Internet of Things

POLITECNICO
MILANO 1863

Waste Management System

MATTEO COLOMBO - 883114 - 10459278

ANDREA TROIANIELLO - 898113 - 10455250

August 25, 2019

Introduction

While developing the project we made the following assumptions:

1. Coordinates are expressed in meters and they are in the range [0,1999].
2. When the truck is traveling to a bin, it ignores all the alert messages.
3. Alert messages are repeated every 10 seconds even by the bin where the truck is moving to.
4. When a bin receives a Move request, it sets a lock, so that other Move requests from other nodes are rejected.
5. 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.

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 truck-bin messages.

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.

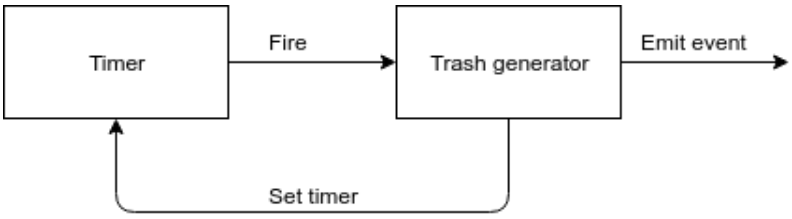
BinSensor: It is a non-bootable component that represents the sensor placed on each bin that signals an event every time some trash is added.

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.



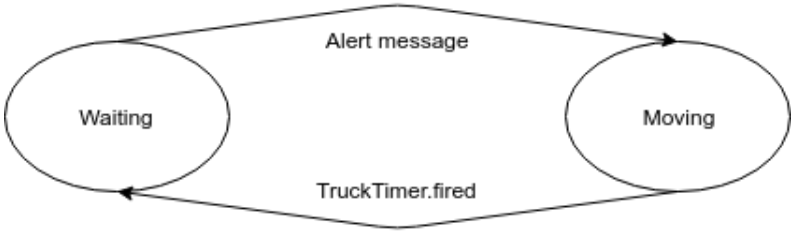
WMSMote

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

This is due to the fact that TOSSIM is not capable of handling heterogeneous networks. Therefore a node whose ID is 0 will be a truck, while a mote with an ID greater than 0 will be a bin.

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

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



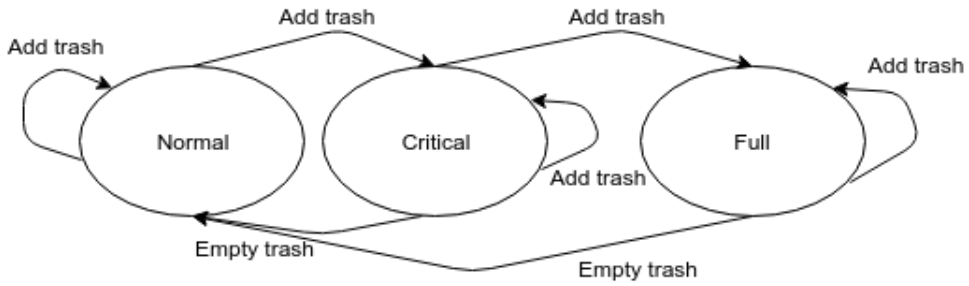
When the truck is waiting, it listen for Alert messages and when the first arrives it starts traveling and sets the variable *moving=true*. If other alerts arrive while the truck is moving, they are ignored.

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 compared to the truck, as it communicates on two channels and has more states.



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 the closer one.

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 Testing

The application was completely implemented with TinyOS, TOSSIM and TOSSIM-Live, moreover its code is 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.