18-748. Lab 3 - Build Your Own Sensor Network.

Kedar Amlad // kamladi. Daniel Santoro // ddsantor. Adam Selevan // aselevan.

The network consists of two types of nodes: the client, and the gateway. This write-up will explain the architecture and design of each node type.

Every node follows the following packet format:

“**[message_type][node_id][sequence_number][hop_number][payload]**”

Client Design.

The client node’s task is as follows:

1. Receives a new packet and parses it.
2. Forwards packet if it is not a copy.
3. Adds the message sender id to its neighbor table if the sender is one hop away.
4. Checks if the sensor sampling timer or neighbor update timers has expired.

Receive / parse / forward packet task:

There are three possible “**message_types**” in the network: light sensor value, neighbor table list, or a gateway message. Message types allow the nodes to parse and handle a received packet appropriately. The “**node_id**” is the originator’s network mac address, which is assigned directly through firmware. The “**sequence_number**” is the value of the total messages sent by the node_id. Each packet contains a sequence number in order to prevent a node from “handling or relaying” a message that has already been received. Each node contains a table correlating a node_id to the last received sequence number. If a received packet’s sequence number is equivalent or lower than the number stored in the receiver's sequence table, the receiver must have already received a “newer” version of the packet, so the received packet can be dropped. The “**hop_number”** is a counter representing the amount of “hops” that particular packet has experienced. Each node increments this value before forwarding the packet. The hop number controls each packet’s Time-To-Live (TTL). Due to the fact that this sensor network only contains 4 nodes, the TTL is inherently limited to 4 hops. The **“payload”** item of the packet varies depending on the message type. A light sensor message will contain the node’s sampled light sensor value. A neighbor table list will contain the node’s sample light value in addition to its neighbor table list. A gateway message will contain the new sampling rate for the light value and neighbor table list.

Update Neighbor table task:

Throughout the task the node updates and checks its neighbor list after a message has been received. If the new packet’s node_id is not in the local neighbor list, a new node_id is added. At the end of the period the node sends its current neighbor list followed by a list reset. The neighbor list is reset to ensure that the node dynamically responds to the nodes entering or leaving the network.

Sample rate task:

Each sensor node has two timers. One is responsible for periodically sampling and sending the node’s light sensor value. The second timer is responsible for periodically sending the node’s neighbor list. Both sample periods can be dynamically updated by a gateway message. If a timer has expired, the node sends either a sensor message or a neighbor table message.

Gateway Design.

The gateway node is organized into three tasks: “transmit”, “receive”, and “ui”.

Transmit task:

The transmit task is responsible for periodically telling the client nodes how often they should be sending the gateway updates to their sensor values and neighbor lists. These update rates are saved as time durations in the gateway's global memory. The ui task will update these values concurrently, so the transmit task is simply responsible for sending whichever value the update rates are currently set to. Since these tasks are pre-emptive, the nano-rk scheduler could switch tasks while the transmit task is still sending the update rates, therefore access to these values are restricted by a semaphore. This way, the ui task will not update the values while they are still being sent.

Receive task:

The receive task is responsible for continuously listening for client messages, and saving their message data. The gateway keeps a list of light sensor values for each client node, and a list of neighbors for each node. The light sensor values are updated when a light sensor message is received, and neighbor table messages are updated when a neighbor table message is received. Once again, access to both the light sensor values list and the neighbor tables list are restricted by a semaphore so the ui will not try to display these values while they are being updated.

UI task:

The ui task is the gateway's interface to the user. It prints lines of text over serial to be displayed on a host computer (via minicom), and listens for text commands from the user on the input serial line. The task will display the available commands to the user, and continuously wait until a command is sent. Five commands are supported:

1. Neighbor Table Update Rate: allows the user to enter a new time interval for client nodes to update the gateway with their neighbor lists.
2. Sensor Sample Update Rate: allows the user to enter a new time interval for client nodes to update the gateway with their light sensor values.
3. Print Map: Displays the topology of the mesh network to the user. By modeling the neighbor lists received from clients as a graph adjacency list, the graph structure can be printed out to display to the user.
4. Print Data: Displays the current light sensor value from each client node to the user.
5. Stream incoming: Displays all incoming packets from each client node.

Performance metrics.

Reliability is an important metric for this course project. The project demo involves user interaction therefore all packets sent by the gateway must be seen by every node. In order to minimize packet loss the gateway and client nodes operate with a high duty cycle. A high duty cycle network also reduces latency which will also enhance the user interface.

Flooded networks can also affect network packet reliability. This network prevents a redundant packet from being propagated by checking the packet's sequence number. Furthermore, the gateway periodically sends the most recent sample rates to ensure each client has the most recent sample rate. Simulation results show that for a set of 1129 packets received at the gateway, 507 of them had been relayed at least once by another node. Also, it was determined that in this set 643 individual messages were sent from all three nodes and only 33 of them were not received at the gateway. That is, approximately 5% of messages were lost.