6TiSCH Minimal Scheduling Function (MSF)
draft-chang-6tisch-msf-00

Tengfei Chang
Malisa Vucinic
Xavi Vilajosana
Oops...

• We missed the deadline (timezone confusion...), thanks to Suresh for publishing the document at:
  • https://tools.ietf.org/html/draft-chang-6tisch-msf-00
Abstract

This specification defines the 6TiSCH Minimal Scheduling Function (MSF). This Scheduling Function describes both the behavior of a node when joining the network, and how the communication schedule is managed in a distributed fashion. MSF builds upon the 6top Protocol (6P) and the Minimal Security Framework for 6TiSCH.
In a nutshell

1. Start with a single cell
   • 6tisch-minimal

2. Perform secure join
   • 6tisch-minimal-security

3. Add/delete cells to parent
   • 6tisch-6top-protocol

Completely defined behavior, fully standardized story
Interaction with 6TiSCH-minimal

- Frames exchanged over the minimal cell:
  1. EBs
  2. DIOs
  3. Join request/response messages between pledge and JP
  4. the first 6P Transaction a node initiates

- Access rules to the minimal cell: cut bandwidth in portions:
  - $1/(3(N+1))$ for EBs (N= number of neighbors)
  - $1/(3(N+1))$ for DIOs
  - Rest for join and 6P (see above)

- Slotframe organization:
  - Slotframe 0 for minimal cell
  - Slotframe 1 for cells added by MSF
Node Behavior at Boot (1/2)

- Start state
  - PSK
  - Any other configuration mentioned in minimal-security
- [7-step join]

- End state
  - node is synchronized to the network
  - node is using the link-layer keying material it learned through the secure joining process
  - node has identified its preferred routing parent
  - node has a single dedicated cell to its preferred routing parent
  - node is periodically sending DIOs, potentially serving as a router for other nodes' traffic
  - node is periodically sending EBs, potentially serving as a JP for new joining nodes
Node Behavior at Boot (2/2)

• Step 1 - Choosing Frequency
  • Listen on random frequency

• Step 2 – Receiving Ebs
  • Listen for multiple neighbors, shoes one as JP

• Step 3 - Join Request/Response
  • First hop over minimal cells, rest over dedicated (same for response)

• Step 4 - Acquiring a RPL rank
  • Select preferred parent

• Step 5 - 6P ADD to Preferred Parent
  • Single TX|RX|SHARED cell to parent

• Step 6 - Send EBs and DIOs
  • Accept children

• Step 7 - Neighbor Polling
  • Keep-alive to each neighbor you have cells to every 10s; remove if dead.
Dynamic Scheduling (1/4)

• 3 reasons for adding/removing/relocating cells:
  • Adapting to Traffic
  • Switching Parent
  • Handling Schedule Collisions

• 6P carries out the work
Dynamic Scheduling (2/4)

• Reason 1/3: Adapting to Traffic
  • A node always has at least one cell to preferred parent
  • Keep counters to preferred parent:
    • NumCellsPassed
    • NumCellsUsed
  • When NumCellsPassed reaches 16:
    • If NumCellsUsed>12, add a cell
    • If NumCellsUsed>4, remove a cell
Dynamic Scheduling (3/4)

• Reason 2/3: Switching parents
  • Count number of cells to old parent
  • Schedule the same number to new parent
  • Remove cells from old parent
Dynamic Scheduling (4/4)

• Reason 3/3: Handling schedule collisions
  • Counter for each cell to preferred parent:
    • NumTx
    • NumTxAck
  • When NumTx==256:
    • NumTx<<1
    • NumTxAck<<1
  • Periodically, compare numbers for all cells to parent
    • If no roll over yet, abort
    • If PDR of one cell <50% of cell with max PDR, relocate
Other “details”

- 6P SIGNAL command
- Rules for CellList
- 6P Timeout Value
- Rule for Ordering Cells
- Meaning of the Metadata Field
- 6P Error Handling
- Schedule Inconsistency Handling
6TiSCH MSF challenge!

• Answer the following questions through implementation:
  • how long does it take a 100-node network to form? what topology?
  • what is the average latency of a packet in a typical 32-node indoor deployment?
  • how does the network handle bursty traffic? up to how much burstiness is acceptable (i.e. no packets lost)?
  • What is the code/memory footprint of an MSF implementation?
  • How resilient a network is to loss of nodes in the middle of its construction? While it's running?
  • What kind of network topology is difficult to handle to such scheduling function? (Linear, Tree, Random, ...).
  • What is the cost of adding a new node to the network once it's running? Does this cost increase with the size of the network?
  • what is the data rate that makes cell allocation too slow? considering the 2-way nature of 6p.
  • how often we get CLEARs due to inconsistency given certain network conditions (different levels of PDR)
  • what is the impact of cell-list size in the allocation probability (5 candidate cells are recommended, what about more or less?)
  • how do queues grow when a node is using its almost maximum capacity and starts requesting more cells.
  • what is the impact of message timeouts in the size of queues?
  • Etc. (ideas welcome!)

• 6TiSCH simulator and OpenWSN implementations ongoing!