Internet of Things

POLITECNICO
MILANO 1863

Waste Management System

MATTEO COLOMBO - 883114 - 10459278

ANDREA TROIANIELLO - 898113 - 10455250

August 25, 2019

Introduction

The project chosen consist in implementing a Waste Management System network, composed by at least 8 motes: one truck and 7 or more bins.

We choose to implement the project with the following tools:

- TinyOS
- TOSSIM
- TOSSIM-Live

Assumptions

1. Coordinates are expressed in meters.
2. Bins and truck coordinates are randomly generated in the range [0,1999].
3. When the truck is traveling to a bin, it ignores all the alert messages.
4. The truck acknowledges the reception of an Alert message, but doesn't respond. If, for example, it is traveling to bin #1, bin #1 will keep sending alert messages.
5. Alert messages are periodically repeated every 10 seconds, after the first one.
6. When a bin receives a Move request, it sets a lock, so that other Move requests from other nodes are rejected.
7. A Move request lock lasts 2s, after that the bin releases it.
8. If a bin in Normal status accepts a Move request and then, with the trash redirected from another mote, becomes full, it will still accept the trash and send itself a new Move request.

General Idea

Due to the limitations with heterogeneous networks of TOSSIM, each node, depending on its ID, when turned on decides how it should act:

- A truck is a node whose ID is 0.
- A bin is a node whose ID is greater than 0.

Communication Channels

To distinguish between bin-bin and bin-truck communication, we decided to use two channels, the first one (Channel 6) dedicated to bin-bin messages and the second one (Channel 7), dedicated to Alert and Truck messages.

- On the truck-bin channel only end to end messages are sent.
- On the bin-bin channel a broadcast messages is sent in case of a Move message, while end to end messages are used for the following responses.

Components

We decided to create two components:

WMSMote: It is the main component that depending on its ID can be either a truck or a bin.

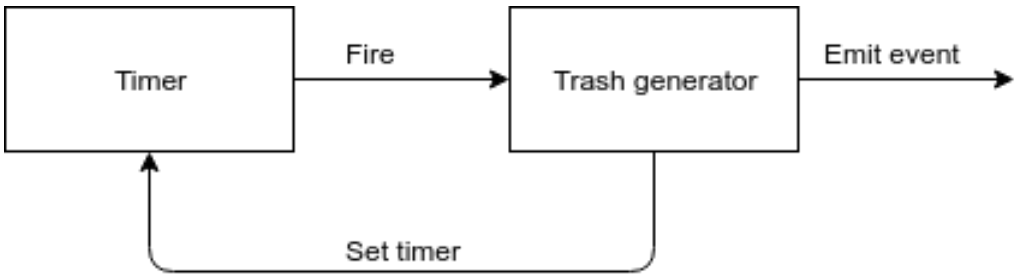
BinSensor: It is a non-bootable component that represents the sensor placed on each bin that triggers an signals an event every time some trash is added.

BinSensor

A BinSensor simulates the sensor which measures the quantity of trash that is added in the bin at every iteration.

Every time some trash is generated, an event containing the number of new units is emitted.

To work it uses a timer, which every time is set with a different (random) value in the range [1,30]s.

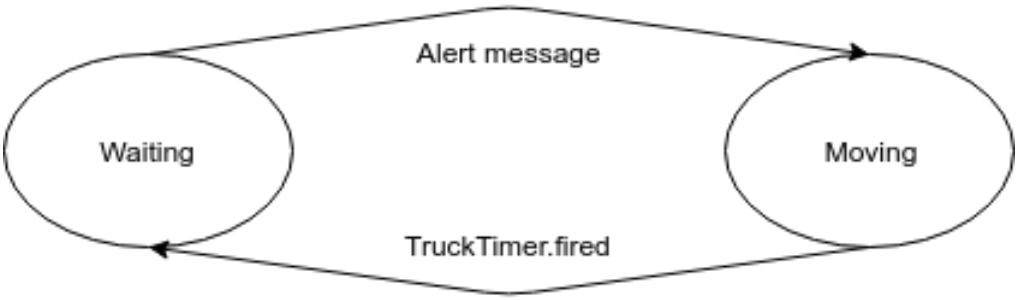


WMSMote

The WMSMote is the main component of the project and includes the logic and implementation of both the truck and the bins.

They both share some code and respond to some common events, but have different behaviour.

The Truck has a simple behaviour which can be represented by this finite state machine:



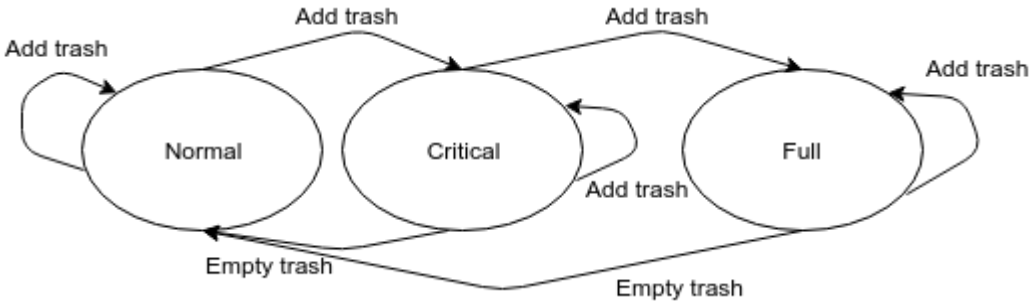
When the truck is waiting, it listen for Alert messages and when the first arrives it starts traveling and sets the variable *moving=true*. If other alerts arrive while the truck is moving, they are ignored.

The *TruckTimer.fired* event is used to simulate the travel, when it fires it means that the truck is arrived to destination.

At destination the truck can empty the trash and notify the bin. Then it acquires the coordinates, releases the lock on *moving* and starts again to listen.

The truck listens only to messages on the truck-bin channel, while even if he receives those on the bin-bin channel, it ignores them.

The Bin has a more complex behaviour within respect to the truck, as it communicates on two channels and has more statuses.



The event *Add trash* is emitted by the BinSensor and depending on the current state, it is handled in different ways.

If there is a state change when the trash level is increased, the mote may start alerting the truck os redirecting trash to neighbours.

A Bin Mote is always listening on two channels, on the truck channel is listening for arrival notifications of the truck so that it can be emptied, while on the bin-bin channel it is waiting for Move requests from other bins.

When the trash outside a bin is greater than zero, the bin sends a broadcast Move message to its neighbours and starts a *MoveTrashTimer*. When this fires, it checks if at least a neighbour bin responded and if so it sends trash to it.

When a bin receives a Move request, it checks its internal status and if its Normal it starts a *MoveResTimer* whose timeout depends on the distance. When it fires it responds to the mote which made the move request.

When a mote accepts to receive trash from another bin, it sets a lock so that if it receives other move requests from other nodes they are ignored. To avoid deadlocks, there is a timer after which the lock is released and the node starts to listen again for Move requests.

Implementation and Simulation

The application was completely implemented as by project requirements and its completely documented.

For the simulation part we did two tests, the first one with the topology and noise trace provided at lesson, and the second one with a completely new topology and noise trace.

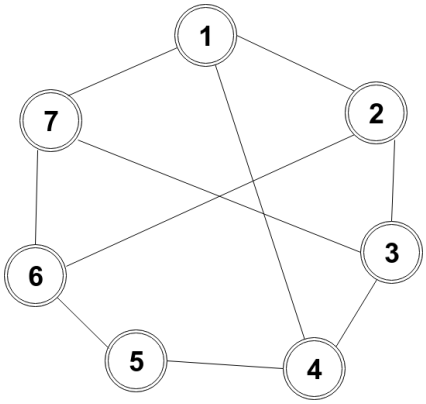


Figure 1: Topology 2