detnet
Internet-Draft
Intended status: Standards Track
Expires: April 26, 2015

N. Finn
P. Thubert
Cisco
October 23, 2014

Deterministic Networking Problem Statement
draft-finn-detnet-problem-statement-01

Abstract

This paper documents the needs in various industries to establish multi-hop paths for characterized flows with deterministic properties .

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Operational Technology (OT) refers to industrial networks that are typically used for monitoring systems and supporting control loops, as well as movement detection systems for use in process control (i.e., process manufacturing) and factory automation (i.e., discrete manufacturing). Due to its different goals, OT has evolved in parallel but in a manner that is radically different from IT/ICT, focusing on highly secure, reliable and deterministic networks, with limited scalability over a bounded area.

The convergence of IT and OT technologies, also called the Industrial Internet, represents a major evolution for both sides. The work has already started; in particular, the industrial automation space has been developing a number of Ethernet-based replacements for existing digital control systems, often not packet-based (fieldbus technologies).

These replacements are meant to provide similar behavior as the incumbent protocols, and their common focus is to transport a fully characterized flow over a well-controlled environment (i.e., a factory floor), with a bounded latency, extraordinarily low frame loss, and a very narrow jitter. Examples of such protocols include PROFINET, ODVA Ethernet/IP, and EtherCAT.

In parallel, the need for determinism in professional and home audio/video markets drove the formation of the Audio/Video Bridging (AVB) standards effort of IEEE 802.1. With the explosion of demand for connectivity and multimedia in transportation in general, the Ethernet AVB technology has become one of the hottest topics, in particular in the automotive connectivity. It is finding application in all elements of the vehicle from head units, to rear seat entertainment modules, to amplifiers and camera modules. While aimed at less critical applications than some industrial networks, AVB networks share the requirement for extremely low packet loss rates and guaranteed finite latency and jitter.

Other instances of in-vehicle deterministic networks have arisen as well for control networks in cars, trains and buses, as well as avionics, with, for instance, the mission-critical "Avionics Full-Duplex Switched Ethernet" (AFDX) that was designed as part of the ARINC 664 standards. Existing automotive control networks such as the LIN, CAN and FlexRay standards were not designed to cover these increasing demands in terms of bandwidth and scalability that we see with various kinds of Driver Assistance Systems (DAS) and new multiplexing technologies based on Ethernet are now getting traction.

The generalization of the needs for more deterministic networks have led to the IEEE 802.1 AVB Task Group becoming the Time-Sensitive Networking (TSN) Task Group (TG), with a much-expanded constituency from the industrial and vehicular markets. Along with this expansion, the networks in consideration are becoming larger and structured, requiring deterministic forwarding beyond the LAN boundaries. For instance, Industrial Automation segregates the network along the broad lines of the Purdue Enterprise Reference Architecture (PERA), using different technologies at each level, and public infrastructures such as Electricity Automation require deterministic properties over the Wide Area. The realization is now coming that the convergence of IT and OT networks requires Layer-3, as well as Layer-2, capabilities.

In order to serve this extended requirement, the IETF and the IEEE must collaborate and define an abstract model that can be applicable both at Layer-2 and Layer-3, and along segments of different technologies. With this new work, a path may span, for instance, across a (limited) number of 802.1 bridges and then a (limited) number of IP routers. In that example, the IEEE802.1 bridges may be operating at Layer-2 over Ethernet whereas the IP routers may be 6TiSCH nodes operating at Layer-2 and/or Layer-3 over the IEEE802.15.4e MAC.

The proposed model should enable a fully scheduled operation orchestrated by a central controller, as well as a more distributed operation with probably lesser capabilities. In any fashion, the model should not compromise the ability of a network to keep carrying the sorts of traffic that is already carried today.

Once the abstract model is agreed upon, the IETF will need to specify the signaling elements to be used to establish a path and the tagging elements to be used identify the flows that are to be forwarded along that path. The IETF will also need to specify the necessary protocols, or protocol additions, based on relevant IETF technologies such as PCE, MPLS and 6TiSCH, to implement the selected model. As a result of this work, it will be possible to establish a multi-hop path over the IP network, for a particular flow with precise timing and throughput requirements, and carry this particular flow along the multi-hop path with such characteristics as low latency and ultra-low jitter, duplication and elimination of packets over non-congruent paths for a higher delivery ratio, and/or zero congestion loss. Depending on the network capabilities and on the current state, requests to establish a path by an end-node or a network management entity may be granted or rejected, and an existing path may be moved or removed.

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. On Deterministic Networking

The Internet is not the only digital network that has grown dramatically over the last 30-40 years. Video and audio entertainment, and control systems for machinery, manufacturing processes, and vehicles are also ubiquitous, and are now based almost entirely on digital technologies. Over the past 10 years, engineers in these fields have come to realize that significant advantages in both cost and in the ability to accelerate growth can be obtained by basing all of these disparate digital technologies on packet networks.

The goals of Deterministic Networking are to enable the migration of applications that use special-purpose fieldbus technologies (HDMI, CANbus, ProfiBus, etc...even RS-232!) to packet technologies in general, and the Internet Protocol in particular, and to support both these new applications, and existing packet network applications, over the same physical network.

Considerable experience ([ODVA],[AVnu], [Profinet],[HSR-PRP], etc...) has shown that these applications need a some or all of a suite of features that includes:

1. Time synchronization of all host and network nodes (routers and/or bridges), accurate to something between 10 nanoseconds and 10 microseconds, depending on the application.

2. Support for critical packet flows that:

   * Can be unicast or multicast;

   * Need absolute guarantees of minimum and maximum latency end-to-end across the network;

   * Need a packet loss ratio in the range of 1.0e-9 to 1.0e-12, or better;

   * Can, in total, absorb more than half of the network's available bandwidth (that is, over-provisioning is ruled out as a solution);

   * Cannot suffer throttling, congestion feedback, or any other network-imposed transmission delay, although the flows can be meaningfully characterized either by a fixed, repeating transmission schedule, or by a maximum bandwidth and packet size.

3. Multiple methods to schedule, shape, limit, and otherwise control the transmission of critical packets at each hop through the network data plane.

4. Robust defenses against misbehaving hosts, routers, or bridges, both in the data and control planes.

5. One or more methods to reserve resources in bridges and routers to carry these flows.

Time synchronization techniques need not be addressed by an IETF Working Group; there are a number of standards available for this purpose, including IEEE 1588, IEEE 802.1AS, and more.

The multicast, latency, loss ratio, and non-throttling needs are made necessary by the algorithms employed by the applications. They are not simply the transliteration of fieldbus needs to a packet-based fieldbus simulation, but reflect fundamental mathematics of the control of a physical system.

When forwarding latency- and loss-sensitive packets across a network, interactions among different critical flows introduce fundamental uncertainties in delivery schedules. The details of the queuing, shaping, and scheduling algorithms employed by each bridge or router to control the output sequence on a given port affect the detailed makeup of the output stream, e.g. how finely a given flow's packets are mixed among those of other flows.

This, in turn, has a strong effect on the buffer requirements, and hence the latency guarantees deliverable, by the next bridge or router along the path. For this reason, the IEEE 802.1 Time-Sensitive Networking Task Group has defined a set of queuing, shaping, and scheduling algorithms (:::reference to section, below :::) that enable each bridge or router to compute the exact number of buffers to be allocated for each flow or class of flows. The present authors assume that these techniques will be used by the DetNet Working Group.

Robustness is a common need for networking protocols, but plays a more important part in real-time control networks, where expensive equipment, and even lives, can be lost due to misbehaving equipment.

Reserving resources before packet transmission is the one fundamental shift in the behavior of network applications that is impossible to avoid. In the first place, a network cannot deliver finite latency and practically zero packet loss to an arbitrarily high offered load. Secondly, achieving practically zero packet loss for unthrottled (though bandwidth limited) flows means that bridges and routers have to dedicate buffer resources to specific flows or to classes of flows. The requirements of each reservation have to be translated into the parameters that control each host's, bridge's, and router's queuing, shaping, and scheduling functions and delivered to the hosts, bridges, and routers.

4. Related IETF work

4.1. Deterministic PHB

[I-D.svshah-tsvwg-deterministic-forwarding] 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].

4.2. 6TiSCH

Industrial process control already leverages deterministic wireless Low power and Lossy Networks (LLNs) to interconnect critical resource-constrained devices and form wireless mesh networks, with standards such as [ISA100.11a] and [WirelessHART].

These standards rely on variations of the [IEEE802154e] timeSlotted Channel Hopping (TSCH) [I-D.ietf-6tisch-tsch] Medium Access Control (MAC), and a form of centralized Path Computation Element (PCE), to deliver deterministic capabilities.

The TSCH MAC benefits include high reliability against interference, low power consumption on characterized flows, and Traffic Engineering capabilities. Typical applications are open and closed control loops, as well as supervisory control flows and management.

The 6TiSCH Working Group focuses only on the TSCH mode of the IEEE802.15.4e standard. The WG currently defines a framework for managing the TSCH schedule. Future work will standardize deterministic operations over so-called tracks as described in [I-D.ietf-6tisch-architecture]. Tracks are an instance of a deterministic path, and the detnet work is a prerequisite to specify track operations and serve process control applications.

[RFC5673] and [I-D.ietf-roll-rpl-industrial-applicability] section 2.1.3. and next discusses application-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 for improved reliability.

5. Problem Statement

5.1. Flow Characterization

Deterministic forwarding can only apply on flows with well-defined characteristics such as periodicity and burstiness. Before a path can be established to serve them, the expression of those characteristics, and how the network can serve them, for instance in shaping and forwarding operations, must be specified.

5.2. Centralized Path Computation and Installation

A centralized routing model, such as provided with a PCE, enables global and per-flow optimizations. The model is attractive but a number of issues are left to be solved. In particular:

o whether and how the path computation can be installed by 1) an end device or 2) a Network Management entity,

o and how the path is set up, either by installing state at each hop with a direct interaction between the forwarding device and the

PCE, or along a path by injecting a source-routed request at one end of the path.

5.3. Distributed Path Setup

Whether a distributed alternative without a PCE can be valuable should be studied as well. Such an alternative could for instance inherit from the Resource ReSerVation Protocol [RFC5127] (RSVP) flows.

6. Security Considerations

Security in the context of Deterministic Networking has an added dimension; the time of delivery of a packet can be just as important as the contents of the packet, itself. A man-in-the-middle attack, for example, can impose, and then systematically adjust, additional delays into a link, and thus disrupt or subvert a real-time application without having to crack any encryption methods employed. See [RFC7384] for an exploration of this issue in a related context.

Security must cover:

- o the protection of the signaling protocol
- o the authentication and authorisation of the controlling nodes
- o the identification and shaping of the flows

7. IANA Considerations

This document does not require an action from IANA.

8. Acknowledgements

The authors wish to thank Jouni Korhonen, Erik Nordmark, George Swallow, Rudy Klecka, Anca Zamfir, David Black, Thomas Watteyne, Shitanshu Shah, Craig Gunther, Rodney Cummings, Wilfried Steiner, Marcel Kiessling, Karl Weber, Ethan Grossman and Pat Thaler, for their various contribution with this work.

9. References

9.1. Normative References

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

9.2. Informative References

[AVnu] http://www.avnu.org/, "The AVnu Alliance tests and certifies devices for interoperability, providing a simple and reliable networking solution for AV network implementation based on the Audio Video Bridging (AVB) standards.", .

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

[HSR-PRP] IEC, "High availability seamless redundancy (HSR) is a further development of the PRP approach, although HSR functions primarily as a protocol for creating media redundancy while PRP, as described in the previous section, creates network redundancy. PRP and HSR are both described in the IEC 62439 3 standard.", .

[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-03 (work in progress), July 2014.

[I-D.ietf-6tisch-tsch]
Watteyne, T., Palattella, M., and L. Grieco, "Using IEEE802.15.4e TSCH in an IoT context: Overview, Problem Statement and Goals", draft-ietf-6tisch-tsch-02 (work in progress), October 2014.

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

[I-D.svshah-tsvwg-deterministic-forwarding]
Shah, S. and P. Thubert, "Deterministic Forwarding PHB", draft-svshah-tsvwg-deterministic-forwarding-02 (work in progress), September 2014.

[IEEE802.1AS-2011]
IEEE, "Timing and Synchronizations (IEEE 802.1AS-2011)", 2011, <http://standards.ieee.org/getieee802/download/802.1AS-2011.pdf>.

[IEEE802.1BA-2011]
IEEE, "AVB Systems (IEEE 802.1BA-2011)", 2011, <http://standards.ieee.org/getieee802/download/802.1BA-2011.pdf>.

[IEEE802.1Q-2011]
IEEE, "MAC Bridges and VLANs (IEEE 802.1Q-2011", 2011, <http://standards.ieee.org/getieee802/download/802.1Q-2011.pdf>.

[IEEE802.1Qat-2010]
IEEE, "Stream Reservation Protocol (IEEE 802.1Qat-2010)", 2010, <http://standards.ieee.org/getieee802/download/802.1Qat-2010.pdf>.

[IEEE802.1Qav]
IEEE, "Forwarding and Queuing (IEEE 802.1Qav-2009)", 2009, <http://standards.ieee.org/getieee802/download/802.1Qav-2009.pdf>.

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

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

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

[ISA100.11a]
ISA/IEC, "ISA100.11a, Wireless Systems for Automation, also IEC 62734", 2011, < http://www.isa100wci.org/en-US/Documents/PDF/3405-ISA100-WirelessSystems-Future-broch-WEB-ETSI.aspx>.

[ODVA] http://www.odva.org/, "The organization that supports network technologies built on the Common Industrial Protocol (CIP) including EtherNet/IP.", .

[Profinet]
http://us.profinet.com/technology/profinet/, "PROFINET is a standard for industrial networking in automation.", <http://us.profinet.com/technology/profinet/>.

[RFC2205] Braden, B., Zhang, L., Berson, S., Herzog, S., and S. Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", RFC 2205, September 1997.

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

[RFC5673] Pister, K., Thubert, P., Dwars, S., and T. Phinney, "Industrial Routing Requirements in Low-Power and Lossy Networks", RFC 5673, October 2009.

[RFC7384] Mizrahi, T., "Security Requirements of Time Protocols in Packet Switched Networks", RFC 7384, October 2014.

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

Authors' Addresses

Norm Finn
Cisco Systems
510 McCarthy Blvd
SJ-24
Milpitas, California 95035
USA

Phone: +1 925 980 6430
Email: nfinn@cisco.com

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

Phone: +33 4 97 23 26 34
Email: pthubert@cisco.com

6TiSCH   Q. Wang, Ed.
Internet-Draft   Univ. of Sci. and Tech. Beijing
Intended status: Informational   X. Vilajosana
Expires: April 30, 2015   Universitat Oberta de Catalunya
                                                              T. Watteyne
                                                        Linear Technology
                                                         October 27, 2014

6TiSCH Operation Sublayer (6top) Interface
draft-ietf-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 of commands for the internal use of the 6top sublayer and or to be used by implementers as an API guideline or basic specification.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document defines a generic data model for the 6TiSCH Operation Sublayer (6top), using the YANG data modeling language defined in [RFC6020]. This data model can be used for future network management solutions defined by the 6TiSCH working group. This 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. Requirements Language

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

3. 6TiSCH Operation Sublayer (6top) Overview

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

Protocol Stack

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

Figure 1

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

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

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

The commands of the management and data interfaces are listed in Section 5. This set of commands is designed to support decentralized, centralized and hybrid scheduling solutions.

3.1. Cell Model

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

b0 = Transmit

b1 = Receive

b2 = Shared

b3 = Timekeeping

b4-b7 = Reserved

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

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

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

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

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

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

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

3.1.1. hard cells

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

3.1.2. soft cells

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

3.2. Data Transfer Model

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

6top Data Transfer Model

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

Figure 2

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

When 6top receives a packet to transmit through a Send.data command (Section 5), the I-MUX module selects a queue in which to insert it. If the packet’s destination address is a unicast (resp. broadcast) address, it will be inserted into a unicast (resp. broadcast) queue.

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

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

> The frame’s layer 2 destination address MUST match the neighbor address associated with the transmit cell.
>
> If the transmit cell is associated with a specific track, the frames in the queue corresponding to the TrackID have the highest priority.
>
> If the transmit cell is not associated with a specific track, i.e., TrackID=(0,0), frames from a queue with a higher priority MUST be sent before frames from a queue with a lower priority.

Further rules can be configured to satisfy specific QoS requirements.

4. Generic Data Model

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

4.1. YANG model of the 6top MIB

```
      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
            "SlotframeID, one in SlotframeList, indicates the slotframe
            the cell belongs to.";
            reference
            "IEEE802154e";
         }
         leaf SlotOffset {
            type uint16;
            description
            "Defined in IEEE802154e.";
            reference
            "IEEE802154e";
         }
         leaf ChannelOffset {
            type uint16;
            description
            "Defined in IEEE802154e.";
            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 one in the TrackList, pointing to a tuple
            (TrackOwnerAddr,InstanceID) , where TrackOwnerAddr is the
            address of the node which initializes the process of
            creating the track, i.e., the owner of the track; and
            InstanceID is an instance identifier given by the owner of
            the track.";
         }
         container Statistic {
            leaf NumOfStatistic {
               type uint8;
               description
               "Number of statistics collected on the cell";
            }
            list MeasureList {
               key "StatisticsMetricsID";
               leaf StatisticsMetricsID{
                  type uint16;
                  description
                  "An index of StatisticsMetricList, which defines how
                  to collect data and get the statistics value";
               }
               leaf StatisticsValue{
                  type uint16;
                  config false;
                  description
                  "updated by 6top according to the statistics method
                  specified by StatisticsMetricsID";
               }
            }
         }
      }

   list SlotframeList {
      key "SlotframeID";
      description
      "List of all of the slotframes used by the node.";

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

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

         leaf MonitoringStatusID {
            type uint16;
         }
         leaf SlotframeID {
            type uint8;
            description
            "SlotframeID, one in SlotframeList, indicates the slotframe
            being monitored";
            reference
            "IEEE802154e";
         }
         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. It indicates the communication
            link being mornitored";
         }
         leaf EnforcePolicy {
            type enumeration {
               enum DISABLE;
               enum BESTEFFORT;
               enum STRICT;
               enum OVERPROVISION;
            }
            description
            "Currently enforced QoS policy. DISABLE-no QoS;
            BESTEFFORT- best effort policy is used; STRICT- Strict
            Priority Queueing; OVERPROVISION- cell overprovision";
         }
         leaf AllocatedHard {
            type uint16;
            config false;
            description
            "Number of hard cells allocated";
         }
         leaf AllocatedSoft {
            type uint16;
            config false;
            description
            "Number of soft cells allocated";
         }
         leaf OverProvision {
            type uint16;
            config false;
            description
            "Overprovisioned cells. 0 if EnforcePolicy is
            DISABLE";
         }
         leaf QoS {
            type uint16;
            config false;
            description
            "Current QoS including overprovisioned cells, i.e. the
            bandwidth obtained including the overprovisioned cells.";
         }
         leaf NQoS {
            type uint16;
            config false;
            description
            "Real QoS without over provisioned cells, i.e. the actual
            bandwidth without taking into account the overprovisioned
            cells.";
         }
      }

list StatisticsMetricsList {
   key "StatisticsMetricsID";
   description
   "List of Statistics Metrics used in the node. Statistics can be set and queri
ed.";

   leaf StatisticsMetricsID {
      type uint16;
   }
   leaf SlotframeID {
      type uint16;
      description
      "SlotframeID, one in SlotframeList, specifies the slotframe to
      which the statistics metrics applies to. If empty, applies to
      all slotframes";
      reference
      "IEEE802154e";
   }
   leaf SlotOffset {
      type uint16;
      description
      "Specific slotOffset to which the statistics metrics applies
      to. If empty, applies to all timeslots";
      reference
      "IEEE802154e";
   }
   leaf ChannelOffset {
      type uint8;
      description
      "Specific channelOffset to which the statistics metrics applies
      to. If empty, applies to all channels";
      reference
      "IEEE802154e";
   }
   leaf TargetNodeAddress {
      type uint64;
      description
      "Specific neighbor nodes to which the statistics metrics
      applies to. If empty, applies to all neighbor nodes.";
   }
   leaf Metrics {
      type enumeration {
         enum macCounterOctets
         enum macRetryCount
         enum macMultipleRetryCount
         enum macTXFailCount
         enum macTXSuccessCount
         enum macFCSErrorCount
         enum macSecurityFailure
         enum macDuplicateFrameCount
         enum macRXSuccessCount
         enum macNACKcount
         enum PDR;
         enum ETX;
         enum RSSI;
         enum LQI;
      }
      description
      "The metric to be monitored. Include those provided by underlying IEEE 802
.15.4e TSCH -- see table 4i (2012). PDR,ETX,RSSI,LQI are maintained by 6top. ";
   }
   leaf Window {
      type uint16;
      description
      "measurement period, in Number of the slotframes. If not specified the met
rics are updated continuously in time. If a period is specified the metric value
s are given for the specified time-window";
   }
   leaf Enable {
      type enumeration {
         enum DISABLE;
         enum ENABLE;
      }
      description
      "indicates the StatisticsMetric is active or not";
   }
}
```

```
      list EBList {
         key "EbID";
         description
         "List of information related with the EBs used by the node";

         leaf EbID {
            type uint8;
         }

         leaf CellID {
            type uint16;
            description
            "CellID, one in CellList, indicates the cell used to send
            EB";
         }
         leaf SlotframeId{
               type uint8;
               description
               "SlotframeID, one in SlotframeList, indicates the
               slotframe to which the EB is send";
         }
         leaf Period {
            type uint16;
            description
            "The EBs period, in seconds, indicates the interval between
            two EB sends";
         }
         leaf Expiration {
            type enumeration {
               enum NEVERSTOP;
               enum EXPIRATION;
            }
            description
            "NEVERSTOP- the period of the EB never stops; EXPIRATION-
            when the Period arrives, the EB will stop.";
         }
         leaf Priority {
            type uint8;
            description
            "The joining priority model that will be used for
            advertisements. Joining priority MAY be for example
            SAME_AS_PARENT, RANDOM, BEST_PARENT+1 or
            DAGRANK(rank).";
         }
      }
```

```
   container TimeSource {
      description
      "specify the timesource selection policy and some relative
      statistics.";

      leaf policy {
         type enumeration {
            enum ALLPARENT;
            enum BESTCONNECTED;
            enum LOWESTJOINPRIORITY;
         }
         description
         "indicates the policy to choose timesource. ALLPARENT- choose
         from all parents; BESTCONNECTED- choose the best-connected
         node; LOWESTJOINPRIORITY- choose the node with lowest priority
         in its EB.";
      }
      leaf TargetNodeAddress {
         type uint64;
         description
         "Address of the time source neighbor";
      }
      leaf MinTimeCorrection {
         type uint16;
         config false;
         description
         "measured in microsecond";
      }
      leaf MaxTimeCorrection {
         type uint16;
         config false;
         description
         "measured in microsecond";
      }
      leaf AveTimeCorrection {
         type uint16;
         config false;
         description
         "measured and computed in microsecond";
      }
   }
   typedef asntype {
      description
         "The type to store ASN. String of 5 bytes";
      type string {
         length "0..5";
      }
   }

   list NeighborList {
      key "TargetNodeAddress";
      description
      "statistics per communication link.";

      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, indicates the most recent timeslot when a
         packet from the neighbor was received";
      }
   }

   list QueueList {
      key "QueueId";
      description
      "List of Queues, including configuration and statistics.";

      leaf QueueId {
         type uint8;
         description
         "Queue Identifier";
      }
      leaf TxqLength {
         type uint8;
         description
         "The TX queue length in number of packets";
      }
      leaf RxqLength {
         type uint8;
         description
         "The RX queue length in number of packets";
      }
      leaf NumrTx {
         type uint8;
         description
         "Number of allowed retransmissions.";
      }
      leaf Age {
         type uint16;
         description
         "In seconds. Discard packet according to its age
          on the queue. 0 if no discards are allowed.";
      }
      leaf RTXbackoff {
         type uint8;
         description
         "retransmission backoff in number of slotframes.
          0 if next available timeslot wants to be used.";
      }
      leaf StatsWindow {
         type uint16;
         description
         "In second, window of time used to compute stats.";
      }
      leaf QueuePriority {
         type uint8;
         description
         "The priority for this queue.";
      }
      list TrackIds {
         key "TrackID";
         leaf TrackID{
            type uint16;
            description
            "The TrackID, one in TrackList, indicates the Track is
            associated with the Queue.";
         }
      }
      leaf MinLenTXQueue {
         type uint8;
         config false;
         description
         "Statistics, lowest TX queue length registered in the window.";
      }
      leaf MaxLenTXQueue {
         type uint8;
         config false;
         description
         "Statistics, largest TX queue length registered in the
         window.";
      }
      leaf AvgLenTXQueue {
         type uint8;
         config false;
         description
         "Statistics, avg TX queue length registered in the window.";
      }
      leaf MinLenRXQueue {
         type uint8;
         config false;
         description
         "Statistics, lowest RX queue length registered in the window.";
      }
      leaf MaxLenRXQueue {
         type uint8;
         config false;
         description
         "Statistics, largest RX queue len registered in the window.";
      }
      leaf AvgLenRXQueue {
         type uint8;
         config false;
         description
         "Statistics, avg RX queue length registered in the window.";
      }
      leaf MinRetransmissions {
         type uint8;
         config false;
         description
         "Statistics, lowest number of retransmissions registered in
         the window.";
      }
      leaf MaxRetransmissions {
         type uint8;
         config false;
         description
         "Statistics, largest number of retransmissions registered
```

```
             in the window.";
         }
         leaf AvgRetransmissions {
            type uint8;
            config false;
            description
            "Statistics, average number of retransmissions registered
             in the window.";
         }
         leaf MinPacketAge {
            type uint16;
            config false;
            description
            "Statistics, in seconds, minimum time a packet stayed in
             the queue during the observed window.";
         }
         leaf MaxPacketAge {
            type uint16;
            config false;
            description
            "Statistics, in seconds, maximum time a packet stayed
             in the queue during the observed window.";
         }
         leaf AvgPacketAge {
            type uint16;
            config false;
            description
            "Statistics, in seconds, average time a packet stayed in
             the queue during the observed window.";
         }
         leaf MinBackoff {
            type uint8;
            config false;
            description
            "Statistics, in number of slotframes, minimum Backoff
             for a packet in the queue during the observed window.";
         }
         leaf MaxBackoff {
            type uint8;
            config false;
            description
            "Statistics, in number of slotframes, maximum Backoff
             for a packet in the queue during the observed window.";
         }
         leaf AvgBackoff {
            type uint8;
            config false;
            description
            "Statistics, in number of slotframes, average Backoff
             for a packet in the queue during the observed window.";
         }
      }

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

         leaf LabelSwitchID {
            type uint16;
         }
         list InputCellIds {
            key "CellID";
            leaf CellID{
               type uint16;
               description
               "The CellID, indicates the Rx cell on which the packet will
               come in.";
            }
         }
         list OutputCellIds {
            key "CellID";
            leaf CellID{
               type uint16;
               description
               "The CellID, indicates the Tx cell on which the received
               packet should be sent out.";
            }
         }
         leaf LoadBalancingPolicy {
            type enumeration {
               enum ROUNDROBIN;
               enum OTHER;
            }
            description
            "The load-balancing policy. ROUNDROBIN- Round robin algorithm
            is used for forwarding scheduling.";
         }
      }
      list TrackList {
         key "TrackId";
         description
         "List of the tracks through the node.";

         leaf TrackId {
            type uint16;
            description
            "Track Identifier, named locally. It is used to refer to the
            tuple (TrackOwnerAddr, InstanceID).";
         }
         leaf TrackOwnerAddr {
            type uint64;
            description
            "The address of the node which initializes the process of
            creating the track, i.e., the owner of the track;";
         }
         leaf InstanceID {
            type uint16;
            description
            "InstanceID is an instance identifier given by the owner of
            the track. InstanceID comes from upper layer; InstanceID could
            for example be the local instance ID defined in RPL.";
         }
      }
```

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

            leaf ChunkId{
               type uint16;
               description
               "The identifier of a chunk";
            }
            leaf SlotframeId{
               type uint8;
               description
               "SlotframeID, one in SlotframeList, indicates the
               slotframe to which the chunk belongs";
            }
            leaf SlotBase {
               type uint16;
               description
               "the base slotOffset of the chunk in the slotframe";
            }
            leaf SlotStep {
               type uint8;
               description
               "the slot incremental of the chunk";
            }
            leaf ChannelBase {
               type 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";
            description
            "List of all of the cells assigned to the node via the
            assignment of chunks.";

            leaf SlotOffset{
               type uint16;
               description
               "The slotoffset of a cell which belongs to a Chunk";
            }
            leaf ChannelOffset{
               type uint16;
               description
               "The channeloffset of a cell which belongs to a chunk.";
            }
            leaf ChunkId {
               type 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 UNSCHEDULED;
                  enum SCHEDULED;
               }
            }
         }
```

4.2. YANG model of the IEEE802.15.4 PIB

This section describes the YANG model of the part of PIB ([IEEE802154] and [IEEE802154e]) used by 6top, such as security related attributes, TSCH related attributes. This part of data will be accessed through the MLME-GET and MLME-SET primitive [IEEE802154] directly, instead of using 6top comannds.

TODO the security related attributes will be added after 6TiSCH WG has consensus on the security scheme of 6top

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

               leaf macMinBE {
                  type uint8;
                  description
                  "defined in Table 52b of IEEE802.15.4e-2012,
                  The minimum value of the backoff exponent (BE) in the
                  CSMA-CA algorithm or the TSCH-CA algorithm. default:
                  3-CSMA-CA, 1-TSCH-CA";
               }
               leaf macMaxBE {
                  type uint8;
                  description
                  "defined in Table 52b of IEEE802.15.4e-2012,
                  The maximum value of the backoff exponent (BE) in the
                  CSMA-CA algorithm or the TSCH-CA algorithm. default:
                  5-CSMA-CA, 7-TSCH-CA";
               }
               leaf macDisconnectTime {
                  type uint16;
                  description
                  "defined in Table 52b of IEEE802.15.4e-2012,
                  Time (in Timeslots) to send out Disassociate frames
                  before disconnecting, default: 0x00ff";
               }
               leaf macJoinPriority {
                  type uint8;
                  description
                  "defined in Table 52b of IEEE802.15.4e-2012,
                  The lowest join priority from the TSCH Synchronization
                  IE in an Enhanced beacon, default: 1";
               }
               leaf macASN {
                  type asntype;
                  description
                  "defined in Table 52b of IEEE802.15.4e-2012,
                  The Absolute Slot Number, i.e., the number of slots
                  that ha elapsed since the start of the network.";
               }
               leaf macNoHLBuffers {
                  type enumeration {
                     enum TRUE;
                     enum FALSE;
                  }
                  description
                  "defined in Table 52b of IEEE802.15.4e-2012,
                  If the value is TRUE, the higher layer receiving the
                  frame payload cannot buffer it, and the device should
                  acknowledge frames with a NACK; If FALSE, the higher
                  layer can accept the frame payload. default: FALSE";
               }
            }

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

               leaf macTimeslotTemplateId {
                  type uint8;
                  description
                  "defined in Table 52e of IEEE802.15.4e-2012.
                  Identifier of Timeslot Template. default: 0";
               }
               leaf macTsCCAOffset {
                  type uint16;
                  description
                  "The time between the beginning of timeslot and start
                  of CCA operation, in microsecond. default: 1800";
               }
               leaf macTsCCA {
                  type uint16;
                  description
                  "Duration of CCA, in microsecond. default: 128";
               }
               leaf macTsTxOffset {
                  type uint16;
                  description
                  "The time between the beginning of the timeslot and
                  the start of frame transmission, in microsecond.
                  default: 2120";
               }
               leaf macTsRxOffset {
                  type uint16;
                  description
                  "Beginning of the timeslot to when the receiver shall
                  be listening, in microsecond. default: 1120";
               }
               leaf macTsRxAckDelay {
                  type uint16;
                  description
                  "End of frame to when the transmitter shall listen for
                  Acknowledgment, in microsecond. default: 800";
               }
               leaf macTsTxAckDelay {
                  type uint16;
                  description
                  "End of frame to start of Acknowledgment, in
                  microsecond.
                  default: 1000";
               }
               leaf macTsRxWait {
                  type uint16;
                  description
                  "The time to wait for start of frame, in microsecond.
                  default: 2200";
               }
               leaf macTsAckWait {
                  type uint16;
                  description
                  "The minimum time to wait for start of an
                  Acknowledgment, in microsecond. default: 400";
               }
               leaf macTsRxTx {
                  type uint16;
                  description
                  "Transmit to Receive turnaround, in microsecond.
                  default: 192";
               }
               leaf macTsMaxAck {
                  type uint16;
                  description
                  "Transmission time to send Acknowledgment,in
                  microsecond. default: 2400";
               }
               leaf macTsMaxTx {
                  type uint16;
                  description
                  "Transmission time to send the maximum length frame,
                  in microsecond. default: 4256";
               }
               leaf macTsTimeslotLength {
                  type uint16;
                  description
                  "The total length of the timeslot including any unused
                  time after frame transmission and Acknowledgment,
                  in microsecond. default: 10000";
               }
            }
            list TSCHHoppingSequence {
               key "macHoppingSequenceID";
               description
               "List of all channel hopping sequences used in the
               nodes";
               reference
               "Table 52f of IEEE802.15.4e-2012";

               leaf macHoppingSequenceID {
                  type uint8;
                  description
                  "defined in Table 52f of IEEE802.15.4e-2012.
                  Each hopping sequence has a unique ID. default: 0";
               }
               leaf macChannelPage {
                  type uint8;
                  description
                  "Corresponds to the 5 MSBs (b27, ..., b31) of a row
                  in phyChannelsSupported. Note this may not correspond
                  to the current channelPage in use.";
               }
               leaf macNumberOfChannels {
                  type uint16;
                  description
                  "Number of channels supported by the PHY on this
                  channelPage.";
               }
               leaf macPhyConfiguration {
                  type uint32;
                  description
                  "For channel pages 0 to 6, the 27 LSBs(b0, b1, ...,
                  b26) indicate the status (1 = to be used, 0 = not to
                  be used) for each of the up to 27 valid channels
                  available to the PHY. For pages 7 and 8, the 27 LSBs
                  indicate the configuration of the PHY, and the channel
                  list is contained in the extendedBitmap.";
               }
               leaf macExtendedBitmap {
                  type uint64;
                  description
                  "For pages 7 and 8, a bitmap of numberOfChannels bits,
                  where bk shall indicate the status of channel k for
                  each of the up to numberOfChannels valid channels
                  supported by that channel page and phyConfiguration.
                  Otherwise field is empty.";
               }
               leaf macHoppingSequenceLength {
                  type uint16;
                  description
                  "The number of channels in the Hopping Sequence.
                  Does not necessarily equal numberOfChannels.";
               }
               list macHoppingSequenceList {
                  key "HoppingChannelID";
                  leaf HoppingChannelID {
                     type uint16;
                     description
                     "channels to be hopped over";
                  }
               }
               leaf macCurrentHop {
                  type uint16;
                  config false;
                  description
                  "Index of the current position in the hopping sequence
                  list.";
               }
            }
```

5. 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 4.1).

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 4.1).

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 4.1).

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 4.1) will be modified.

DELETE.hardcell: To remove a hard cell. This removes the hard cell from the node’s schedule, from CellList (Section 4.1).

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 4.1) 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 4.1) on both sides.

CREATE.slotframe: Creates a new slotframe. Adds a entry to the SlotframeList (Section 4.1).

READ.slotframe: Returns the information of a slotframe given its slotframeID from SlotframeList (Section 4.1).

UPDATE.slotframe: Change the number of timeslots in a slotframe given its slotframeID in SlotframeList (Section 4.1).

DELETE.slotframe: Deletes a slotframe, remove it from SlotframeList (Section 4.1).

CONFIGURE.monitoring: Configures the level of QoS the Monitoring process MUST enforce, i.e. config MonitoringStatusList (Section 4.1).

READ.monitoring: Reads the current Monitoring status from MonitoringStatusList (Section 4.1).

CONFIGURE.statistics: Configures the statistics process in StatisticsMetricsList(Section 4.1). 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 4.1).

RESET.statistics: Resets the gathered statistics in CellList (Section 4.1).

CONFIGURE.eb: Configures EBs, i.e. configures EBlist (Section 4.1).

READ.eb: Reads the EBs configuration from EBList (Section 4.1).

CONFIGURE.timesource: Configures the Time Source Neighbor selection process, i.e. configure TimeSource (Section 4.1).

READ.timesource: Retrieves information about the time source neighbors of that node from TimeSource (Section 4.1).

CREATE.neighbor: Creates an entry for a neighbor in the neighbor table, i.e. NeighborList (Section 4.1).

READ.all.neighbor: Returns the list of neighbors of that node according to NeighborList (Section 4.1).

READ.neighbor: Returns the information of a specific neighbor of that node specified by its neighbor address according to NeighborList (Section 4.1).

UPDATE.neighbor: Updates the last status for a given TargetNodeAddress in the NeighborList (Section 4.1).

DELETE.neighbor: Deletes a neighbor given its address from NeighborList (Section 4.1).

CREATE.queue: Creates and Configures a queue in QueueList (Section 4.1).

READ.queue: Reads the queue configuration for given QueueId from QueueList (Section 4.1).

READ.queue.stats: For a given QueueId, reads the queue statistics information from the QueueList (Section 4.1).

UPDATE.queue: For a given QueueId, update its configuration in the QueueList (Section 4.1).

DELETE.queue: Deletes a Queue for a given QueueId from the QueueList (Section 4.1).

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 4.1).

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 4.1).

CREATE.chunk: Creates a chunk which consists of one or more unscheduled cells, i.e. add an entry to the ChunkList (Section 4.1).

READ.chunk: Returns the information of a chunk given its ChunkID from ChunkList (Section 4.1).

DELETE.chunk: For given ChunkId, removes a chunk from the ChunkList (Section 4.1), which also causes all of the scheduled cells in the chunk to be deleted from the TSCH schedule and CellList (Section 4.1).

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 4.1). In addition, 6top will change the attributes corresponding to the cell in the ChunkCellList (Section 4.1), i.e. its CellID is changed to the same CellID in the CellList, and its Status is changed to SCHEDULED.

READ.chunkcell: Returns the information of all cells in a chunk given its ChunkID from ChunkCellList (Section 4.1).

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 4.1). In addition, it changes the attributes corresponding to the cell in the ChunkCellList (Section 4.1), i.e. its CellID is changed back to 0xFFFF, and its Status is changed to UNSCHEDULED.

6. References

6.1. Normative References

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

6.2. Informative References

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

[I-D.ietf-6tisch-tsch]
Watteyne, T., Palattella, M., and L. Grieco, "Using IEEE802.15.4e TSCH in an IoT context: Overview, Problem Statement and Goals", draft-ietf-6tisch-tsch-02 (work in progress), October 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-03 (work in progress), July 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-02 (work in progress), July 2014.

[I-D.ietf-6tisch-minimal]
Vilajosana, X. and K. Pister, "Minimal 6TiSCH Configuration", draft-ietf-6tisch-minimal-03 (work in progress), October 2014.

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

[I-D.ietf-6tisch-coap]
Sudhaakar, R. and P. Zand, "6TiSCH Resource Management and Interaction using CoAP", draft-ietf-6tisch-coap-01 (work in progress), July 2014.

6.3. External Informative References

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

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

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

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

Authors’ Addresses

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

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

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

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

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

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

6TiSCH   X. Vilajosana, Ed.
Internet-Draft   Universitat Oberta de Catalunya
Intended status: Informational   K. Pister
Expires: April 29, 2015   University of California Berkeley
                                                        October 26, 2014

Minimal 6TiSCH Configuration
draft-ietf-6tisch-minimal-03

Abstract

This document describes the minimal set of rules to operate a [IEEE802154e] Timeslotted Channel Hopping (TSCH) network. This minimal mode of operation can be used during network bootstrap, as a fall-back mode of operation when no dynamic scheduling solution is available or functioning, or during early interoperability testing and development.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The nodes in a [IEEE802154e] TSCH network follow a communication schedule. The entity (centralized or decentralized) responsible for building and maintaining that schedule has very precise control over the trade-off between the network's latency, bandwidth, reliability and power consumption. During early interoperability testing and development, however, simplicity is often more important than efficiency. One goal of this document is to define the simplest set of rules for building a [IEEE802154e] TSCH-compliant network, at the necessary price of lesser efficiency. Yet, this minimal mode of operation MAY also be used during network bootstrap before any schedule is installed into the network so nodes can self-organize and the management and configuration information be distributed. In addition, as outlined in [I-D.phinney-roll-rpl-industrial-applicability], the minimal configuration MAY be used as a fall-back mode of operation, ensuring connectivity of nodes in case that dynamic scheduling mechanisms fail or are not available. [IEEE802154e] provides a mechanism whereby the details of slotframe length, timeslot timing, and channel hopping pattern are communicated at synchronization to a node. This document describes specific settings for these parameters. Nodes MUST broadcast properly formed Enhanced Beacons to announce these values.

2. Requirements Language

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

3. Minimal Schedule Configuration

In order to form a network, a minimum schedule configuration is required so nodes can advertise the presence of the network, and allow other nodes to join.

3.1. Slotframe

The slotframe, as defined in [I-D.ietf-6tisch-terminology], is an abstraction of the link layer that defines a collection of time slots of equal length, and which repeats over time. In order to set up a minimal TSCH network, nodes need to be synchronized with the same slotframe configuration so they can exchange Enhanced Beacons (EBs) and data packets. This document recommends the following slotframe configuration.

Minimal configuration

Property	Value
Number of time slots per Slotframe	Variable
Number of available frequencies	16
Number of scheduled cells	1 (slotOffset 0) (macLinkType NORMAL)
Number of unscheduled cells	The remainder of the slotframe
Number of MAC retransmissions (max)	3 (4 attempts to tx)

The slotframe is composed of a configurable number of time slots. Choosing the number of time slots per slotframe needs to take into account network requirements such as density, bandwidth per node, etc. In the minimal configuration, there is only a single active slot in slotframe, used to transmit data and EBs, and receive information. The trade-off between bandwidth, latency and energy consumption can be controlled by choosing a different slotframe length. The active slot MAY be scheduled at the slotOffset 0x00 and channelOffset 0x00 and MUST be announced in the EBs. EBs are sent using this active slot to the link-layer broadcast address (and are therefore not acknowledged). Data packets, as described in Section 3.2, use the same active slot. Per [IEEE802154e], data packets sent unicast on this cell are acknowledged by the receiver. The remaining cells in the slotframe are unscheduled, and MAY be used by dynamic scheduling solutions. Details about such dynamic scheduling solution are out of scope of this document.

The slotframe length (expressed in number of time slots) is configurable. The length used determines the duty cycle of the network. For example, a network with a 0.99% duty cycle is composed of a slotframe of 101 slots, which includes 1 active slot. The present document RECOMMENDS the use of a default slot duration set to 10ms and its corresponding default timeslot timings defined by the [IEEE802154e] macTimeslotTemplate. The use of the default macTimeslotTemplate MUST be announced in the EB by using the Timeslot IE containing only the default macTimeslotTemplateId. Other time slot durations MAY be supported and MUST be announced in the EBs. If one uses a timeslot duration different than 10ms, it is RECOMMENDED to use a power-of-two of 10ms (i.e. 20ms, 40ms, 80ms, etc.). In this case, EBs MUST contain the complete TimeSlot IE as described in

Section 3.4. This document also recommends to manufacturers to clearly indicate nodes not supporting the default timeslot value.

Example schedule with 0.99% duty cycle

```
   Chan.  +----------+----------+          +----------+
   Off.0  | TxRxS/EB |   OFF    |          |   OFF    |
   Chan.  +----------+----------+          +----------+
   Off.1  |          |          |   ...    |          |
          +----------+----------+          +----------+
              .
              .
              .
   Chan.  +----------+----------+          +----------+
   Off.15 |          |          |          |          |
          +----------+----------+          +----------+

slotOffset     0          1                    100
```

```
EB:  Enhanced Beacon
Tx:  Transmit
Rx:  Receive
S:   Shared
OFF: Unscheduled (MAY be used by a dynamic scheduling mechanism)
```

3.2. Cell Options

Per [IEEE802154e] TSCH, each scheduled cell has an associated bitmap of cell options, called LinkOptions. The scheduled cell in the minimal schedule is configured as a Hard cell [I-D.ietf-6tisch-tsch][I-D.ietf-6tisch-6top-interface]. Additional available cells MAY be scheduled by a dynamic scheduling solution. The dynamic scheduling solution is out of scope, and this specification does not make any restriction on the LinkOption associated with those dynamically scheduled cells (i.e. they can be hard cells or soft cells).

The active cell is assigned the bitmap of cell options below. Because both the "Transmit" and "Receive" bits are set, a node transmits if there is a packet in its queue, listens otherwise. Because the "shared" bit is set, the back-off mechanism defined in [IEEE802154e] is used to resolve contention when transmitting. This results in "Slotted Aloha" behavior. The "Timekeeping" flag is never set, since the time source neighbor is selected using the DODAG structure of the network (detailed below).

```
   b0 = Transmit = 1 (set)
```

b1 = Receive = 1 (set)

b2 = Shared = 1 (set)

b3 = Timekeeping = 0 (clear)

b4-b7 = Reserved (clear)

All remaining cells are unscheduled. In unscheduled cells, the nodes SHOULD keep their radio off. In a memory-efficient implementation, scheduled cells can be represented by a circular linked list. Unscheduled cells SHOULD NOT occupy any memory.

3.3. Retransmissions

The maximum number of link layer retransmissions is set to 3. For packets which require an acknowledgment, if none is received after a total of 4 attempts, the transmissions is considered failed and the link layer MUST notify the upper layer. Packets sent to the broadcast MAC address (including EBs) are not acknowledged and therefore not retransmitted.

3.4. Time Slot timing

The figure below shows an active timeslot in which a packet is sent from the transmitter node (TX) to the receiver node (RX). A link-layer acknowledgment is sent by the RX node to the TX node when the packet is to be acknowledged. The TsTxOffset duration defines the instant in the timeslot when the first byte of the transmitted packet leaves the radio of the TX node. The radio of the RX node is turned on tsRxWait/2 before that instant, and listens for at least tsRxWait. This allows for a de-synchronization between the two nodes of at most tsRxWait. The RX node needs to send the first byte of the MAC acknowledgment exactly TsTxAckDelay after the end of the last byte of the received packet. TX’s radio has to be turned on tsAckWait/2 before that time, and keep listening for at least tsAckWait. The TX node can perform a Clear Channel Assessment (CCA) if required, this does not interfere with the scope of this draft. As for a minimal configuration, CCA is not mandatory.

Time slot internal timing diagram

```
      /---------------------- Time Slot Duration ----------------------/
      |                                                  / (5) /       |
      |                   |              / tsRxAckDelay /|  |  |       |
      |-------------------+--------------+------------------+------+---|
   TX |/(1)/  (2)  / (3) /|   TX packet  |                  |RX ack|   |
      |----+-------+------+--------------+------------------+------+---|
      |/    tsTxOffset   /|              |                  |      |   |
      |                   |              |                  |      |   |
      |-------------------+--------------+------------------+------+---|
   RX |                |  |  | RX packet |                  |TX ack|   |
      |----------------+--+--+-----------+------------------+------+---|
      |                |  |  |           |                  |      |   |
      |                / (4) /           /   tsTxAckDelay   /      |   |
      Start                                                          End
      of                                                              of
      Slot                                                          Slot
   /(1)/ tsCCAOffset
   /(2)/ tsCCA
   /(3)/ tsRxTx
   /(4)/ tsRxWait
   /(5)/ tsAckWait
```

A 10ms time slot length is the default value defined by [IEEE802154e]. Section 6.4.3.3.3 of [IEEE802154e] defines a default macTimeslotTemplate, i.e. the different duration within the slot. These values are summarized in the following table and MUST be used when utilizing the default time slot duration. In this case, the Timeslot IE only transports the macTimeslotTemplateId (0x00) as the timing values are well-known. If a timeslot template other than the default is used, the EB MUST contain a complete TimeSlot IE indicating the timeslot duration and the corresponding timeslot timings, requiring 25 bytes.

Default timeslot durations (per [IEEE802154e], Section 6.4.3.3.3)

IEEE802.15.4e TSCH parameter	Value (us)
tsCCAOffset	1800
tsCCA	128
tsTxOffset	2120
tsRxOffset	1120
tsRxAckDelay	800
tsTxAckDelay	1000
tsRxWait	2200
tsAckWait	400
tsRxTx	192
tsMaxAck	2400
tsMaxTx	4256
Time Slot duration	10000

4. Enhanced Beacons Configuration and Content

[IEEE802154e] does not define how often EBs are sent, nor their contents. The choice of the duration between two EBs needs to take into account whether EBs are used as the only mechanism to synchronize devices, or whether a Keep-Alive (KA) mechanism is also used. For a minimal TSCH configuration, a mote SHOULD send an EB every EB_PERIOD. For additional reference see [I-D.ietf-6tisch-tsch] where different synchronization approaches are summarized.

EBs MUST be sent with the Beacon IEEE802.15.4 frame type and this EBs MUST carry the Information Elements (IEs) listed below.

The content of the IEs is presented here for completeness, however this information is redundant with [I-D.ietf-6tisch-tsch] and [IEEE802154e].

4.1. Sync IE

Contains synchronization information such as ASN and Join Priority. The value of Join Priority is discussed in Section 6.2.

4.1.1. IE Header

Length (b0-b7) = 0x06

Sub-ID (b8-b14) = 0x1a

Type (b15) = 0x00 (short)

4.1.2. IE Content

ASN Byte 1 (b16-b23)

ASN Byte 2 (b24-b31)

ASN Byte 3 (b32-b39)

ASN Byte 4 (b40-b47)

ASN Byte 5 (b48-b55)

Join Priority (b56-b63)

4.2. TSCH Timeslot IE

Contains the timeslot template identifier. This specification uses the default timeslot template as defined in [IEEE802154e], Section 5.2.4.15.

4.2.1. IE Header

Length (b0-b7) = 0x01

Sub-ID (b8-b14) = 0x1c

Type (b15) = 0x00 (short)

4.2.2. IE Content

Timeslot Template ID (b0-b7) = 0x00

4.3. Channel Hopping IE

Contains the channel hopping template identifier. This specification uses the default channel hopping template, as defined in [IEEE802154e], Section 5.2.4.16.

4.3.1. IE Header

Length (b0-b7) = 0x01

Sub-ID (b8-b14) = 0x1d

Type (b15) = 0x00 (short)

4.3.2. IE Content

Channel Hopping Template ID (b0-b7) = 0x00

4.4. Frame and Link IE

Each node MUST indicate the schedule in each EB through a Frame and Link IE. This enables nodes which implement [IEEE802154e] to configure their schedule as they join the network.

4.4.1. IE Header

Length (b0-b7) = variable

Sub-ID (b8-b14) = 0x1b

Type (b15) = 0x00 (short)

4.4.2. IE Content

# Slotframes (b16-b23) = 0x01

Slotframe ID (b24-b31) = 0x01

Size Slotframe (b32-b47) = variable

# Links (b48-b55) = 0x01

For the active cell in the minimal schedule:

Channel Offset (2B) = 0x00

Slot Number (2B) = 0x00

LinkOption (1B) = as described in Section 3.2

5. Acknowledgment

Link-layer acknowledgment frames are built according to [IEEE802154e]. Data frames and command frames sent to a unicast MAC destination address request an acknowledgment. The acknowledgment frame is of type ACK (0x10). Each acknowledgment contains the following IE:

5.1. ACK/NACK Time Correction IE

The ACK/NACK time correction IE carries the measured de-synchronization between the sender and the receiver.

5.1.1. IE Header

Length (b0-b7) = 0x02

Sub-ID (b8-b14) = 0x1e

Type (b15) = 0x00 (short)

5.1.2. IE Content

Time Synchronization Information and ACK status (b16-b31)

The possible values for the Time Synchronization Information and ACK status are described in [IEEE802154e] and reproduced in the following table:

ACK status and Time Synchronization Information.

ACK Status	Value
ACK with positive time correction	0x0000 - 0x07ff
ACK with negative time correction	0x0800 - 0x0fff
NACK with positive time correction	0x8000 - 0x87ff
NACK with negative time correction	0x8800 - 0x8fff

6. Neighbor information

[IEEE802154e] does not define how and when each node in the network keeps information about its neighbors. Keeping the following information in the neighbor table is RECOMMENDED:

6.1. Neighbor Table

The exact format of the neighbor table is implementation-specific, but it SHOULD contain the following information for each neighbor:

Neighbor statistics:

numTx: number of transmitted packets to that neighbor

numTxAck: number of transmitted packets that have been acknowledged by that neighbor

numRx: number of received packets from that neighbor

The EUI64 of the neighbor.

Timestamp when that neighbor was heard for the last time. This can be based on the ASN counter or any other time base. Can be used to trigger a keep-alive message.

RPL rank of that neighbor.

A flag indicating whether this neighbor is a time source neighbor.

Connectivity statistics (e.g., RSSI), which can be used to determine the quality of the link.

In addition to that information, each node has to be able to compute some RPL Objective Function (OF), taking into account the neighbor and connectivity statistics. An example RPL objective function is the OF Zero as described in [RFC6552] and Section 9.1.1.

6.2. Time Source Neighbor Selection

Each node MUST select at least one Time Source Neighbor among the nodes in its RPL routing parent set. When a node joins a network, it has no routing information. To select its time source neighbor, it uses the Join Priority field in the EB, as described in Section 5.2.4.13 and Table 52b of [IEEE802154e]. The Sync IE contains the ASN and 1 Byte field named Join Priority. The Join Priority of any node is equivalent to the result of the function DAGRank(rank) as defined by [RFC6550] and Section 9.1.1. The Join

Priority of the DAG root is zero, i.e., EBs sent from the DAG root are sent with Join Priority equal to 0. A lower value of the Join Priority indicates higher preference to connect to that device. When a node joins the network, it MUST NOT send EBs before having acquired a RPL rank. This avoids routing loops and matches RPL topology with underlying mesh topology. As soon as a node acquires a RPL rank (see [RFC6550] and Section 9.1.1), it SHOULD send Enhanced Beacons including a Sync IE with Join Priority field set to DAGRank(rank), where rank is the node’s rank. If a node receives EBs from different nodes with equal Join Priority, the time source neighbor selection SHOULD be assessed by other metrics that can help determine the better connectivity link. Time source neighbor hysteresis SHOULD be used, according to the rules defined in Section 9.2.3. If connectivity to the time source neighbor is lost, a new time source neighbor MUST be chosen among the neighbors in the RPL routing parent set.

The decision for a node to select one Time Source Neighbor when multiple EBs are received is open to implementers. For example, a node MAY wait until one EB from NUM_NEIGHBOURS_TO_WAIT neighbors have been received to select the best Time Source Neighbor. This condition MAY apply unless a second EB is not received after MAX_EB_DELAY seconds. This avoids initial hysteresis when selecting a first Time Source Neighbor.

Optionally, some form of hysteresis SHOULD be implemented to avoid frequent changes in time source neighbors.

7. Queues and Priorities

[IEEE802154e] does not define the use of queues to handle upper layer data (either application or control data from upper layers). The use of a single queue with the following rules is RECOMMENDED:

When the node is not synchronized to the network, higher layers are not able to insert packets into the queue.

Frames generated by the MAC layer (e.g., EBs and ACK) have a higher priority than packets received from a higher layer.

IEEE802.15.4 frame types Beacon and Command have a higher priority than IEEE802.15.4 frame types Data and ACK.

One entry in the queue is reserved at all times for an IEEE802.15.4 frames of types Beacon or Command frames.

8. Security

A minimal security configuration inherits the security considerations defined in the Section 19 of [RFC6550]. Other specific security mechanisms described in Section 10 of [RFC6550] are OPTIONAL in this scope. As this document refers to the interaction between Layer 3 and Layer 2 protocols, this interaction MUST be secured by L2 security mechanisms which include a CCM* [RFC3610], [CCM] ,[CCM-Star], architecture. Yet, as RPL is a distributed routing protocol, a peer-wise security mechanism might be used, rather than a centralized one. Key distribution is out of scope of this document, but examples include pre-configured keys at the nodes, shared keys amongst peers or well-known keys. Refer to the 6TiSCH architecture document [I-D.ietf-6tisch-architecture] for further details on security aspects. This document RECOMMENDS the use of shared keys and a CCM* architecture. It also RECOMMENDS the strict application of RPL consideration introduced above.

9. RPL on TSCH

Nodes in the network MUST use the RPL routing protocol [RFC6550].

9.1. RPL Objective Function Zero

Nodes in the network MUST use the RPL routing protocol [RFC6550] and implement the RPL Objective Function Zero [RFC6552].

9.1.1. Rank computation

The rank computation is described at [RFC6552], Section 4.1. Briefly, a node rank is computed by the following equation:

R(N) = R(P) + rank_increase

rank_increase = (Rf*Sp + Sr) * MinHopRankIncrease

Where:

R(N): Rank of the node.

R(P): Rank of the parent obtained as part of the DIO information.

rank_increase: The result of a function that determines the rank increment.

Rf (rank_factor): A configurable factor that is used to multiply the effect of the link properties in the rank_increase computation. If none is configured, rank_factor of 1 is used. In this specification, a rank_factor of 1 MUST be used.

Sp (step_of_rank): (strictly positive integer) - an intermediate computation based on the link properties with a certain neighbor. In this specification, 2*ETX (Expected Transmissions) as defined by [decouti03high] and [RFC6551] MUST be used. The ETX is computed as the inverse of the Packet Delivery Ratio (PDR), and MAY be computed as the number of acknowledged packets, divided by the number of transmitted packets to a certain node. E.g: Sp=2*numTX/numTXAck

Sr (stretch_of_rank): (unsigned integer) - the maximum increment to the step_of_rank of a preferred parent, to allow the selection of an additional feasible successor. If none is configured to the device, then the step_of_rank is not stretched. In this specification, stretch_of_rank MUST be set to 0.

MinHopRankIncrease: the MinHopRankIncrease is set to the fixed constant DEFAULT_MIN_HOP_RANK_INCREASE [RFC6550]. DEFAULT_MIN_HOP_RANK_INCREASE has a value of 256.

DAGRank(rank): Equivalent to the floor of (Rf*Sp + Sr) as defined by [RFC6550]. Specifically, when an Objective Function computes Rank, this is defined as an unsigned integer (i.e., a 16-bit value) Rank quantity. When the Rank is compared, e.g. to determine parent relationships or loop detection, the integer portion of the Rank is used. The integer portion of the Rank is computed by the DAGRank() macro as floor(x) where floor(x) is the function that evaluates to the greatest integer less than or equal to x. DAGRank(rank) = floor(rank/MinHopRankIncrease)

Rank computation scenario

```
    +-------+
    |   P   | R(P)
    |       |
    +-------+
        |
        |
    +-------+
    |   N   | R(N)=R(P) + rank_increase
    |       | rank_increase = (Rf*Sp + Sr) * MinHopRankIncrease
    +-------+ Sp=2*ETX
```

9.1.2. Rank computation Example

This section illustrates with an example the use of the Objective Function Zero. Assume the following parameters:

```
Rf = 1

Sp = 2* ETX

Sr = 0

minHopRankIncrease = 256 (default in RPL)

ETX=(numTX/numTXAck)

r(n) = r(p) + rank_increase

rank_increase = (Rf*Sp + Sr) * minHopRankIncrease

rank_increase = 512*numTx/numTxACK
```

Rank computation example for 5 hop network where numTx=100 and numTxAck=75 for all nodes

```
    +-------+
    |   0   | R(0)=0
    |       | DAGRank(R(0)) = 0
    +-------+
        |
        |
    +-------+
    |   1   | R(1)=R(0)+683=683
    |       | DAGRank(R(1)) = 2
    +-------+
        |
        |
    +-------+
    |   2   | R(2)=R(1)+683=1366
    |       | DAGRank(R(2)) = 5
    +-------+
        |
        |
    +-------+
    |   3   | R(3)=R(2)+683=2049
    |       | DAGRank(R(3)) = 8
    +-------+
        |
        |
    +-------+
    |   4   | R(4)=R(3)+683=2732
    |       | DAGRank(R(4)) = 10
    +-------+
        |
        |
    +-------+
    |   5   | R(5)=R(4)+683=3415
    |       | DAGRank(R(5)) = 13
    +-------+
```

9.2. RPL Configuration

In addition to the Objective Function (OF), a minimal configuration for RPL SHOULD indicate the preferred mode of operation and trickle timer operation so different RPL implementations can inter-operate. RPL information and hop-by-hop extension headers MUST be compressed according to the specification described in [I-D.thubert-6lo-rpl-nhc]

9.2.1. Mode of Operation

For downstream route maintenance, in a minimal configuration, RPL SHOULD be set to operate in the Non-Storing mode as described by [RFC6550] Section 9.7. Storing mode ([RFC6550] Section 9.8) MAY be supported in less constrained devices.

9.2.2. Trickle Timer

RPL signaling messages such as DIOs are sent using the Trickle Algorithm [RFC6550] (Section 8.3.1) and [RFC6206]. For this specification, the Trickle Timer MUST be used with the RPL defined default values [RFC6550] (Section 8.3.1). For a description of the Trickle timer operation see Section 4.2 on [RFC6206].

9.2.3. Hysteresis

According to [RFC6552], [RFC6719] recommends the use of a boundary value (PARENT_SWITCH_THRESHOLD) to avoid constant changes of parent when ranks are compared. When evaluating a parent that belongs to a smaller path cost than current minimum path, the candidate node is selected as new parent only if the difference between the new path and the current path is greater than the defined PARENT_SWITCH_THRESHOLD. Otherwise the node MAY continue to use the current preferred parent. As for [RFC6719] the recommended value for PARENT_SWITCH_THRESHOLD is 192 when ETX metric is used, the recommendation for this document is to use PARENT_SWITCH_THRESHOLD equal to 394 as the metric being used is 2*ETX. This is mechanism is suited to deal with parent hysteresis in both cases routing parent and time source neighbor selection.

9.2.4. Variable Values

The following table presents the RECOMMENDED values for the RPL-related variables defined in the previous section.

Recommended variable values

Variable	Value
EB_PERIOD	10s
MAX_EB_DELAY	180
NUM_NEIGHBOURS_TO_WAIT	2
PARENT_SWITCH_THRESHOLD	394

10. Acknowledgments

The authors would like to acknowledge the guidance and input provided by the 6TiSCH Chairs Pascal Thubert and Thomas Watteyne.

11. References

11.1. Normative References

[RFC6719] Gnawali, O. and P. Levis, "The Minimum Rank with Hysteresis Objective Function", RFC 6719, September 2012.

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

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

[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.

[RFC6206] Levis, P., Clausen, T., Hui, J., Gnawali, O., and J. Ko, "The Trickle Algorithm", RFC 6206, March 2011.

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

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

11.2. Informative References

[I-D.ietf-6tisch-tsch]
Watteyne, T., Palattella, M., and L. Grieco, "Using IEEE802.15.4e TSCH in an IoT context: Overview, Problem Statement and Goals", draft-ietf-6tisch-tsch-02 (work in progress), October 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-03 (work in progress), July 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-02 (work in progress), July 2014.

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

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

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

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

11.3. 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.

[CCM] National Institute of Standards and Technology, "Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality. SP 800-38C", May 2004.

[CCM-Star]
Struik, R., "Formal Specification of the CCM* Mode of Operation, IEEE P802.15 Working Group for Wireless Personal Area Networks (WPANs).", September 2005.

[decouti03high]
De Couto, D., Aguayo, D., Bicket, J., and R. Morris, "A High-Throughput Path Metric for Multi-Hop Wireless Routing", ACM International Conference on Mobile Computing and Networking (MobiCom) , June 2003.

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

Authors' Addresses

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

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

Kris Pister
University of California Berkeley
490 Cory Hall
Berkeley, California 94720
USA

Email: pister@eecs.berkeley.edu

6TiSCH
Internet-Draft
Intended status: Informational
Expires: April 20, 2015

T. Watteyne, Ed.
Linear Technology
MR. Palattella
University of Luxembourg
LA. Grieco
Politecnico di Bari
October 17, 2014

Using IEEE802.15.4e TSCH in an IoT context: Overview, Problem Statement and Goals
draft-ietf-6tisch-tsch-02

Abstract

This document describes the environment, problem statement, and goals for using the IEEE802.15.4e TSCH MAC protocol in the context of LLNs. The set of goals enumerated in this document form an initial set only.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

IEEE802.15.4e [IEEE802154e] was published in 2012 as an amendment to the Medium Access Control (MAC) protocol defined by the IEEE802.15.4-2011 [IEEE802154] standard. IEEE802.15.4e will be rolled into the next revision of IEEE802.15.4, scheduled to be published in 2015. The Timeslotted Channel Hopping (TSCH) mode of IEEE802.15.4e is the object of this document.

This document describes the main issues arising from the adoption of the IEEE802.15.4e TSCH in the LLN context, following the terminology defined in [I-D.ietf-6tisch-terminology].

TSCH was designed to allow IEEE802.15.4 devices to support a wide range of applications including, but not limited to, industrial ones [IEEE802154e]. At its core is a medium access technique which uses time synchronization to achieve ultra low-power operation and channel hopping to enable high reliability. Synchronization accuracy impacts power consumption, and can vary from micro-seconds to milli-seconds depending on the solution. This is very different from the "legacy" IEEE802.15.4 MAC protocol, and is therefore better described as a "redesign". TSCH does not amend the physical layer; i.e., it can operate on any IEEE802.15.4-compliant hardware.

IEEE802.15.4e is the latest generation of ultra-lower power and reliable networking solutions for LLNs. [RFC5673] discusses industrial applications, and highlights the harsh operating conditions as well as the stringent reliability, availability, and security requirements for an LLN to operate in an industrial environment. In these environments, vast deployment environments with large (metallic) equipment cause multi-path fading and interference to thwart any attempt of a single-channel solution to be reliable; the channel agility of TSCH is the key to its ultra high reliability. Commercial networking solutions are available today in which motes consume 10’s of micro-amps on average [CurrentCalculator] with end-to-end packet delivery ratios over 99.999% [doherty07channel].

Bringing industrial-like performance into the LLN stack developed by Internet of Things (IoT) related IETF working groups such as 6Lo, ROLL and CoRE opens up new application domains for these networks. Sensors deployed in smart cities [RFC5548] will be able to be installed for years without needing battery replacement. "Umbrella" networks will interconnect smart elements from different entities in smart buildings [RFC5867]. Peel-and-stick switches will obsolete the need for costly conduits for lighting solutions in smart homes [RFC5826].

IEEE802.15.4e TSCH focuses on the MAC layer only. This clean layering allows for TSCH to fit under an IPv6 enabled protocol stack for LLNs, running 6LoWPAN [RFC6282], IPv6 Routing Protocol for Low power and Lossy Networks (RPL) [RFC6550] and the Constrained Application Protocol (CoAP) [RFC7252]. What is missing is a Logical Link Control (LLC) layer between the IP abstraction of a link and the TSCH MAC, which is in charge of scheduling a timeslot for a given packet coming down the stack from the upper layer.

While [IEEE802154e] defines the mechanisms for a TSCH mote to communicate, it does not define the policies to build and maintain the communication schedule, match that schedule to the multi-hop paths maintained by RPL, adapt the resources allocated between neighbor nodes to the data traffic flows, enforce a differentiated treatment for data generated at the application layer and signaling messages needed by 6LoWPAN and RPL to discover neighbors, react to topology changes, self-configure IP addresses, or manage keying material.

In other words, IEEE802.15.4e TSCH is designed to allow optimizations and strong customizations, simplifying the merging of TSCH with a protocol stack based on IPv6, 6LoWPAN, and RPL.

2. Requirements Language

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

3. TSCH in the LLN Context

To map the services required by the IP layer to the services provided by the link layer, an adaptation layer is used [palattella12standardized]. The 6LoWPAN working group started working in 2007 on specifications for transmitting IPv6 packets over IEEE802.15.4 networks [RFC4919]. Low-power WPANs are characterized by small packet sizes, support for addresses with different lengths, low bandwidth, star and mesh topologies, battery powered devices, low cost, large number of devices, unknown node positions, high unreliability, and periods during which communication interfaces are turned off to save energy. Given these features, it is clear that the adoption of IPv6 on top of a Low-Power WPAN is not straightforward, but poses strong requirements for the optimization of this adaptation layer.

For instance, due to the IPv6 default minimum MTU size (1280 bytes), an un-fragmented IPv6 packet is too large to fit in an IEEE802.15.4 frame. Moreover, the overhead due to the 40-byte long IPv6 header wastes the scarce bandwidth available at the PHY layer [RFC4944]. For these reasons, the 6LoWPAN working group has defined an effective adaptation layer [RFC6282]. Further issues encompass the auto-configuration of IPv6 addresses [RFC2460][RFC4862], the compliance with the recommendation on supporting link-layer subnet broadcast in shared networks [RFC3819], the reduction of routing and management overhead [RFC6606], the adoption of lightweight application protocols (or novel data encoding techniques), and the support for security mechanisms (confidentiality and integrity protection, device bootstrapping, key establishment, and management).

These features can run on top of TSCH. There are, however, important issues to solve, as highlighted in Section 4.

Routing issues are challenging for 6LoWPAN, given the low-power and lossy radio links, the battery-powered nodes, the multi-hop mesh topologies, and the frequent topology changes due to mobility. Successful solutions take into account the specific application requirements, along with IPv6 behavior and 6LoWPAN mechanisms [palattella12standardized]. The ROLL working group has defined RPL in [RFC6550]. RPL can support a wide variety of link layers, including ones that are constrained, potentially lossy, or typically utilized in conjunction with host or router devices with very limited resources, as in building/home automation [RFC5867][RFC5826], industrial environments [RFC5673], and urban applications [RFC5548]. RPL is able to quickly build up network routes, distribute routing knowledge among nodes, and adapt to a changing topology. In a typical setting, motes are connected through multi-hop paths to a small set of root devices, which are usually responsible for data collection and coordination. For each of them, a Destination Oriented Directed Acyclic Graph (DODAG) is created by accounting for link costs, node attributes/status information, and an Objective Function, which maps the optimization requirements of the target scenario.

The topology is set up based on a Rank metric, which encodes the distance of each node with respect to its reference root, as specified by the Objective Function. Regardless of the way it is computed, the Rank monotonically decreases along the DODAG towards the root, building a gradient. RPL encompasses different kinds of traffic and signaling information. Multipoint-to-Point (MP2P) is the dominant traffic in LLN applications. Data is routed towards nodes with some application relevance, such as the LLN gateway to the larger Internet, or to the core of private IP networks. In general, these destinations are the DODAG roots and act as data collection points for distributed monitoring applications. Point-to-Multipoint (P2MP) data streams are used for actuation purposes, where messages are sent from DODAG roots to destination nodes. Point-to-Point (P2P) traffic allows communication between two devices belonging to the same LLN, such as a sensor and an actuator. A packet flows from the source to the common ancestor of those two communicating devices, then downward towards the destination. RPL therefore has to discover both upward routes (i.e. from nodes to DODAG roots) in order to enable MP2P and P2P flows, and downward routes (i.e. from DODAG roots to nodes) to support P2MP and P2P traffic.

Section 4 highlights the challenges that need to be addressed to use RPL on top of TSCH.

Several open-source initiatives have emerged around TSCH. The OpenWSN project [OpenWSN][OpenWSNETT] is an open-source implementation of a standards-based protocol stack, which aims at evaluating the applicability of TSCH to different applications. This implementation was used as the foundation for an IP for Smart Objects Alliance (IPSO) [IPSO] interoperability event in 2011. In the absence of a standardized scheduling mechanism for TSCH, a "slotted Aloha" schedule was used.

4. Problems and Goals

As highlighted in Appendix A, TSCH differs from traditional low-power MAC protocols because of its scheduled nature. TSCH defines the mechanisms to execute a communication schedule, yet it is the entity that sets up that schedule which controls the topology of the network. This scheduling entity also controls the resources allocated to each link in that topology.

How this entity should operate is out of scope of TSCH. The remainder of this section highlights the problems this entity needs to address. For simplicity, we refer to this entity by the generic name "LLC". Note that the 6top sublayer, currently being defined in [I-D.wang-6tisch-6top-sublayer], can be seen as an embodiment of this generic "LLC".

Some of the issues the LLC needs to target might overlap with the scope of other protocols (e.g., 6LoWPAN, RPL, and RSVP). In this case, it is entailed that the LLC will profit from the services provided by other protocols to pursue these objectives.

4.1. Network Formation

The LLC needs to control the way the network is formed, including how new motes join, and how already joined motes advertise the presence of the network. The LLC needs to:

1. Define the Information Elements included in the Enhanced Beacons advertising the presence of the network.

2. For a new mote, define rules to process and filter received Enhanced Beacons.

3. Define the joining procedure. This might include a mechanism to assign a unique 16-bit address to a mote, and the management of initial keying material.

4. Define a mechanism to secure the joining process and the subsequent optional process of scheduling more communication cells.

4.2. Network Maintenance

Once a network is formed, the LLC needs to maintain the network’s health, allowing for motes to stay synchronized. The LLC needs to:

1. Manage each mote’s time source neighbor.

2. Define a mechanism for a mote to update the join priority it announces in its Enhanced Beacon.

3. Schedule transmissions of Enhanced Beacons to advertise the presence of the network.

4.3. Multi-Hop Topology

RPL, given a weighted connectivity graph, determines multi-hop routes. The LLC needs to:

1. Define a mechanism to gather topological information, which it can then feed to RPL.

2. Ensure that the TSCH schedule contains cells along the multi-hop routes identified by RPL.

3. Where applicable, maintain independent sets of cells to transport independent flows of data.

4.4. Routing and Timing Parents

At all times, a TSCH mote needs to have a time source neighbor it can synchronize to. The LLC therefore needs to assign a time source neighbor to allow for correct operation of the TSCH network. A time source neighbors could, or not, be taken from the RPL routing parent set.

4.5. Resource Management

A cell in a TSCH schedule is an atomic "unit" of resource. The number of cells to assign between neighbor motes needs to be appropriate for the size of the traffic flow. The LLC needs to:

1. Define a mechanism for neighbor nodes to exchange information about their schedule and, if applicable, negotiate the addition/ deletion of cells.

2. Allow for an entity (e.g., a set of devices, a distributed protocol, a PCE, etc.) to take control of the schedule.

4.6. Dataflow Control

TSCH defines mechanisms for a mote to signal it cannot accept an incoming packet. It does not, however, define the policy which determines when to stop accepting packets. The LLC needs to:

1. Define a queuing policy for incoming and outgoing packets.

2. Manage the buffer space, and indicate to TSCH when to stop accepting incoming packets.

3. Handle transmissions that have failed. A transmission is declared failed when TSCH has retransmitted the packet multiple times, without receiving an acknowledgment. This covers both dedicated and shared cells.

4.7. Deterministic Behavior

As highlighted in [RFC5673], in some applications, data is generated periodically and has a well understood data bandwidth requirement, which is deterministic and predictable. The LLC needs to:

1. Ensure timely delivery of such data.

2. Provide a mechanism for such deterministic flows to coexist with bursty or infrequent traffic flows of different priorities.

4.8. Scheduling Mechanisms

Several scheduling mechanisms can be envisioned, and possibly coexist in the same network. For example, [I-D.phinney-roll-rpl-industrial-applicability] describes how the allocation of bandwidth can be optimized by an external Path Computation Element (PCE). Alternatively, two neighbor nodes can adapt the number of cells autonomously by monitoring the amount of traffic, and negotiating the allocation to extra cell when needed. This mechanism can be used to establish multi-hop paths in a fashion similar to RSVP. The LLC needs to:

1. Provide a mechanism for two 6TiSCH devices to negotiate the allocation and deallocation of cells between them.

2. Provide a mechanism for device to monitor and manage the 6TiSCH capabilities of a node several hops away.

3. Define an mechanism for these different scheduling mechanisms to coexist in the same network.

4.9. Secure Communication

Given some keying material, TSCH defines mechanisms to encrypt and authenticate MAC frames. It does not define how this keying material is generated. The LLC needs to:

1. Define the keying material and authentication mechanism needed by a new mote to join an existing network.

2. Define a mechanism to allow for the secure transfer of application data between neighbor motes.

3. Define a mechanism to allow for the secure transfer of signaling data between motes and the LLC.

5. IANA Considerations

This memo includes no request to IANA.

6. Security Considerations

This memo is an informational overview of existing standards, and does define any new mechanisms or protocols.

It does describe the need for the 6TiSCH WG to define a secure solution. In particular, Section 4.1 describes security in the join process. Section 4.9 discusses data frame protection.

7. Acknowledgments

Special thanks to Jonathan Simon for his review and valuable comments. Thanks to the IoT6 European Project (STREP) of the 7th Framework Program (Grant 288445).

8. References

8.1. Normative References

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

8.2. Informative References

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, June 2014.

[RFC6606] Kim, E., Kaspar, D., Gomez, C., and C. Bormann, "Problem Statement and Requirements for IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Routing", RFC 6606, May 2012.

[RFC6568] Kim, E., Kaspar, D., and JP. Vasseur, "Design and Application Spaces for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)", RFC 6568, April 2012.

[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.

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

[RFC5867] Martocci, J., De Mil, P., Riou, N., and W. Vermeylen, "Building Automation Routing Requirements in Low-Power and Lossy Networks", RFC 5867, June 2010.

[RFC5826] Brandt, A., Buron, J., and G. Porcu, "Home Automation Routing Requirements in Low-Power and Lossy Networks", RFC 5826, April 2010.

[RFC5673] Pister, K., Thubert, P., Dwars, S., and T. Phinney, "Industrial Routing Requirements in Low-Power and Lossy Networks", RFC 5673, October 2009.

[RFC5548] Dohler, M., Watteyne, T., Winter, T., and D. Barthel, "Routing Requirements for Urban Low-Power and Lossy Networks", RFC 5548, May 2009.

[RFC4944] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", RFC 4944, September 2007.

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

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

[RFC3819] Karn, P., Bormann, C., Fairhurst, G., Grossman, D., Ludwig, R., Mahdavi, J., Montenegro, G., Touch, J., and L. Wood, "Advice for Internet Subnetwork Designers", BCP 89, RFC 3819, July 2004.

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

[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-01 (work in progress), July 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-03 (work in progress), July 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-02 (work in progress), July 2014.

[I-D.ietf-6tisch-minimal]
Vilajosana, X. and K. Pister, "Minimal 6TiSCH Configuration", draft-ietf-6tisch-minimal-02 (work in progress), July 2014.

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

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

[I-D.ietf-6tisch-coap]
Sudhaakar, R. and P. Zand, "6TiSCH Resource Management and Interaction using CoAP", draft-ietf-6tisch-coap-01 (work in progress), July 2014.

[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.tsao-roll-security-framework]
Tsao, T., Alexander, R., Daza, V., and A. Lozano, "A Security Framework for Routing over Low Power and Lossy Networks", draft-tsao-roll-security-framework-02 (work in progress), March 2010.

[I-D.thubert-roll-asymlink]
Thubert, P., "RPL adaptation for asymmetrical links", draft-thubert-roll-asymlink-02 (work in progress), December 2011.

[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.ietf-roll-p2p-rpl]
Goyal, M., Baccelli, E., Philipp, M., Brandt, A., and J. Martocci, "Reactive Discovery of Point-to-Point Routes in Low Power and Lossy Networks", draft-ietf-roll-p2p-rpl-17 (work in progress), March 2013.

[I-D.ietf-roll-trickle-mcast]
Hui, J. and R. Kelsey, "Multicast Protocol for Low power and Lossy Networks (MPL)", draft-ietf-roll-trickle-mcast-09 (work in progress), April 2014.

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

[I-D.sarikaya-core-sbootstrapping]
Sarikaya, B., Ohba, Y., Moskowitz, R., Cao, Z., and R. Cragie, "Security Bootstrapping Solution for Resource-Constrained Devices", draft-sarikaya-core-sbootstrapping-04 (work in progress), April 2012.

[I-D.gilger-smart-object-security-workshop]
Gilger, J. and H. Tschofenig, "Report from the ’Smart Object Security Workshop’, 23rd March 2012, Paris, France", draft-gilger-smart-object-security-workshop-00 (work in progress), October 2012.

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

8.3. 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.

[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/>.

[OpenWSNETT]
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.

[IPSO]   "IP for Smart Objects Alliance Homepage", <http://www.ipso-alliance.org/>.

[CurrentCalculator]
Linear Technology, "Application Note: Using the Current Calculator to Estimate Mote Power", August 2012, <http://cds.linear.com/docs/en/application-note/Application_Note_-_Using_the_Current_Calculator_to_Estimate_Mote_Power.pdf>.

[doherty07channel]
Doherty, L., Lindsay, W., and J. Simon, "Channel-Specific Wireless Sensor Network Path Data", IEEE International Conference on Computer Communications and Networks (ICCCN) 2008, 2007.

[tinka10decentralized]
Tinka, A., Watteyne, T., and K. Pister, "A Decentralized Scheduling Algorithm for Time Synchronized Channel Hopping", Ad Hoc Networks 2010, 2010, < http://robotics.eecs.berkeley.edu/˜pister/publications/2008/TSMP%20DSN08.pdf>.

[watteyne09reliability]
Watteyne, T., Mehta, A., and K. Pister, "Reliability Through Frequency Diversity: Why Channel Hopping Makes Sense", International Conference on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks (PE-WASUN) 2009, Oct. 2009, <http://www.ietf.org/mail-archive/web/roll/current/pdfa_EzmuDIv3.pdf>.

[kerkez09feasibility]
Kerkez, B., Watteyne, T., and M. Magliocco, "Feasibility analysis of controller design for adaptive channel hopping", International Workshop on Performance Methodologies and Tools for Wireless Sensor Networks (WSNPERF) 2009, Oct. 2009, <http://www-bsac.eecs.berkeley.edu/publications/search/send_publication_pdf2client.php?pubID=1249681245>.

[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>.

[TASA-SENSORS]
Palattella, MR., Accettura, N., Dohler, M., Grieco, LA., and G. Boggia, "Traffic-Aware Time-Critical Scheduling In Heavily Duty-Cycled IEEE 802.15.4e For An Industrial IoT", IEEE SENSORS 2012, Oct. 2012, < http://www.cttc.es/resources/ doc/120821-sensors2012-4396981770946977737.pdf>.

[TASA-WCNC]
Accettura, N., Palattella, MR., Dohler, M., Grieco, LA., and G. Boggia, "Standardized Power-Efficient and Internet-Enabled Communication Stack for Capillary M2M Networks", IEEE WCNC 2012, Apr. 2012, < http://www.cttc.es/resources/ doc/120109-1569521283-submitted-58230.pdf>.

[palattella12standardized]
Palattella, MR., Accettura, N., Vilajosana, X., Watteyne, T., Grieco, LA., Boggia, G., and M. Dohler, "Standardized Protocol Stack For The Internet Of (Important) Things", IEEE Communications Surveys and Tutorials 2012, Dec. 2012, < http://www.cttc.es/resources/doc/121025- completestackforiot-clean-4818610916636121981.pdf>.

[PANA]   Kanda, M., Ohba, Y., Das, S., and S. Chasko, "PANA applicability in constrained environments", Febr. 2012, <h ttp://www.lix.polytechnique.fr/hipercom/SmartObjectSecurit y/papers/MitsuruKanda.pdf>.

Appendix A. TSCH Protocol Highlights

This appendix gives an overview of the key features of the IEEE802.15.4e Timeslotted Channel Hopping (TSCH) amendment. It makes no attempt at repeating the standard, but rather focuses on the following:

o Concepts which are sufficiently different from traditional IEEE802.15.4 networking that they may need to be defined and presented precisely.

o Techniques and ideas which are part of IEEE802.15.4e and which might be useful for the work of the 6TiSCH WG.

A.1. Timeslots

All motes in a TSCH network are synchronized. Time is sliced up into timeslots. A timeslot is long enough for a MAC frame of maximum size to be sent from mote A to mote B, and for mote B to reply with an acknowledgment (ACK) frame indicating successful reception.

The duration of a timeslot is not defined by the standard. With IEEE802.15.4-compliant radios operating in the 2.4GHz frequency band, a maximum-length frame of 127 bytes takes about 4ms to transmit; a shorter ACK takes about 1ms. With a 10ms slot (a typical duration), this leaves 5ms to radio turnaround, packet processing and security operations.

A.2. Slotframes

Timeslots are grouped into one of more slotframes. A slotframe continuously repeats over time. TSCH does not impose a slotframe size. Depending on the application needs, these can range from 10s to 1000s of timeslots. The shorter the slotframe, the more often a timeslot repeats, resulting in more available bandwidth, but also in a higher power consumption.

A.3. Node TSCH Schedule

A TSCH schedule instructs each mote what to do in each timeslot: transmit, receive or sleep. The schedule indicates, for each scheduled (transmit or receive) cell, a channelOffset and the address of the neighbor to communicate with.

Once a mote obtains its schedule, it executes it:

- o For each transmit cell, the mote checks whether there is a packet in the outgoing buffer which matches the neighbor written in the schedule information for that timeslot. If there is none, the mote keeps its radio off for the duration of the timeslot. If there is one, the mote can ask for the neighbor to acknowledge it, in which case it has to listen for the acknowledgment after transmitting.

- o For each receive cell, the mote listens for possible incoming packets. If none is received after some listening period, it shuts down its radio. If a packet is received, addressed to the mote, and passes security checks, the mote can send back an acknowledgment.

How the schedule is built, updated and maintained, and by which entity, is outside of the scope of the IEEE802.15.4e standard.

A.4. Cells and Bundles

Assuming the schedule is well built, if mote A is scheduled to transmit to mote B at slotOffset 5 and channelOffset 11, mote B will be scheduled to receive from mote A at the same slotOffset and channelOffset.

A single element of the schedule characterized by a slotOffset and channelOffset, and reserved for mote A to transmit to mote B (or for mote B to receive from mote A) within a given slotframe, is called a "scheduled cell".

If there is a lot of data flowing from mote A to mote B, the schedule might contain multiple cells from A to B, at different times. Multiple cells scheduled to the same neighbor can be equivalent, i.e. the MAC layer sends the packet on whichever of these cells shows up first after the packet was put in the MAC queue. The union of all cells between two neighbors, A and B, is called a "bundle". Since the slotframe repeats over time (and the length of the slotframe is typically constant), each cell gives a "quantum" of bandwidth to a given neighbor. Modifying the number of equivalent cells in a bundle modifies the amount of resources allocated between two neighbors.

A.5. Dedicated vs. Shared Cells

By default, each scheduled transmit cell within the TSCH schedule is dedicated, i.e., reserved only for mote A to transmit to mote B. IEEE802.15.4e allows also to mark a cell as shared. In a shared cell, multiple motes can transmit at the same time, on the same frequency. To avoid contention, TSCH defines a back-off algorithm for shared cells.

A scheduled cell can be marked as both transmitting and receiving. In this case, a mote transmits if it has an appropriate packet in its output buffer, or listens otherwise. Marking a cell as [transmit,receive,shared] results in slotted-Aloha behavior.

A.6. Absolute Slot Number

TSCH defines a timeslot counter called Absolute Slot Number (ASN). When a new network is created, the ASN is initialized to 0; from then on, it increments by 1 at each timeslot. In detail:

ASN = (k*S+t)

where k is the slotframe cycle (i.e., the number of slotframe repetitions since the network was started), S the slotframe size and t the slotOffset. A mote learns the current ASN when it joins the network. Since motes are synchronized, they all know the current value of the ASN, at any time. The ASN is encoded as a 5-byte number: this allows it to increment for hundreds of years (the exact value depends on the duration of a timeslot) without wrapping over. The ASN is used to calculate the frequency to communicate on, and can be used for security-related operations.

A.7. Channel Hopping

For each scheduled cell, the schedule specifies a slotOffset and a channelOffset. In a well-built schedule, when mote A has a transmit cell to mote B on channelOffset 5, mote B has a receive cell from mote A on the same channelOffset. The channelOffset is translated by both nodes into a frequency using the following function:

```
frequency = F {(ASN + channelOffset) mod nFreq}
```

The function F consists of a look-up table containing the set of available channels. The value nFreq (the number of available frequencies) is the size of this look-up table. There are as many channelOffset values as there are frequencies available (e.g. 16 when using IEEE802.15.4-compliant radios at 2.4GHz, when all channels are used). Since both motes have the same channelOffset written in their schedule for that scheduled cell, and the same ASN counter, they compute the same frequency. At the next iteration (cycle) of the slotframe, however, while the channelOffset is the same, the ASN has changed, resulting in the computation of a different frequency.

This results in "channel hopping": even with a static schedule, pairs of neighbors "hop" between the different frequencies when communicating. A way of ensuring communication happens on all available frequencies is to set the number of timeslots in a slotframe to a prime number. Channel hopping is a technique known to efficiently combat multi-path fading and external interference.

A.8. Time Synchronization

Because of the slotted nature of communication in a TSCH network, motes have to maintain tight synchronization. All motes are assumed to be equipped with clocks to keep track of time. Yet, because clocks in different motes drift with respect to one another, neighbor motes need to periodically re-synchronize.

Each mote needs to periodically synchronize its network clock to another mote, and it also provides its network time to its neighbors. It is up to the entity that manages the schedule to assign an adequate time source neighbor to each mote, i.e., to indicate in the schedule which of neighbor is its "time source neighbor". While setting the time source neighbor, it is important to avoid synchronization loops, which could result in the formation of independent clusters of synchronized motes.

TSCH adds timing information in all packets that are exchanged (both data and ACK frames). This means that neighbor motes can resynchronize to one another whenever they exchange data. In detail, two methods are defined in IEEE802.15.4e-2012 for allowing a device to synchronize in a TSCH network: (i) Acknowledgment-Based and (ii) Frame-Based synchronization. In both cases, the receiver calculates the difference in time between the expected time of frame arrival and its actual arrival. In Acknowledgment-Based synchronization, the receiver provides such information to the sender mote in its acknowledgment. In this case, it is the sender mote that synchronizes to the clock of the receiver. In Frame-Based synchronization, the receiver uses the computed delta for adjusting its own clock. In this case, it is the receiver mote that synchronizes to the clock of the sender.

Different synchronization policies are possible. Motes can keep synchronization exclusively by exchanging EBs. Motes can also keep synchronized by periodically sending valid frames to a time source neighbor and use the acknowledgment to resynchronize. Both method (or a combination thereof) are valid synchronization policies; which one to use depends on network requirements.

A.9. Power Consumption

There are only a handful of activities a mote can perform during a timeslot: transmit, receive, or sleep. Each of these operations has some energy cost associated to them, the exact value depends on the the hardware used. Given the schedule of a mote, it is straightforward to calculate the expected average power consumption of that mote.

A.10. Network TSCH Schedule

The schedule entirely defines the synchronization and communication between motes. By adding/removing cells between neighbors, one can adapt a schedule to the needs of the application. Intuitive examples are:

- o Make the schedule "sparse" for applications where motes need to consume as little energy as possible, at the price of reduced bandwidth.

- o Make the schedule "dense" for applications where motes generate a lot of data, at the price of increased power consumption.

- o Add more cells along a multi-hop route over which many packets flow.

A.11. Join Process

Motes already part of the network can periodically send Enhanced Beacon (EB) frames to announce the presence of the network. These contain information about the size of the timeslot used in the network, the current ASN, information about the slotframes and timeslots the beaconing mote is listening on, and a 1-byte join priority. Even if a node is configured to send all EB frames on the same channel offset, because of the channel hopping nature of TSCH described in Appendix A.7, this channel offset translates into a different frequency at different slotframe cycles. As a result, EB frames are sent on all frequencies.

A mote wishing to join the network listens for EBs. Since EBs are sent on all frequencies, the joining node can listen on any frequency until it hears an EB. What frequency it listens on, and whether it slowly changes frequency during this joining period is implementation-specific. Using the ASN and the other timing information of the EB, the new mote synchronizes to the network. Using the slotframe and cell information from the EB, it knows how to contact other nodes in the network.

The IEEE802.15.4e TSCH standard does not define the steps beyond this network "bootstrap".

A.12. Information Elements

TSCH introduces the concept of Information Elements (IEs). An information element is a list of Type-Length-Value containers placed at the end of the MAC header. A small number of types are defined for TSCH (e.g., the ASN in the EB is contained in an IE), and an unmanaged range is available for extensions.

A data bit in the MAC header indicates whether the frame contains IEs. IEs are grouped into Header IEs, consumed by the MAC layer and therefore typically invisible to the next higher layer, and Payload IEs, which are passed untouched to the next higher layer, possibly followed by regular payload. Payload IEs can therefore be used for the next higher layers of two neighbor motes to exchange information.

A.13. Extensibility

The TSCH standard is designed to be extensible. It introduces the mechanisms as "building block" (e.g., cells, bundles, slotframes, etc.), but leaves entire freedom to the upper layer to assemble those. The MAC protocol can be extended by defining new Header IEs. An intermediate layer can be defined to manage the MAC layer by defining new Payload IEs.

Appendix B. TSCH Gotchas

This section lists features of TSCH which we believe are important and beneficial to the work of 6TiSCH.

B.1. Collision Free Communication

TSCH allows one to design a schedule which yields collision-free communication. This is done by building the schedule with dedicated cells in such a way that at most one node communicates with a specific neighbor in each slotOffset/channelOffset cell. Multiple pairs of neighbor motes can exchange data at the same time, but on different frequencies.

B.2. Multi-Channel vs. Channel Hopping

A TSCH schedule looks like a matrix of width "slotframe size", S, and of height "number of frequencies", nFreq. For a scheduling algorithm, these can be considered atomic "units" to schedule. In particular, because of the channel hopping nature of TSCH, the scheduling algorithm should not worry about the actual frequency communication happens on, since it changes at each slotframe iteration.

B.3. Cost of (continuous) Synchronization

When there is traffic in the network, motes which are communicating implicitly re-synchronize using the data frames they exchange. In the absence of data traffic, motes are required to synchronize to their time source neighbor(s) periodically not to drift in time. If they have not been communicating for some time (typically 30s), motes can exchange an dummy data frame to re-synchronize. The frequency at which such messages need to be transmitted depends on the stability of the clock source, and on how "early" each mote starts listening for data (the "guard time"). Theoretically, with a 10ppm clock and a 1ms guard time, this period can be 100s. Assuming this exchange causes the mote’s radio to be on for 5ms, this yields a radio duty cycle needed to keep synchronized of 5ms/100s=0.005%. While TSCH does requires motes to resynchronize periodically, the cost of doing so is very low.

B.4. Topology Stability

The channel hopping nature of TSCH causes links to be very "stable". Wireless phenomena such as multi-path fading and external interference impact a wireless link between two motes differently on each frequency. If a transmission from mote A to mote B fails, retransmitting on a different frequency has a higher likelihood of succeeding that retransmitting on the same frequency. As a result, even when some frequencies are "behaving bad", channel hopping "smoothens" the contribution of each frequency, resulting in more stable links, and therefore a more stable topology.

B.5. Multiple Concurrent Slotframes

The TSCH standard allows for multiple slotframes to coexist in a mote’s schedule. It is possible that, at some timeslot, a mote has multiple activities scheduled (e.g. transmit to mote B on slotframe 2, receive from mote C on slotframe 1). To handle this situation, the TSCH standard defines the following precedence rules:

1. Transmissions take precedence over receptions;

2. Lower slotframe identifiers take precedence over higher slotframe identifiers.

In the example above, the mote would transmit to mote B on slotframe 2.

Authors’ Addresses

Thomas Watteyne (editor)
Linear Technology
30695 Huntwood Avenue
Hayward, CA 94544
USA

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

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

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

Phone: +39 08 05 96 3911
Email: a.grieco@poliba.it

Network Working Group   M. Richardson
Internet-Draft   SSW
Intended status: Informational   October 26, 2014
Expires: April 29, 2015

6tisch secure join using 6top
draft-richardson-6tisch--security-6top-03

Abstract

This document details a security architecture that permits a new 6tisch compliant node to join an 802.15.4e network. The process bootstraps the new node authenticating the node to the network, and the network to the node, and configuring the new node with the required 6tisch schedule. Any resemblance to WirelessHART/IEC62591 is entirely intentional.

Requirements Language

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

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

A challenging part with constructing an LLN with nodes from multiple vendors is providing enough security context to each node such that the network communication can form and remain secure. Most LLNs are small and have no operator interfaces at all, and even if they have debug interfaces (such as JTAG) with personnel trained to use that, doing any kind of interaction involving electrical connections in a dirty environment such as a factory or refinery is hopeless.

It is necessary to have a way to introduce new nodes into a 6tisch network that does not involve any direct manipulation of the nodes themselves. This act has been called "zero-touch" provisioning, and it does not occur by chance, but requires coordination between the manufacturer of the node, the service operator running the LLN, and the installers actually taking the devices out of the shipping boxes.

1.1. Assumptions

For the process described in this document to work, some assumptions about available infrastructure are made. These are perhaps more than assumptions, but rather architectural requirements; the exact operation of said infrastructure to be defined in a subsequent document.

In the diagrams and text that follows entities are named (and defined in the terminology section). Unless otherwise stated these are roles, not actual machines/systems. The roles are seperated by network protocols in order that they roles can be performed by different systems, not because they have to be. Different deployments will have different scaling requirements for those entities. Smaller deployments might co-located many roles together into a single ruggedized platform, while other deployments might operate all of the roles on distinct, multiply-redundant server classes located in a fully equipped datacentre.

2. Terminology and Roles

Most terminology should be taken from [I-D.ietf-6tisch-architecture] and from [I-D.ietf-6tisch-6top-interface] and [I-D.wang-6tisch-6top-sublayer]. As well, many terms are taken from [RFC6775].

The following roles/things are defined:

PCE	the Path Computation Engine. This entity reaches out to each of the nodes in the LLN, and configures an appropriate schedule using 6top.
Authz Server/ACE	the Authorization Server. This offloads calculation of access control lists and other access control decisions for constrainted nodes. See [I-D.seitz-ace-problem-description]
6top	the 6top protocol is defined abstractly in [I-D.ietf-6tisch-6top-interface] and mapped to run over CoAP in [I-D.ietf-6tisch-coap]. The 6top protocol is defined primarily to provision the 6TiSCH schedule; this document proposes to extend it for also provisioning of layer-2 security parameters.
JCE	the Join Coordination Entity. This acronym is chosen to parallel the PCE.
joining node	The newly unboxed constrained node that needs to join a network.
join protocol	the protocol which secures initial communication between the joining node and the JCE
join assistant	A constrained node near the joining node that will act as it’s first 6LR, and will relay traffic to/from the joining node.
join network	A 802.15.4e network whose encryption and authentication key is "JOIN6TISCH".

unique join key	a key shared between a newly joining node, and the JCE
production network	A 802.15.4e network whose encryption/ authentication keys are determined by some algorithm. There may have network-wide group keys, or per-link keys.
production network key	A shared L2-key known by all authorized nodes. This key can be used to derive other keys.
per-peer L2 key	a key that results from an exchange (such as MLE) that creates a pair-wise L2 key which is known only to the two nodes involved, [I-D.piro-6tisch-security-issues] calls this a LinkKey

The following terms are used in this document and come from other documents:

DevID	[IEEE.802.1AR] defines the secure DEVice IDentifier as a device identifier that is cryptographically bound to the device and is composed of the Secure Device Identifier Secret and the Secure Device Identifier Credential.
IDevID	The Initial secure DEVice IDentifier (IDevID) is the Device Identifier which was installed on the device by the manufacturer.
LDevID	A Locally significant secure DEVice IDentifiers (LDevIDs) is a Secure Device Identifier credential that is unique in the local administrative domain in which the device is used. The LDevID is usually a new certificate provisioned by some local means, such as the 6top mechanism defined in this document.
CoAP	The CoAP protocol, defined in [RFC7252] is an HTTP-like resource access protocol. CoAP runs over UDP.
DTLS	The datagram version of TLS, defined in [RFC6347], and which can be used to secure CoAP in the same way that TLS secures HTTP.

ARO	[RFC6775]defines a number of new Neighbor Discovery options including the Address Registration Option
DAR/DAC	[RFC6775]defines the Duplicate Address Request and Duplicate Address Confirmation options to turn the multicasted Duplicate Address Detection protocol into a client/server process
EARO	[I-D.thubert-6lo-rfc6775-update-reqs]extends the ARO option to include some additional fields necessary to distinguish duplicate addresses from nodes that have moved networks when there are mulitple LLNs linked over a backbone.

3. Architectural requirements of join protocol

This section works from the ultimate goal, and goes backwards to prerequisite actions. Section 6 presents the protocol from beginning to end order.

The ultimate goal of the join protocol is to provide a new node with enough locally significant security credentials that it is able to take part in the network directly. The credentials may vary by deployment. They can be one of:

1) a network-wide shared symmetric key (this is the production network key, or MasterKey)

2) a locally significant (one-level only) 802.11AR type DevID certificate (which allows it to negotiate a pair-wise key)

One of these items is communicated by the JCE to the joining node using the 6top protocol. The authentication of this communication channel is the subject of the Join Protocol as explained below.

Given one of the the above, there are a number of possible protocols that can be used to generate layer-2 sessions keys for the node, including:

1) Mesh Link Exchange [I-D.kelsey-intarea-mesh-link-establishment] (IMPORTANT, a good option. Uses certificates from common CA)

2) work in 802.15.9 (uses certificates from common CA)

3) Security Framework and Key Management Protocol Requirements for 6TiSCH [I-D.ohba-6tisch-security] (this document provides the phase 0 required, using the network-wide shared key)

4) Layer-2 security aspects for the IEEE 802.15.4e MAC [I-D.piro-6tisch-security-issues]: the MasterKey is used to derive per-peer L2 keys

Per-peer L2 keying is critical when doing peer2peer schedule negotiation over 15.4 Information Elements. Therefore a network-wide layer-2 key is inappropriate for the self-organizing networks, and a protocol (MLE, 802.15.9) SHOULD be used to derive per-peer L2 keys.

For networks where there is a PCE present and will do all schedule computation, then the only trust relationship necessary is between the individual node and the PCE, and it MAY be acceptable to have a network-wide L2 key derived in ways such as [I-D.piro-6tisch-security-issues] describes in section ?

The intermediate goal of the join protocol is to enable a Join Coordination Entity (JCE) to reach out to the new node, and install the credentials detailed above. The JCE must authenticate itself to the joining node so that the joining node will know that it has joined the correct network, and the joining node must authenticate itself to the JCE so that the JCE will know that this node belongs in the network. This two way authentication occurs in the 6top/CoAP/DTLS session that is established between the JCE and the joining node.

[I-D.ietf-6tisch-6top-interface] presents a way to interface to a 6top information model (defined in YANG). [I-D.ietf-6tisch-coap] explains how to access that information model using CoAP. That model is to be extended to include security attributes for the network. The JCE would therefore reach out to the joining node and simply provision appropriate security properties into the joining node, much like the PCE will provision schedules.

This 6top-based secure join protocol has defined a push model for security provisioning by the JCE. This has been done for three reasons:

1) 6tisch nodes already have to have a 6top CoAP server for schedule provising

2) this permits the JCE to manage how many nodes are trying to join at the same time, and limit how much bandwidth/energy is used for the join operation, and also for the JCE to prioritize the join order for nodes.

3) making the JCE initiate the DTLS connection significantly simplies the certificate chains that must be exchanged as the most constrained side (the joining node) provides it’s

credentials first, and lets the less constrained JCE figure out what kind of certificate chain will be required to authenticate the JCE to the joining node. In EAP-TLS/802.1x situations, the TLS channel is created in the opposite direction, and it would have to complete in a tentative way, and then further authorization occur in-band.

In order for a 6top/DTLS/CoAP connection to occur between the JCE and the joining node, there needs to be end-to-end IPv6 connectivity between those two entities. The joining node will not participate in the route-over RPL mesh, but rather will be seen by the network as being a 6lowpan only leaf-node.

There are some alternatives to having full end to end connectivity which are discussed in the security considerations section.

The specific mechanism to enable end to end connectivity with the JCE are still open but will consist of one of:

(1) IPIP tunnel between Join Assistant and JCE (least preferred)

(2) using straight RPL routing: the Join Assistant sends a DAO (moderate preference)

(3) using a separate RPL DODAG for join traffic (could be a non-storing, best practice)

(4) establishing a specific multi-hop 6tisch track for join traffic for each Join Assistant (not always practical)

Of these mechanisms, the only one which does not require additional state on the Join Assistant (which is also a constrained device) is (1) and (2). Mechanism (2) additionally requires no specific state on the Join Assistant. Mechanism (2), in a non-storing DODAG requires additional state on the DODAG root (6LBR) only; while mechanism (1) requires a similar amount of state on the JCE. For deployments where the JCE is part of the 6LBR, the amount of state is similar, but in any case, the 6LBR is assumed to be a non-constrained node.

As long as the Join Assistant does not do any kind of stateful firewalling, the IPIP tunnel and the DAO (2) method can be done by the Join Assistant statelessly. Upward traffic from the Join Network must be restricted to a 6tisch slotframe(s) to which join traffic is welcome, no tunnelling is necessary as the upwards routes are all in place. A destination address ACL on traffic from the Join Network restricts the Joining Nodes to sending traffic only to the address of the JCE. (If JCE and 6LBR are colocated, then this is the address in the ABRO, if they are not colocated, then this address needs to have been provisioning in the Join Assistant when it joined, or could be carried in a new RA option)

When using option (2), networks that have storing mode DODAGs will consume routing resources on all intermediate nodes between the Join Assistant and the DODAG root. This resource will be depleted without any authentication, and this threat is detailed below.

Continuing to work backwards, in order the JCE reach out to provision the Joining Node, it needs to know that the new node is present. This is done by taking advantage of the 6lowPAN Address Resolution Option (ARO) (section 4.1 [RFC6775]). The ARO causes the new address to also be sent up to the 6LBR for duplicate detection using the DAR/ DAC mechanism. The 6LBR simply needs to tell the JCE about this using a protocol that needs to be defined, but could be either DAR or NS.

In addition to needing to know the joining devices address from the DAR/NS, the JCE also needs to know the joining node’ IDevID. If the serialNumber attribute of the IDevID is less than 64 bits, then it is possible that it could be placed into the EUI-64 option of the ARO, or the OUI of the [I-D.thubert-6lo-rfc6775-update-reqs] EARO. The JCE needs to know the joining node’s serialNumber to know if this is device that it should even attempt to provision; and if so, it may need to retrieve an appropriate certificate chain (see [I-D.richardson-6tisch-idevid-cert]) from the Factory in order for the JCE to prove it is the legitimate owner of the joining node.

Neither 802.1AR nor [RFC5280] provide any structure for the serialNumber, except that they are positive integers of up-to 20 octets in size (numbers up to 2^160). This specification would require that the serialNumber encoded in the IDevID is the same as the EUI-64 used by the device. Some consideration needs to be given as to whether there are privacy considerations to doing this: any observer that can see the join traffic, can also see the source MAC address of the node as well.

Prior to being able to announce itself in a NS, the joining node needs to find the Join Network. This is done by listening to an extended beacon which are broadcast in designated slotframes by Join Assistants. The Extended Beacon provides a way for the Joining Node to synchronize itself to the overall timeslot schedule and provides an Aloha period in which the Joining Node can send a Router Soliticiation, and receive an appropriate Router Advertisement giving the Joining Node a prefix and default route to which to send join traffic.

It may be possible to eliminate a message exchange if space for a Router Advertisement can be found as part of the Join Network Extended Beacon. This Enhanced Beacon would be distinct to the Join Network, and would be encrypted with the well-known Join Network key.

3.1. prefixes to use for join traffic

What prefix would the joining node for communication? There are three options:

(1) just use link-local addresses (requires all traffic be tunneled)

(2) use a prefix specifically for join traffic (may be easier with a join-only DODAG)

(3) use the same prefix as the rest of the traffic (may require more complex ACLs, and leaks information to attackers)

4. security requirements

4.1. threat model

There are three kinds of threats that a join process must deal with: threats to the joining node, threats to the resources of the network, and threats to other joining nodes.

4.1.1. threats to the joining node

A node may be taken out of it’s box by a malicious entity and powered on. This could happen during shipping, while being stored in a warehouse. The device may be subject to physical theft, or the goal of the attacker may be to turn the device into a trojan horse of some kind. Physical protection of the device is out of scope for this document; this document will henceforth assume that the device is sealed in some tamper-evident way and this document deals with attacks over the network.

An attacker may attempt to convince the joining node that it is the legitimate Production Network; this is done by putting up a legitimate looking Join Network, and following the protocol as described in this document. The Joining Node can not know if it has the corrrect Production Network until steps 11-13, when it attempts to validate the ClientCertificate provided by the JCE.

When the joining node determines that this is the incorrect network, it must remember the PANID of the network that it has attempted to join, and then look for another network to try. It SHOULD have some limit as the number of times it will try before going back to sleep, or shutting down, and it SHOULD take care not to consume more than some specified percentage of any battery it might have.

Should a malicious production network be present at the same time/ place as the legitimate production network, a the malicious agent could intercept and replay various packets from the proper join network, but ultimately this either results in a jamming-like denial of service, and/or the the ClientCertificate will not validate.

It is a legitimate situation for there to be multiple possible join networks, and the joining node may have to try each one before it finds the network that it the right one for it. The incorrect, but non-malicious networks will not attempt the 6top provisioning step, and SHOULD return a negative result in steps 8/9, refusing the node's NS. Those incorrect networks will be recognize that the node does not belong to them, because they will be able to see the Joining Node's IDevID in the ARO of step 4.

4.1.2. threats to the resources of the network

The production network has two important resources that may be attacked by malicious Joining nodes: 1) energy/bandwidth, 2) memory for routing entries.

A malicious joining node could send many NS messages to the Join Assistant (from many made up addresses), which would send many NS/DAR messages to the 6LBR, and this would consume bandwidth, and therefore energy from the members of the mesh along the path to the 6LBR. This can be mitigated by limited the total bandwidth available for joining.

A malicious joining node could send many NS messages, and if the 6LBR agreed to accept the new node (by IDevID), then the Join Assistant would MAY inject routing information into mesh for the Joining node. Non-storing DODAGs store are routing information in the DODAG Root (probably the 6LBR), which is generally not a constrained node. Storing DODAGs store routing entries at all nodes up to the DODAG, and those are constrained nodes. Using a separate Join DODAG, and having that DODAG be non-storing will reduce any impact on intermediate nodes, but it does cause resources to be used for the second DODAG, and it may have a code impact if the nodes otherwise would not implement non-storing RPL.

4.1.3. threats to other joining nodes

A joining node (or the nodes of a malicious network, co-located near the legitimate production network) may mount attacks on legitimate nodes which have not yet joined.

The malicious nodes may attempt to perform 6top operations against the joining node to keep it from being able to respond to the legitimate 6top session from the legitimate JCE. During the Join phase, the Joining node MUST have all other resources and protocols turned off, even if they would normally be accessible as read-only unauthenticated CoAP resources.

Malicious nodes could use the Join Network to mount various DTLS based attacks against the joining node, such as sending very long certificate chains to validate. One might think to limit the length of such chains, but as shown in [I-D.richardson-6tisch-idevid-cert] the chain may as a long as the supplier chain, plus may include additional certificates due to resales of plants/equipment/etc. Validating from a trusted certificate down to the specific certificate which proves ownership would eliminate random certificate chains, but the attacker could just feed the joining node legitimate chains that it observed (and replayed) from the legitimate JCE. This does no good; the Joining node finds that the DTLS connection is invalid, but it may significantly run batteries down.

4.2. implementation cost

(storage of security material, computational cost)

4.3. denial of service

other communication impacts of security protocol mechanics

5. protocol requirements/constraints/assumptions

5.1. inline/offline

dependencies on centralized or external functionality, inline and offline

6. time sequence diagram

```
   +-----+   +------+           +----------+             +-----------+
   |     |   |      |           |  JOIN    |             |  Joining  |
   | JCE |   | 6LBR |           | Assistant|             |   Node    |
   +-----+   +------+           |  (proxy) |             |           |
      |         |               +----------+             +-----------+
      |         |                     |                            |
      |         |                     |-------BEACON (1)---------->|
      |         |                     |                            |
      |         |                     |<----Router Solicitation----|
      |         |                     |---Router Advertisement---->|
      |         |                     |                            |
      |         |                     |<------CERT CACHE-----------|
      |         |                     |         LOAD    (2)        |
      |         |                     |                            |
      |         |                     |                            |
      |         |                     |-------CERT CACHE---------->|
      |         |                     |        RESPONSE (multiple) |
      |         |                     |                 (packets)  |
      |         |                     |                            |
      |         |                     |<-----JOIN REQUEST (4) -----|
      |         |                     |      (NS w/ARO )           |
      |         |                     |                            |
      |         |<---NS (DAR) (5)-----|                            |
      |<--??(6)-|                     |                            |
      |         |                     |                            |
      |--??(7)->|                     |                            |
      |         |                     |                            |
      |         |----NS (DAC)-(8)---->|                            |
      |         |     +------+        |                            |
      |         |<DAO-| mesh |<--DAO--|                            |
      |         |-DAO-| node |--DACK->|                            |
      |         | ACK +------+        |                            |
      |         |                     |-------JOIN ACK (9)-------->|
      |         |                     |                            |
      |         |                     |                            |
      |================(10)==========>|----------6top----(11)----->|
      |         |                     |          DTLS              |
      |<===============(13)===========|<---------CoAP----(12)------|
      |         |                     |        (many packets)      |
```

Figure 1: Message sequence for JOIN message

6.1. explanation of each step

6.1.1. step (1): enhanced beacon

A 6tisch join/synchronization beacon is broadcast periodically, and is authenticated with a symmetric "beacon key":

well known JOIN key, such "JOIN6TISCH"

another key, provisioned in advance (OOB)

a shared symmetric key derived from public part of top level certificate (a closely held "secret")

The purpose of this key is not to provide a high level of assurance, but rather to filter out 6tisch traffic from another random traffic that may be sharing the same radio frequencies.

These beacons are used for JOIN purpose only, and are not related to the Enhanced Beacons used in the rest of 6tisch.

6.1.2. step (1B): send router solicitation

The joining node sends a router solicitation during the Aloha period of the beacon.

6.1.3. step (1C): receive router advertisement

The joining node receives a router advertisement from the Join Assistant. It could include 6CO options to help compress packets, and should contain a prefix appropriate for join traffic.

6.1.4. step (2): certificate cache load

At step 10, the JCE will need to present a certificate chain anchored at a trusted CA built into the joining node. It has been speculated that a significant amount of traffic could be avoided at step (10) if the common parts of the certificate chains could be cached in the join assistant.

This optional step involves the joining node asking for certificates from the join assistant.

6.1.5. step (3): receive certificate cache

the proxy neighbour sends requested cached certificates to the joining node

6.1.6. step (4): join request

A regular Neighbour Solicitation is sent. This should contain an ARO (or EARO) option containing the Joining Nodes' IDevID. The ARO/EARO will be proxied by the Join Assistant as part of normal 6LowPAN processing for leaf nodes (non-RPL nodes) upwards to the 6LBR

6.1.7. step (5): NS duplicate address request (DAR)

6.1.8. step (7): 6LBR informs JCE of new node

6.1.9. step (8): JCE informs/acks to 6LBR of new node

The JCE could reply in the negative, and this would cause a DAC failure, TBD

6.1.10. step (9): NS duplicate address confirmation (DAC)

6.1.11. step (10): JCE initiates connection to joining node

The double lines indicate that an IPIP tunnel operation may be required. If a straight DAO or seperate Join DODAG is used, then this is just a straight forwarding root to leaf node forwarding operation, and involves either using source routes (non-storing), or just forwarding for storing DODAGs.

A specific bandwidth allocation would be used for this join traffic

The production network encryption keys would be used for the join traffic

6.1.12. step (11): Join Assistant forwards packet to joining node

The JOIN Assistant would forward traffic to the Joining Node. Recognizing that this traffic the JOIN Network, the JOIN Assistant would use the JOIN Network key.

6.1.13. step (12): Joining node replies

The joining node replies, using JOIN Network key.

6.1.14. step (13): Join Assistant forwards reply to JCE

The JOIN Assistant, recognizing that the traffic came from the JOIN Network, restricts the destination that can be reached to the the JCE only. It can do this in a stateless way, and it does NOT need to track the traffic at (10) to open pinhole, etc.

Recognizing that the traffic came from the JOIN Network, the traffic would be placed into a bandwidth allocation (track?) that allows such traffic.

6.2. size of each packet

and number of frames needed to contain it.

7. resulting security properties obtained from this process

An end to end IPv6 CoAP/DTLS connection is created between the JCE and the Joining Node. This connection carries 6top commands to update security parameters. This results in either deployment of a single-level, locally relevant certificate (LDevID), or deployment of a network-wide symmetric "Master Key"

8. deployment scenarios underlying protocol requirements

9. device identification

The JCE authenticates the joining node using a certificate chain provided inline during the DTLS negotiation. The certificate chain is rooted in a vendor certificate that the JCE must have preloaded, and is a statement as to the node’s 802.1AR IDevID. The joining node authenticates the

9.1. PCE/Proxy vs Node identification

9.2. Time source authentication / time validation

Note: RPL Root authentication is a chartered item

9.3. description of certificate contents

9.4. privacy aspects

The EUI-64 of the Joining node is transmitted using a Well Known layer-2 encryption key. Within the ARO/EARO of the Neighbour Solicitation is an OUI, which may be identical to the EUI-64 of the Joining node, or it might be an unrelated IDevID.

An eavesdropper can therefore learn something about the manufacturer of every device as it joins.

10. slotframes to be used during join

how is this communicated in the (extended) beacon.

11. configuration aspects

(allocation of slotframes after join, network statistics, neighboetc.)

12. authorization aspects

lifecycle (key management, trust management)

12.1. how to determine a proxy/PCE from a end node

12.2. security considerations

what prevents a node from transmitting when it is not their turn (part one: jamming)

can a node successfully communicate with a peer at a time when not supposed to, may be tied to link layer security, or will it be policed by receiver?

13. security architecture

security architecture and fit of e.g. join protocol and provisioning into this

14. Posture Maintenance

(SACM related work)

15. Security Considerations

16. Other Related Protocols

17. IANA Considerations

18. Acknowledgements

19. References

19.1. Normative references

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

[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.

[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.

[IEEE.802.1AR]
Institute of Electrical and Electronics Engineers, "Secure Device Identity", IEEE 802.1AR, 2009, <http://www.ieee802.org/1/pages/802.1ar.html>.

[I-D.ietf-6tisch-coap]
Sudhaakar, R. and P. Zand, "6TiSCH Resource Management and Interaction using CoAP", draft-ietf-6tisch-coap-00 (work in progress), May 2014.

[I-D.ietf-6tisch-6top-interface]
Wang, Q., Vilajosana, X., and T. Watteyne, "6TiSCH Operation Sublayer (6top) Interface", draft-ietf-6tisch-6top-interface-00 (work in progress), March 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.irtf-nmrg-autonomic-network-definitions]
Behringer, M., Pritikin, M., Bjarnason, S., Clemm, A., Carpenter, B., Jiang, S., and L. Ciavaglia, "Autonomic Networking - Definitions and Design Goals", draft-irtf-nmrg-autonomic-network-definitions-00 (work in progress), December 2013.

[I-D.seitz-ace-problem-description]
Seitz, L. and G. Selander, "Problem Description for Authorization in Constrained Environments", draft-seitz-ace-problem-description-00 (work in progress), May 2014.

[I-D.richardson-6tisch-idevid-cert]
Richardson, M., "X509.v3 certificate extension for authorization of device ownership", draft-richardson-6tisch-idevid-cert-00 (work in progress), May 2014.

[I-D.wang-6tisch-6top-sublayer]
Wang, Q., Vilajosana, X., and T. Watteyne, "6TiSCH Operation Sublayer (6top)", draft-wang-6tisch-6top-sublayer-00 (work in progress), February 2014.

[I-D.thubert-6lo-rfc6775-update-reqs]
Thubert, P., "Requirements for an update to 6LoWPAN ND", draft-thubert-6lo-rfc6775-update-reqs-04 (work in progress), August 2014.

19.2. Informative references

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, May 2008.

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, June 2014.

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

[I-D.thubert-6lowpan-backbone-router]
Thubert, P., "LoWPAN Backbone Router", draft-thubert-6lowpan-backbone-router-00 (work in progress), March 2008.

[I-D.ietf-netconf-zerotouch]
Watsen, K., Hanna, S., Clarke, J., and M. Abrahamsson, "Zero Touch Provisioning for NETCONF Call Home (ZeroTouch)", draft-ietf-netconf-zerotouch-00 (work in progress), July 2014.

[I-D.kelsey-intarea-mesh-link-establishment]
Kelsey, R., "Mesh Link Establishment", draft-kelsey-intarea-mesh-link-establishment-05 (work in progress), February 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", draft-ohba-6tisch-security-01 (work in progress), March 2014.

[I-D.piro-6tisch-security-issues]
Piro, G., Boggia, G., and L. Grieco, "Layer-2 security aspects for the IEEE 802.15.4e MAC", draft-piro-6tisch-security-issues-02 (work in progress), June 2014.

Author's Address

Michael C. Richardson
Sandelman Software Works
470 Dawson Avenue
Ottawa, ON K1Z 5V7
CA

Email: mcr+ietf@sandelman.ca
URI: http://www.sandelman.ca/

6TiSCH R. Struik
Internet-Draft Struik Security Consultancy
Intended status: Informational Y. Ohba
Expires: April 30, 2015 Toshiba
S. Das
ACS
October 27, 2014

6TiSCH Security Architectural Elements, Desired Protocol Properties, and Framework
draft-struik-6tisch-security-architecture-elements-01

Abstract

This document describes 6TiSCH security architectural elements with high level requirements and the security framework that are relevant for the design of the 6TiSCH security solution.

Status of This Memo

Copyright Notice

Table of Contents

1. Security Architecture Elements

1.1. Device Types and Roles

There are two types of devices (or nodes) that are involved in the 6TiSCH security architecture: end devices that intend to join the LLN (commonly known as joining nodes) and network devices that help the joining node to be authenticated and authorized by the network. From a security operations perspective, each device has a distinct role in the network. An end device has normally a client role, while the network device can be a proxy or assume a server role. A proxy is an intermediate node that helps the end device to establish a communication with the server. An end device may move in and out of networks (that may be alien to it) and may have little network management functionality on board. However, it usually does have the right credential required for initializing the network joining process. A proxy is an intermediary node that that may be more tied into a relatively stable infrastructure and may have more support for network management functionality and generally has reliable access to back-end systems of the network. A server provides stable, highly available infrastructure and network management support and is capable of authenticating and authorizing a joining node.

It is important to note that a network node may assume multiple roles at the same time and that a particular role may be assumed by multiple network nodes. Furthermoe, the roles of a network node may change over time and can be dynamic in nature along a node or a network's lifecycle.

1.2. Device Enrollment Phases

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

Authorization: The network node decides on whether/how to authorize a joining node (if denied, this may result in loss of bandwidth). Authorization decisions may involve other nodes in the network.

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

1.3. Desired Protocol Properties

Security-Related:

1. Parties executing a security protocol should be explicitly aware of its security properties;

2. Compromise of keys or devices should have limited effect on security of other devices or services;

3. Attacks should not have a serious impact beyond the time interval/space during/in which these take place;

4. Security protocols should minimize the impact of network outages, denial of service attacks.

Communication Flows:

1. Security protocols should allow to be run locally, without third party involvement, wherever possibl;

2. The number of message exchanges for a joining device should be reduced;

3. Message exchanges should be structured so as to allow parallel execution of protocol steps, wherever possible.

Computational Cost:

1. Security protocols should not impose an undue computational burden, especially on joining devices (An exception here may arise, when recovering from an event seriously impacting availability of the network.)

Device Capabilities:

1. Dependency on an accurate time-keeping mechanism should be reduced;

2. Computational/time latency trade-offs should be tweaked to benefit those of joining node, wherever possible;

3. Dependency on "homogeneous trust models" should be reduced, without jeopardizing the security properties;

4. Dependency on on-board trusted platforms and trusted I/O interfaces should be reduced.

2. Security Framework

2.1. Single-Stage Authentication Framework

In the single-stage authentication and authorization framework, depicted in Figure 1, it is assumed that devices have access to certificates and that entities have access to the root CA certificate of their communicating parties (initial set-up requirement). Under these assumptions, the authentication step of the device enrollment process does not require online involvement of a third party. Authentication is performed between the joining node and the proxy using their certificates. Upon successful authentication, link-layer keys are established between the client and the proxy. The proxy will deny bandwidth if authorization is not successful. After successful authentication and authorization, configuration information is exchanged.

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

```
{joining node}      {neighbor}                 {server, etc.}
+---------+         +---------+                 +---------+
|  Node   |         |  Proxy  |              +--|    CA   |e.g., certificate
|    A    |         |    B    |              |  +---------+       issuance
+---------+         +---------+              |  +---------+
    |                    |                   +--|Authoriz.|e.g., membership
    |<----Beaconing------|                   |  +---------+         test
    |                    |                   |  +---------+
    |<--Authentication-->|                   +--| Routing |e.g., IP address
    |                    |<--Authorization-->|  +---------+       assignment
    |<-------------------|                   |  +---------+
    |                    |                   +--| Gateway |e.g., backbone,
    |------------------->|                   |  +---------+      cloud
    |                    |<--Configuration-->|  +---------+
    |<-------------------|                   +--|Bandwidth|e.g., PCE
                                                +---------+      schedule
    .                    .                   .
    .                    .                   .
```

Figure 1: Single-stage authentication/authorization

2.2. Two-Stage Authentication Framework

In the two-stage authentication and authorization framework, depicted in Figure 2, a joining node performs two authentication and authorization steps. The first step, called Phase-1 authentication, is performed between the joining node and the server via a proxy. Phase-1 authentication and authorization uses deployment-specific enrollment credentials and results in issuance of a certificate by the CA to the joining node. Here, the node’s certificate and root CA certificates of its communicating parties are distributed from the server to the client.

The second step, called Phase-2 authentication, follows the successful completion of Phase-1 authentication and authorization. Phase-2 authentication is performed between the joining node and the proxy using their certificates. Upon successful authentication, link-layer keys are established between the joining node and the proxy. The proxy will deny bandwidth if Phase-2 authorization is not successful. After successful authentication and authorization, configuration information is exchanged.

Once a joining node obtains a certificate for Phase-2 authentication, no additional Phase-1 authentication and authorization is needed, i.e., only Phase-2 authentication and the configuration are required for rejoining the network via a proxy under the same authorization domain. This reduces to the single-stage authentication framework discussed in the previous section.

```
{joining node}       {neighbor}             {server, etc.}
+---------+          +---------+            +---------+
|  Node   |          |  Proxy  |         +--|    CA   |e.g., certificate
|    A    |          |    B    |         |  +---------+      issuance
+---------+          +---------+         |  +---------+
    |                      |             +--|Authoriz.|e.g., membership
    |<----Beaconing--------|             |  +---------+        test
    |                      |             |  +---------+
    |<-Ph1 Authentication & Authorization->+--| Routing |e.g., IP address
    |                      |             |  +---------+      assignment
    |<-Ph2 Authentication->|             |  +---------+
    |                      |             +--| Gateway |e.g., backbone,
    |--------------------->|             |  +---------+     cloud
    |                      |<-- Config. -->|  +---------+
    |<---------------------|             +--|Bandwidth|e.g., PCE schedule
    .                      .             .  +---------+
    .                      .             .
```

Figure 2: Two-stage authentication/authorization

3. Security Considerations

In this section, security issues that can potentially impact the operation of IEEE 802.15.4e TSCH MAC are described.

In TSCH MAC, time synchronization and channel hopping information are advertised in Enhanced Beacon (EB) frames [I-D.ietf-6tisch-terminology]. The advertised information is used by mesh nodes to determine the timeslots available for transmission and reception of MAC frames. A rogue node can inject forged EB frames and can cause replay and DoS attacks to TSCH MAC operation. To mitigate such attacks, all EB frames MUST be integrity protected. While it is possible to use a pre-installed static key for protecting EB frames to every node, the static key becomes vulnerable when the associated MAC frame counter continues to be used after the frame counter wraps. Therefore, the 6TiSCH solution MUST provide a mechanism by which mesh nodes can use the available time slots to run authentication protocols and provide integrity protection to EB frames.

For use cases where certificates are used for authentication, pre-provisioning of absolute time to devices from a trustable time source using an out-of-band (OOB) mechanism is a general requirement.

Accuracy of time depends on the OOB mechanism, including use of the time hard-coded into the installed firmware. The less time accuracy is, the more attack opportunities during Phase-1. In addition, use of CRL is another requirement for authentication employing certificates to avoid an attack that can happen by a compromised server or CA certificate.

4. IANA Considerations

There is no IANA action required for this document.

5. Acknowledgments

TBD.

6. Normative References

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

[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-02 (work in progress), July 2014.

Authors' Addresses

Rene Struik
Struik Security Consultancy

Email: rstruik.ext@gmail.com

Yoshihiro Ohba
Toshiba Corporate Research and Development Center
1 Komukai-Toshiba-cho
Saiwai-ku, Kawasaki, Kanagawa 212-8582
Japan

Phone: +81 44 549 2127
Email: yoshihiro.ohba@toshiba.co.jp

Subir Das
Applied Communication Sciences
1 Telcordia Drive
Piscataway, NJ 08854
USA

Email: sdas@appcomsci.com