Tree-Based Routing

Sensor Networks and Internet of Things

Sébastien Vaucher and Benjamin Bediako

Université de Neuchâtel
Neuchâtel, Switzerland
{sebastien.vaucher,benjamin.bediako}@unine.ch

1 Introduction

This report presents the results obtained during the project of the Sensor Networks and Internet of Things lecture taught at the University of Bern. The goal of the project is to implement a tree-based routing protocol on top of the Contiki platform. The implementation is programmed in the C programming language.

This document is structured as follows. The first section contains a theoretical introduction of the protocol. In the second part, the implementation is discussed. Finally, we evaluate different variants of the protocol and discuss the results.

2 Protocol Description

This section describes the routing protocol which we were tasked to implement.

In a wireless sensor network, a number of nodes collaborate to provide sensor data to a more powerful computer for further analysis. Our deployment considers a single aggregator, commonly called the *sink*. A number of low-power sensor nodes will sense data at their location and then send values towards the sink. As it is not realistically feasible to reach the sink from any location, more distant nodes will forward their data to an intermediate node that is then tasked to send the data towards the sink.

The tree-based routing protocol is elegantly simple. It only requires broadcast and unicast primitives in the sensor node firmware. Packet routing, however, is only possible in one direction, viz. from individual nodes to the central sink node.

The tree-based routing protocol works in two phases. First, the discovery phase will shape paths from each leaf node to the root node. After this phase is completed, each node knows a route towards the sink that it can use to transfer useful sensor data.

2.1 Discovery Phase

The discovery phase of the algorithm works by *flooding*. Discovery messages will be exchanged from the sink to distant nodes. A discovery message contains the following fields:

```
Input:
WaitTime: Time to wait between each discovery phase
NodeId: ID of the root node
begin
    SequenceEmitted ← 0 ;
    while $\mathcal{T}$ do
        Sleep(WaitTime) ;
        SequenceEmitted ← SequenceEmitted + 1 ;
        DiscoveryMessage.$ParentId$ ← NodeId ;
        DiscoveryMessage.$HopCount$ ← 1 ;
        DiscoveryMessage.$SequenceNumber$ ← SequenceEmitted ;
        Broadcast(DiscoveryMessage) ;
    end
end
```

Algorithm 1: Sink node discovery algorithm

```
Input:
NodeId: ID of the current node
DiscoveryMessage: An incoming discovery packet
PreviousSequence: Most recent sequence number heard
Output:
NewParent: New parent node
begin
    ShouldForward ← $\mathcal{F}$ ;
    if DiscoveryMessage is the 1$^{st}$ message to ever arrive then
        NewParent ← DiscoveryMessage.$ParentId$ ;
        ShouldForward ← $\mathcal{T}$ ;
    else
        if DiscoveryMessage.$SequenceNumber$ > PreviousSequence then
            ShouldForward ← $\mathcal{T}$ ;
        end
        if DiscoveryMessage.$ParentId$ would be a better parent then
            NewParent ← DiscoveryMessage.$ParentId$ ;
            ShouldForward ← $\mathcal{T}$ ;
        end
    end
    if ShouldForward = $\mathcal{T}$ then
        DiscoveryMessage.$HopCount$ ← DiscoveryMessage.$HopCount$ + 1 ;
        DiscoveryMessage.$ParentId$ ← NodeId ;
        Broadcast(DiscoveryMessage) ;
    end
end
```

Algorithm 2: Sensor node discovery algorithm

– The parent node ID
– A hop counter
– A sequence number

The parent node ID is the identifier of the node sending a particular packet. The hop-count value states how many nodes a packet has traversed. When a packet originates from any node, the hop-count is always 0. The value is incremented whenever a node forwards a packet. The sequence number identifies an instance of a packet. Nodes that forward a packet must never change this value. The sequence number is incremented by 1 for each new discovery message created by the root node. Using this information, sensor nodes can identify whether an incoming packet is new.

The root node constitutes what we will call "level 0" of the tree. Its operations are described in algorithm 1. In short, the root will broadcast a new discovery packet at regular intervals.

When a sensor node that is not the root receives a discovery packet, it proceeds as shown is algorithm 2. Basically, a node will compare the information in the discovery packet with the characteristics of its current parent. If the parent advertised in the packet would be better (smaller hop-count, better signal strength, etc.), the node assigns it as its parent. Then, if the packet would be of interest to other nodes – if it is a new packet coming from the root, or if the hop-count is smaller than a previously forwarded packet – then, the node will broadcast it. The process is periodical, so broken links in the system will automatically heal when the next discovery iteration happens.

2.2 Sending Sensor Data

The role of the discovery phase was to establish a routing tree. Sending data to the sink is then trivially simple for the sensor nodes. Once a sensor node knows a parent, it will routinely send sensor data to it. The parent node will then forward the packet to its own parent until the sink is reached.

3 Analysis

In our implementation, we have two configuration options: the MAC protocol and the metric used to qualify if a parent node is "better" than another. We compared two MAC protocols: NullMAC and X-MAC [1]; and two metrics: hop-count and RSSI. The hop-count metric favorises nodes that are topologically closer to the sink. The RSSI metric favorises the quality of the reception from the parent node.

We ran our algorithm of the TARWIS testbed of the University of Bern, by batches of 30 min.

			Packet delivery rate	
MAC	Strategy	Average hop-count	Global	Connected only
NullMAC	Hop-count			
	RSSI			
X-MAC	Hop-count			
	RSSI			

Table 1. Comparative table of different configurations

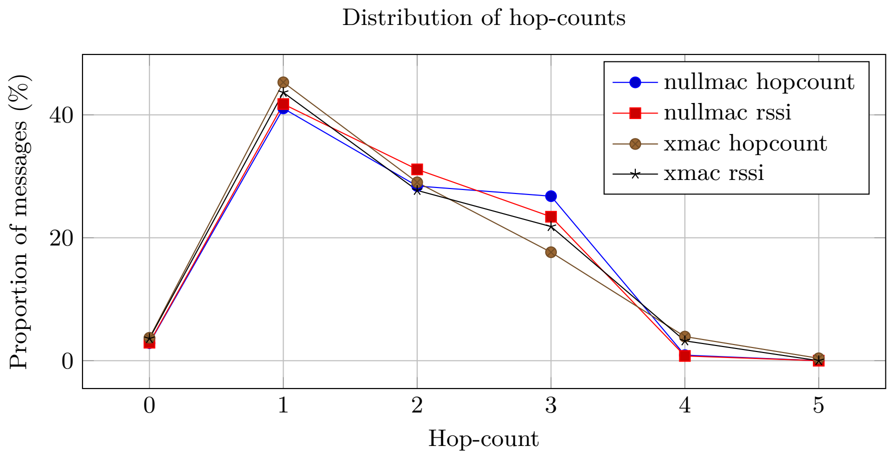

Figure 1. Distribution of hop-counts with different strategies

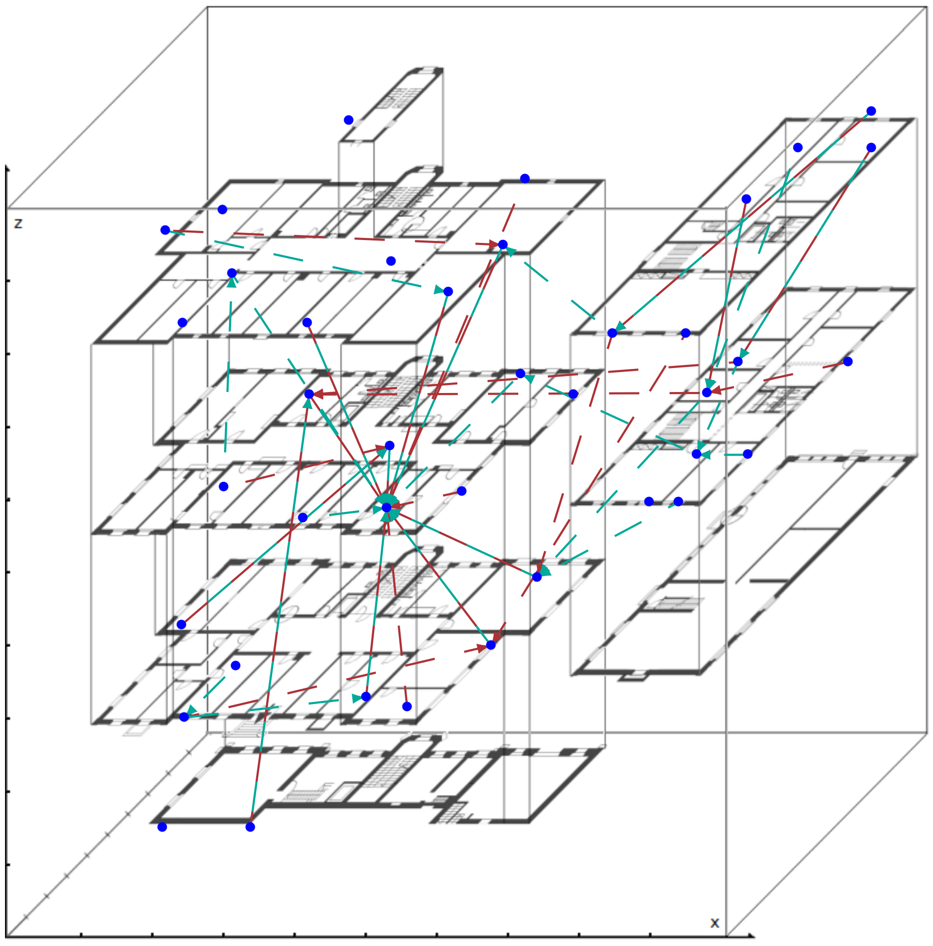

Figure 2. Parent relations with X-MAC, using the hop-count or RSSI strategies

References

1. Buettner, M., Yee, G.V., Anderson, E., Han, R.: X-MAC: A Short Preamble MAC Protocol for Duty-cycled Wireless Sensor Networks. In: Proceedings of the 4th International Conference on Embedded Networked Sensor Systems, SenSys '06, pp. 307–320. ACM, Boulder, Colorado, USA (2006)