Internet of Things (IoT)

6TiSCH Technology

6TiSCH Survey Papers:
https://ieeexplore.ieee.org/document/8823863
https://doi.org/10.1016/j.comnet.2024.110759

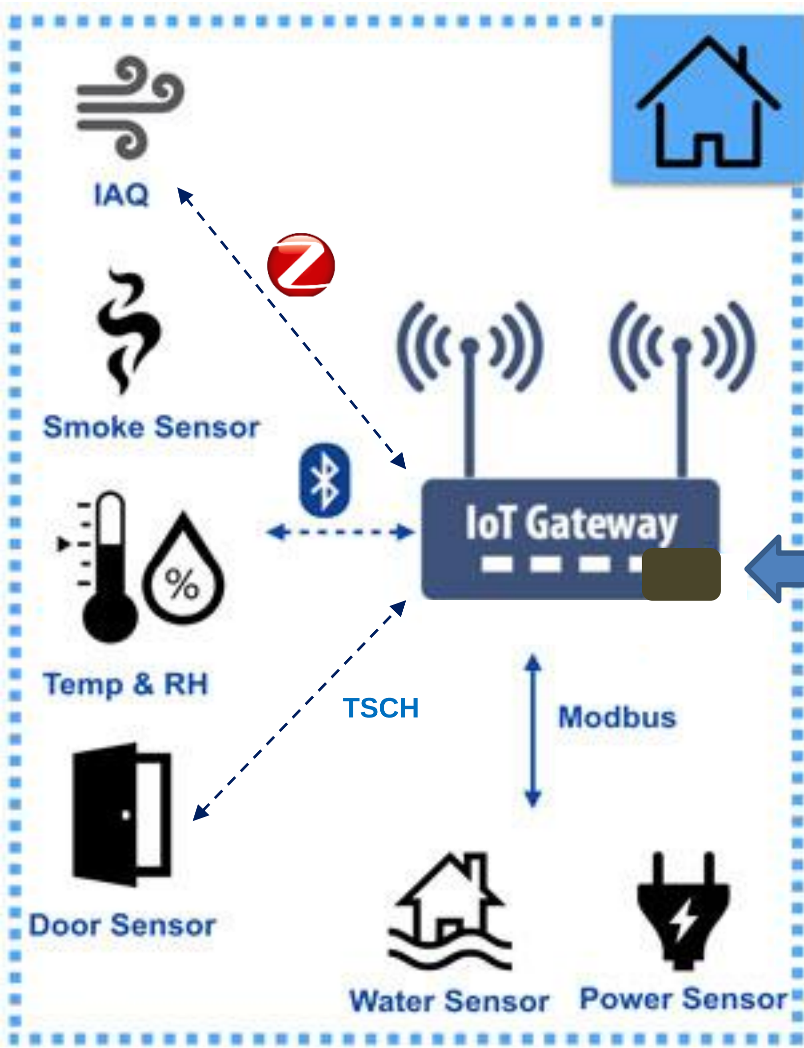
Dr. Manas Khatua
Associate Professor
Dept. of CSE, IIT Guwahati
E-mail: manaskhatua@iitg.ac.in

“All Birds find shelter during a rain. But Eagle avoids rain by flying above the Clouds” – **APJ Abdul Kalam**

IoT Access Technologies

- there are many IoT technologies in the market today

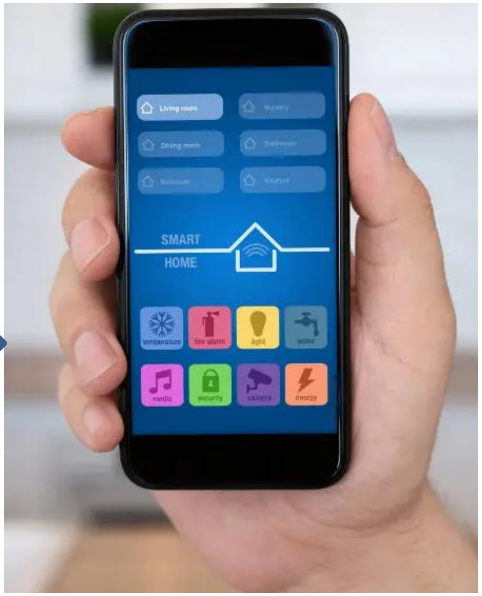
Application Viewpoint



Remote Real-Time Building Monitoring

How?

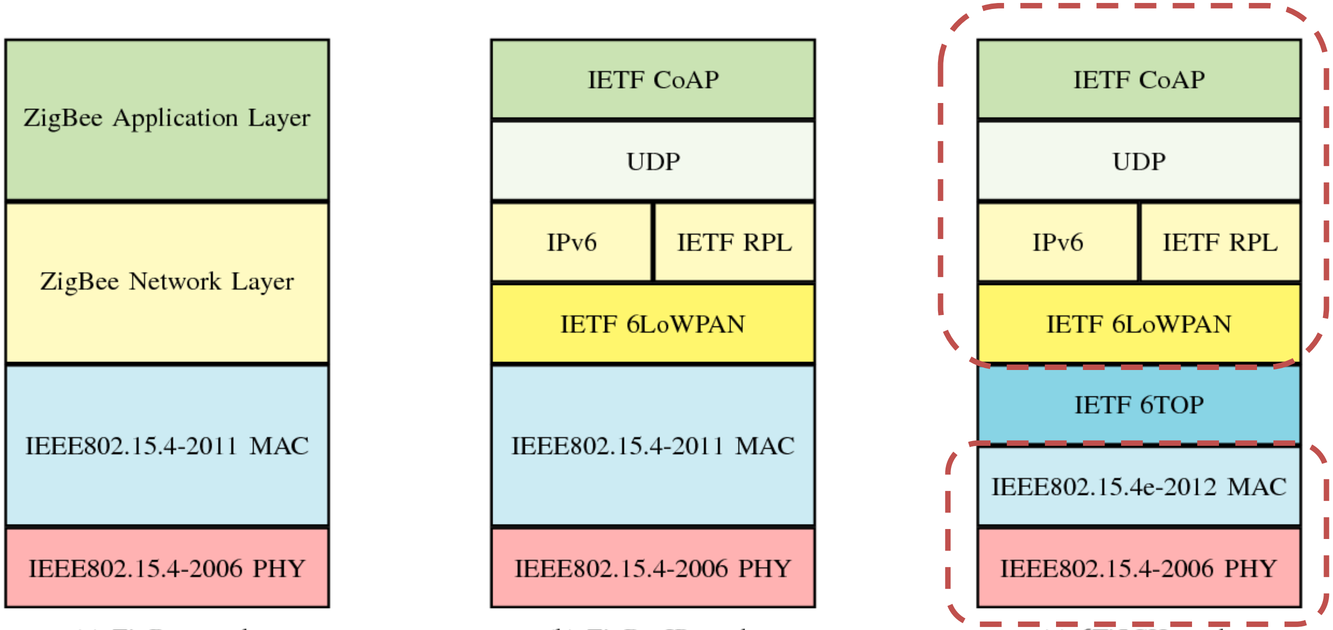
using Existing Internetworking Infrastructure (e.g. IP)



6TiSCH Working Group

- 6TiSCH Working Group created by IETF in October 2013

- **Goal**: To integrate **TSCH** with the **IPv6** through the IETF upper stack
  - To enable IPv6 over TSCH mode of IEEE 802.15.4e
  - Defining a new functional entity in charge of TSCH scheduling



(a) ZigBee stack. (b) ZigBeeIP stack. (c) 6TiSCH stack.

Survey Article: "IETF 6TiSCH: A Tutorial" https://ieeexplore.ieee.org/document/8823863

6TiSCH Architecture (RFC 9030)

- Considers low-power lossy-network (LLN)
- Allow more than 1000 nodes
- Nodes are in same IPv6 subnet
- 6LoWPAN header compression (HC) is used to transmit packet
- Presence of high-speed backbone (e.g. WiFi mesh) to connect all nodes
- Backbone is connected to the Internet through a Gateway
- Constrained nodes are attached to backbone through backbone router (BBR) or 6LBR

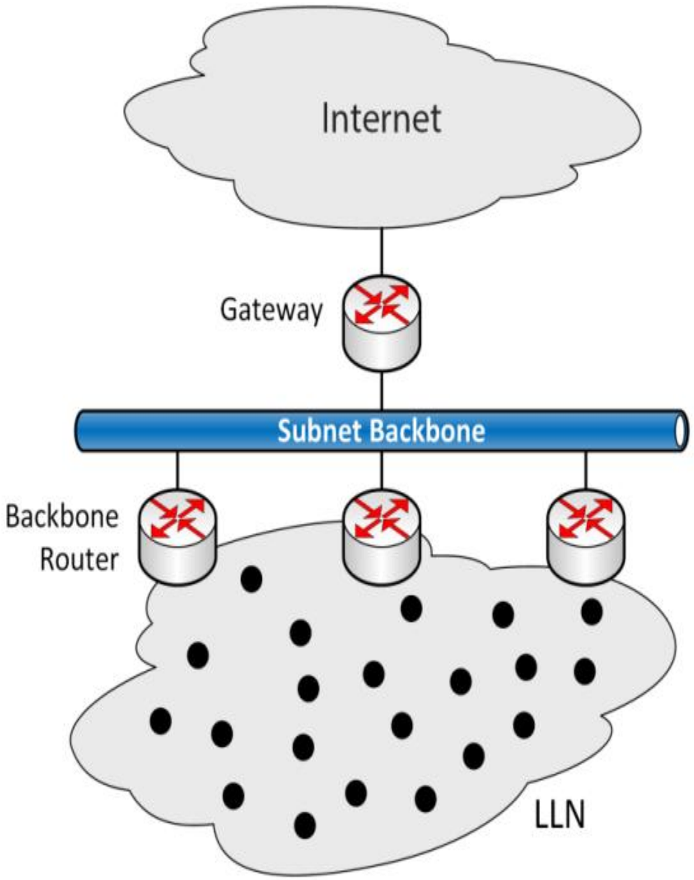


Fig. 6TiSCH Architecture

RFC 9030: An Architecture for IPv6 over the Time-Slotted Channel Hopping Mode of IEEE 802.15.4 (6TiSCH)

Need for 6TiSCH Operation Sub-Layer

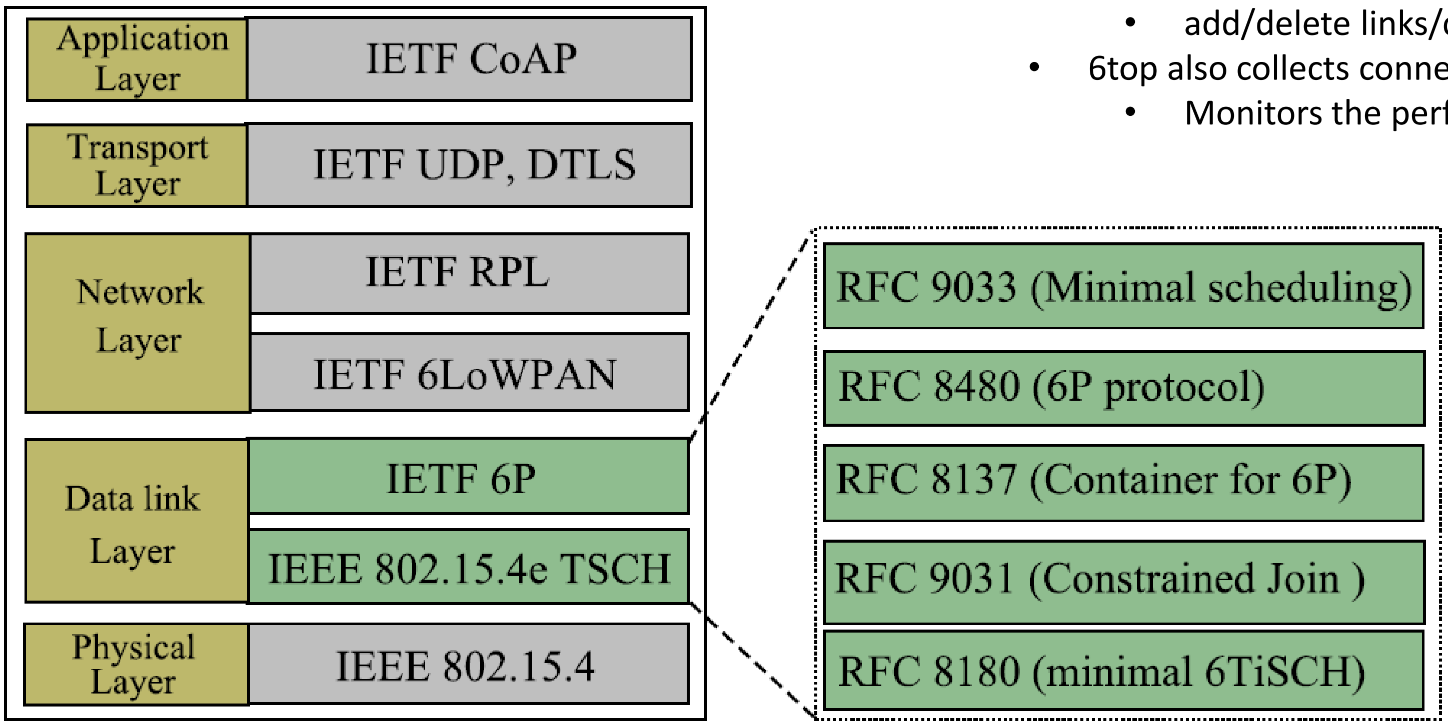
- In 6TiSCH, the TSCH MAC mode is placed under an IPv6-enabled protocol stack:
  - Constrained Application Protocol (CoAP)
  - IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL)
  - IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN)

- TSCH **does not** define
  - Policies to build and maintain the data communication schedule
  - Mechanisms to adapt the resources allocated between neighbor nodes as per the data traffic flow features – change in data rates, neighbor change, etc.
  - Techniques to allow differentiated treatment of packets – data and control packets
  - Mechanisms to match the schedule to the multi-hop paths maintained by RPL

6Top Sub-Layer

A new sublayer, called 6Top

- Works on top of TSCH
- Build and manage TSCH schedule
  - add/delete links/cells
- 6top also collects connectivity information
  - Monitors the performance of cells



6TiSCH Protocol Stack

6TiSCH Operational sublayer

Source: Kalita and Khatua, “6TiSCH – IPv6 Enabled Open Stack IoT Network Formation: A Review” *ACM Transactions on Internet of Things,3(3)*, 2022, pp. 24:1–24:36.

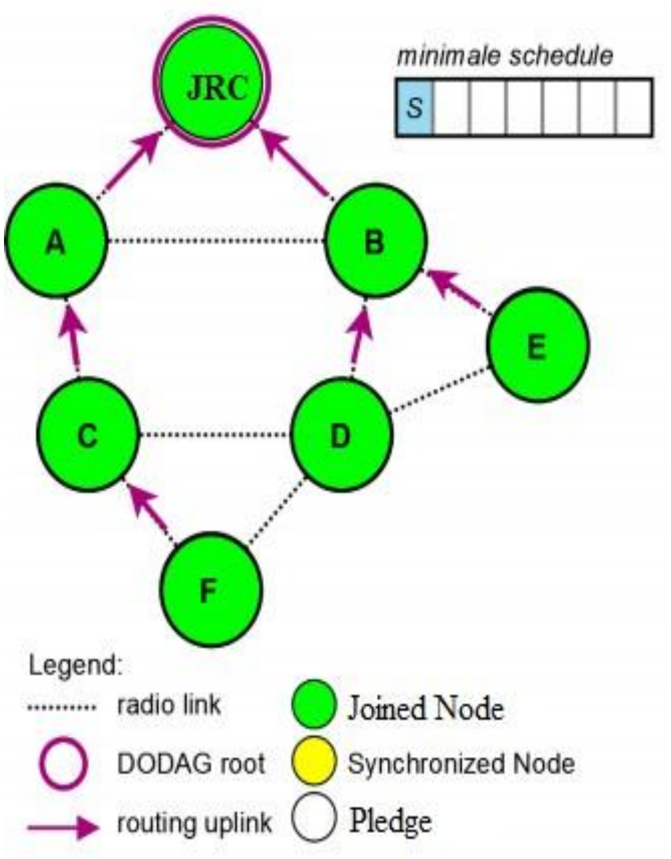
6TiSCH Protocol Stack

Application (CoAP)	**RFC8613**	(2019)	*object security extension to CoAP*
	RFC7252	(2014)	*base CoAP specification*
Routing (RPL)	**RFC6554**	(2012)	*header format for routing header*
	RFC6553	(2012)	*header format for RPL option*
	RFC6552	(2012)	*Objective Function, RPL algorithm*
	RFC6550	(2012)	*base RPL specification*
Adaptation (6LoWPAN)	**RFC8505**	(2018)	neighbor discovery and registration
	RFC8138	(2017)	*routing header* compression
	RFC8025	(2016)	*mechanism for extending 6LoWPAN*
	RFC6282	(2011)	*updated base 6LoWPAN* specification
	RFC4944	(2007)	*base 6LoWPAN specification*
Scheduling (6TiSCH)	**draft-ietf-6tisch-msf** **(RFC 9033)**	(WIP)	*distributed scheduling algorithm*
	RFC8480	(2018)	*6P, distributed scheduling protocol*
	RFC8137	(2017)	*container for 6P*
	draft-ietf-6tisch-minimal-security **(RFC 9031)**	(WIP)	*security framework for 6TiSCH*
	RFC8180	(2017)	*minimal 6TiSCH*
Physical layer	**IEEE802.15.4**	(2015)	*2.4 GHz, 50-200 m range, 250 kbps, 127 byte frames*

Source: Xavier Vilajosana et al., "IETF 6TiSCH: A Tutorial" *IEEE Communications Surveys & Tutorials*, 22(1), 2020, pp. 595– 615.

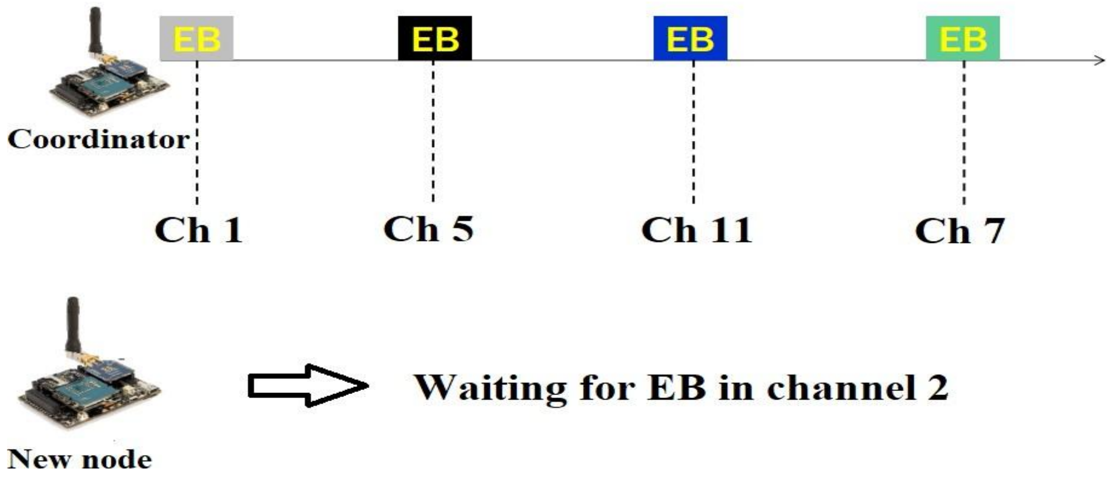
6TiSCH Network Formation Process

- Join Registrar/Coordinator (JRC) starts the formation process
  - Enhanced Beacon (EB)
  - Routing Information (DIO)
- Pledge (new node) scans for EB on a random channel (RDC 100%)
- After receiving an EB, pledge synchronized with the underlying TSCH network (RDC ~1%)
- Synchronized node waits for DIO, after exchanging JRQ & JRS
- After receiving DIO, pledge becomes joined node; can transmit own packets
- Network formation completes when all the pledge join the network



Why Network formation is an issue?

- Channel hopping feature of TSCH
  - A pledge does not know in which channel transmission of control packets is happening



- Limited resource allocated for control packets
  - Only one shared cell in a slotframe

TSCH v/s 6TiSCH network formation

- TSCH formation/synchronization time
  - A pledge gets synchronized with a TSCH network after receiving a valid EB frame
  - The time when a pledge receives its first EB frame is considered as TSCH synchronization time or TSCH formation time

- 6TiSCH formation time
  - When a TSCH synchronized node receives a valid DIO packet, it becomes a 6TiSCH joined node
  - The DIO receiving time is considered as 6TiSCH joining time

Goals during Network formation

- Reduce pledge joining time
  - To immediately transmit data

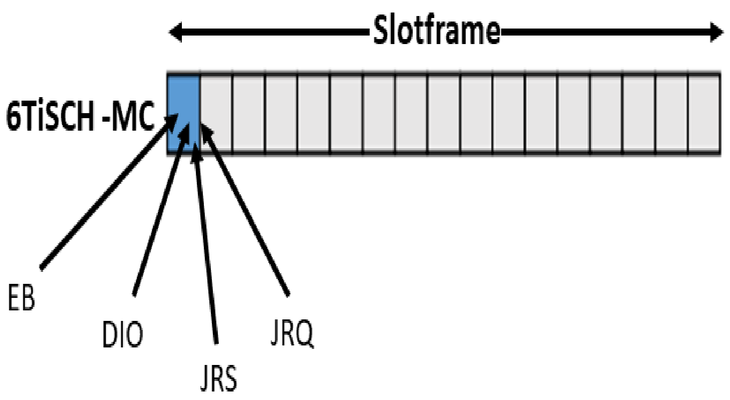
- Save energy consumption
  - Radio duty cycle is 100% before TSCH synchronization
  - Maximum energy consumption

6TiSCH minimal configuration (RFC 8180)

- In 2017, 6TiSCH Working Group released the 6TiSCH minimal configuration standard in order to provide details about the minimal resource usage during network bootstrapping
  - Only one shared cell per slotframe can be used for transmission of control packets by all the nodes
  - Both EB and DIO packets are required to complete joining process
  - EB has the highest priority over other control frames like DIO, DIS, etc.
  - Control packets (JRQ and JRS) for secure enrolment of a node are also exchanged in shared cell

<u>Shortcomings</u>

- Static Allocation
- Joining time is more



MSF: Minimal Scheduling Function

- **MSF**: 6TiSCH Minimal Scheduling Function
  - RFC 9033, Year: 2021

- **Objective of MSF:**
  - To manage the communication schedule in 6TiSCH network in a distributed manner.
  - To describe the behavior of a node when it joins the network

- 6TiSCH carry dynamic scheduling on top of minimal profile
  - Scheduling functions
    - ✓ Decision-making entity
    - ✓ Add/delete/relocate
  - 6top protocol (6P)
    - ✓ Managing entity
    - ✓ responsible for pairwise negotiation
    - ✓ 2-step or 3-step transaction

- Joined node relies on MSF & 6top

- 3 slotframes used
  - Slotframe 0 (Minimal cell)
  - Slotframe 1 (Autonomous cells)
  - Slotframe 2 (6P Negotiated cells)

❑ https://datatracker.ietf.org/doc/rfc9033/

MSF (Cont…)

- A node implementing MSF should implement 6TiSCH minimal configuration.

➢ **Minimal cell** is for broadcast frames(EB,DIO)

✓ A single shared cell- provides minimal connectivity

➢ **Negotiated cells**

✓ managed by 6P to meet traffic requirements

➢ **Autonomous cells**

- Maintained autonomously by node without 6P negotiation

- AutoTxCell (cell options Tx=1,Rx=0,shared=1)(added/deleted on demand)
  - ✓ When there is a frame to send and there is no negotiated Tx cell and uninstall after sending out the frame

- AutoRxCell (cell options Tx=0,Rx=1,shared=0)(permanent)
  - ✓ Always remain scheduled after synchronization

- SlotOffset = 1+ hash (EUI64, Slotframe1 –1)

- ChannelOffset = hash (EUI64, NumberOfChannels)

- For **Tx cell**, EUI64 of **destination node**

- For **Rx cell**, hash of EUI64 **of node itself**

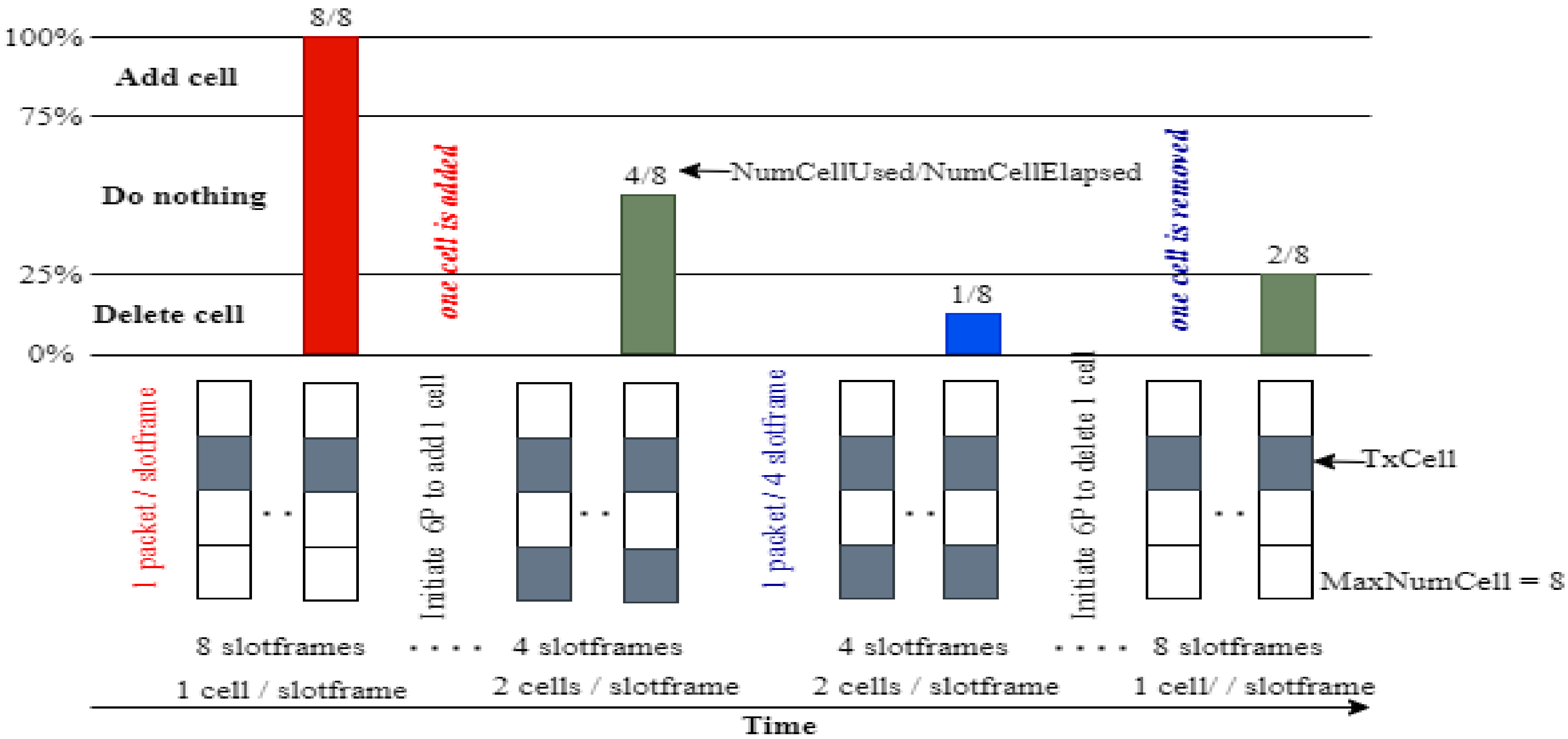
MSF (Cont…)

- **Rules for adding/deleting cells (Negotiated cells)**
    - **Adapting to traffic**
    - For a node, monitors current usage of the cells it has with one of its neighbors

- **Initially 1 negotiated cell**
- Maintains two separate pairs of NumCellsElapsed and NumCellsUsed For a node, monitors current usage of the cells it has with one of its neighbors
    - ✓ CellUsage= **NumCellsUsed / NumCellsElapsed**

- Both initialized to zero when node boots
- when **NumCellsElapsed** reaches **MaxNumCell**
    - ✓ If **CellUsage > LIM_NUMCELLSUSED_HIGH**
        - Triggers 6P to add a single cell
    - ✓ If **CellUsage < LIM_NUMCELLSUSED_LOW**
        - Triggers 6P to remove a single cell
- ✓ Reset to zero

MSF (Cont…)



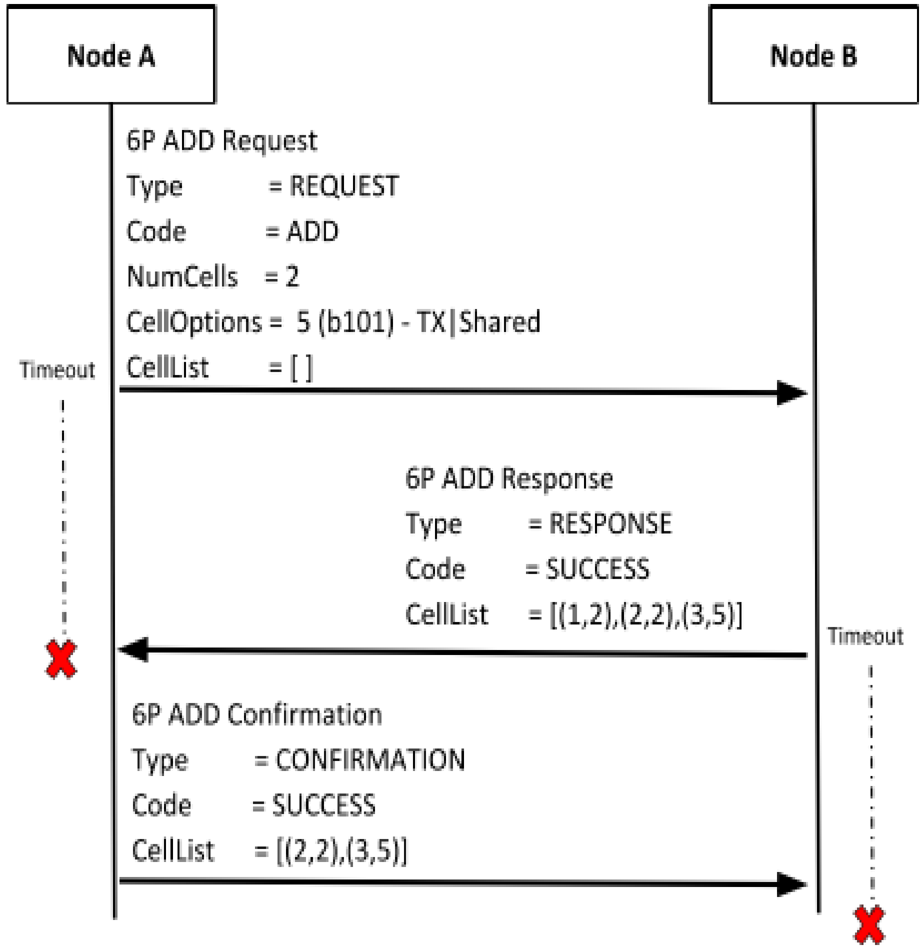
Here, MAX_NUMCELL is 8. It adds a cell when cell usage is more than 75% and deletes a cell if cellusage is less than 25%.

MSF (Cont…)

```
                 +----------+                                  +----------+
                 |  Node A  |                                  |  Node B  |
                 +----+-----+                                  +-----+----+
                      |                                              |
                      |  6P ADD Request                              |
                      |    Type        = REQUEST                     |
                      |    Code        = ADD                         |
                      |    SeqNum      = 123                         |
    cells             |    NumCells    = 2                           |
    locked            |    CellList    = [(1,2),(2,2),(3,5)]         |
     +---             |--------------------------------------->|
     |                |                                   L2 ACK     |
     |  6P Timeout    |<- - - - - - - - - - - - - - - - - - - -     |
     |         |      |                                              |
     |         |      |  6P Response                                 |
     |         |      |    Type        = RESPONSE                    |
     |         |      |    Code        = RC_SUCCESS                  |
     |         |      |    SeqNum      = 123                         |   cells
     |         |      |    CellList    = [(2,2),(3,5)]               |   locked
     +->       X      |<---------------------------------------|   --+
                      |  L2 ACK                                      |     |
                      |  - - - - - - - - - - - - - - - - - - - ->|   <-+
                      |                                              |
```

An example of 2 step 6P transaction

MSF (Cont…)



An example of 3 step 6P transaction

- **Rules for Cell list**
  - ✓ To have at least NumCells in CellList
  - ✓ Each cell must have different slot offset value
  - ✓ Must not have any scheduled cell on the same slot offset
  - ✓ Can't be with slotoffset 0
  - ✓ Should be randomly chosen among all slotoffset values
  - ✓ Channel offset is chosen randomly from [0...NoOfFrequencies]

- **IETF 6TiSCH:A Tutorial**
- **https://tools.ietf.org/html/draft-ietf-6tisch-6top-protocol**

MSF (Cont…)

- **Handling Schedule Collisions**
  - if a node has several cells to the selected parent, all should exhibit the same PDR.
  - A cell having PDR significantly lower than the others - collisions on that cell.
  - **PDR = NumTXAck / NumTx**

- **Rules for relocation of cell (Negotiated cells)**

  Every Housekeeping period the node execute

  - For each negotiated Tx cell with that parent compute its PDR
  - Identifies cell with highest PDR
  - For other cell, find the difference with highest PDR.
  - If difference in PDR> Relocate_PDRTHRESH then it triggers relocation command

MSF (Cont…)

- **Switching parent**
  - ✓ Counts the number of negotiated Tx cell it has with the old parent per slot frame.
  - ✓ Triggers one or more 6p ADD request with same cell options to the new parent.
  - ✓ Then issues 6p CLEAR command to its old parent.

Thanks!

Figures and slide materials are taken from the following sources:

1. David Hanes *et al.*, "IoT Fundamentals: Networking Technologies, Protocols, and Use Cases for the Internet of Things", 1st Edition, 2018, Pearson India.