.wang-6tisch-6top], in Sec. 2.4.9.1). The set of parameters handled by this primitive are set as in the sequel:

b.1) enable = true;

b.2) macAutoRequestSecurityLevel = security level expected for the beacon message and stored within the GlobalSecurityLevelsTable;

b.3) macAutoRequestKeyIdMode = 0x03;

b.4) macAutoRequestKeySource = MAC address of the PAN coordinator;

b.5) macAutoRequestKeyIndex = 1;

b.6) macDefaultKeySource = MAC address of the PAN coordinator;

c) CONFIGURE.security.macSecurityLevelTable primitive is generated by the 6top layer and sent to the MAC entity to initialize macSecurityLevelTable (as discussed in [I-D.wang-6tisch-6top], in Sec. 2.4.9.3). Parameters stored into this command are taken from the GlobalSecurityLevelsTable.

d) The DefaultKey, D_k, is obtained from the MasterKey, M_k, by using a 128-bit hash function, H_128(.)

D_k= H_128(PAN_ID | shortMACaddress | M_k).

e) A new KeyIdLookupList data structure is created. A KeyIdLookupDescriptor is generated and stored into the KeyIdLookupList data structure. The KeyIdMode, the KeySource, and the KeyIndex variables of this KeyIdLookupDescriptor are set to 0x03, the MAC address of the PAN coordinator, and 1, respectively. Instead, DeviceAddrMode, DevicePANId, and DeviceAddress are not set due to the selected KeyIdMode (see Tab. 65 of the IEEE 802.15.4 standard for more details [IEEE802154]).

f) A KeyUsageList data structure is created. One KeyUsageDescriptor for each kind of broadcast messages is create and stored into the KeyUsageList data structure.

g) A new DeviceDescriptor element, associated to the PAN coordinator (i.e., the FFD node that sent the Beacon message) is created and stored into the macDeviceTable. It is built considering these specifications (see Tab. 64 of the IEEE 802.15.4 standard [IEEE802154] for more details):

g.1) The PANId variable is associated to the PAN_ID value extracted from the Beacon message.

g.2) The ShortAddress is set to the MAC address of the coordinator whenever the short addressing mode is used. This parameter is set to 0xfffe if only the extended addressing mode is used. If its value is unknown, the ShortAddress parameter is set to 0xfff.

g.3) The ExtAddress is set to the IEEE MAC address of the coordinator.

g.4) The FrameCounter parameter is set to the FrameCounter value extracted from the Beacon message.

g.5) The Extempt boolean flag is set to the allowed value of the DeviceOverriddeSecurityMinimum variable described in Fig. 2.

h) The DeviceDescriptorHandleList is created and populated with the DeviceDescriptor created at the previous step.

i) A KeyUsageList data structure is created and stored within the KeyDescriptor element. One KeyUsageDescriptor for each broadcast message is create and stored into the KeyUsageList data structure.

Then, the 6top layer deliver the DefaultKey, the KeyIdLookupList, the KeyUsageList, and the DeviceDescriptorHandleList to the MAC layer by using the CONFIGURE.security.macKeyTable primitive (as discussed in [I-D.wang-6tisch-6top], in Sec. 2.4.9.2).

Triggered by the CONFIGURE.security.macKeyTable primitive, the MAC layer will create a KeyDescriptor associated to the DefaultKey, D_k, in which storing all the parameters received by the 6top layer, and and will store it within the macKeyTable.

The Bootstrap Phase for a remote mote in a beacon-enabled network has been summarized in Fig. 8.

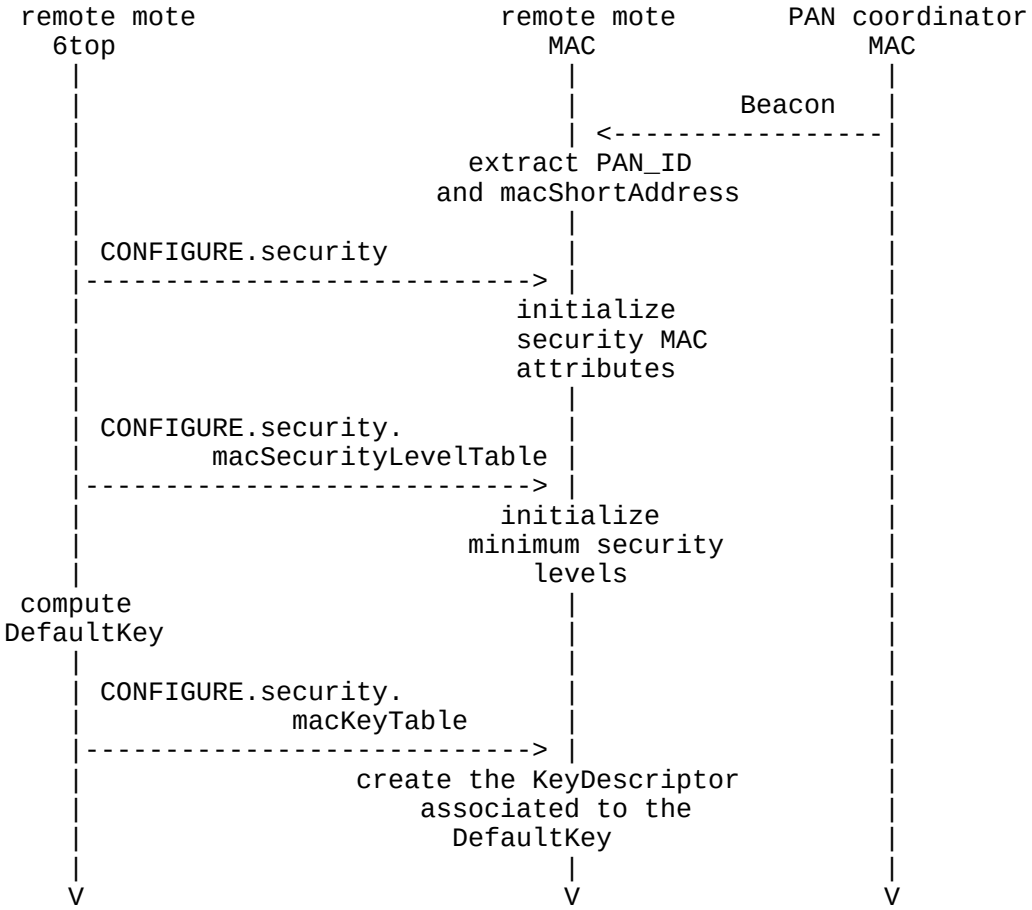



Figure 8. Bootstrap Phase for the remote mote in an enabled-beacon network.

### 6.2.3 Bootstrap phase for a mote in a not-Beacon-enabled network

In the case the not-beacon-enabled scheme is enabled, the mote must explicitly requests its generation to the coordinator. The payload of the Beacon Request packet must be protected using an ephemeral key, phi_k, obtained from the MasterKey, M_k, and the source address of the device, srcMACaddrMote, as

$$phi\_k = H\_128(srcMACaddrMote \mid M\_K).$$

The KeyIdMode of the Beacon Request packet is set to 00, thus enabling the PAN coordinator to implicitly obtain the ephemeral key.

Once received the Beacon frame, the remote mote will execute all the steps described in Sec. 6.3.

The Bootstrap Phase for a remote mote in a not-beacon-enabled network has been summarized in Fig. 9.

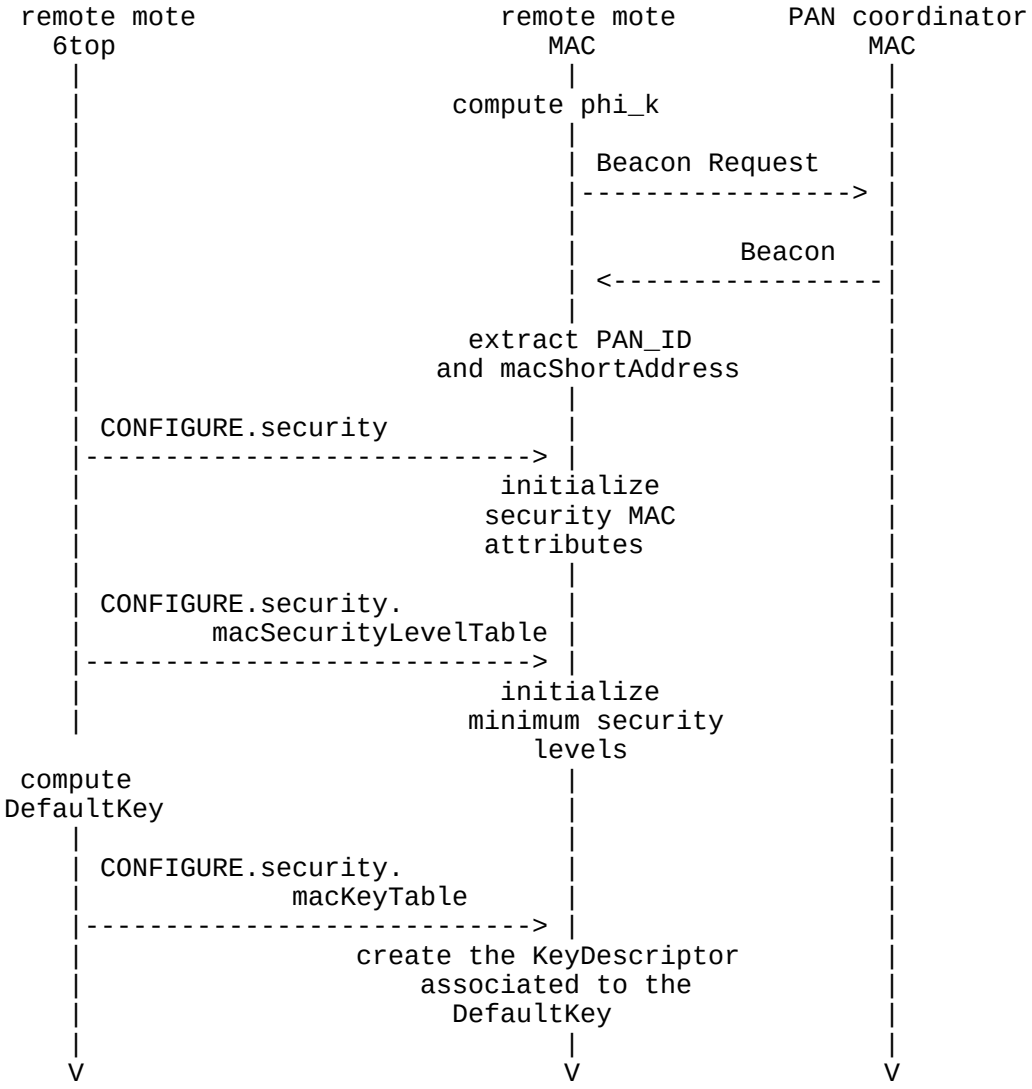



Figure 9. Bootstrap Phase for the remote mote in an not-enabled-beacon network.

## 6.3 Key Negotiation Phase

Since resource-constrained devices are unable to perform complex algorithms and protocols [Altolini2013][Riaz2009], a simple key agreement protocol is adopted during the execution of the key negotiation phase.

The KMP adopted by the secured framework presented in this draft is based on both DH algorithm [DH] and Station-To-Station protocol [StsProtocol].

Both anonymous and certified DH schemes are supported. The former one does not require that a mote will deliver its public key through a certificate and, for this reason, it doe snot support the node authentication. The latter one, instead, supports the use of certificates to authenticate the public key of each mote. With the anonymous DH scheme, a mote is able to deliver its public key by means of only one packet. When the certified DH scheme is used, instead, the public key will be delivered through multiple packets because of the high size of certificate (i.e., the key material generated by the mote needs to be fragmented).

To handle the Key Negotiation Phase, a new command MAC frame and two new 6top commands have been defined.

### 6.3.1 The new command MAC frame

A new command MAC frame, which is identified with a CommandFrameIdentifier set to 0xAA, has been introduced.

It is composed by four different fields: KeyGenControlField, Rand, KeyMaterial, and AuthenticationField.

The structure of the new command MAC frame has been reported in Fig. 10. The structure of the KeyGenControlField, instead, are shown in Fig. 11. The introduction of these new fields respects the constraints imposed by the standard about the maximum packet size.

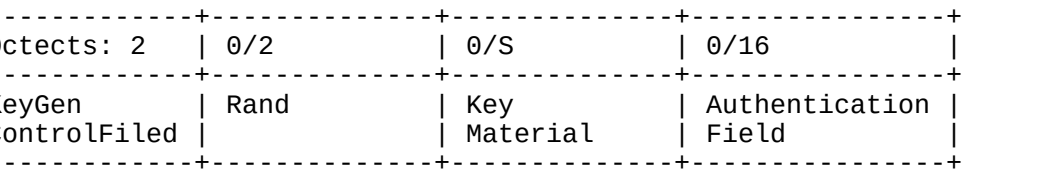



Figure 10. A new command MAC frame adopted during the key negotiation phase.

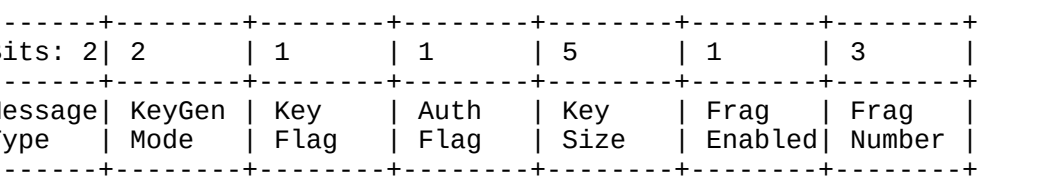



Figure 11. KeyGenControlField of the new command MAC frame adopted during the key negotiation phase.

The KeyGenControlField (2 bytes long) stores details about the content of the message. It is composed by the following fields:

- the MessageType (2 bits long), which identifies the type of message exchanged during the procedure. It may assume these values:

  - MessageType=00 identifies a message storing key materials (i.e., DH parameters).

  - MessageType=01 identifies final messages belonging to the Key Negotiation Phase that are used to verify the mutual authentication of nodes.

  - MessageType=10 and MessageType=11 are reserved for future upgrades.

- the KeyGenMode (2 bits long), which describes the algorithm adopted for key generation. It is set to 00 and 01 when the key is computed through the anonymous DH and the certified DH algorithms, respectively. Other values, i.e., 10 and 11, are reserved and can be used for future upgrades.

- the boolean KeyFlag (1 bit long), which is set to TRUE in the case the message delivers key materials or to FALSE otherwise.

- the boolean AuthFlag (1 bit long) ), which is set to TRUE in the case the message delivers an authentication field or to FALSE otherwise.

- the KeySize (5 bits long), which indicates the size of the transported key material, expressed in bytes. Its value is set to 0 in the case the message does not contain any key materials.

- the boolean FragEnabled (1 bit long), which is set to TRUE when the key material contains a fragment of the certificate storing the public key of a mote;

- the FragNumber (3 bit long), which indicates the fragment number associated to the key material field.

The Rand field (0/2 bytes long) contains a random value used for generating the PreLinkKey, P_k, and for verifying the authenticity of the remote device. It is present only if MessageType is equal to 00 or 01.

The KeyMaterial field (0/S bytes long, where S is the size of the prime number) contains key materials, such as DH parameters. It is present only if MessageType is equal to 00 or 01.

AuthenticationField field (0/16 bytes long) is used to verify the authenticity of the remote device. It is present only if MessageType is equal to 10.

### 6.3.2 New 6top commands

The following 6top commands have been designed to perform the KMP:

- CONFIGURE.security.startKMP: it is used to send the initial key material that will be exploited by the DH protocol to generate the key. The command requires:
  - KeyGenMode, which represents the algorithm adopted to negotiate the key;
  - KeyMaterial, which represent the public key or a certificate storing the public key;
  - Rand, which is a random number exploited during the mutual authentication process.
- CONFIGURE.security.completeKMP: it is used to complete the KMP by handling the mutual authentication between motes according to the Station-To-Station protocol. The command requires:
  - KeyGenMode, which represents the algorithm adopted to negotiate the key;

- AuthenticationField, which is used to verify the mutual authentication.

### 6.3.3 KMP implementation when the anonymous DH scheme is used

The KPM is initialized by the 6top adaptation layer of the remote device connected to the coordinator, that has already completed the joining procedure and that wants to establish a secured link with the PAN coordinator.

The procedure assumes that both motes store into the PrimeNumbersTable the same set of N prime numbers and their primitive roots, each one having size equal to S (see Sec. 6.1 for more details).

The number of bits needed to identify each prime number of the PrimeNumbersTable, i.e., P_bits, is equal to

$$P\_bits = \log2 (N).$$

We note that the security level of the proposed approach does not depend from P_bits, but it is only influenced by the length of each prime number, S. See Appendix A.1 for more details.

The KMP is performed through the execution of these operations:

a) The 6top adaptation layer of the remote mote performs the following steps:

a.1) a prime number, P, and the corresponding primitive root, g, are identified in the PrimeNumbersTableby by extracting the latest P_bits bits from the output of the following hash function:

$$H\_128(PAN\_ID \mid D\_k).$$

a.2) two random numbers are generated: the former one represents its private key, i.e., PVK_remoteMote; the latter one is used for the mutual authentication, i.e., RAND_remoteMote.

a.3) the public key, i.e., PBK_remoteMote, is generated according to the DH algorithm:

$$PBK\_remoteMote = g^{PVK\_remoteMote} * \bmod P$$

a.4) A CONFIGURE.security.startKMP command is released and delivered to the MAC layer. KeyGenMode, KeyMaterial, and Rand parameters of this command are ser to 00, PBK_remoteMote, and RAND_remoteMote, respectively.

b) Triggered by the CONFIGURE.security.startKMP command, the MAC layer of the remote mote generates a command MAC frame, which is composed by the following fields: MessageType=00, KeyGenMode=00, KeyFalg=TRUE, AuthFlag=FALSE, KeySize=S, Rand=RAND_remoteMote, and KeyMaterial=PBK_remoteMote. In the case the Fully Secured Configuration is enabled, this message is encrypted with the DefaultKey, D_k. Otherwise it is sent in clear.

c) The 6top adaptation layer of the PAN coordinator performs the following steps:

c.1) a prime number, P, and the corresponding primitive root, g, are identified in the PrimeNumbersTableby by extracting the latest P_bits bits from the output of the following hash function:

H_128(PAN_ID | D_k).

c.2) two random numbers are generated: the former one represents its private key, i.e., PVK_coordinator; the latter one is used for the mutual authentication, i.e., RAND_coordinator.

c.3) the public key, i.e., PBK_coordinator, is generated according to the DH algorithm:

PBK_coordinator = g^PVK_coordinator * mod P

c.4) A CONFIGURE.security.startKMP command is released and delivered to the MAC layer. KeyGenMode, KeyMaterial, and Rand parameters of this command are ser to 00, PBK_coordinator, and RAND_coordinator, respectively.

d) Triggered by the CONFIGURE.security.startKMP command, the MAC layer of the remote mote generates a command MAC frame, which is composed by the following fields: MessageType=00, KeyGenMode=00, KeyFalg=TRUE, AuthFlag=FALSE, KeySize=S, Rand=RAND_coordinator, and KeyMaterial=PBK_coordinator. In the case the Fully Secured Configuration is enabled, this message is encrypted with the DefaultKey, D_k. Otherwise it is sent in clear.

e) The 6top adaptation layer of the remote mote computes the PreLinkKey, P_k

P_k = PBK_coordinator^PVK_remoteMote * mod P.

f) The 6top adaptation layer of the PAN coordinator computes the PreLinkKey, P_k,

P_k = PBK_remoteMote^PVK_coordinator * mod P.

g) Both remote motes and PAN coordinator compute the LinkKey by using the procedure described in Sec. 6.3.5.

h) The 6top adaptation layer of the remote mote computes the authentication parameter, AUTH_remoteMote, through the 128-bit hash function, H_128(.), as in the sequel

AUTH_remoteMote=H_128(P_k || RAND_coordinator || RAND_remoteMote).

Then, it releases a CONFIGURE.security.completeKMP command with KeyGenMode and AuthenticationField set to 00 and AUTH_remoteMote, respectively.

i) The MAC layer of the remote mote sends to the coordinator a new MAC command message to complete the mutual authentication. This message is composed by the following fields: MessageType=10, KeyGenMode=00, KeyFalg=FALSE, AuthFlag=TRUE, KeySize=0, and AuthenticationField=AUTH_remoteMote. This message is protected by using the LinkKey computed before.

j) The 6top adaptation layer of the PAN coordinator verifies the validity of the received AUTH_remoteMote parameter. In affirmative case, it computes the authentication parameter, AUTH_coordinator, through the 128-bit hash function, H_128(.), as in the sequel:

AUTH_coordinator=H_128(P_k || RAND_remoteMote || RAND_coordinator).

Then, it releases a CONFIGURE.security.completeKMP command with KeyGenMode and AuthenticationField set to 00 and AUTH_coordinator, respectively.

k) The MAC layer of the PAN coordinator sends to the remote mote a new MAC command message to complete the mutual authentication. This message is composed by the following fields: MessageType=10, KeyGenMode=00, KeyFalg=FALSE, AuthFlag=TRUE, KeySize=0, and AuthenticationField=AUTH_coordinator. This message is protected by using the LinkKey computed before.

l) The remote motes verifies the validity of the received AUTH_coordinator parameter.

m) PAN coordinator and remote mote update MAC security attributes according to procedures described in Sec. 6.3.6 and Sec 6.3.7, respectively.

The aforediscussed KMP has been summarized in Fig. 12.

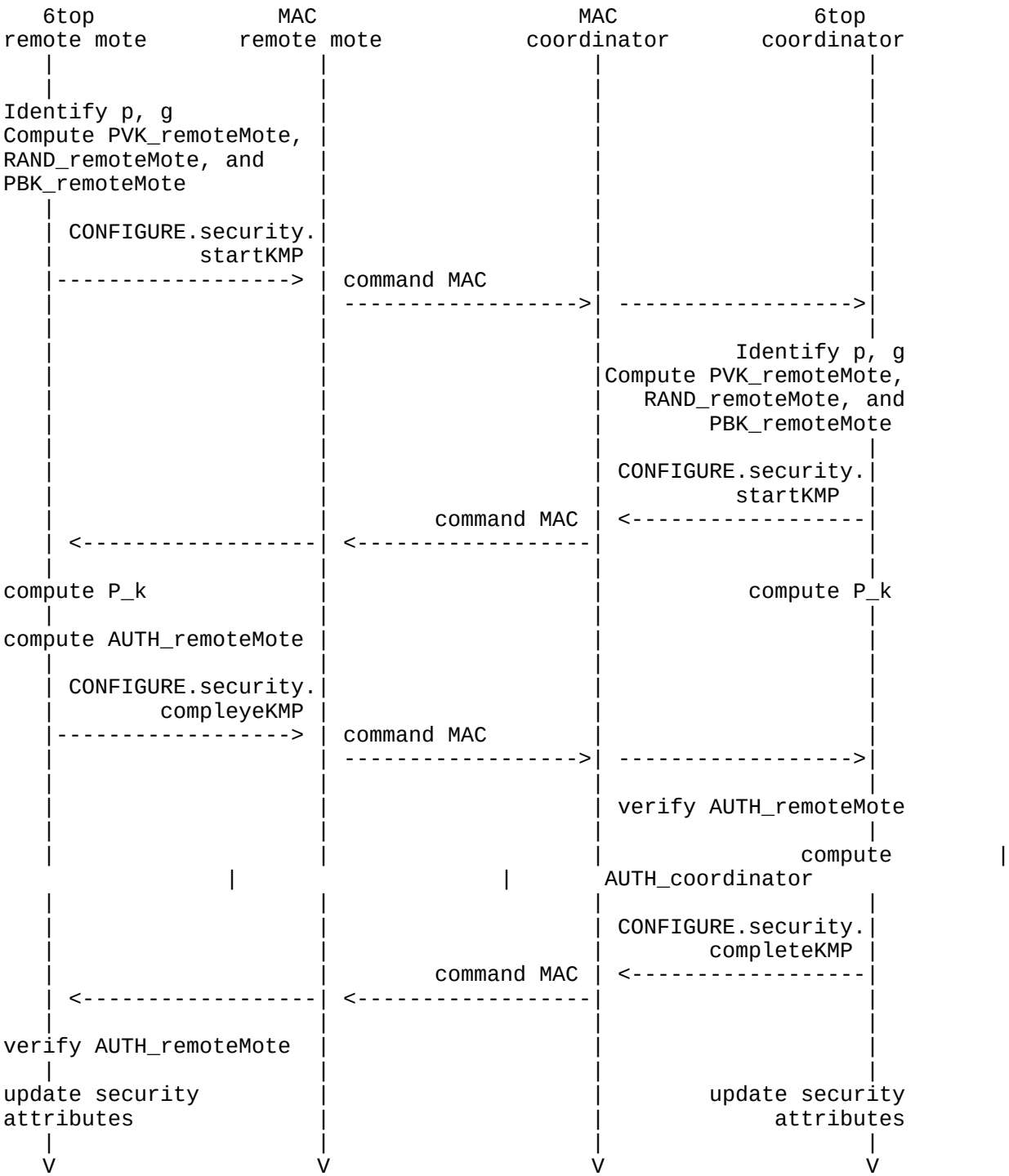

6top
remote mote
MAC
remote mote
MAC
coordinator
6top
coordinator
Identify p, g
Compute PVK_remoteMote,
RAND_remoteMote, and
PBK_remoteMote
CONFIGURE.security.
startKMP
command MAC
Identify p, g
Compute PVK_remoteMote,
RAND_remoteMote, and
PBK_remoteMote
CONFIGURE.security.
startKMP
command MAC
compute P_k
compute P_k
compute AUTH_remoteMote
CONFIGURE.security.
compleyeKMP
command MAC
verify AUTH_remoteMote
compute
AUTH_coordinator
CONFIGURE.security.
completeKMP
command MAC
verify AUTH_remoteMote
update security
attributes
update security
attributes

Figure 12. KMP implementation when the anonymous DH scheme is used

### 6.3.4 KMP implementation when the certified DH scheme is used

When the certified DH scheme is used, is is assumed that all devices have its certificate in which is stored its public key.

Let CERT_remoteMote, CERT_coordinator, PBK_remoteMote, PBK_coordinator be the certificate of the remote mote, the certificate of the coordinator, the public key of the remote mote, and the public key of the coordinator, respectively.

The KMP is performed through the execution of these operations:

a) The 6top adaptation layer of the remote mote extracts a random number, RAND_remoteMote, that will be used to complete the mutual authentication procedure. Then, it releases a CONFIGURE.security.startKMP command and delivers it to the MAC layer. KeyGenMode, KeyMaterial, and Rand parameters of this command are ser to 00, CERT_remoteMote, and RAND_remoteMote, respectively.

b) Triggered by the CONFIGURE.security.startKMP command, the MAC layer generates Z fragments of the certificate, with size T. For each i-th fragment, it generates a command MAC frame, which is composed by the following fields: MessageType=00, KeyGenMode=01, KeyFalg=TRUE, AuthFlag=FALSE, KeySize=T,, FlagEnabled=TRUE, FragNumber=i, Rand=RAND_remoteMote, and KeyMaterial=fragment_i. In the case the Fully Secured Configuration is enabled, this message is encrypted with the DefaultKey, D_k. Otherwise it is sent in clear.

c) The 6top adaptation layer of the PAN coordinator extracts a random number, RAND_coordinator, that will be used to complete the mutual authentication procedure. Then, it releases a CONFIGURE.security.startKMP command and delivers it to the MAC layer. KeyGenMode, KeyMaterial, and Rand parameters of this command are ser to 00, CERT_coordinator, and RAND_coordinator, respectively.

d) Triggered by the CONFIGURE.security.startKMP command, the MAC layer generates Z fragments of the certificate, with size T. For each i-th fragment, it generates a command MAC frame, which is composed by the following fields: MessageType=00, KeyGenMode=01, KeyFalg=TRUE, AuthFlag=FALSE, KeySize=T, FlagEnabled=TRUE, FragNumber=i, Rand=RAND_coordinator, and KeyMaterial=fragment_i.

In the case the Fully Secured Configuration is enabled, this message is encrypted with the DefaultKey, D_k. Otherwise it is sent in clear.

e) The 6top adaptation layer of the remote mote extracts the public key of the PAN coordinator from its certificate and computes the PreLinkKey, P_k

P_k = PBK_coordinator^PVK_remoteMote * mod P.

f) The 6top adaptation layer of the PAN coordinator extracts the public key of the remote mote and computes the PreLinkKey, P_k,

P_k = PBK_remoteMote^PVK_coordinator * mod P.

g) Both remote motes and PAN coordinator compute the LinkKey by using the procedure described in Sec. 6.3.4.

h) The 6top adaptation layer of the remote mote computes the authentication parameter, AUTH_remoteMote, by following these steps:

h.1) generate an authenticator message:
authMsg_remoteMote = P_k || RAND_coordinator || RAND_remoteMote

h.2) sign with the private key the 128-hash function of the authenticator message:

sign=S(PVK_remoteMote, H_128(authMsg_remoteMote))

h.3) obtain AUTH_remoteMote by encrypting with the preLinkKey, P_k, the sign computed before:

AUTH_remoteMote=E(P_k, sign)

Then, it releases a CONFIGURE.security.completeKMP command with KeyGenMode and AuthenticationField set to 00 and AUTH_remoteMote, respectively.

i) The MAC layer of the remote mote sends to the coordinator a new MAC command message to complete the mutual authentication. This message is composed by the following fields: MessageType=10, KeyGenMode=01, KeyFalg=FALSE, AuthFlag=TRUE, KeySize=0, and AuthenticationField=AUTH_remoteMote. This message is protected by using the LinkKey computed before.

j) The 6top adaptation layer of the PAN coordinator verifies the

validity of the received AUTH_remoteMote parameter. In affirmative case, it computes the authentication parameter, AUTH_coordinator, by following these steps:

j.1) generate an authenticator message:

authMsg = P_k || RAND_remoteMote || RAND_coordinator

j.2) sign with the private key the 128-hash function of the authenticator message:

sign=S(PVK_remoteMote, H_128(authMsg))

j.3) obtain AUTH_remoteMote by encrypting with the preLinkKey, P_k, the sign computed before:

AUTH_coordinator=E(P_k, sign)

Then, it releases a CONFIGURE.security.completeKMP command with KeyGenMode and AuthenticationField set to 00 and AUTH_coordinator, respectively.

k) The MAC layer of the PAN coordinator sends to the remote mote a new MAC command message to complete the mutual authentication. This message is composed by the following fields: MessageType=10, KeyGenMode=01, KeyFalg=FALSE, AuthFlag=TRUE, KeySize=0, and AuthenticationField=AUTH_coordinator. This message is protected by using the LinkKey computed before.

l) The remote motes verifies the validity of the received AUTH_coordinator parameter.

m) PAN coordinator and remote mote update MAC security attributes according to procedures described in Sec. 6.3.6 and Sec 6.3.7, respectively.

The aforediscussed KMP has been summarized in Fig. 13.

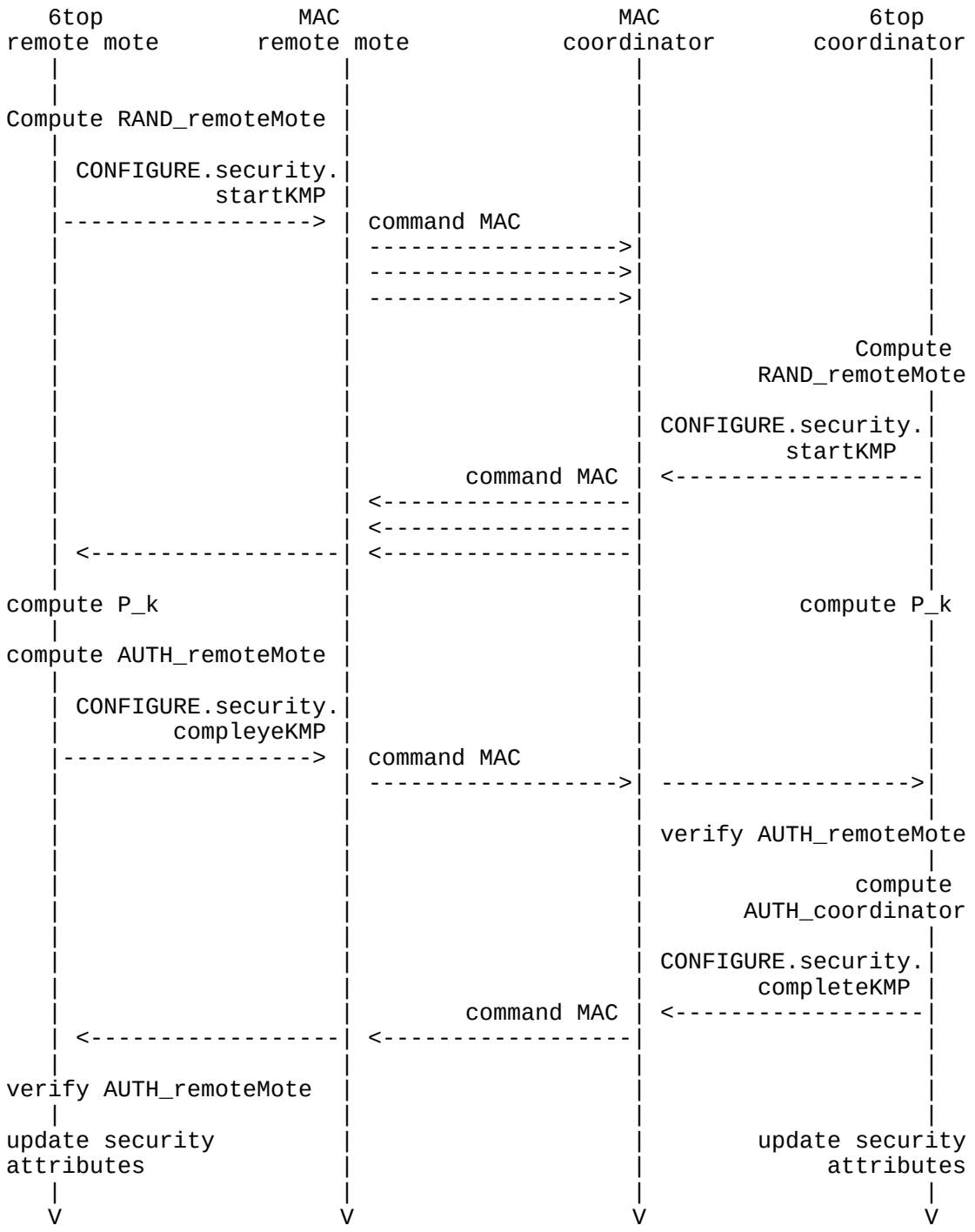

6top
remote mote
MAC
remote mote
MAC
coordinator
6top
coordinator
Compute RAND_remoteMote
CONFIGURE.security.
startKMP
command MAC
Compute
RAND_remoteMote
CONFIGURE.security.
startKMP
command MAC
compute P_k
compute P_k
compute AUTH_remoteMote
CONFIGURE.security.
compleyeKMP
command MAC
verify AUTH_remoteMote
compute
AUTH_coordinator
CONFIGURE.security.
completeKMP
command MAC
verify AUTH_remoteMote
update security
attributes
update security
attributes

Figure 13. KMP implementation when the certified DH scheme is used

### 6.3.5 Generation of the LinkKey

The standard imposes to use the CCM* algorithm and a 128-bit key to protect MAC frames. At the same time, the CCM* algorithm assumes that each key must be used for a specific number of block ciphers [IEEE802154].

For each i-th group of block ciphers, the LinkKey, L_k, is computed as in the following:

$$L_k = H_{128} (i \mid PAN\_ID \mid P_k).$$

### 6.3.6 Update of MAC security attributes for the PAN coordinator after the generation of the LinkKey

After the calculation of the i-th LinkKey, the 6top adaptation layer of the PAN coordinator updates its MAC security attributes as described in what follows.

a) If i=1, a new DeviceDescriptor element, associated to the remote mote with which it has negotiated a link key, is created. It is composed by:

a.1) the PANId, which is set to the PAN_ID value.

a.2) The ShortAddress, which is set to the MAC address of the RFD node whenever the short addressing mode is used. This parameter is set to 0xfffe if only the extended addressing mode is used. In the case its value is unknown, this parameter is set to 0xfff.

a.3) The ExtAddress, which is set to the IEEE MAC address of the RFD node.

a.4) The FrameCounter, which is set to the FrameCounter value extracted from the latest packet received by the RFD node.

a.5) The Extempt boolean flag, which is set to the allowed value of the DeviceOverriddeSecurityMinimum variable described in Fig. 2.

b) A new KeyIdLookupList data structure is created. A KeyIdLookupDescriptor is generated and stored into the KeyIdLookupList data structure. The KeyIdMode, the KeySource, and the KeyIndex variables of this KeyIdLookupDescriptor are set to 0x03, the MAC address of the remote mote that initialized the Key Negotiation Phase, and 1, respectively. DeviceAddrMode, DevicePANId, and DeviceAddress are not set because of the selected KeyIdMode (see Tab. 65 of the IEEE 802.15.4 standard for more details [IEEE802154]).

c) A KeyUsageList data structure is created and stored within the KeyDescriptor element. One KeyUsageDescriptor associated to data MAC frames is created and stored into the KeyUsageList data structure.

d) A DeviceDescriptorHandleList is created and populated with the pointer to the DeviceDescriptor created at the point a).

Then, the 6top layer delivers the LinkKey, the KeyIdLookupList, the KeyUsageList, and the DeviceDescriptorHandleList to the MAC layer by using the CONFIGURE.security.macKeyTable command (as discussed in [I-D.wang-6tisch-6top], in Sec. 2.4.9.2).

Triggered by the CONFIGURE.security.macKeyTable command, the MAC layer will create a KeyDescriptor associated to the LinkKey, L_k,in which storing all the parameters received by the 6top layer, and and will store it within the macKeyTable.

### 6.3.7 Update of MAC security attributes for the remote mote after the generation of the LinkKey

After the calculation of the i-th LinkKey, the 6top adaptation layer of the remote mote updates its MAC security attributes as described in what follows.

a) A new KeyIdLookupList data structure is created. A KeyIdLookupDescriptor is generated and stored into the KeyIdLookupList data structure. The KeyIdMode, the KeySource, and the KeyIndex variables of this KeyIdLookupDescriptor are set to 0x03, the MAC address of the PAN coordinator that initialized the Key Negotiation Phase, and 1, respectively. DeviceAddrMode, DevicePANId, and DeviceAddress are not set because of the selected KeyIdMode (see Tab. 65 of the IEEE 802.15.4 standard for more details [IEEE802154]).

b) A KeyUsageList data structure is created and stored within the

KeyDescriptor element. One KeyUsageDescriptor associated to data MAC frames is created and stored into the KeyUsageList data structure.

c) A DeviceDescriptorHandleList is created and populated with the pointer to the DeviceDescriptor associated to the PAN coordinator and created during the Bootstrap Phase.

Then, the 6top layer delivers the LinkKey, the KeyIdLookupList, the KeyUsageList, and the DeviceDescriptorHandleList to the MAC layer by using the CONFIGURE.security.macKeyTable command (as discussed in [I-D.wang-6tisch-6top], in Sec. 2.4.9.2).

Triggered by the CONFIGURE.security.macKeyTable command, the MAC layer will create a KeyDescriptor associated to the LinkKey, L_k,in which storing all the parameters received by the 6top layer, and and will store it within the macKeyTable.

## 7 Additional features

There is the possibility to switch from the Flexible Secured to the Hybrid Secure configuration.

To this aim, during the join process, a mote without security capabilities sends to the PAN coordinator a Beacon Request message with the SecurityEnabled flag set to FALSE.

The PAN coordinator, if properly configures, switches to the Hybrid Secure configuration and update all the MAC security attributes accordingly.

From this moment on, the coordinator will send broadcast messages in clear.

## 8 Security Considerations

There are no security considerations for this document.

## 9 IANA Considerations

There is no IANA action required for this document.

## 10 References

### 10.1 Normative References

[I-D.watteyne-6tisch-tsch] Watteyne, T., "Using IEEE802.15.4e TSCH in an LLN context: Overview, Problem Statement and Goals", Internet-Draft draft-watteyne-6tisch-tsch-00, (work in progress) October 2013.

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

[I-D.draft-palattella-6tisch-terminology] Palattella, MR., Ed., Thubert, P., Watteyne, T., and Q. Wang, "Terminology in IPv6 over Time Slotted Channel Hopping". Internet Draft draft-palattella-6tisch-terminology-00, (work in progress) October 2013.

[DH] W. Diffie and M. Hellman, "New directions in cryptography," IEEE Trans. Inf. Theor. 22, 6 Sep., 2006.

[StsProtocol] Whitfield Diffie, Paul C. van Oorschot and Michael J, "Wiener, Authentication and authenticated key exchange", Designs, Codes, and Cryptography, 1987.

### 10.2 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.

[ZIGBEEIP] ZigBee Public Document 15-002r00, "ZigBee IP Specification", 2013.

[Camtepe2005] Seyit A. Camtepe and Bulent Yener, "Key Distribution Mechanisms for Wireless Sensor Networks: a Survey", Technical Report 2005.

[Walters07] John Paul Walters, Zhengqiang Liang, Weisong Shi, and Vipin Chaudhary, "Wireless sensor network security: A survey," in book chapter of Security", Proc. of Distributed, Grid, and Pervasive Computing, CRC Press, 2007.

[Wang2006] Yong Wang, Garhan Attebury, and Byrav Ramamurthy, "A survey of security issues in wireless sensor networks", IEEE Communications Surveys & Tutorials, 2006

[Cayirci2007] Security in Wireless Ad Hoc and Sensor Networks. John Wiley & Sons, 2007.

[I-D.roll-security-framework] Tzeta Tsao, Roger Alexander, Mischa Dohler, Vanesa Daza, and Angel Lozano, "A Security Framework for Routing over Low Power and Lossy Networks", Internet Draft draft-ietf-roll-security-framework-07, Jan 2013.

[I-D.garcia-core-security] O. Garcia-Morchon, S. Keoh, S. Kumar, R. Hummen, and R. Struik, "Security Considerations in the IP-based Internet of Things," IETF, Internet Draft, Sep. 2013.

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

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

[HIPDEX] Moskowitz, R., "HIP Diet EXchange (DEX)", draft-moskowitzhip-rg-dex-06 (work in progress), May 2012.

[PalattellaSurvey] Maria Rita Palattella, Nicola Accettura, Xavier Vilajosana, Thomas Watteyne, Luigi Alfredo Grieco, Gennaro Boggia, and Mischa Dohler," Standardized Protocol Stack For The Internet Of (Important) Things", IEEE Communications Surveys & Tutorials, December, 2012

[StallingsSecurityBooks] William Stallings: Cryptography and network

security - principles and practice. Prentice Hall 2010.

[Becher2006] Alexander Becher, Zinaida Benenson, and Maximillian Dornseif, "Tampering with motes: real-world physical attacks on wireless sensor networks", In Proc. of conf. on Security in Pervasive Computing (SPC), Berlin, 2006

[TELOSB] "Crossbow Technology, TelosB Datasheet." [Online]. Available: http://www.willow.co.uk/TelosB_Datasheet.pdf

[Riaz2009] Riaz, R.; Ki-Hyung Kim; Ahmed, H.F., "Security analysis survey and framework design for IP connected LoWPANs," Autonomous Decentralized Systems, 2009. ISADS '09. International Symposium on , vol., no., pp.1,6, 23-25 March 2009

[Altolini2013] Altolini, D.; Lakkundi, V.; Bui, N.; Tapparello, C.; Rossi, M., "Low power link layer security for IoT: Implementation and performance analysis," Wireless Communications and Mobile Computing Conference (IWCMC), 2013 9th International , vol., no., pp.919,925, 1-5 July 2013

[Watteyne2012] Thomas Watteyne, Xavier Vilajosana, Branko Kerkez, Fabien Chraim, Kevin Weekly, Qin Wang, Steven D. Glaser, Kris Pister: OpenWSN: a standards-based low-power wireless development environment. Trans. Emerging Telecommunications Technologies 23(5): 480-493 (2012)

## Appendix A. DH protocol

### A.1 Security considerations about the DH protocol

As discussed in Sec. 6.5.5, the CCM* transformation requires a 128-bit key.

According to the DH algorithm, and considering properties of the modular arithmetic [DH], the length of both the public key of a mote and the prime number used for its generation must be at least equal to 128 bits.

The security level of the proposed approach does not depend from the number of prime numbers stored into the PrimeNumberTable (because these numbers are not the keys), but it coincides with the security level of the DH protocol. hence, it is strictly related to length of prime numbers, S.

In particular, the total number of keys that a mote can use during the Key Negotiation Phase are equal to 2^S. Supposing to have S >= 128, the total number of keys is higher than 3*10^38. This should guarantee a very high resilience to any kind of brute force attack.

Authors' Addresses

G. Piro
DEI, Dep. of Electrical and Information Engineering
Politecnico di Bari
Via Orabona 4, 70125, Bari, ITALY
Phone: +39 0805963301

Email: g.piro@poliba.it

G. Boggia
DEI, Dep. of Electrical and Information Engineering
Politecnico di Bari
Via Orabona 4, 70125, Bari, ITALY
Phone: +39 0805963913

Email: g.boggia@poliba.it

L.A. Grieco
DEI, Dep. of Electrical and Information Engineering
Politecnico di Bari
Via Orabona 4, 70125, Bari, ITALY
Phone: +39 0805963911

Email: a.grieco@poliba.it

Network Working Group S. Shah
Internet-Draft P. Thubert
Intended status: Informational Cisco Systems
Expires: July 12, 2014 January 08, 2014

# Deterministic Forwarding PHB
# draft-svshah-tsvwg-deterministic-forwarding-00

## Abstract

This document defines a Differentiated Services Per-Hop-Behavior (PHB) Group called Deterministic Forwarding (DF). The document describes the purpose and semantics of this PHB. It also describes creation and forwarding treatment of the service class. The document also describes how the code-point can be mapped into one of the aggregated Diffserv service classes [RFC5127].

## Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on July 12, 2014.

## Copyright Notice

Table of Contents

## 1. Introduction

IP Networks typically implement Diffserv to provide differentiated forwarding behavior to different class of traffic. Networks that implement Diffserv relies on DSCP code-point in the IP header of a packet to select PHB as a specific forwarding treatment for that packet [RFC2474, RFC2475]. This document describes a particular PHB called Deterministic Forwarding (DF). The proposed new code-point defines a service class for the purpose of forwarding treatment of a packet at determined/fixed scheduled time providing no jitter service to the class of traffic (updates RFC4594 with the addition of a new Service Class).

DF PHB can be used for the network services that require the capability to ensure a predictable interaction between networked systems and guarantee a very strict time scheduled services. Applications of such networks may be able to absorb a loss but are very sensitive to end to end latency and jitter. Examples of such networks include Machine to Machine (M2M) control and monitoring deployment with IP over varieties of Layer 2 networks.

The definition of Expedited Forwarding (EF) [RFC2598] PHB is low latency and thus one can envision use of EF code-point for such service. However, even though EF defines low latency and low jitter, it does not guarantee deterministic/fixed scheduled time service. Depending on co-existence of the other traffic in the network, EF traffic may have more or less variance on jitter and thus not suitable for the deterministic service. DF PHB thus is more suitable for deterministic time sensitive traffic.

Typically for an application where end to end deterministic service is important, relevant traffic should be provisioned through DF PHB at every hop in that end to end path. However, in cases where intermediate hops (or DS domains) either do not support DF PHB or supports only aggregated service classes described in RFC5127, DF traffic in those DS domains MUST be mapped to Real Time Treatment class (EF PHB) defined in RFC5127. Traffic in such scenario MUST be conditioned at the Edge before entering and after exiting such DS domains. This is described further in later section.

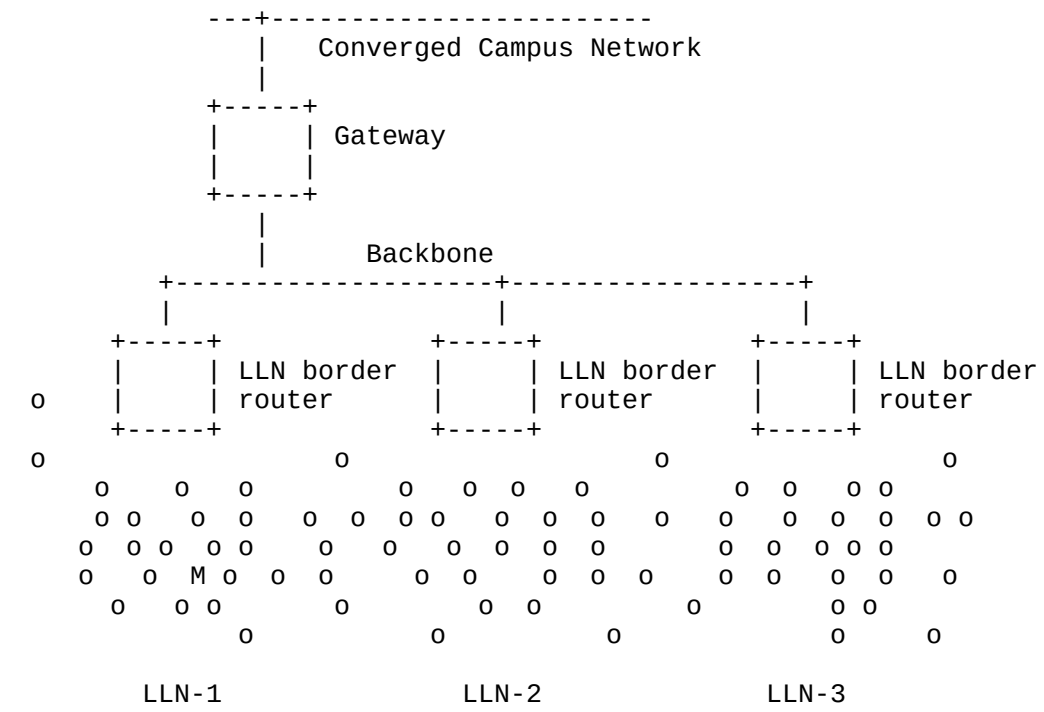
### 1.1. Use-cases

With an introduction of machine to machine networks over IP, a new set of applications are emerging. Traffic types from such applications/networks are some-what different from the traditional traffic types. Though most traffic types have characteristics similar to that of traditional ones [LLN-DIFF], certain control signals for some of the applications are extremely sensitive to

latency and jitter. Such control signals demand much stricter latency and jitter, at pretty much decisive time scheduled delivery, end to end. Industrial automation, Smart cities and automobiles/ planes/trains built around such networks are examples of such use-cases.

Machine to machine networks may be implemented on varieties of Layer 2 protocols. 802.3 and 802.15e [TiSCH] are examples of layer 2 that are enhancing their capabilities to allow time scheduled delivery of packets.

In a wireless sensor networks, that are implemented over IP, multiple LLN (Low power and Lossy Networks) may be connected through Backbone.




As shown in the diagram, multiple LL Networks are connected to each other via Backbone through LLN Border routers. Each LL Network consist of many nodes. There are different types of traffic forwarded through each LL node and from one LL Network to another. Most LLN traffic types have characteristics similar to that of traditional ones and thus can be supported through existing Diffserv classes except time sensitive control signals. Without segregating

such control signals to a specific Diffserv class would require Intserv support for LLN traffic in such networks. All traffic would be subject to flow classification to differentiate time sensitive control signals which can be a big scale concern. Supporting time sensitive control signals via newly proposed DF Diffserv class allows implementation of Diffserv in LLN Networks.

## 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.

## 3. DF code-point Behavior

The DF PHB is to implement time scheduled forwarding treatment. Provisioning of such a service has two parts,

1) Provisioning of the fixed/relative time for scheduling of such service
2) Provisioning of the max size of the data to be transmitted at each scheduled time

Provisioned scheduled time may be absolute or relative. For example, a DF class may be provisioned to schedule packets (or bytes) at every fixed time. Fixed time can be time of a day or any other absolute definition. In a multi hop forwarding of DF traffic, absolute time service provisioning at each hop may require to be dependent on the clock synchronization (clock synchronization is not in the scope of this specification). In relative time scheduling, packets to be scheduled at every specific interval or it could be relative to any other specific event/trigger. The definition of the time interval or any other event is relevant to that specific provisioned node only.

The size of the data, to be transmitted at each scheduled time service, provisioned can be in the unit of bytes or time. Once DF PHB is provisioned and enabled, forwarding treatment MUST service packets (bytes) from this class at the scheduled time for max allowable data. Scheduling MUST pre-empt any other service, including EF, during the schedule time service for the DF class. In order to avoid incurred latency to EF class of traffic, it is expected to carefully provision DF class to limit scheduled time service to as minimal data transmission that would prevent larger than expected delays to EF class of traffic.

Provisioning can be done via any of multiple possible methods. It could be via command interface, or could be via external provisioning agents, or could be via some sort of signaling that may dynamically pre-negotiate time window of transmission at each node in a network path.

### 3.1. Potential implementation of DF scheduling

Following are examples of potential implementations. They are not any form of guidelines or recommendations.

There are at least two ways to implement scheduling for DF traffic class.

1) One queue to buffer and schedule all DF traffic (from all flows),

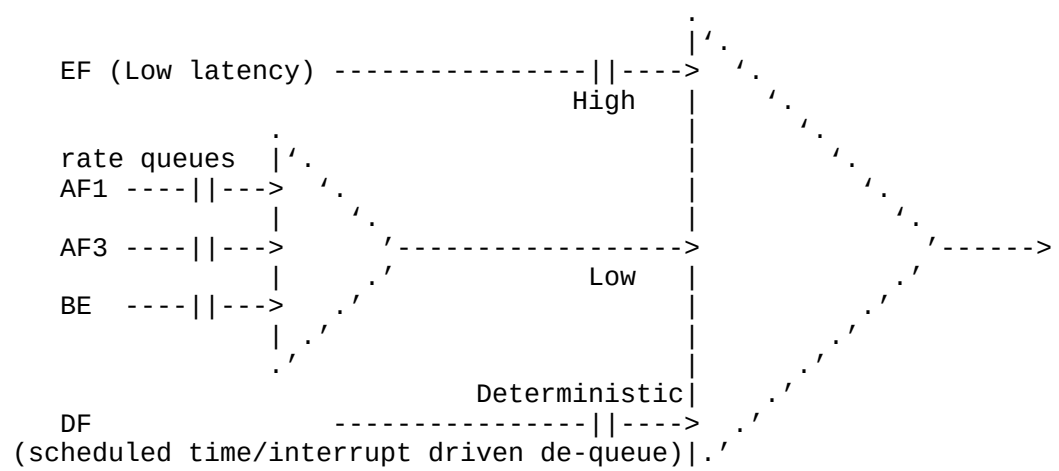
2) Multiple sub-queues for DF traffic class, one queue for each DF provisioned flow

Flow here represents macro definition, it does not have to be only 5-tuple.

Any chosen DF scheduling implementation MUST run traffic conditioning at enqueue to decide if packets to be enqueued or discarded. Discussed more in later section.

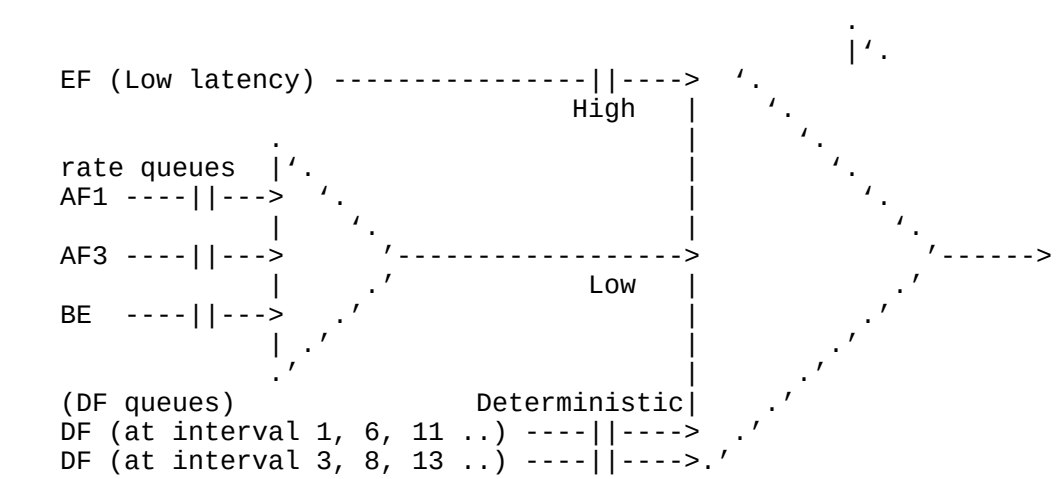
1) One data-plane queue to buffer all DF traffic

This one queue maintains, possibly a circular, indexed buffer list. Every time scheduled slot is an index in the buffer list. If enqueue conditioning decides not to discard a packet, packet gets en-queued at the relevant index in the buffer list in such a way that relevant index pointer, and thus buffered relevant packets for that index, at the head of the list is ready to be de-queued at next scheduled time. Subsequent buffer index is scheduled for the subsequent scheduled time slot and so forth. If a specific flow has not received any packet for a scheduled time then buffer index for that flow remains empty. A packet from other flows do not get buffered at that empty index. That means during dequeue, at a schedule time service, an empty index results in no packets to dequeue and thus nothing to be transmitted from the DF queue at that point in time.




2) multiple sub-queues for each DF flows

If enqueue conditioning decides not to discard a packet, packet gets enqueued in the relevant DF sub-queue designated for that flow. At a scheduled time slot, scheduler dequeues a packet from the respective sub-queue. Every scheduled time service interrupt is mapped to a specific DF sub-queue to dequeue a packet from.



### 3.2. Conditioning DF traffic at Enqueue

DF traffic MUST be conditioned at the enqueue. As per PHB definition, packets are required to be scheduled and delivered at a precise absolute or relative time interval. Any packet that has missed the window of its service time MUST be discarded. That would also mean any packet coming from the previous hop MUST be conditioned at the enqueue for validity of its scheduled service. For example if a DF queue is provisioned to serve a packet with less than x ms of jitter and for an arrived packet, if next scheduled time for a packet results in more than x ms of jitter then such packet MUST be discarded. The enqueued packet MUST also be checked against the size of the data. If size of the data to be enqueued in a DF queue is bigger than what scheduled time slot is provisioned for then such packet MUST be discarded.

## 4. Diffserv behavior through non-DF DS domains

In cases where DF traffic is forwarded through multiple DS domains, DS domains close to the source and receiver understand application's deterministic service requirement well and so MUST be provisioned for the precise time scheduled forwarding treatment. Intermediate DS domains MAY support DF PHB. Intermediate domains that can not support DF PHB, DF traffic from such domains SHOULD get EF treatment, as defined in RFC5127 for Real Time Service aggregation. Sender and Receiver DS domains, in such cases, MUST condition DF traffic at the respective Edge. If EF service through intermediate DS domains can have a predictable upper bound, receiver DS domain Edge can add a correction to an incurred latency/jitter with its own defined time interval for DF service.

## 5. Updates to RFC4594 and RFC5127

This specification updates RFC4594 with an addition of a new Diffserv Class. It also updates RFC5127 to aggregate DF class of traffic to Real Time Aggregation Class.

## 6. IANA Considerations

This document defines a new DSCP code-point DF. IANA maintains the list of existing DSCPs. Proposal is to allocate a new one for the DF code-point.

## 7. Security Considerations

There is no security considerations required besides ones already understood in the context of Differentiated services architecture

## 8. Acknowledgements

Fred Baker and Norm Finn.

## 9. References

### 9.1. Normative References

[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.

[RFC2475] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., and W. Weiss, "An Architecture for Differentiated Services", RFC 2475, December 1998.

[RFC2598] Jacobson, V., Nichols, K., and K. Poduri, "An Expedited Forwarding PHB", RFC 2598, June 1999.

[RFC5127] Chan, K., Babiarz, J., and F. Baker, "Aggregation of Diffserv Service Classes", RFC 5127, February 2008.

### 9.2. Informative References

[TiSCH] Thubert, P., Watteyne, T., and R. Assimiti, "An Architecture for IPv6 over the TSCH mode of IEEE 802.15.4e, I-D.draft-ietf-6tisch-architecture", Nov 2013.

[LLN-DIFF] Shah, S. and P. Thubert, "Differentiated Service Class Recommendations for LLN Traffic, I-D.draft-svshah-tsvwg-lln-diffserv-recommendations", Aug 2013.

Authors' Addresses

Shitanshu Shah
Cisco Systems
170 W. Tasman Drive
San Jose, CA 95134
US

Email: svshah@cisco.com

Pascal Thubert
Cisco Systems
Village d'Entreprises Green Side
400, Avenue de Roumanille
Batiment T3
Biot - Sophia Antipolis 06410
FRANCE

Email: pthubert@cisco.com

6TiSCH
Internet-Draft
Intended status: Informational
Expires: August 16, 2014

Q. Wang, Ed.
Univ. of Sci. and Tech. Beijing
X. Vilajosana
Universitat Oberta de Catalunya
T. Watteyne
Linear Technology
February 12, 2014

# 6TiSCH Operation Sublayer (6top) Interface
draft-wang-6tisch-6top-interface-02

## 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 future network management solutions defined by the 6TiSCH working group. This document also defines a list commands internal to the 6top sublayer.

## Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 16, 2014.

## 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. This data model can be used for future network management solutions defined by the 6TiSCH working group. This document also defines a list commands internal to the 6top sublayer. 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. 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. It includes monitoring and statistics collection, both of which are configurable through its management interface.

Protocol Stack

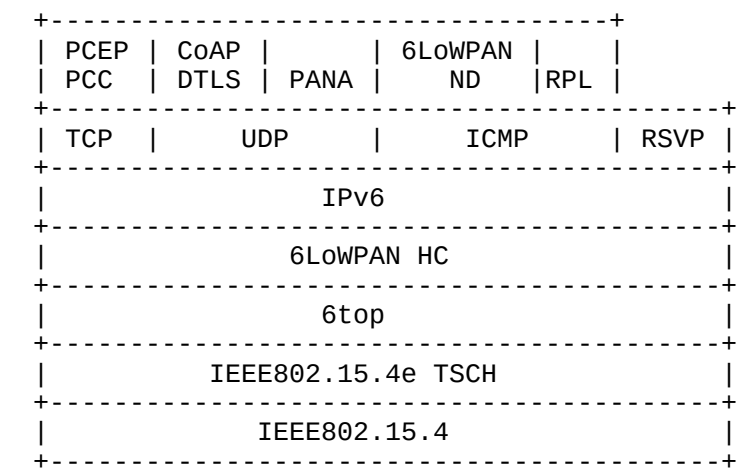



Figure 1

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 2.1.

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 2.2.

Section 3 contains a generic data model for the 6top sublayer, described in the YANG data modeling language.

The commands of the management and data interfaces are listed in Section 4. This set of commands is designed to support decentralized, centralized and hybrid scheduling solutions.

## 2.1. 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 ask a node manage a 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.

#### 2.1.1. hard cells

A hard cell is a cell that cannot be dynamically reallocated by 6top. A hard cell is uniquely identified by the following tuple:

slotframe ID: ID of the slotframe this cell is part of.

slotOffset: the slotOffset for the cell.

channelOffset: the channelOffset for the cell.

LinkOption bitmap: bitmap as defined in [IEEE802154].

CellType: MUST be set to 1.

#### 2.1.2. 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 "draft-wang-6tisch-6top-sublayer".

### 2.2. 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

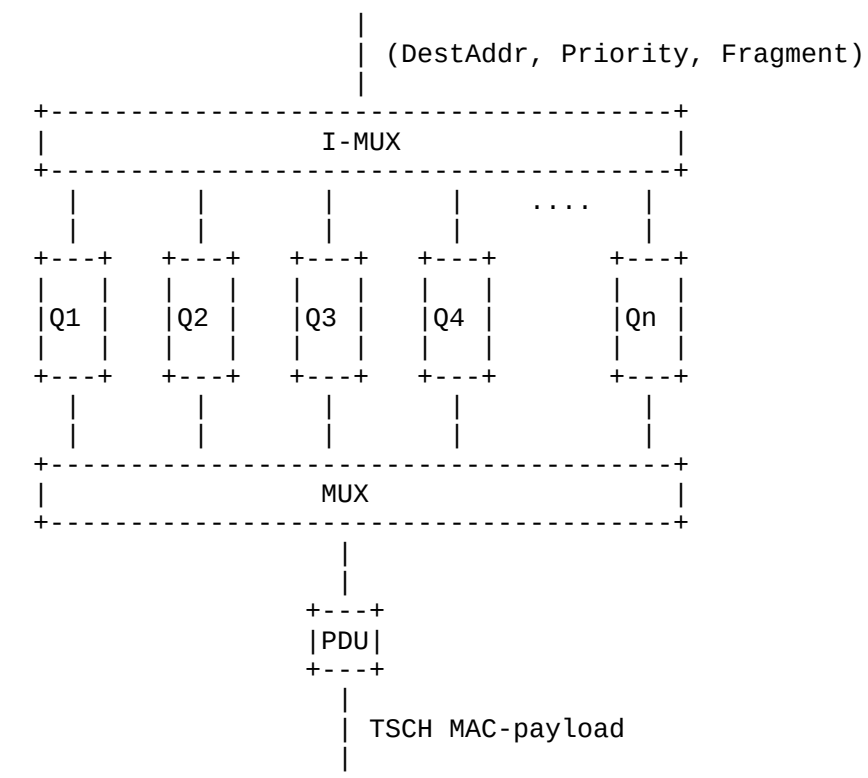



Figure 2

In Figure 2, Qi represents a queue, which is either broadcast or unicast, and is assigned a 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 through a Send.data command (Section 4), 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.

## 3. 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 dada model consists of three parts: 6top MIB, part of the [IEEE802154e] PIB, and part of the [IEEE802154] PIB.

### 3.1. YANG model of the 6top MIB

```
list CellList {
   key "CellID";
   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 uint8;
      description
      "Equal to SlotframeHandle defined in TSCH";
      reference
      "IEEE802154e";
   }
   leaf SlotOffset {
      type uint16;
      description
```

```
            "Defined in IEEE802154e.";
            reference
            "IEEE802154e";
         }
         leaf ChannelOffset {
            type uint8;
            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;
               }
               bit Reserved1 {
                  position 4;
               }
               bit Reserved2 {
                  position 5;
               }
               bit Reserved3 {
                  position 6;
               }
               bit Reserved4 {
                  position 7;
               }
            }
            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 TargetNodeAddress {
            type uint64;
            description
            "Defined by 6top, but being constrained by TSCH
            macNodeAddress size, 2-octets. If using TSCH as MAC,
            higher 6-octets should be filled with 0, and lowest
            2-octets is neighbor address";
         }
         leaf TrackID {
            type uint16;
            description
            "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.";
         }
         container Statistic {
            leaf NumOfStatistic {
               type uint8;
               description
               "Number of statistics collected on the cell";
            }
            list MeasureList {
               key "StatisticsMetricsID";
               leaf StatisticsMetricsID{
                  type uint16;
               }
               leaf StatisticsValue{
                  type uint16;
                  config false;
               }
            }
         }
      }
```

```
   list SlotframeList {
      key "SlotframeID";
      leaf SlotframeID {
         type uint8;
      }
      leaf NumOfSlots {
         type uint16;
      }
   }

      list MonitoringStatusList {
         key "MonitoringStatusID";
         leaf MonitoringStatusID {
            type uint16;
         }
         leaf SlotframeID {
            type uint8;
         }
         leaf TargetNodeAddress {
            type uint64;
         }
         leaf EnforcePolicy {
            type enumeration {
               enum DISABLE;
               enum BESTEFFORT;
               enum STRICT;
               enum OVERPROVISION;
            }
            description
            "Currently enforced QoS policy";
         }
         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;
            description
            "Overprovisioned cells. 0 if CONFIGURE.qos enforce 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 provisioned cells, i.e. the actual
            bandwidth without taking into account the overprovisioned
            cells.";
         }
      }

   list StatisticsMetricsList {
      key "StatisticsMetricsID";
      leaf StatisticsMetricsID {
         type uint16;
      }
      leaf SlotframeID {
         type uint16;
         description
         "ID of the slotframe.  If empty, monitors all slotframe IDs";
         reference
         "IEEE802154e";
      }
      leaf SlotOffset {
         type uint16;
         description
         "Specific slotOffset to be monitored.  If empty all timeslots
         are monitored";
         reference
         "IEEE802154e";
      }
      leaf ChannelOffset {
         type uint8;
         description
         "Specific channelOffset to be monitored.  If empty all
         channels are monitored";
         reference
         "IEEE802154e";
      }
      leaf TargetNodeAddress {
```

```
         type uint64;
         description
         "If empty, all neighbor nodes are monitored.";
      }
      leaf Metrics {
         type enumeration {
            enum PDR;
            enum ETX;
            enum RSSI;
            enum LQI;
         }
         description
         "The metric to be monitored.";
      }
      leaf Window {
         type uint16;
         description
         "measurement period, in Number of the slotframe size";
      }
      leaf Enable {
         type enumeration {
            enum DISABLE;
            enum ENABLE;
         }
      }
   }
```

```
      list EBList {
         key "EbID";
         leaf EbID {
            type uint8;
         }
         leaf CellID {
            type uint16;
            description
            "Equal to LinkHandle in IEEE802154e";
         }
         leaf Peroid {
            type uint16;
            description
            "The EBs period, in seconds";
         }
         leaf Expiration {
            type enumeration {
               enum NEVERSTOP;
               enum EXPIRATION;
            }
            description
            "Which Period to indicate when the EBs periodicity will
            stop. If Zero the period never stops.";
         }
         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 {
      leaf policy {
         type enumeration {
            enum ALLPARENT;
            enum BESTCONNECTED;
            enum LOWESTJOINPRIORITY;
         }
      }
      leaf TargetNodeAddress {
         type uint64;
         description
         "Address of the time source neighbor";
      }
      leaf MinTimeCorrection {
         type uint16;
         description
         "In microsecond";
      }
      leaf MaxTimeCorrection {
         type uint16;
         description
         "In microsecond";
      }
      leaf AveTimeCorrection {
         type uint16;
         description
         "In microsecond";
      }
   }
```

```
   typedef asntype {
      description
         "The type to store ASN. String of 5 bytes";
      type string {
         length "0..5";
      }
   }

   list NeighborList {
      key "TargetNodeAddress";
      leaf TargetNodeAddress {
         type uint64;
         description
         "Address of the time source 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";
      }
   }

   list QueueList {
      key "QueueId";
      leaf QueueId {
         type uint8;
         description
         "Address of the time source neighbor";
      }
      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 uint16;
            description
            "The TrackID.";
         }
      }
      leaf MinLenTXQueue {
         type uint8;
         config false;
         description
         "Statistics, lowest TX queue len registered in the window.";
      }
      leaf MaxLenTXQueue {
         type uint8;
         config false;
```

```
         description
         "Statistics, largest TX queue len registered in the window.";
      }
      leaf AvgLenTXQueue {
         type uint8;
         config false;
         description
         "Statistics, avg TX queue len registered in the window.";
      }
      leaf MinLenRXQueue {
         type uint8;
         config false;
         description
         "Statistics, lowest RX queue len 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 len
          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";
      leaf LabelSwitchID {
         type uint16;
      }
      list InputCellIds {
         key "CellID";
         leaf CellID{
            type uint16;
            description
            "The CellID.";
         }
      }
      list OutputCellIds {
         key "CellID";
         leaf CellID{
            type uint16;
            description
            "The CellID.";
         }
      }
      leaf LoadBalancingPolicy {
         type enumeration {
            enum ROUNDROBIN;
            enum OTHER;
         }
         description
         "The load-balancing policy.";
      }
   }
```

```
   list TrackList {
      key "TrackId";
      leaf TrackId {
         type uint16;
      }
      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";
            leaf ChunkId{
               type uint16;
               description
               "The id of a chunk";
            }
            leaf SlotframeId{
               type uint8;
               description
               "The id of the slotframe that is mapped to this chunk";
            }
            leaf SlotBase {
               type uint16;
               description
               "the base slotOffset of the chunk";
            }
            leaf SlotStep {
               type uint8;
               description
               "the slot incremental of the chunk";
            }
            leaf ChannelBase {
               type uint8;
               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";
            leaf SlotOffset{
               type uint16;
               description
               "The slotoffset.";
            }
            leaf ChannelOffset{
               type uint16;
               description
               "The channeloffset.";
            }
            leaf ChunkId {
               type uint16;
               description
               "Identifier of the chunk the cell belongs to";
            }
            leaf CellID{
               type uint16;
               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 UNUSED;
                  enum USED;
               }
            }
         }
```

### 3.2. YANG model of the IEEE802.15.4 PIB

This section describes the YANG model of the part of [IEEE802154] PIB used by 6top, such as security related attributes. This part of data will be accessed through the MLME-GET and MLME-SET [IEEE802154] primitive.

TODO

### 3.3. YANG model of the IEEE802.15.4e PIB

This section describes the YANG model of the part of [IEEE802154e] PIB used in 6top, such as TSCH related attributes. This part of data will be accessed through the MLME-GET and MLME-SET [IEEE802154] primitive.

TODO

## 4. Commands

6top provides a set of commands as the interface with the higher layer. Most of these commands are related to the management of slotframes, cells and scheduling information. 6top also provides an interface allowing an upper layer to retrieve status information and statistics. The command set aims to facilitate 6top implementation by describing the main operations that higher layers may use to interact with 6top. The listed commands aim at providing semantics to manipulate 6top MIB, IEEE802.15.4 PIB and IEEE802.15.4e PIB programmatically.

CREATE.hardcell: Creates one or more hard cells in the schedule. Fails if the cell already exists. A cell is uniquely identified by the tuple (slotframe ID, slotOffset, channelOffset). 6top schedules the cell and marks it as a hard cell, indicating that it cannot reschedule this cell. The return value is CellID and the created cell is also filled in CellList(Section 3.1).

CREATE.softcell: To create soft cell(s). 6top is responsible for picking the exact slotOffset and channelOffset in the schedule, and ensure that the target node chooses the same cell and TrackID. 6top marks these cells as soft cell, indicating that it will continuously monitor their performance and reschedule if needed. The return value is CellID, and the created cell is also filled in CellList (Section 3.1).

READ.cell: Given a (slotframe ID, slotOffset, channelOffset), retrieves the cell information. A read command can be issued for any cell, hard or soft. 6top gets cell information from CellList (Section 3.1).

UPDATE.cell: Update a hard cell, i.e., re-allocate it to a different slotOffset and/or channelOffset. Fails if the cell does not exist. CellList (Section 3.1) will be modified.

DELETE.hardcell: To remove a hard cell. This removes the hard cell from the node’s schedule, from CellList (Section 3.1).

DELETE.softcell: To remove a (number of) soft cell(s). This command leads the pair of nodes figure out the specific cell(s) to be removed. After that, the cell(s) will be removed from the CellLists (Section 3.1) on both sides.

REALLOCATE.softcell: To force a re-allocation of a soft cell. The reallocated cell will be installed in a different slotOffset,

channelOffset but slotframe and TrackID remain the same. Hard cells MUST NOT be reallocated. This command will result in the modificaition of CellLists (Section 3.1) on both sides.

CREATE.slotframe: Creates a new slotframe. Adds a entry to the SlotframeList (Section 3.1).

READ.slotframe: Returns the information of a slotframe given its slotframeID from SlotframeList (Section 3.1).

UPDATE.slotframe: Change the number of timeslots in a slotframe given its slotframeID in SlotframeList (Section 3.1).

DELETE.slotframe: Deletes a slotframe, remove it from SlotframeList (Section 3.1).

CONFIGURE.monitoring: Configures the level of QoS the Monitoring process MUST enforce, i.e. config MonitoringStatusList (Section 3.1).

READ.monitoring: Reads the current Monitoring status from MonitoringStatusList (Section 3.1).

CONFIGURE.statistics: Configures the statistics process in StatisticsMetricsList(Section 3.1). The CONFIGURE.statistics enables flexible configuration and supports empty parameters that will force 6top to conduct statistics on all members of that dimension. For example, if ChannelOffset is empty and metric is set as PDR, then, 6top will conduct the statistics of PDR on all of channels.

READ.statistics: Reads a metric for the specified dimension. Information is aggregated according to the parameters from CellList (Section 3.1).

RESET.statistics: Resets the gathered statistics in CellList (Section 3.1).

CONFIGURE.eb: Configures EBs, i.e. configures EBlist (Section 3.1).

READ.eb: Reads the EBs configuration from EBList (Section 3.1).

CONFIGURE.timesource: Configures the Time Source Neighbor selection process, i.e. configure TimeSource (Section 3.1).

READ.timesource: Retrieves information about the time source neighbors of that node from TimeSource (Section 3.1).

CREATE.neighbor: Creates an entry for a neighbor in the neighbor table, i.e. NeighborList (Section 3.1).

READ.all.neighbor: Returns the list of neighbors of that node according to NeighborList (Section 3.1).

READ.neighbor: Returns the information of a specific neighbor of that node specified by its neighbor address according to NeighborList (Section 3.1).

UPDATE.neighbor: Updates the last status for a given TargetNodeAddress in the NeighborList (Section 3.1).

DELETE.neighbor: Deletes a neighbor given its address from NeighborList (Section 3.1).

CREATE.queue: Creates and Configures a queue in QueueList (Section 3.1).

READ.queue: Reads the queue configuration for given QueueId from QueueList (Section 3.1).

READ.queue.stats: For a given QueueId, reads the queue statistics information from the QueueList (Section 3.1).

UPDATE.queue: For a given QueueId, update its configuration in the QueueList (Section 3.1).

DELETE.queue: Deletes a Queue for a given QueueId from the QueueList (Section 3.1).

LabelSwitching.map: Maps an input cell or a bundle of input cells to an output cell or a bundle of output cells, i.e. adds a entry to the LabelSwitchList (Section 3.1).

LabelSwitching.unmap: Unmap one input cell or a bundle of input cells to an output cell or a bundle of output cells, i.e. modifies the LabelSwitchList (Section 3.1).

CREATE.chunk: Creates a chunk which consists of one or more unused cells, i.e. add an entry to the ChunkList (Section 3.1).

READ.chunk: Returns the information of a chunk given its ChunkID from ChunkList (Section 3.1).

DELETE.chunk: For given ChunkId, removes a chunk from the ChunkList (Section 3.1), which also causes all of the scheduled

cells in the chunk to be deleted from the TSCH schedule and CellList (Section 3.1).

CREATE.hardcell.fromchunk: Creates one or more hard cells from a chunk. 6top schedules the cell and marks it as a hard cell, indicating that it cannot reschedule this cell. The cell will be added into the CellList (Section 3.1). In addition, 6top will change the attributes corresponding to the cell in the ChunkCellList (Section 3.1), i.e. its CellID is changed to the same CellID in the CellList, and its Status is changed to USED.

READ.chunkcell: Returns the information of all cells in a chunk given its ChunkID from ChunkCellList (Section 3.1).

DELETE.hardcell.fromchunk: To remove a hard cell which comes from a chunk. This removes the hard cell from the node's schedule and CellList (Section 3.1). In addition, it changes the attributes corresponding to the cell in the ChunkCellList (Section 3.1), i.e. its CellID is changed back to 0xFFFF, and its Status is changed to UNUSED.

# 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 LLN context: Overview, Problem Statement and Goals", draft-ietf-6tisch-tsch-00 (work in progress), November 2013.

[I-D.ietf-6tisch-architecture] Thubert, P., Watteyne, T., and R. Assimiti, "An Architecture for IPv6 over the TSCH mode of IEEE 802.15.4e", draft-ietf-6tisch-architecture-00 (work in progress), November 2013.

[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-00 (work in progress), November 2013.

[I-D.ietf-6tisch-minimal]
Vilajosana, X. and K. Pister, "Minimal 6TiSCH Configuration", draft-ietf-6tisch-minimal-00 (work in progress), November 2013.

## 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] "Berkeley’s OpenWSN Project Homepage", <http://www.openwsn.org/>.

[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

6TiSCH
Internet-Draft
Intended status: Informational
Expires: August 18, 2014

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

February 14, 2014

# 6TiSCH Operation Sublayer (6top)
# draft-wang-6tisch-6top-sublayer-00

## Abstract

The recently published [IEEE802154e] standard formalizes the concept of link-layer resources in LLNs. Nodes are synchronized and follow a schedule. A cell in that schedule corresponds to an atomic link-layer resource, and can be allocated to any pair of neighbors in the network. This allows the schedule to be built to tightly match each node's bandwidth, latency and energy constraints. The [IEEE802154e] standard does not, however, present a mechanism to do so, as building and managing the schedule is out of scope of the standard. This document describes the 6TiSCH Operation Sublayer (6top) and the commands it provides to upper network layers such as RPL or GMPLS. The set of functionalities includes feedback metrics from cell states so network layers can take routing decisions, TSCH configuration and control procedures, and the support for decentralized and centralized scheduling. In addition, 6top can be configured to enable packet switching at layer 2.5, analogous to GMPLS.

Table of Contents

## 1. Introduction

As presented in [I-D.ietf-6tisch-tsch], the [IEEE802154e] standard defines the mechanisms for a TSCH node to communicate, given a schedule. It does not, however, define the mechanism to build and maintain the TSCH schedule, match that schedule to the multi-hop paths maintained by a network layer such as RPL or a 2.5 layer such as GMPLS, adapt the resources allocated between neighbor nodes to the data traffic flows, enforce a differentiated treatment for data generated at the application layer and signalling messages needed by 6LoWPAN and RPL to discover neighbors, react to topology changes, self-configure IP addresses, or manage keying material.

In a TSCH network, the MAC layer is not in charge of setting up the schedule that controls the connectivity graph of the network and the resources allocated to each cell in that topology. This responsibility is left to the next-higher layer, defined in this document, called "6top".

This document describes the 6TiSCH Operation Sublayer (6top) and the main commands provided to upper network layers such as RPL or GMPLS. The set of functionalities include feedback metrics from cell state so the network layer can take routing decisions, TSCH configuration and control procedures, and support for the different scheduling mechanisms defined in [I-D.ietf-6tisch-architecture]. 6top addresses the set of functionalities described in [I-D.ietf-6tisch-tsch].

For example, network formation in a TSCH network involves the transmission 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. 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 [label-switching-154e], 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. 6TiSCH Operation Sublayer (6top) Overview

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

Protocol Stack

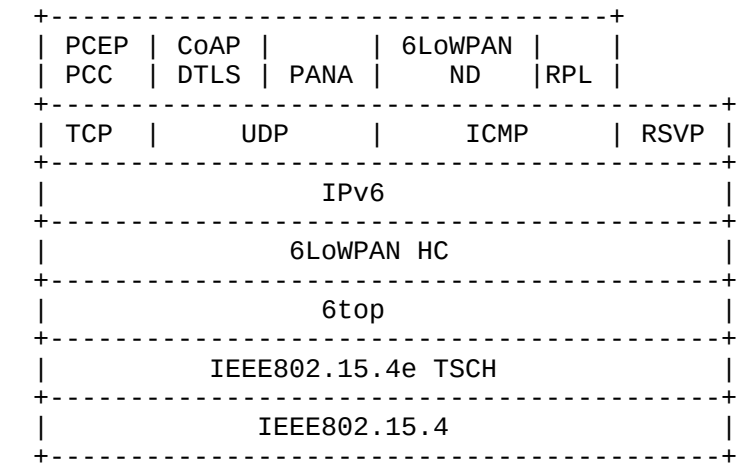



Figure 1

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 2.1.

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 2.2.

All 6top commands of the management and data interfaces are detailed in Section 3. This set of commands is designed to support decentralized, centralized and hybrid scheduling solutions. They form a conceptual interface an upper layer can used; implementations can use this set of commands, or any equivalent alternative.

6top defines TSCH Information Elements (IEs) for neighbors nodes to negotiate scheduling cells in the TSCH schedule. The format of those IEs is given in Section 4.1. Example data exchanges between neighbor nodes are given in Section 4.2.

Section 5 defines how 6top gathers statistics (e.g. link quality, energy level, queue usage), and what commands an upper layer can use to configure and retrieve those statistics.

6top can be configured to monitor the cells it has scheduled in order to detect cells with poor performance. It can automatically re-allocate those cells inside the TSCH schedule. This behavior is described in Section 6

## 2.1. 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 and channelOffset of the corresponding cell(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). TrackOwnerAddr is the address of the node which initiates the process of creating the track, i.e. the owner of the track. InstanceID is an instance identifier given by the owner of the track. InstanceID comes from the upper layer; it 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, called a 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 cells in a chunk can be appropriated by the node, i.e. the node is in charge of managing 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.

#### 2.1.1. hard cells

A hard cell is a cell that cannot be dynamically reallocated by 6top. A hard cell is uniquely identified by the following tuple:

slotframe ID: ID of the slotframe this cell is part of.

slotOffset: the slotOffset for the cell.

channelOffset: the channelOffset for the cell.

LinkOption bitmap: bitmap as defined in [IEEE802154].

CellType: MUST be set to 1.

#### 2.1.2. 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 Section 4.2.

### 2.2. Data Transfer Model

The TSCH MAC layer is decoupled from the upper layer; the interaction between the upper layer and TSCH is asynchronous. This means that the MAC layer executes a schedule and checks at each timeslot according to the type of cell, whether there is something to send or receive. If that is the case, the packet is transmitted and the MAC layer continues its operation. When an upper layer sends a packet, this packet is pushed into a queue waiting for the MAC layer to read it and send it in a particular timeslot according to its destination and priority. 6top provides a set of queue management operations which enable upper layers to create different queues and set their priorities. This allows different classes of traffic to be handled by the forwarding plane by inserting a packet into the queue appropriate for its priority.

A 6top implementation MUST provide at least a Broadcast Queue and a Transmit Queue. The Broadcast Queue is associated with cells with LinkType=ADVERTISING in the sender’s schedule, and LinkOption="Receive" and "Timekeeping" in all its neighbors’ schedule. For example, NodeA uses slotOffset=5 and channelOffset=12 as Broadcast cell to its neighbors NodeB and NodeC. Then, in the schedule of NodeA the cell will be featured with neighbor address is Broadcast address, LinkType=ADVERTISING; and in the schedules of both nodeB and nodeC the cell will be featured with nodeA address as neighbor address, and LinkOption="Receive" and "Timekeeping", which ensure nodeB and nodeC will be active at least one time in the cell to receive broadcast packet during a Timekeeping period. A Transmit Queue is associated with the dedicated Transmit cells or Shared Cells.

Data Communication Commands (Section 3.12) can be used to send control messages and data messages. The operation is used to insert a message into a specific queue.

For example, a configuration can include two Broadcast Queues with priority High and Low, and three Transmit Queues with priority High, Mid, and Low.

When DestAddr is the broadcast address, its related MAC layer packets will be pushed into the Broadcast Queue with the corresponding priority. 6top is responsible for feeding these packets into broadcast cells.

When DestAddr is a unicast address, its related MAC layer packets will be pushed into the Transmit Queue with the corresponding priority. 6top is responsible for feeding these packets into Transmit or Shared Cells.

The QoS policy enforced by 6top is out of scope. As an example, packets in higher priority queues could be transmitted before the packets in lower priority queue. As a result, when there is an available broadcast/unicast cell, 6top checks the broadcast/unicast queue with higher priority first. 6top continues this search until it finds a broadcast/unicast packet, or finds that all of broadcast/ unicast queues are empty.

Figure Figure 2 shows how 6top shapes data from the upper layer (e.g., RPL, 6LoWPAN), and feeds it to TSCH. The properties associated with a packet/fragment from the upper layer includes the next hop neighbor (DestAddr), the packet priority, and TrackID(s).

6top Data Transfer Model

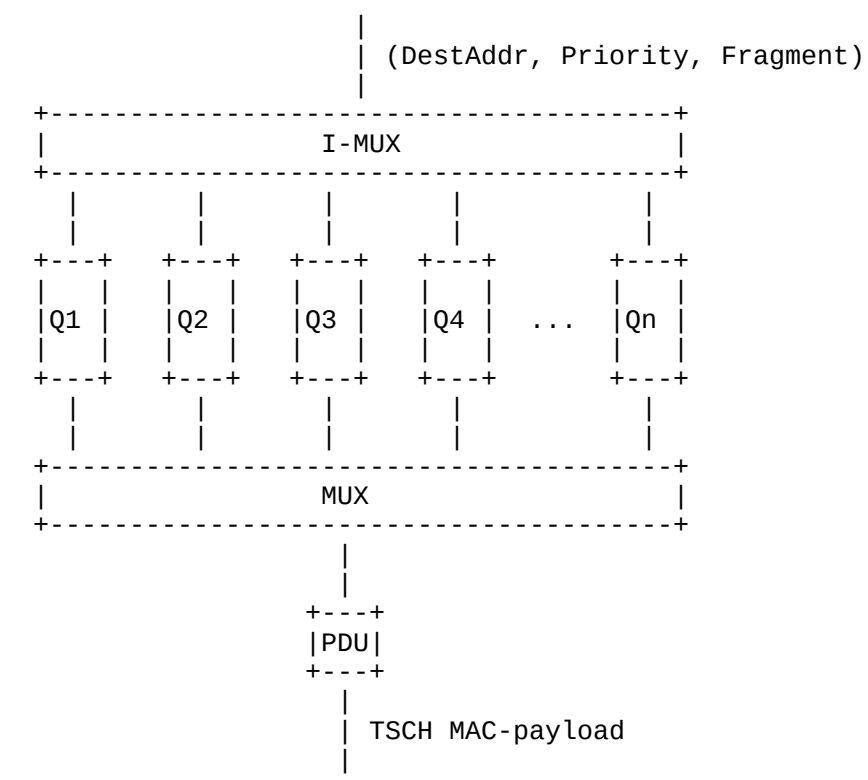



Figure 2

In Figure 2, Qi represents a queue, which is either broadcast or unicast, and is assigned a 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 through a Send.data command (Section 3.12), 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 is 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.

## 3. 6top Commands

6top provides a set of commands as the interface with the higher layer. Most of these commands are related to the configuration of slotframes, cells and scheduling information. 6top also provides an interface allowing an upper layer to retrieve status information and statistics. The management commands provided by 6top are listed below. Note that this set defines a conceptual interface only; an implementation can choose to use this exact set of commands, or any equivalent alternative.

CREATE.hardcell: Section 3.1.1

CREATE.softcell: Section 3.1.2

READ.cell: Section 3.1.3

UPDATE.cell: Section 3.1.4

DELETE.hardcell: Section 3.1.5

DELETE.softcell: Section 3.1.6

REALLOCATE.softcell: Section 3.1.7

CREATE.slotframe: Section 3.2.1

READ.slotframe: Section 3.2.2

READ.chunk: Section 3.10.2

DELETE.chunk: Section 3.10.3

CREATE.hardcell.fromchunk: Section 3.11.1

READ.chunkcell: Section 3.11.2

DELETE.hardcell.fromchunk: Section 3.11.3

Besides management commands, 6top provides the following data commands:

Send.data: Section 3.12.1

Receive.data: Section 3.12.2

In addition, 6top offers a delegation interface allowing an upper layer to configure TSCH. 6top only delegates the functionalities to the MAC security services. In other words, 6top allows an upper layer to access the security PIB (Table 60, Table 61, Table 63 in [IEEE802154]) by using MLME-GET/MLME-SET primitives defined in [IEEE802154].

## 3.1. Cell Commands

6top provides the following commands to manage TSCH cells.

### 3.1.1. CREATE.hardcell

Creates one or more hard cells in the schedule. Fails if the cell already exists. A cell is uniquely identified by the tuple (slotframe ID, slotOffset, channelOffset).

To create a hard cell, the upper layer specifies:

slotframe ID: ID of the slotframe this timeslot will be scheduled in.

slotOffset: the slotOffset for the cell.

channelOffset: channelOffset for the cell.

LinkOption bitmap: bitmap as defined in [IEEE802154e]

LinkType : as defined in section 6.2.19.3 of [IEEE802154e].

CellType: as defined in Section 2.1

target node address: the address of that node to communicate with over this cell. In case of broadcast cells this is the broadcast address.

TrackID: ID of the track the cell will belong to.

6top schedules the cell and marks it as a hard cell, indicating that it cannot reschedule this cell. The return value is CellID and the created cell is also filled in CellList ([I-D.wang-6tisch-6top-interface]).

The interaction between 6top and MAC layer caused by CREATE.hardcell is as follows.

Firstly, 6top calls the premitive MLME-SET-LINK.request defind in section 6.2.19.3 of [IEEE802154e]. The premitive parameters are set as follows.

| MLME-SET-LINK.request parameter | set by 6top |
|---|---|
| operation | ADD-LINK |
| LinkHandle | CellID |
| slotframeHandle | slotframe ID |
| timeslot | slotOffset |
| channelOffset | channelOffset |
| LinkOptions | LinkOption bitmap |
| LinkType | LinkType |
| nodeAddr | target node address |

Secondly, if the status from MLME-SET-LINK.confirm defined in section 6.2.19.4 of [IEEE802154e] is SUCCESS, then add the LinkHandle to the BundleList specified by TrackID, and confirm to upper layer with status = SUCCESS; otherwise, confirm to upper layer with status = FAIL.

#### 3.1.2. CREATE.softcell

To create soft cell(s), the upper layer specifies:

slotframe ID: ID of the slotframe the cell(s) will be scheduled in

number of cells: the required number of soft cells.

LinkOption bitmap: bitmap as defined in [IEEE802154e]

CellType: as defined in Section 2.1

target node address: the address of the node to communicate with over the cell(s). In case of broadcast cells this is the broadcast address.

TrackID: ID of the track the cell(s) will belong to.

QoS level: the cell redundancy policy. The policy can be for example STRICT, BEST_EFFORT, etc.

6top is responsible for picking the exact slotOffset and channelOffset in the schedule, and ensure that the target node choose the same cell and TrackID. 6top marks these cells as soft cell, indicating that it will continuously monitor their performance and reschedule if needed. The return value is CellID, and the created cell is also filled in CellList ([I-D.wang-6tisch-6top-interface]).

6top deals with the allocation process by negotiation with the target node. The command returns the list of created cells defined by (slotframe ID, slotOffset, channelOffset). It fails if the required number of cells is higher than the available number of cells in the schedule. It fails if the negotiation with the target node fails. It fails if the LinkOption bitmap indicates that the cell(s) MUST be Hard.

The interaction between 6top and TSCH happens on both sides described as follows.

For example, after neigotiation, node A and node B find a specifical cell, slotOffset=10, channelOffset=12, as a Tx cell and Rx cell, respectively, then the 6top in node A and node B will call the premitive MLME-SET-LINK.request defind in section 6.2.19.3 of [IEEE802154e], respectively. The premitive parameters are set in node A and node B as follows.

| MLME-SET-LINK.request parameter | set by A's 6top | set by B's top |
|---|---|---|
| operation | ADD-LINK | ADD-LINK |
| LinkHandle | CellID | CellID |
| slotframeHandle | slotframe ID | slotframe ID |
| timeslot | 10 | 10 |
| channelOffset | 12 | 12 |
| LinkOptions | Tx | Rx |
| LinkType | NORMAL | NORMAL |
| nodeAddr | Node A | Node B |

If the Status from MLME-SET-LINK.confirm defined in section 6.2.19.4 of [IEEE802154e], 6top will notify upper layer failure.

### 3.1.3. READ.cell

Given a (slotframe ID, slotOffset, channelOffset), retrieves the cell information. Fails if the cell does not exist. The returned information contains:

slotframe ID: ID of the slotframe where this cell is installed.

slotOffset: the slotOffset for the cell.

channelOffset: the selected channelOffset for the cell.

LinkOption bitmap: bitmap as defined in [IEEE802154e]

CellType: as defined in Section 2.1

target node address: the target address of that cell. In case of broadcast cells this is the broadcast address.

TrackID: ID of the track the cell will belong to.

NumOfStatistics: Number of elements in the following list of tuple (StatisticsMetriceID and StatisticsValue)

list of (StatisticsMetriceID, StatisticsValue): StatisticsMetriceID is the index to Statistics Metric defined in Section 3.4, StatisticsValue is the value corresponding to the metric indexd by StatisticsMetriceID

A read command can be issued for any cell, hard or soft. 6top gets cell information from CellList ([I-D.wang-6tisch-6top-interface]).

#### 3.1.4. UPDATE.cell

Update a hard cell, i.e., re-allocate it to a different slotOffset and/or channelOffset. Fails if the cell does not exist. Requires both old (slotframe ID, slotOffset, channelOffset) and new (slotframe ID, slotOffset, channelOffset) as parameters. And, the type of cell, target node address and TrackID are the fields that cannot be updated. Soft cells MUST NOT be updated by the UPDATE.cell command. REALLOCATE.softcell (Section 3.1.7) MUST be used instead.

It causes a old cell being removed and a new cell being created.

#### 3.1.5. DELETE.hardcell

To remove a hard cell, the upper layer specifies:

slotframe ID: the ID of the slotframe where this cell is installed.

slotOffset: the slotOffset for the cell.

channelOffset: the selected channelOffset for the cell.

LinkOption bitmap: bitmap as defined in [IEEE802154e]

LinkType : as defined in in section 6.2.19.3 of [IEEE802154e].

CellType: as defined in Section 2.1

target node address: the target address of that cell. In case of broadcast cells this is the broadcast address.

TrackID: ID of the track the cell will belong to.

This removes the hard cell from the node’s schedule, from CellList ([I-D.wang-6tisch-6top-interface])as well.

The interaction between 6top and MAC layer caused by DELETE.hardcell is as follows.

Firstly, 6top calls the premitive MLME-SET-LINK.request defind in section 6.2.19.3 of [IEEE802154e]. The premitive parameters are set as follows.

| MLME-SET-LINK.request parameter | set by 6top |
|---|---|
| operation | DELETE-LINK |
| LinkHandle | CellID |
| slotframeHandle | slotframe ID |
| timeslot | slotOffset |
| channelOffset | channelOffset |
| LinkOptions | LinkOption bitmap |
| LinkType | LinkType |
| nodeAddr | target node address |

Secondly, if the status from MLME-SET-LINK.confirm defined in section 6.2.19.4 of [IEEE802154e] is SUCCESS, then remove the LinkHandle from its BundleList specified by TrackID, and confirm to upper layer with status = SUCCESS; otherwise, confirm to upper layer with status = FAIL.

### 3.1.6. DELETE.softcell

To remove a (number of) soft cell(s), the upper layer specifies:

slotframe ID: ID of the slotframe where this cell is installed.

number of cells: the number of cells to be removed

LinkOption bitmap: bitmap as defined in [IEEE802154e]

CellType: as defined in Section 2.1

target node address: the target address of that cell. In case of broadcast cells this is the broadcast address.

TrackID: ID of the track the cell will belong to.

In the case a soft cell wants to be re-allocated from the allocated cell so a hard cell can be installed instead, the REALLOCATE.softcell (Section 3.1.7) MUST be used.

After the pair of nodes figure out the specific cell(s) to be removed, the interaction between 6top and TSCH on both sides will be similar to that caused by DELETE.hardcell, except LinkType should be set to NORMAL.

#### 3.1.7. REALLOCATE.softcell

To force a re-allocation of a soft cell, the upper layer specifies:

slotframe ID: ID of the slotframe where the cell is allocated.

slotOffset: the slotOffset for that cell.

channelOffset: the channelOffset for that cell.

The reallocated cell will be installed in a different slotOffset, channelOffset but slotframe and TrackID remain the same. Hard cells MUST NOT be reallocated.

The interaction between 6top and TSCH caused by this command includes that described in Section 3.1.6 and Section 3.1.2.

### 3.2. Slotframe Commands

6top provides the following commands to manage TSCH slotframes.

#### 3.2.1. CREATE.slotframe

Creates a new slotframe. The command requires:

slotframe ID: unique identifier of the slotframe, corresponding to its priority.

number of timeslots: the required number of timeslots in the slotframe.

Fails if the number of required timeslots is less than zero.

The interaction between 6top and MAC layer caused by CREATE.slotframe is as follows.

Firstly, 6top calls the premitive MLME-SET-SLOTFRAME.request defind in section 6.2.19.1 of [IEEE802154e]. The premitive parameters are set as follows.

| MLME-SET-SLOTFRAME.request parameter | set by 6top |
|---|---|
| slotframeHandle | slotframe ID |
| operation | ADD |
| size | number of timeslot |

Secondly, if the status from MLME-SET-SLOTFRAME.confirm defined in section 6.2.19.2 of [IEEE802154e] is SUCCESS, then confirms to upper layer with status = SUCCESS; otherwise, confirm to upper layer with status = FAIL.

### 3.2.2. READ.slotframe

Returns the information of a slotframe given its slotframe ID. The command returns:

slotframe ID: ID of the slotframe. (SlotFrameHandle)

number of timeslots: the number of timeslots in the slotframe.

Fails if the slotframe ID does not exist.

### 3.2.3. UPDATE.slotframe

Change the number of timeslots in a slotframe. The command requires:

slotframe ID: ID of the slotframe.

number of timeslots: the number of timeslots to be updated.

Fails if the number of required timeslots is less than zero. Fails if the slotframe ID does not exist.

The interaction between 6top and MAC layer caused by UPDATE.slotframe is as follows.

Firstly, 6top calls the premitive MLME-SET-SLOTFRAME.request defind in section 6.2.19.1 of [IEEE802154e]. The premitive parameters are set as follows.

| MLME-SET-SLOTFRAME.request parameter | set by 6top |
|---|---|
| slotframeHandle | slotframe ID |
| operation | MODIFY |
| size | number of timeslot |

Secondly, if the status from MLME-SET-SLOTFRAME.confirm defined in section 6.2.19.2 of [IEEE802154e] is SUCCESS, then confirms to upper layer with status = SUCCESS; otherwise, confirm to upper layer with status = FAIL.

### 3.2.4. DELETE.slotframe

Deletes a slotframe. The command requires:

slotframe ID: ID of the slotframe.

number of timeslot: the number of timeslots in the slotframe.

Fails if the slotframe ID does not exist.

The interaction between 6top and MAC layer caused by DELETE.slotframe is as follows.

Firstly, 6top calls the premitive MLME-SET-SLOTFRAME.request defind in section 6.2.19.1 of [IEEE802154e]. The premitive parameters are set as follows.

| MLME-SET-SLOTFRAME.request parameter | set by 6top |
|---|---|
| slotframeHandle | slotframe ID |
| operation | DELETE |
| size | number of timeslot |

Secondly, if the status from MLME-SET-SLOTFRAME.confirm defined in section 6.2.19.2 of [IEEE802154e] is SUCCESS, then confirms to upper layer with status = SUCCESS; otherwise, confirm to upper layer with status = FAIL.

### 3.3. Monitoring Commands

Monitoring commands provide the means for upper layers to configure whether 6top must ensure the required bandwidth. This procedure is achieved through overprovisioning according to cell status feedback. Monitoring is also in charge of reallocating soft cells that are under the required QoS.

#### 3.3.1. CONFIGURE.monitoring

Configures the level of QoS the Monitoring process MUST enforce. The command requires:

slotframe ID: ID of the slotframe.

target node address: the target neighbor address.

enforce policy: The policy used to enforce the QoS requirements. Can be for example DISABLE, BEST_EFFORT, STRICT, OVER-PROVISION, etc.

Fails if the slotframe ID does not exist.

#### 3.3.2. READ.monitoring.status

Reads the current Monitoring status. Requires the following parameters.

slotframe ID: the ID of the slotframe.

target node address: the target neighbor address.

Returns the QoS levels for that Target node on that slotframe.

allocated_hard: Number of hard cells allocated.

allocated_soft: Number of soft cells allocated.

provisioned: the extra provisioned cells. 0 if CONFIGURE.qos enforce is DISABLE.

QoS: the current QoS. Including overprovisioned cells, i.e what bandwidth is being obtained including the overprovisioned cells.

RQoS: the real QoS without provisioned cells. What is the actual bandwidth without taking into account the overprovisioned cells.

Fails if the slotframe ID does not exist.

### 3.4. Statistics Commands

6top keeps track of TSCH statistics for upper layers to adapt correctly to medium changes. The exact metrics for statistics are out of scope but the present commands SHOULD be used to configure and read monitored information regardless of the specific metric.

#### 3.4.1. CONFIGURE.statistics

Configures Statistics process. The command requires:

slotframe ID: ID of the slotframe. If empty monitors all slotframe IDs

slotOffset: specific slotOffset to be monitored. If empty all timeslots are monitored

channelOffset: specific channelOffset to be monitored. If empty all channels are monitored.

target node address: the target neighbor address. If empty, all neighbor nodes are monitored.

metric: metric to be monitored. This MAY be PDR, ETX, queuing statistics, energy-related metrics, etc.)

window: time window to be considered for the calculations. If 0 all historical data is considered.

enable: Enables statistics or disables them.

Fails if the slotframe ID does not exist. The statistics service can be configured to retrieve statistics at different levels. For example to aggregate information by slotframe ID, or to retrieve statistics for a particular timeslot, etc. The CONFIGURE.statistics enables flexible configuration and supports empty parameters that will force 6top to conduct statistics on all members of that dimension. For example, if ChannelOffset is empty and metric is set as PDR, then, 6top will conduct the statistics of PDR on all of channels.

#### 3.4.2. READ.statistics

Reads a metric for the specified dimension. Information is aggregated according to the parameters. The command requires:

slotframe ID: ID of the slotframe. If empty aggregates information of all slotframe IDs

slotOffset: the specific slotOffset for which the information is required. If empty all timeslots are aggregated

channelOffset: the specific channelOffset for which the information is required. If empty all channels are aggregated.

target node address: the target neighbor address. If empty all neighbor addresses are aggregated.

metric: metric to be read.

Returns the value for the requested metric.

Fails if empty metric or metric does not exits.

#### 3.4.3. RESET.statistics

Resets the gathered statistics. The command requires:

slotframe ID: ID of the slotframe. If empty resets the information of all slotframe IDs

slotOffset: the specific slotOffset for which the information wants to be reset. If empty statistics from all timeslots are reset

channelOffset: the specific channelOffset for which the information wants to be reset. If empty all statistics for all channels are reset.

target node address: the target neighbor address. If empty all neighbor addresses are aggregated.

metric: metric to be reset.

Fails if empty metric or metric does not exits.

### 3.5. Network Formation Commands

EBs need to be configured, including their transmission period, the slotOffset and channelOffset that they SHOULD be sent on, and the join priority they contain. The parameters for that command are optional and enable flexible configuration of EBs. If slotframe ID is specified, the EBs will be configured to use that specific slotframe; if not, they will use the first slotframe where the

configured slotOffset is allocated. The slotOffset enforces the EB to a specific timeslot. In case slotOffset parameter is not present, the EB is sent in the first available transmit timeslot. In case channelOffset parameter is not set, the EB is configured to use the first available channel.

#### 3.5.1. CONFIGURE.eb

Configures EBs. The command requires:

slotframe ID: ID of the slotframe where the EBs MUST be sent. Zero if any slotframe can be used.

slotOffset: the slotOffset where the EBs MUST be sent. Zero if any timeslot can be used.

channelOffset: the channelOffset where the EBs MUST be sent. Zero if any channelOffset can be used.

period: the EBs period, in seconds.

Expiration: when the EBs periodicity will stop. If Zero the period never stops.

priority: the joining priority model that will be used for advertisement. Joining priority MAY be for example SAME_AS_PARENT, RANDOM, BEST_PARENT+1 or DAGRANK(rank) as decribed in in [I-D.ietf-6tisch-minimal].

Fails if the tuple (slotframe ID, slotOffset, channelOffset) is already scheduled.

#### 3.5.2. READ.eb

Reads the EBs configuration. No parameters are required.

Returns the current EBs configuration for that slotframe, which contains:

slotframe ID: the slotframe where the EB is being sent.

slotOffset: the slotOffset where the EBs is being sent.

channelOffset: the channelOffset the EBs is being sent on.

period: the EBs period.

Expiration: when the EBs periodicity stops. If 0 the period never stops.

priority: the joining priority that this node advertises.

Fails if the slotframe ID does not exist.

### 3.6. Time Source Neighbor Commands

Commands to select time source neighbors.

#### 3.6.1. CONFIGURE.timesource

Configures the Time Source Neighbor selection process. More than one time source neighbor can be selected. The command requires:

selection policy: The policy used to select the time source neighbor. The policy MAY be for example ALL_PARENTS, BEST_CONNECTED, LOWEST_JOIN_PRIORITY, etc.

Fails if any of the time source neighbors do not exist or it is not reachable.

#### 3.6.2. READ.timesource

Retrieves information about the time source neighbors of that node. The command does not require any parameter.

Returns the following information for each of the time sources:

target node: address of the time source neighbor.

statistics: includes for example minimum, maximum, average time correction for that time source neighbor

policy: the used policy

Fails if the slotframe ID or no time source neighbors exist.

### 3.7. Neighbor Commands

Commands to manage neighbor table. The commands SHOULD be used by the upper layer to query the neighbor related information and by the lower layer to keep track of neighbors information.

#### 3.7.1. CREATE.neighbor

Creates an entry for a neighbor in the neighbor table.

neighbor address: The address of the neighbor.

neighbor stats: for example, RSSI of the last received packet from that neighbor, ASN when that neighbor has been added, etc.

Returns whether the neighbor is created or not.

#### 3.7.2. READ.all.neighbor

Returns the list of neighbors of that node. Fails if empty. For each neighbor in the list it returns:

neighbor address: The address of the neighbor.

neighbor stats: for example, RSSI of the last received packet from that neighbor, ASN when that neighbor has been added, packets received from that neighbor, packets sent to it, etc.

#### 3.7.3. READ.neighbor

Returns the information of a specific neighbor of that node specified by its neighbor address. Fails if it does not exists. For that neighbor it returns:

neighbor address: The address of the neighbor.

neighbor stats: for example, RSSI of the last received packet from that neighbor, ASN when that neighbor has been added, packets received from that neighbor, packets sent to it, etc.

#### 3.7.4. UPDATE.neighbor

Updates an entry for a neighbor in the neighbor table. Fails if the neighbor does not exist. Updates stats parameters. Requires:

neighbor address: The address of the neighbor.

neighbor stats: for example, RSSI of the last received packet from that neighbor, ASN when that neighbor has been added, etc.

Returns whether the neighbor is updated or not.

#### 3.7.5. DELETE.neighbor

Deletes a neighbor given its address. Fails if the neighbor does not exists.

### 3.8. Queueing Commands

Queues need to be configured. This includes queue length, retransmission policy, discarding of packets, etc.

#### 3.8.1. CREATE.queue

Creates and Configures Queues. The command SHOULD be applied for each required queue. The command requires:

txqlength: the desired transmission queue length.

rxqlength: the desired reception queue length.

numrtx: number of allowed retransmissions.

age: discard packet according to its age on the queue. 0 if no discards are allowed.

rtxbackoff: retransmission backoff in number of slotframes. 0 if next available timeslot wants to be used.

statswindow: window of time used to compute stats.

queue priority: the priority of this queue.

TrackIDs: a set of TrackIDs. While it is empty, no specific track is associated with the queue

Returns the queue ID.

#### 3.8.2. READ.queue

Reads the queue configuration. Requires the queue ID.

The command returns:

txqlength: the transmission queue length.

rxqlength: the reception queue length.

numrtx: number of allowed retransmissions.

age: maximum age of a packet before being discarded. 0 if no discards are allowed.

rtxbackoff: retransmission backoff in number of slotframes. 0 if next available timeslot is used.

#### 3.8.3. READ.queue.stats

Reads the queue stats. Requires queue ID.

The command returns:

txqlengthstats: average, maximum, minimum length of the transmission queue.

rxqlengthstats: average, maximum, minimum length of the reception queue.

numrtxstats: average, maximum, minimum number of retransmissions.

agestats: average, maximum, minimum age of a packet in the queue.

rtxbackoffstats: average, maximum, minimum retransmission backoff.

queue priority: the priority of this queue.

TrackIDs: a set of TrackIDs.

#### 3.8.4. UPDATE.queue

Update a Queue. The command requires:

queueid: the queue ID.

txqlength: the desired transmission queue length.

rxqlength: the desired reception queue length.

numrtx: number of allowed retransmissions.

age: discard packet according to its age on the queue. 0 if no discards are allowed.

rtxbackoff: retransmission backoff in number of slotframes. 0 if next available timeslot wants to be used.

statswindow: window of time used to compute stats.

queue priority: the desired priority of this queue.

TrackIDs: the desired set of TrackIDs.

#### 3.8.5. DELETE.queue

Deletes a Queue. The command requires the queue ID. All packets in the queue are discarded and the queue is deleted.

### 3.9. Label Switching Commands

6top is responsible for maintaining the mapping of input cells and output cells in the same track in a particular node. By keeping that mapping, layer 3 routing can be avoided as packets are forwarded by the 6top sublayer according to the input cells they were received on. The selected output cell is one of the cells that forward the packet to the subsequent hop in the track.

#### 3.9.1. LabelSwitching.map

The command used by an upper layer to map an input cell or a bundle of input cells to an output cell or a bundle of output cells. 6top stores this mapping and makes sure that the packets are forwarded at the specific output cell/bundle. Label Switching is enabled by the specified bundle as soon as the mapping is installed.

The required parameters are:

input cells: list of input cells (one or more cells in a bundle). Each input cells is described by an unique tuple (slotOffset, channelOffset, destination address).

output cells: list of output cells (one or more cells in a bundle). Each output cells is described by an unique tuple (slotOffset, channelOffset, destination address).

load balancing policy: A policy for load balance cell usage. The policy is out of scope, however an example can be use ROUND ROBIN policy within the cells of the same bundle.

#### 3.9.2. LabelSwitching.unmap

The command used by upper layers to unmap one input cell or a bundle of input cells to an output cell or a bundle of output cells. The mapping is removed from the state kept by 6top.

The required parameters are:

input cells: list of input cells (one or more cells in a bundle). Each input cells is described by an unique tuple (slotOffset, channelOffset, destination address).

output cells: list of output cells (one or more cells in a bundle). Each output cells is described by an unique tuple (slotOffset, channelOffset, destination address).

### 3.10. Chunk Command

#### 3.10.1. Create.chunk

Create a chunk which consists of one or more unappropriated cells.

To create a chunk, upper layer specifies:

slotframe ID: ID of the slotframe which this chunk belongs to.

ChunkSize: number of the cells which the chunk includes.

SlotBase : the base slotOffset of the chunk.

SlotStep : the incremental of slotOffset in the chunk.

ChannelBase: the base channelOffset of the chunk.

ChannelStep: the incremental of channalOffset in the chunk.

ChunkID is the return value of the command ([I-D.wang-6tisch-6top-interface]). The chunk is a set of cells in the given slotframe, consisting of (slotOffset(i),channelOffset(i)), i=0..Chunksize-1, slotOffset(i)= (slotBase + i * slotStep) % slotframeLen, channelOffset(i) = (channelBase + i * channelStep) % 16". Those cells will be added into ChunkCellList ([I-D.wang-6tisch-6top-interface]) also.

#### 3.10.2. READ.chunk

Returns the information of a chunk given its ChunkId. The command returns:

slotframe ID: ID of the slotframe which this chunk belongs to.

ChunkSize: number of the cells which the chunk includes.

SlotBase : the base slotOffset of the chunk.

SlotStep : the incremental of slotOffset in the chunk.

ChannelBase: the base channelOffset of the chunk.

ChannelStep: the incremental of channalOffset in the chunk.

Fails if the ChunkId does not exist.

#### 3.10.3. Delete.chunk

To delete a chunk, upper layer specifies ChunkID.

It removes the chunk from ChunkList ([I-D.wang-6tisch-6top-interface]), and also remove those entries corresponding to the cells of the chunk from ChunkCellList([I-D.wang-6tisch-6top-interface]). In addition, it also causes all of the scheduled cells in the chunk are deleted from CellList ([I-D.wang-6tisch-6top-interface]) and TSCH schedule as well.

### 3.11. Chunk Cell Command

#### 3.11.1. CREATE.hardcell.fromchunk

Creates one or more hard cells from a chunk. Fails if the cell already exists. A cell is uniquely identified by the tuple (slotframe ID, slotOffset, channelOffset).

To create a hard cell from a chunk which is corresponding to a specific slotframe ID, the upper layer specifies:

chunkID: ID of the chunk which this cell belongs to.

slotOffset: the slotOffset for the cell.

channelOffset: channelOffset for the cell.

LinkOption bitmap: bitmap as defined in [IEEE802154e]

LinkType : as defined in section 6.2.19.3 of [IEEE802154e].

CellType: as defined in Section 2.1

target node address: the address of that node to communicate with over this cell. In case of broadcast cells this is the broadcast address.

TrackID: ID of the track the cell will belong to.

6top schedules the cell and marks it as a hard cell, indicating that it cannot reschedule this cell. In addition, 6top will change the attributes corresponding to the cell in the ChunkCellList, i.e. its CellID is changed to the same CellID in the CellList, and its Status is changed to USED ([I-D.wang-6tisch-6top-interface]).

The interaction between 6top and MAC layer caused by CREATE.hardcell.fromchunk is same as that caused by CREATE.hardcell (Section 3.1.1).

#### 3.11.2. READ.chunkcell

Returns the cell information of a chunk given its ChunkId. For each cell of the chunk, the command returns:

slotOffset: the slotOffset of the cell.

channelOffset: channelOffset of the cell.

cellId: the cellID in the CellList if scheduled.

Status: USED/UNUSED

Fails if the ChunkId does not exist.

#### 3.11.3. DELETE.hardcell.fromchunk

To remove a hard cell which comes from a chunk, the upper layer specifies:

slotframe ID: the ID of the slotframe where this cell is installed.

slotOffset: the slotOffset for the cell.

channelOffset: the selected channelOffset for the cell.

LinkOption bitmap: bitmap as defined in [IEEE802154e]

LinkType : as defined in in section 6.2.19.3 of [IEEE802154e].

CellType: as defined in Section 2.1

target node address: the target address of that cell. In case of broadcast cells this is the broadcast address.

TrackID: ID of the track the cell will belong to.

This removes the hard cell from the node's schedule and CellList ([I-D.wang-6tisch-6top-interface]). In addition, it changes the attributes corresponding to the cell in the ChunkCellList, i.e. its CellID is changed back to FFFF, and its Status is changed to UNUSED ([I-D.wang-6tisch-6top-interface]).

The interaction between 6top and MAC layer caused by DELETE.hardcell is same as that caused by DELETE.hardcell (Section 3.1.5).

### 3.12. Data Commands

#### 3.12.1. Send.data

The command used by upper layers to queue a packet so underlying TSCH sends it. According to the specific priority, the packet is pushed into a Queue with the equivalent priority or following a criteria out of scope. Once a packet is inserted into a queue it waits to be transmitted by TSCH according to the model defined in Section 2.2. If the queue is full or destination address is not a L2 neighbor of the node, failure to enqueue will be indicated to the caller.

The required parameters are:

src address: L2 address

dest address: L2 unicast or broadcast address

priority: packet priority, usually is consistent with queue priority

message length: the length of the message

message: control message or data message

securityLevel:As defined by [IEEE802154e].

#### 3.12.2. Receive.data

The command is invoked whenever a packet is received and inserted into a reception queue. The method acts as a callback function to notify to the upper layers the received message. Upper layers MUST terminate this indication.

The function has the following parameters:

src address: L2 source address

dest address: L2 unicast or broadcast destination address

priority: packet priority, usually is consistent with queue priority

message length: the length of the message.

message: control message or data message

# 4. 6top Communication Protocol

This section defines the Information Element (IE) based message formats, and the 6top-to-6top communication time sequences.

## 4.1. Message Formats

6top has to negotiate the scheduling of soft cells with neighbor nodes. This negotiation happens through 6top-specific TSCH Information Elements, the format of which is defined in this section. For completeness, this section also details the formats of the IEs already defined in [IEEE802154e] and presented here without modification.

6top messages can contain one or more IEs. Section 4.1.1 defines the different IEs used by 6top, both the ones used without modification from [IEEE802154e], and the new ones defined by this document. Section 4.1.2 shows how several IEs are assembled to form the different frames used by 6top.

### 4.1.1. Information Elements

[IEEE802154e] defines Information elements (IEs). IEs are formatted data objects consisting of an ID, a length, and a data payload used to pass data between layers or devices. [IEEE802154e] defines Header IEs and Payload IEs; 6top only uses Payload IEs. A Payload IE includes one or more IEs, and ends with a termination IE (ID = 0x0f, see [IEEE802154e]).

6top uses the following Information Elements, some defined in [IEEE802154e], others introduced in this document.

Defined in [IEEE802154e] and used by 6top without modification:

TSCH Synchronization IE (Section 4.1.1.1)

TSCH Slotframe and Link IE (Section 4.1.1.2)

TSCH Timeslot Template IE (Section 4.1.1.3)

TSCH Channel Hopping IE (Section 4.1.1.4)

Defined by 6top:

6top Opcode IE (Section 4.1.1.5)

6top Bandwidth IE (Section 4.1.1.6)

6top TrackID IE (Section 4.1.1.7)

6top Generic Schedule IE (Section 4.1.1.8)

#### 4.1.1.1. TSCH Synchronization IE

A Synchronization IE (SyncIE) contains Information allowing a node to synchronize to a TSCH network, including the current ASN and a join priority. Synchronization IE MUST be included in all TSCH Enhanced Beacons.

6top re-uses this IE as defined in [IEEE802154e].

Format of a TSCH Synchronization IE (SyncIE).

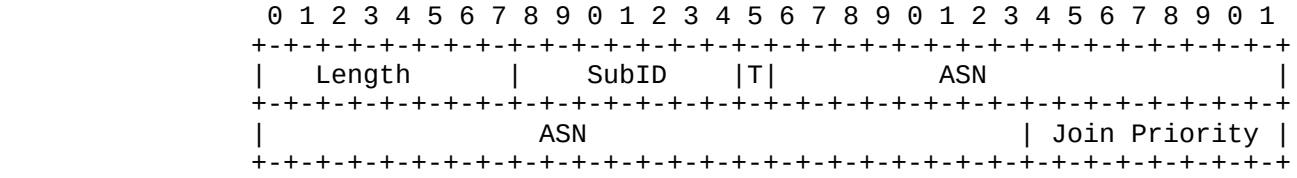



Figure 3

Length=6

SubID=0x1a

T=0, i.e., short type

ASN (5 octets) contains the Absolute Slot Number corresponding to the timeslot in which the TSCH Enhanced Beacon is sent.

The Join Priority can be used by a joining device to select among beaconing devices when multiple beacons are heard. The PAN coordinator’s join priority is zero. A lower value of join priority indicates that the device is the preferred one to connect to. As

suggested by [I-D.ietf-6tisch-minimal], the beaconing device's join priority is its DAGRank(rank).

#### 4.1.1.2. TSCH Slotframe and Link IE

The Slotframe and Link IE (FrameAndLinkIE) contains one or more slotframes and their respective cells that a beaconing device advertises to allow other devices to join the network.

6top re-uses this IE as defined in [IEEE802154e].

Format of a TSCH Slotframe and Link IE (FrameAndLinkIE).

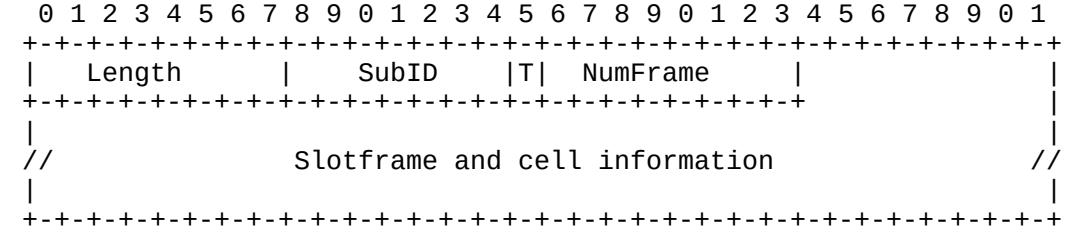



Figure 4

Length=variable

SubID=0x1b

T=0, i.e., short type

NumFrame is set to the total number of slotframe descriptors contained in the TSCH Enhanced Beacon.

Format of a slotframe descriptor.

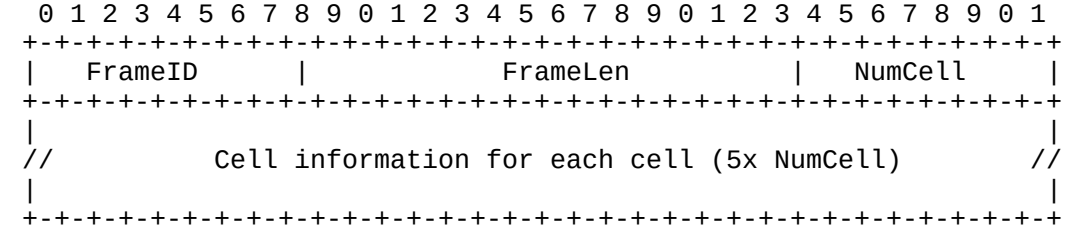



Figure 5

The FrameID field shall be set to the slotframeHandle that uniquely identifies the slotframe.

The FrameLen field shall be set to the size of the slotframe in number of timeslots.

The NumCell field shall be set to the number of cells that belong to the specific slotframe identified by the slotframeHandle.

Format of a Cell information.

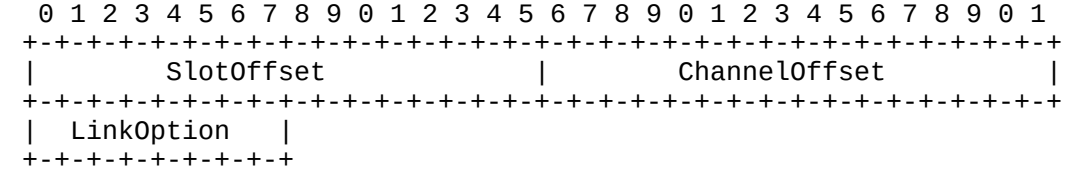



Figure 6

SlotOffset shall be set to the slotOffset of this cell.

ChannelOffset shall be set to the channelOffset of this cell.

LinkOption indicates whether this cell is a TX cell, an RX cell, or a SHARED TX cell, whether the device to which it is being linked is to be used for clock synchronization, and whether this cell is hard cell.

#### 4.1.1.3. TSCH Timeslot Template IE

Timeslot Template IE (SlotTemplateIE) defines Timeslot template being used by the TSCH device.

6top re-uses this IE as defined in [IEEE802154e].

Format of a TSCH Timeslot Template IE (SlotTemplateIE).

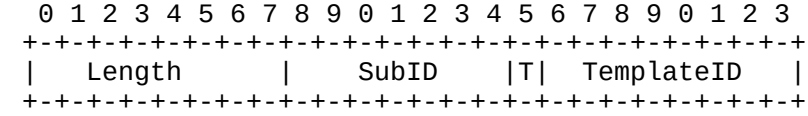



Figure 7

Length=1

SubID=0x1c

T=0, i.e., short type

TemplateID shall be set to a Timeslot template handle. The full timeslot template, which contains the macTimeslotTemplate of TSCH (total 25 octets), MAY be included.(see [IEEE802154e]).

#### 4.1.1.4. TSCH Channel Hopping IE

Channel Hopping IE (ChHoppingIE) defines the Hopping Sequence being used by the TSCH device.

6top re-uses this IE as defined in [IEEE802154e].

Format of a TSCH Channel Hopping IE (ChHoppingIE).

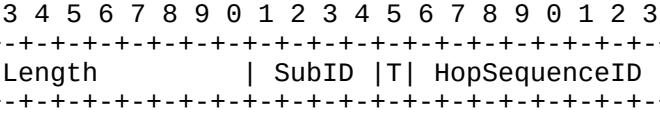



Figure 8

Length=1

SubID=0x09

T=1, i.e., long type

HopSequenceID shall be set to a Hopping Sequence handle. The full Hopping Sequence information MAY be included. (see [IEEE802154e]).

#### 4.1.1.5. 6top Opcode IE

6top Opcode IE (OpcodeIE) defines operation codes of packets in 6top sublayer.

This IE is not present in [IEEE802154e] and is defined by 6top.

Format of a 6top Opcode IE (OpcodeIE).

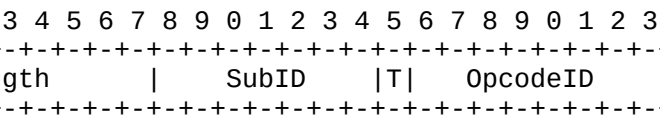



Figure 9

Length=1

SubID=0x41

T=0, i.e., short type

OpcodeID field shall be set to one of the following codes.

0x00: Reserve Soft Cell Request

0x01: Reserve Soft Cell Response

0x02: Remove Soft Cell Request

0x03: Reserve Hard Cell Request

0x04: Remove Hard Cell Request

#### 4.1.1.6. 6top Bandwidth IE

Bandwidth IE (BwIE) defines the number of cells to be reserved or actually be reserved.

This IE is not present in [IEEE802154e] and is defined by 6top.

Format of a 6top Bandwidth IE (BwIE).

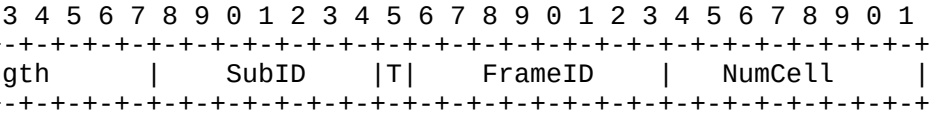



Figure 10

Length=2

SubID=0x42

T=0, i.e., short type

FrameID MAY be set to the SlotFrameHandle to identify the slotframe from which cells are reserved. FrameID field MAY be set to NOP, which means no specific slotframe is associated.

NumCell shall be set to the number of cells. When BwIE is combined with the OpecodeID of Reserve Soft Cell Request, NumCell presents how many cells are required to reserve; and when BwIE is combined with the OpecodeID of Reserve Soft Cell Response, NumCell presents how many cells are reserved successfully.

#### 4.1.1.7. 6top TrackID IE

TrackID IE (TrackIdIE) describes the track which the reserved/removed cell(s) are associated with.

This IE is not present in [IEEE802154e] and is defined by 6top.

Format of a 6top TrackID IE (TrackIdIE).

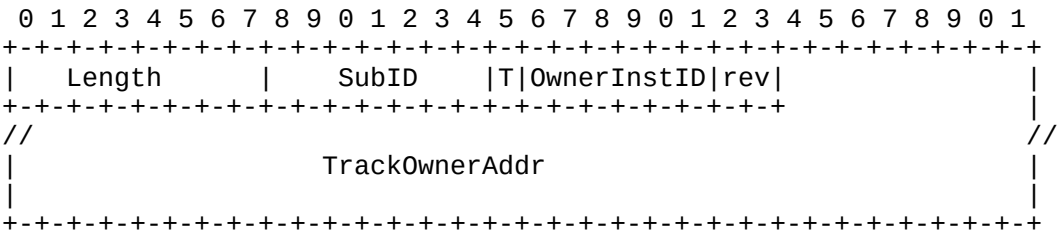



Figure 11

Length=3 or 7. When length=3, TrackOwnerAddr is 2 bytes short address, and when length=7, TrackOwnerAddr is 6 bytes long address.

SubID=0x43

T=0, i.e., short type

The combination of TrackOwnerAddr and OwnerInstId represents a specific TrackID.

#### 4.1.1.8. 6top Generic Schedule IE

Generic Schedule IE (ScheduleIE) describes cell sets. In different packets, ScheduleIE represents different information. See Section 4.1.2 for more detail.

This IE is not present in [IEEE802154e] and is defined by 6top.

Format of a 6top Generic Schedule IE (ScheduleIE).

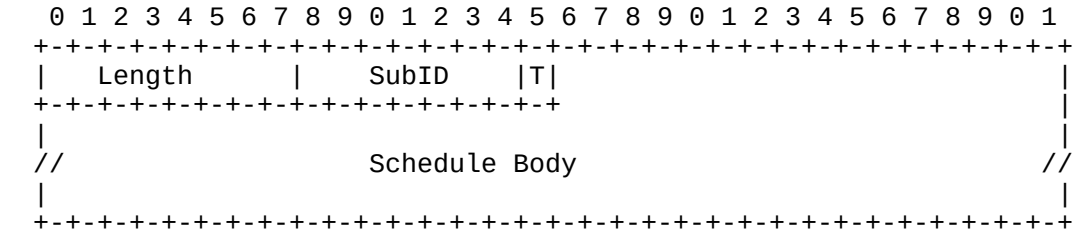



Figure 12

Length=variable

SubID=0x44

T=0, i.e., short type

Schedule Body carries one or more schedule object. An object MAY carry a TLV (Type-Length-Value), which MAY itself comprise other TLVs. TLV format is as follows. Type: 1 byte, Length: 1 byte, Value: variable

The following are some examples of schedule object TLV.

Example 1. Cell Set TLV

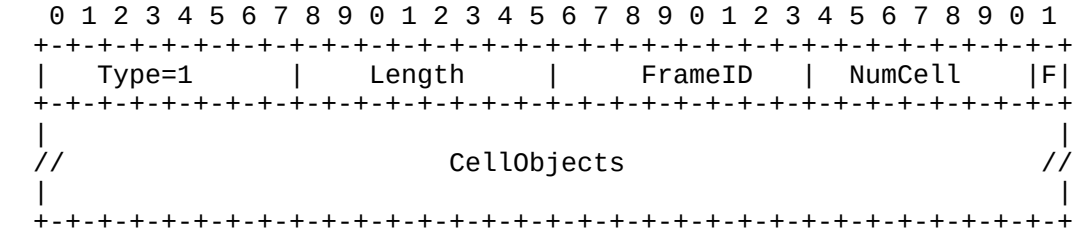



Figure 13

FrameID shall be set to the slotframeHandle that uniquely identifies the slotframe.

NumCell shall be set to the number of cells that belong to the specific slotframe identified by the slotframeHandle.

F=1 means the specified cells equals to what are listed in CellObjects, and F=0 means the specified cells equals to what are not listed in CellObjects.

CellObjects carries the information for one or more cells, including SlotOffset, ChannelOffset, LinkOption (Figure 6).

Example 2. Schedule Matrix TLV

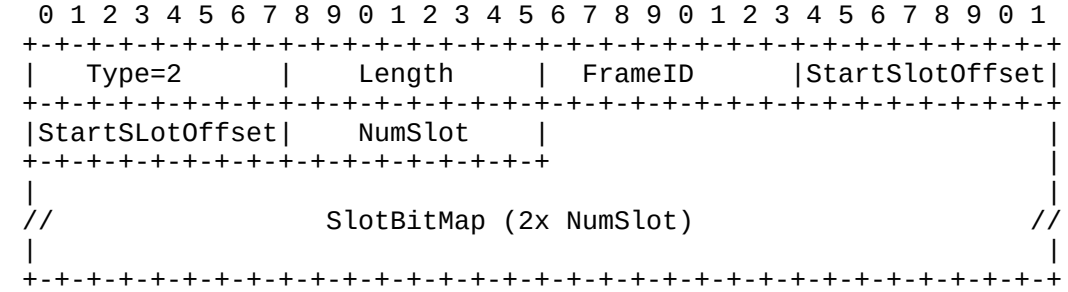



Figure 14

FrameID field MUST be set to the slotframeHandle that uniquely identifies the slotframe.

StartSlotOffset field (2 octets) MUST be set to the slotOffset in the specific slotframe identified by the slotframeHandle.

NumSlot field MUST be set to the number of timeslots from StartSlotOffset in the specific slotframe identified by the slotframeHandle.

SlotBitMap (per timeslot) indicates for the given timeslot which channels are specified. For the 16 channels in 2.4GHz band, 2-octets are used to indicate which channel is specified. For example, given a timeslot and a SlotBitmap with value (10001000,00010000); the bitmap represents that ChannelOffset-0, ChannelOffset-4, ChannelOffset-11 are specified.

#### 4.1.2. Packet Formats

This section describes the packets used in 6top to form a network, reserve/maintain bandwidth using soft cells, and reserve/remove hard cells in both the transmitter side and receiver sides. Each of these packets uses one or more IEs defined in Section 4.1.1.

##### 4.1.2.1. TSCH Enhanced Beacon

The TSCH Enhanced Beacon is used to announce the presence of the network and allows new nodes to join. It is an Enhanced Beacon packet defined in [IEEE802154e] with the following Payload IEs:

TSCH Synchronization IE (Section 4.1.1.1)

TSCH Timeslot Template IE (Section 4.1.1.3)

TSCH Channel Hopping IE (Section 4.1.1.4)

TSCH Slotframe and Link IE (Section 4.1.1.2)

Payload IE of TSCH Enhanced Beacon Packet

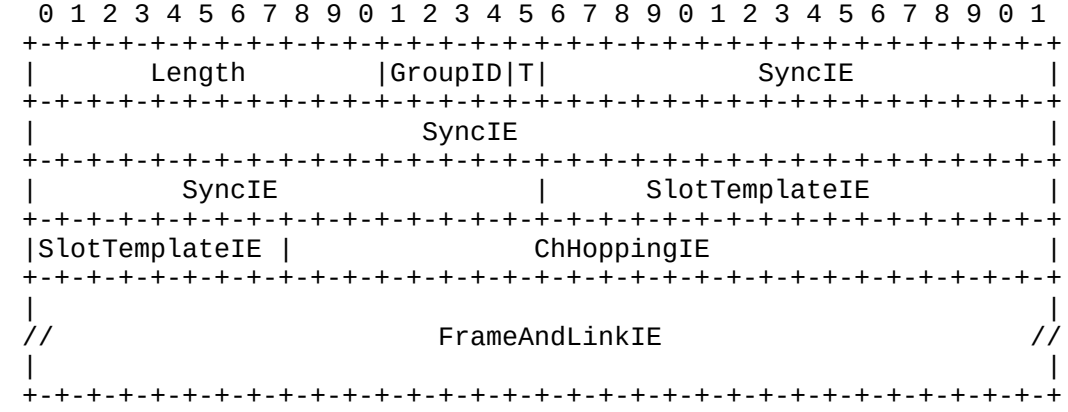



Figure 15

Length=variable

GroupID=0x1, i.e., MLME IE

T=1, i.e., payload IE

See Section 4.1.1.1, Section 4.1.1.3, Section 4.1.1.4,Section 4.1.1.2 for SyncIE, SlotTemplateIE, ChHoppingIE and FrameAndLinkIE.

#### 4.1.2.2. Soft Cell Reservation Request

A Soft Cell Reservation Request packet is a DATA packet defined in [IEEE802154e] with the following payload IE.

Payload IE of Soft Cell Reservation Request

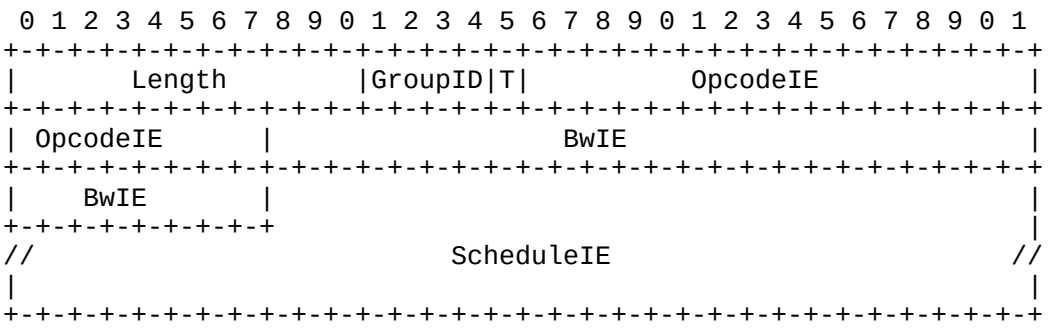



Figure 16

Length=variable

GroupID=0x1, i.e., MLME IE

T=1, i.e., payload IE

The OpcodeID field in the 3-octet OpcodeIE SHOULD be set to 0x00, indicates Reserve Soft Cell Request operation.

The NumCell field in 4-octet BwIE SHOULD be set to the number of cells needed to be reserved.

The ScheduleIE specifies a candidate cell set, from which the cells SHOULD be reserved. ScheduleIE MAY be empty, means there is no constrain on which cells SHOULD not be reserved.

In addition, TrackIdIE can be added in the packet to associate the reserved soft cells to a specific TrackID.

#### 4.1.2.3. Soft Cell Reservation Response

Soft Cell Reservation Response is a DATA packet defined in [IEEE802154e] with the following payload IE.

Payload IE of Soft Cell Reservation Response

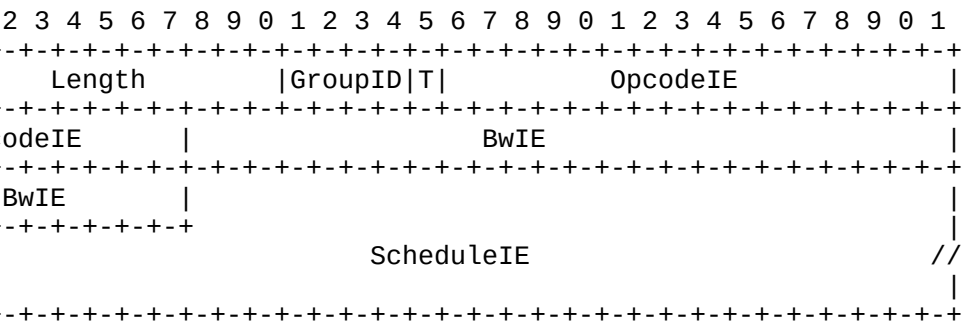



Figure 17

Length=variable

GroupID=0x1, i.e., MLME IE

T=1, i.e., payload IE

The OpcodeID field in the 3-octet OpcodeIE SHOULD be set to 0x01, indicates Reserve Soft Cell Response operation.

The NumCell field in 4-octet BwIE SHOULD be set to the number of cells which have been reserved successfully.

The ScheduleIE SHOULD specify all of the cells which have been reserved successfully.

In addition, TrackIdIE can be added in the packet to associate the reserved soft cells to a specific TrackID.

4.1.2.4. Soft Cell Remove Request

Soft Cell Remove Request is a DATA packet defined in [IEEE802154e] with the following payload IE.

Payload IE of Soft Cell Remove Request

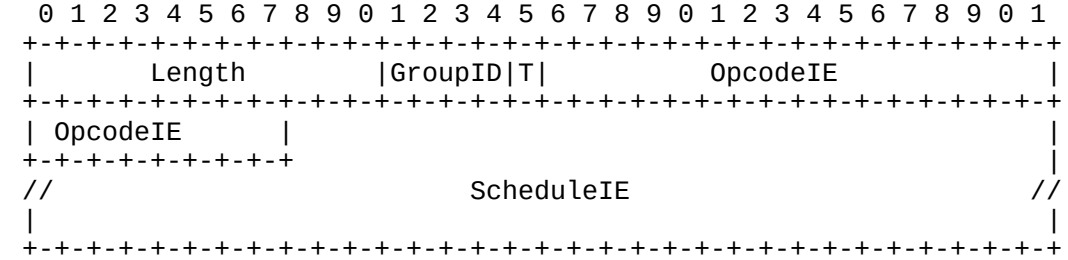



Figure 18

Length=variable

GroupID=0x1, i.e., MLME IE

T=1, i.e., payload IE

The OpcodeID field in the 3-octet OpcodeIE SHOULD be set to 0x02, indicates Remove Soft Cell Request operation.

The ScheduleIE SHOULD specify all the cells that need to be removed.

#### 4.1.2.5. Hard Cell Reservation Request

Hard Cell Reservation Request packet is a DATA packet defined in [IEEE802154e] with the following payload IE.

Payload IE of Hard Cell Reservation Request

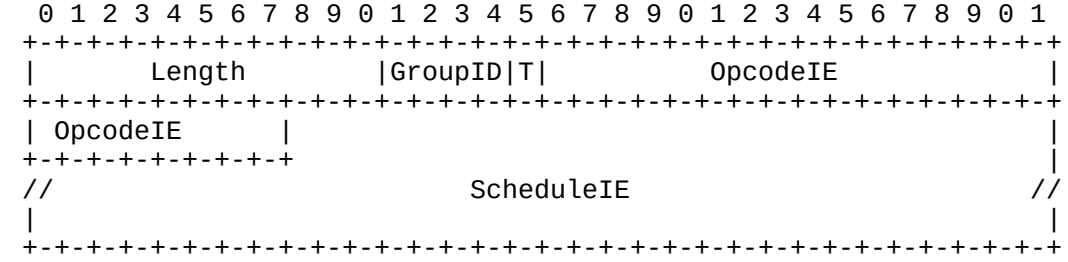



Figure 19

Length=variable

GroupID=0x1, i.e., MLME IE

T=1, i.e., payload IE

The OpcodeID field in the 3-octet OpcodeIE SHOULD be set to 0x03, indicates Reserve Hard Cell Request operation.

The ScheduleIE SHOULD specify all the cell that need to be reserved.

In addition, TrackIdIE can be added in the packet to associate the reserved hard cells to a specific TrackID.

#### 4.1.2.6. Hard Cell Remove Request

Hard Cell Remove Request is a DATA packet defined in [IEEE802154e] with the following payload IE.

Payload IE of Hard Cell Remove Request

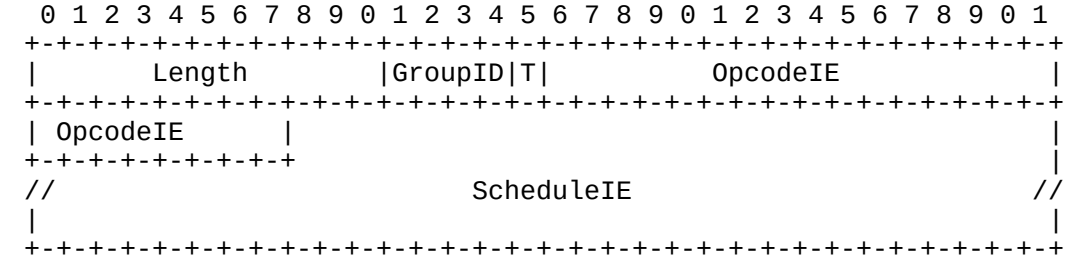



Figure 20

Length=variable

GroupID=0x1, i.e., MLME IE

T=1, i.e., payload IE

The OpcodeID field in the 3-octet OpcodeIE SHOULD be set to 0x04, indicates Remove Hard Cell Request operation.

The ScheduleIE SHOULD specify all the cells that need to be removed.

### 4.2. Time Sequences

6top neighbors exchange 6top-specific packets in the following cases, each detailed in a subsection.

Network formation (Section 4.2.1)

Creating soft cells (Section 4.2.2)

Deleting soft cells (Section 4.2.3)

Maintaining soft cells (Section 4.2.4)

Creating hard cells (Section 4.2.5)

Deleting hard cells (Section 4.2.6)

#### 4.2.1. Network Formation

Network formation consists of two processes: joining and maintenance.

##### 4.2.1.1. Joining

A node already in the network sends out TSCH Enhanced Beacons periodically.

When a node is joining an existing network, it listens for TSCH Enhanced Beacons. After collecting one or more TSCH Enhanced BEACONs (the format of which is detailed in Section 4.1.2.1), the joining node MUST do the following.

Initialize a neighbor table. Establish a neighbor table and record all of the information described in the TSCH Enhanced BEACONs as its initial schedule with those neighbors.

Select a time source neighbor. According to the Joining Priority described by SyncIEs, the joining node chooses time source neighbors. 6top does not specify the criteria to choose time source neighbors from the Enhanced BEACONs.

Select cells for Enhanced Beacons. The joining node selects one or more cells to indicate in its own Enhanced Beacons, which MAY be the same as the cells used by its neighbors for Enhanced Beacon broadcast, and record those cell(s) into the TSCH schedule with LinkType=ADVERTISING.

Its Enhanced Beacons SHOULD include the cell(s) selected for EB purposes. The EB cells MUST be configured with LinkOption to "Receive" and "Timekeeping", telling its neighbors that the cell is used for broadcast.

Start broadcasting Enhanced Beacon and communicate with neighbors.

#### 4.2.1.2. Maintenance

Nodes MAY broadcast Enhance Beacons on the cells marked with LinkType=ADVERTISING, and listen for Enhanced Beacons from neighbors on the cells with LinkOptions "Receive" and "Timekeeping". If a cell with LinkType=ADVERTISING has both the "Receive" and "Timekeeping" LinkOptions set, which means that the cell is shared by neighbors and itself for broadcasting, then broadcasting Enhanced Beacon has higher priority.

Whenever a node receives an Enhanced Beacon, it SHOULD update its schedule if there is a difference regarding to the cells used for synchronizing with the advertiser of the Enhanced Beacon.

### 4.2.2. Creating soft cells

The upper layer instructs 6top to schedule one or more soft cells by calling the Create soft cell command. This command can also be called by the monitoring process internal to 6top.

When receiving a Create soft cell command, Node A's 6top sublayer forms a Soft Cell Reservation Request packet which includes the BwIE and ScheduleIE Information Elements. The BwIE indicates the number of cells to be reserved (N1); the ScheduleIE indicates set of a candidate cells from which the new cells SHOULD be selected. If the ScheduleIE is empty, Node A indicates there is no constraint on cell selection.

The Soft Cell Reservation Request is sent to the neighbor (Node B) with whom new cells need to be scheduled. After receiving the Soft Cell Reservation Request, Node B selects the cells from the candidate cell set defined by the ScheduleIE in the Soft Cell Reservation Request, and forms a Soft Cell Reservation Response packet. In the Cell Reservation Response packet, the BwIE indicates the number of cells actually being reserved (N2); the ScheduleIE indicates those reserved cells. If N2 is smaller than N1, node B indicates to node A that there are not enough qualified cells to be reserved. Node B MUST record the reserved cells into its local schedule when sending the Soft Cell Reservation Response. After receiving the Soft Cell Reservation Response, Node A MUST record the reserved cells into its local schedule.

The policy to build a candidate cell set and the policy to select cells from the candidate cell set to reserve are out of scope.

The format of Schedule Body is flexible. For example, Node A can use Cell Set TLV defined in Figure 13 with field 'F' set to '0', and the CellObjects includes all of the cells being used by Node A. In

another word, the cell candidate set is all of the cells not being included in the list defined by CellObjects.

The behavior of the nodes when the soft cells negotiation fails is out of scope.

#### 4.2.3. Deleting soft cells

The upper layer instructs 6top to delete one or more soft cells by calling the Delete soft cell command (Section 3.1.6). This command can also be called by the monitoring process internal to 6top (Section 6).

When receiving a Delete soft cell command, Node A's 6top sublayer selects cells to be removed from its local schedule, and creates a Soft Cell Remove Request, which includes a ScheduleIE Information Element. The ScheduleIE indicates which specific cells to remove with a neighbor (Node B). The cells specified in the ScheduleIE SHOULD be removed from local schedule of Node A when the Soft Cell Remove Request is sent to Node B. When receiving the Soft Cell Remove Request, the cells specified in the ScheduleIE SHOULD be removed from the local schedule of Node B.

The policy to select cells corresponding to a Delete soft cell command is out of scope.

#### 4.2.4. Maintaining soft cells

The monitoring process internal to 6top (Section 6) is responsible for monitoring and re-scheduling soft cells to meet some QoS requirements. The monitoring process MAY issue a soft cell Maintenance command, which indicate a set of cells to be re-allocated in the TSCH schedule.

When receiving a soft cell Maintenance command, 6top initializes a Soft Cell Remove Request (Section 4.2.3) with the neighbor in question, followed by a Soft Cell Reservation Request (Section 4.2.2).

#### 4.2.5. Creating hard cells

The upper layer instructs 6top to create one or more hard cells by calling the Create hard cell command.

When receiving a Create hard cell command, Node A's 6top sublayer creates a Hard Cell Reservation Request, including a ScheduleIE. The ScheduleIE indicates which specific cells with a neighbor (Node B) to be added. The cells specified in the ScheduleIE SHOULD be added in

local schedule of Node A while the Hard Cell Reserve Request is sent to Node B. When receiving the Hard Cell Reserve Request, the cells specified in the ScheduleIE SHOULD be added in the local schedule of Node B.

#### 4.2.6. Deleting hard cells

The upper layer instructs 6top to delete one or more hard cells by calling the Delete hard cell command.

When receiving a Delete hard cell command, Node A's 6top sublayer creates a Hard Cell Remove Request, including a ScheduleIE. The ScheduleIE indicates which specific cells with a neighbor (Node B) to be removed. The cells specified in the ScheduleIE SHOULD be removed from local schedule of Node A while the Hard Cell Remove Request is sent to Node B. When receiving the Hard Cell Remove Request, the cells specified in the ScheduleIE SHOULD be removed from the local schedule of Node B.

## 5. Statistics

The 6top Statistics Function (SF) is responsible for collecting statistics, which it can provide to an upper layer and the Monitoring Function (Section 6).

### 5.1. Statistics Metrics

6top is in charge of keeping statistics from a set of metrics gathered from the behavior of the TSCH layer.

The statistics data related to node states and cell metrics SHOULD be provided to upper layer for management, e.g., for RPL to calculate the node's Rank or for GMPLS to the required bandwidth is met. The specific algorithm to generate the statistics is out of scope. However, the statistics component SHOULD include the following metrics:

1. LinkThroughput: associated with a link, Node A->Node B. For example, LinkThroughput can be calculated with: SUM(NumOfCell(i)*NumOfBytePerPacket)/(FrameLen(i)*SlotDuration) where NumOfCell(i) is the total number of cells from Node A to Node B in Slotframe-i, FrameLen(i) is the length of Slotframe-i. The unit is Byte/second.

2. Latency: associated with a link, Node A->Node B. For example, latency can be expressed as Minimum and Maximum Latency. Minimum Latency = Min(MinNumOfSlot(i),i=1..) * SlotDuration and Maximum Latency = Max(MaxNumOfSlot(i),i=1..) * SlotDuration where,

MinNumOfSlot(i) and MaxNumOfSlot(i) are the minimum or maximum number of timeslots between two dedicated cells from Node A to Node B in Slotframe-i, respectively.

3. LinkQuality. For example, average LQI, ETX, PDR, RSSI.

4. TrafficLoad. For example, Queue Full Rate, Queue Empty Rate.

5. NodeEnergy. For example, E_E=E_bat / [E_0 (T-t)/T].

## 5.2. Statistics Configuration

The Statistics Function SHOULD be configurable. The configuration parameters SHOULD include:

LinkQualityStatisticsEn

TafficLoadStatisticsEn

DeviceStatisticsEn

6top statistics function is enabled/disabled and configured by the commands defined in Section 3.4

# 6. Monitoring

The 6top Monitoring Function (MF) is responsible for monitoring cell quality, traffic load, and issuing soft cell Maintenance commands, or Create/Delete soft cell commands. The data provided by the Statistics Function MAY be used as an input of MF in taking a monitoring decision.

## 6.1. Monitor Configuration

Monitoring Function SHOULD be configurable. The configuration parameters SHOULD include:

MaintainCellEn.

CreateDeleteCellEn.

QosLevel. QosLevel SHOULD associate with specific neighbor address. QosLevel MAY reflect the latency constraint, cell quality constraint, and so on. The value of QosLevel works as the bandwidth redundancy coefficient.

The 6top monitoring function is enabled/disabled and configured by the commands defined in Section 3.3

### 6.2. Actuation

The cell quality statistics MAY be used to generate soft a cell Maintenance command, which triggers a soft cell Maintenance procedure (see Section 4.2.4). The traffic load statistics MAY be used to generate internal Create (resp. Delete) soft cell commands, which trggiers a soft cell Reservation (resp. Remove) process (see Section 4.2.2 and Section 4.2.3).

The policy to generate the soft cell Maintenance command and the policy to generate Create/Delete soft cell commands is out of scope.

The policy to generate Create/Delete soft cell commands MAY take QosLevel into account. For example, there are two slotframes existing, Slotframe-1 consists of 32 timeslots, Slotframe-2 consists of 96 timeslots; timeslot duration is 10ms; QosLevel=1.5. If, from the traffic load statistics, MF determines that 2 packet/second SHOULD be added, then the MF generates a Create soft cell command, where FrameID=2, NumCell=3.

## 7. References

### 7.1. Normative References

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

### 7.2. Informative References

[I-D.ietf-6tisch-tsch]
Watteyne, T., Palattella, M., and L. Grieco, "Using IEEE802.15.4e TSCH in an LLN context: Overview, Problem Statement and Goals", draft-ietf-6tisch-tsch-00 (work in progress), November 2013.

[I-D.ietf-6tisch-architecture]
Thubert, P., Watteyne, T., and R. Assimiti, "An Architecture for IPv6 over the TSCH mode of IEEE 802.15.4e", draft-ietf-6tisch-architecture-01 (work in progress), February 2014.

[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-01 (work in progress), February 2014.

[I-D.ietf-6tisch-minimal]
Vilajosana, X. and K. Pister, "Minimal 6TiSCH Configuration", draft-ietf-6tisch-minimal-00 (work in progress), November 2013.

[I-D.ohba-6tsch-security]
Chasko, S., Das, S., Lopez, R., Ohba, Y., Thubert, P., and A. Yegin, "Security Framework and Key Management Protocol Requirements for 6TSCH", draft-ohba-6tsch-security-01 (work in progress), July 2013.

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

## 7.3. 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.

[label-switching-154e]
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