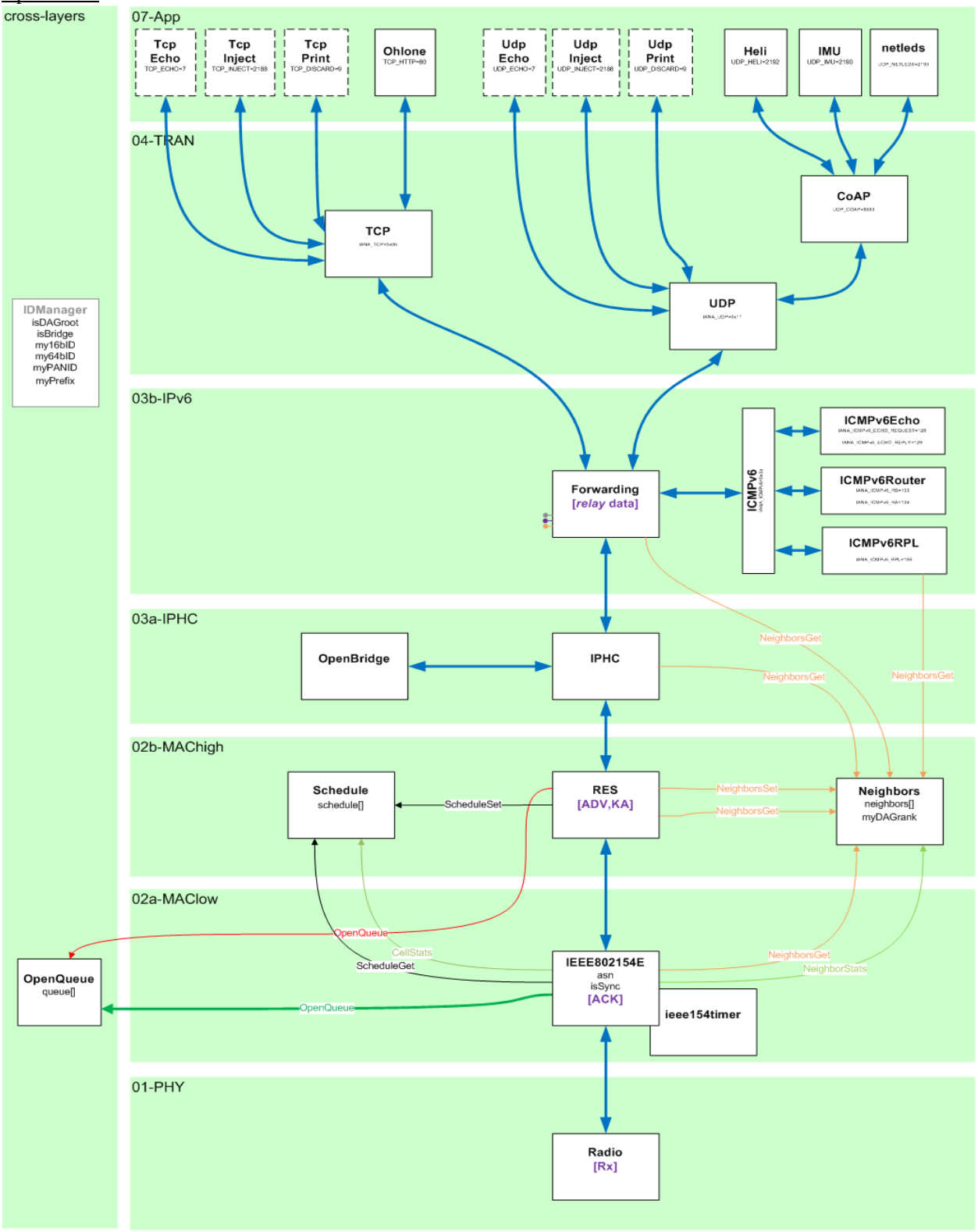
Software Design:

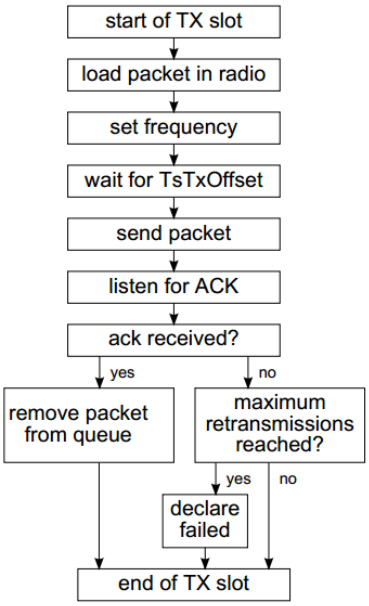
We are mainly concerned with introducing tree structured channel hopping using existing 802.15.4.e TSCH, which already implemented in OpenStack.

OpenStack:

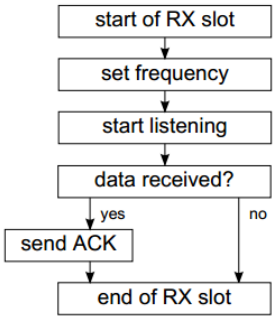


In 02a-MAClow layer, we are looking into IEEE802154E submodule.
It consist of 2 files: /openwsn-fw/openstack/02a-MAClow/IEEE802154E.c and IEEE802154E.h

State diagram of transmit (TX) and receiving (RX) slot is described below:
For TX:



For RX:



Already existing collection is done using COAP.
A mote is connected to a physical sensor, and an application runs on top of the OpenWSN stack to sample that sensor and initiate a transmission to the CoAP UDP port of a data server on the Internet. Sensor data is passed to the CoAP protocol, which adds a header indicating which "resource" this data comes from. This payload is prepended with UDP, 6LoWPAN and IEEE802.15.4e headers. The resulting frame is then scheduled for transmission using the IEEE802.15.4e TSCH MAC layer. When reaching the edge of the network, the packet is forwarded to the LBR, which inflates the 6LoWPAN into a full IPv6 header, and transmits into the IPv6 Internet. A data server, which listens to the well-known CoAP UDP port, receives the data and stores it in a database, where the data can be displayed or processed.

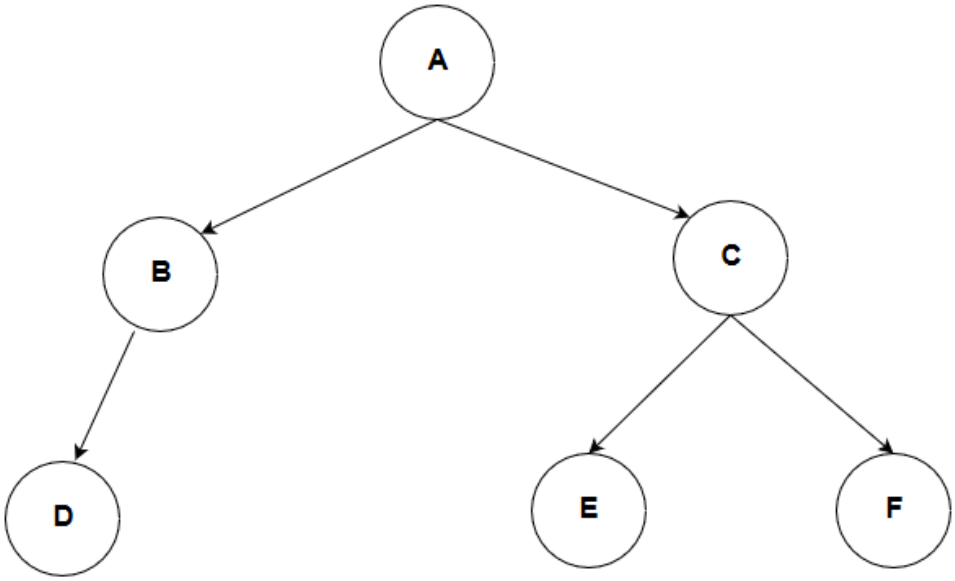
For building the tree structure we consider the already existing 'neighbors' submodule in 02b-MAChigh. This neighbor system is adaptive and uses DAG rank and neighbor preference.
Planned header for tree topology:

Control Part - time sync, - topology control, - slots handling	Schedulded state - yes/no - Completion state - yes/no	Optional: Dynamic Channel hopping using TSCH

Complete frame /header - to be added (Planned)

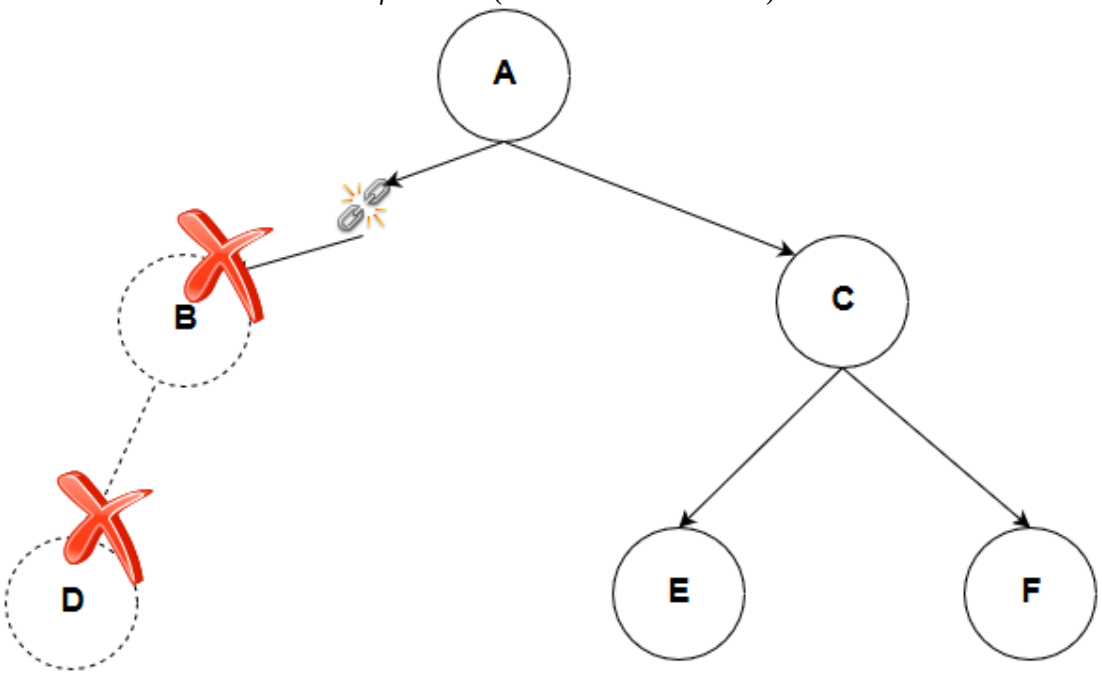
For the task given in the lab we have 2 options:
a. Use 02b-MAChigh submodule 'neighbors' and update its header.
b. Use 03b-IPv6 submodule 'ICMPv6RPL' for making the packet.
Final decision of which one (or some other submodules) to use would be taken after more detailed code analysis of existing OpenWSN code.

Adaptive tree topology:
We are considering adaptive topology for dynamic joining of nodes and unplanned sudden leaving of nodes.
For example, *initial topology*:



Normal tree topology (no connection breaking)

And after a *sudden disconnection of one link*: (i.e. between node A & B):



Normal tree topology (with connection breaks)

From previous submission:

Goals & Milestones:

1. *Creation and maintenance of tree structure for collection protocol. It should be adaptive to topology changes.*
   - 1.1. *Go through existing implementation in OpenWSN and check how the current version of collection protocol is implemented.*
   - 1.2. *Tree structured collection protocol might be maintained in the following way:*
     - 1.2.1. *All nodes send periodic message to the tree (apart from the root itself)*
     - 1.2.2. *Root can be collection point of the tree.*
2. *Scheduling of data transmissions (sending)*
   - 2.1. *Ignore channel hopping at the beginning. It is pseudo random.*
3. *Assignment of more free slots to the nodes for transmission.*
   - 3.1. *Assignment of slots can be contention based or reserve based.*

[Note: Goal #3 is a bonus goal and it could be done only if here are any time left after completion of first 2 goals.]