IoT project: a waste management system

Giuseppe Leone, 10518770
Alessandro Petocchi, 10661410

Abstract

Using TinyOS, we provided an implementation, simulated by TOSSIM, of a smart waste management system. This document illustrates the various operation modes, such as trash filling, trash collection by a custom truck mote and, if needed, garbage transfer between bins.

1 Overview

The network is composed by two types of motes: the *bin motes*, that collect the generated garbage, and a *truck mote*. The latter is used by the bins to empty their garbage levels, once certain conditions are met.
An additional capability of the bin motes is the possibility to communicate to its neighbors when they reach full capacity, so that any incoming garbage can be transferred to the nearest available bin, thus avoiding garbage overflow.
The whole network is emulated using TOSSIM. Serial communication is implemented in the application, in order to attach the simulation to either TOSSIM-Live or NodeRed, where the output of relevant messages circulating in the network will be visible.

2 Technical details

At network startup, specifically when the `SplitControl.startDone` event is signaled without errors, all motes randomly generate their X and Y coordinates, that are inserted into a `mote` struct.
When startup is completed, network operation is divided into two phases:

- **Garbage phase:** In this phase the bin motes will collect the trash. This is implemented via a timer called `TimerTrashThrown`, that is oneshot-started with a random value between 1 and 30 seconds.

  When the timer is fired, a random amount of garbage (1 to 10 units) is generated and added to the bin's `mote.trash` value. Then the same timer is restarted with a newly generated random value, so that this operation repeats "periodically".

- **Sensing phase:** The bins check their filling level:
  1. `mote.trash` < 85: **NORMAL MODE**.
  2. $85 \leq$ `mote.trash` < 100: the bin is almost full. If this condition is met, the bin will go in **ALERT MODE**.
  3. `mote.trash` $= 100$: the bin cannot store any more garbage and will go in **NEIGHBOR MODE**.

2.1 Alert mode

This mode is triggered by setting the `alertMode` flag to true. If this condition is met, the bin mote will send periodic `ALERT` type messages to the truck. To accomplish this a `TimerAlert` with a period of ten seconds is started: when fired, the mote will call the `sendAlertMsg` task.

The `alertMsg` type is defined in the header file, and it contains the mote's X and Y coordinates along with its ID number. The message is sent only to the truck mote (identified in the topology with `TOS_NODE_ID` = 8).

2.2 The truck

The truck mote has an infinite trash capacity and empties the bin's filling level when requested. Upon the receiving of the `ALERT` message, the truck will fetch the bin's coordinates and will start a `TimerTruck` to simulate the trip towards to the bin.
Furthermore, it sets its `truckBusy` flag to true, in order to ignore other incoming `ALERT` messages from different bins, as the truck cannot simultaneously travel towards two different locations.

The travel time used in the timer is defined as

$$t = \alpha_{truck \to bin} \sqrt{(x_{mote2} - x_{mote1})^2 + (y_{mote2} - y_{mote1})^2} \tag{1}$$

where the multiplicative constant is defined in the header file and is set to 100. This is needed to differentiate the truck travel time from the trash transfer time between bins in neighbor mode, as we will illustrate later.

When the timer is fired, the mote will send a `TRUCK` message to the requesting bin and refresh its coordinates with the ones of the bin. Upon the message's arrival, the bin will empty its filling level and send an ACK.

2.3 Neighbor mode

If a bin is full it cannot store newly generated garbage. In order to solve the problem, the trash is redirected towards other bins with available capacity. A bin can also generate more garbage than its residual capacity (consider the case of 10 generated garbage to be thrown in a bin with a 99 filling level). The program handles this case by filling the bin completely and sending the remaining excess trash to other bins.

To locate the destination bin, the mote sends a `moveMsg` of type `MOVEREQ` to its neighbors and wait for two seconds (`TimerListeningMoveResp`). The message contains the mote's coordinates and the amount of trash (`mote.excessTrash`) to be thrown away.
The receiving node, if in `NORMAL` mode, will set a `TimerMoveTrash` where time is provided by equation (1), where the multiplicative constant is replaced with $\alpha_{bin \to bin} = 2$. As noted earlier, this constant is much smaller to better modelize a real network (as the garbage truck might encounter traffic on its way, or be very far away from the motes).

The neighbors will then send a `MOVERESP` type message containing their coordinates. When the receiving window expires, the requesting bin will collect all messages, compute neighbor distances, and send the excess trash to the closest one with a `MOVETRASH` message. If no neighbor replies, the excess trash is deleted.

3 Serial port communication

The program is able to send relevant messages to the serial port. A Java application was written in order to decode the messages and print them to the terminal.

3.1 Technical details

The serial port will receive two kinds of messages: the arrival of the truck at a bin and trash transfer to a neighbor, along with the excess trash value.
The payload of the serial message is generated as such:

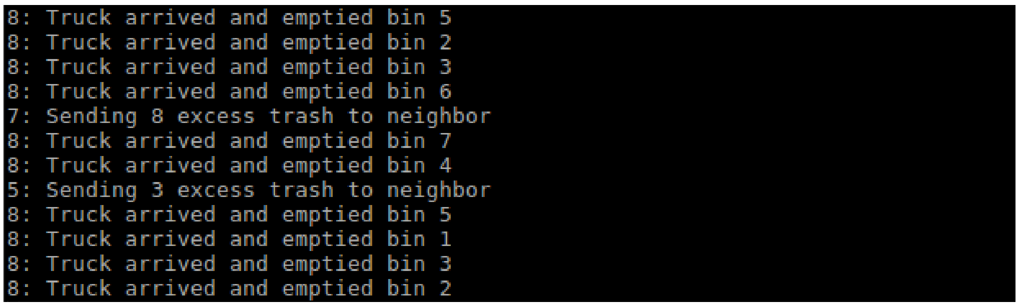
- **Truck messages:** `sfpayload = TOS_NODE_ID << 8 | fullBinID`
- **Neighbor messages:** = `sfpayload = TOS_NODE_ID << 8 | mote.excessTrash;`

where `<<` is the left bit shift operator and `|` is the bitwise `OR` operator.

Upon the receiving of the message, the Java application will fetch the payload by calling `get_sample_value()` and decode it into the two original values (right bit shift for the first eight bits and `AND` with `0xFF` for the last eight bits).

If the ID in the payload is equal to 8 (the truck's `TOS_NODE_ID`), the application will print a message containing the ID of the emptied bin. If it is any other value (the IDs of the bins), it will print a message containing the value of excess trash sent to the neighbor node.

To receive this messages, the TinyOs SerialForwarder must be opened on port 9001, while the SendAck application on port 9002. After launching the simulation, the output will look like this:



```
8: Truck arrived and emptied bin 5
8: Truck arrived and emptied bin 2
8: Truck arrived and emptied bin 3
8: Truck arrived and emptied bin 6
7: Sending 8 excess trash to neighbor
8: Truck arrived and emptied bin 7
8: Truck arrived and emptied bin 4
5: Sending 3 excess trash to neighbor
8: Truck arrived and emptied bin 5
8: Truck arrived and emptied bin 1
8: Truck arrived and emptied bin 3
8: Truck arrived and emptied bin 2
```