Gradient routing

Victor Cionca

Use the wsnsimpy network simulator to implement a network application where nodes periodically send data packets to the network sink. Communication is multi-hop and the gradient routing protocol is implemented.

Details

- the network sink (ID 0) will not be sending data packets
- when the network sink receives a data packet it should print the address of the node that sent it
- gradient routing should use the hop-count metric
- nodes should keep track of
  - best distance to the sink
  - rank
  - preferred parent
  - list of potential parents - these are neighbours whose rank is lower or equal to that of the node
    - for each potential parent you will need to monitor its rank and its best distance, therefore you should use a dictionary to store the potential parents.
- nodes should implement the Trickle timer to send gradient updates to their neighbours, with minimum period 100ms, maximum period 2s
  - gradient packets are sent whenever the Trickle timer expires
  - each time the Trickle expires the Trickle period is doubled
  - if the node's internal state changes the Trickle period should be reset
  - updates should **NOT** be sent if the node's rank is infinity
  - **OBS** use a large number to represent infinity
- parent monitoring
  - nodes should monitor the *liveness* of their potential parents
  - if a gradient update is not received from a potential parent within two Trickle maximum periods, it should be assumed that the parent is no longer available, and the parent should be removed from the list
  - if the preferred parent is removed, the node from the list of potential parents that provides the shortest distance should be selected instead
  - if there are no available parents, the node should set its rank to infinity (i.e. a large number).
- implement the routing protocol as a network layer
  - nodes should instantiate the routing protocol and have a reference to it
  - the routing protocol should implement a `send_pdu` method that takes as parameter a PDU
  - when a node sends a packet it does not specify destination address; instead the PDU is passed to the routing layer's `send_pdu`, and not to the phy layer as you've done so far
  - the routing protocol processes two types of packets: gradient and data; you must use a `type` field in the PDUs send by the routing layer to distinguish between the two
  - the routing layer will identify the destination (the preferred parent) and use the physical layer to send the packet onward towards the sink
  - the routing layer should implement the `on_receive_pdu` method where it receives pdus from the physical layer
  - when receiving PDUs, the routing layer has the following options
    - if it is the sink node, it is the destination of the packet; the PDU must be provided to the

node's application layer through the node's `on_receive_pdu` method
        * if it is not the sink, it has to continue forwarding the packet towards the sink by sending it to the preferred parent.
    - the routing protocol should not allow sending packets when there is no parent towards the sink (i.e., current preferred parent is `None`)
        * the `send_pdu` method in the routing layer should return `False` in these cases, otherwise return `True`
- the routing protocol is periodically sending packets from the moment the node starts up; this means that
    - the routing protocol should implement a method (e.g. `run`) that performs the periodical activity
    - in the `run` method of the node you should call the above method of the routing protocol so that the routing protocol can commence the periodical activity.

Implementation hints

- the routing layer should be initialised with a reference to the node
    - this reference will be used when the routing layer must notify the node that a PDU was received (and the node is the sink)
- the routing layer should instantiate in its constructor a physical layer
- the node does not use the physical layer to send PDUs, instead it uses the routing layer
- after instantiating the routing layer, the node should also set it as the MAC:

```
self.routing = GradientRouting(self)
self.mac = self.routing
self.phy = self.routing.phy
```

- if the routing layer has to use simulator functions (`delayed_exec`) or logging, it should make use of the node object (e.g. `self.node.delayed_exec`)

Parent monitoring

- monitoring of potential parents should be performed whenever a gradient update is received
- store for each potential parent (in the dictionary):
    - the distance to the root through this parent
    - the parent's rank
    - the time when the last update was received from this parent; use `node.sim.now`
    - these three items can be grouped in a dictionary - one dictionary per parent
- define a function that takes a parent id as parameter and checks if the last update time of the parent occurred more than `2*trickle max period` before
    - this function should be scheduled (`delayed_exec`) when an update is received from the parent.

Marking scheme

Total: 25 points

- basic gradient routing, nodes only store distance and current parent, gradient packets are propagated in the network, and the gradient is built providing for each node the best parent to reach the sink **5pts**
- Trickle timer used to control the transmission of gradient packets **5pts**
- integration of the gradient routing as a network layer **5pts**
- extended gradient routing **10pts**
    - support for list of potential parents and their parameters
    - monitoring of the potential parents
    - updating of potential parent list following loop-prevention rules.