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 [1]. 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, we evaluate different implementation 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 intermediate nodes that are 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, that is from individual nodes to the central sink node.

The tree-based routing protocol works in two phases. First, the discovery phase will discover 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:
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 1: Sensor node discovery algorithm executed by nodes at level > 0 when they receive a broadcast packet

- 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 role is to broadcast a new discovery packet at regular intervals. It will also receive all sensor data from all the nodes in the system.

When a sensor node at level > 0 receives a discovery packet, it proceeds as shown in algorithm 1. 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.

Sensor data packets contain the following fields:

- The ID of the origin sensor node
- A hop counter
- Sensor data

The hop counter starts at 1. The value is incremented each time the packet gets forwarded by an intermediate node.

3 Implementation and methodology

This section discusses our implementation of the protocol and how we proceeded to collect statistics about each run.

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 [2]; in addition to two metrics: hop-count and RSSI [3]. The hop-count metric favorises nodes that are topologically closer to the sink. The Received Signal Strength Indication metric favorises the quality of the reception from the parent node.

We ran our algorithm on the TARWIS/WISEBED testbed of the University of Bern, by batches of 30 min. The sink periodically prints the distribution of the hop-counts of the packets it received. Each sensor node prints the identifier of its parent everytime it assigns a new parent. Using these data, and combined with the metadata contained in the XML output from TARWIS, we were able to compute various statistics related to each execution.

The analysis of the logs is totally automated thanks to a custom-written Python script. It computes the values shown in Table 1, and generates the Ti*k*Z code that draws Figure 2.

4 Analysis

In this section, we analyze the results we got after running our implementation on the testbed.

Table 1. Comparative table of different configurations

MAC	Strategy	Average hop-count	Avg. link distance	Connected nodes (%)	Packet delivery rate (%) Global	Packet delivery rate (%) Connected only
NullMAC	Hop-count	2.73	*n/a*	*n/a*	85.0	*n/a*
	RSSI	2.77	*n/a*	*n/a*	87.2	*n/a*
X-MAC	Hop-count	2.72	49.5	72.5	66.3	89.1
	RSSI	2.61	46.1	67.5	68.6	99.0

Some raw statistics are found in Table 1. Some values related to NullMAC executions are absent. The reason is because most of the prints executed on remote sensor nodes did not arrive to the testbed log collection system. This problem never occurred using X-MAC, but systematically happened with NullMAC. The cause of this problem is unknown.

By looking at Table 1, we see that NullMAC yields a better global packet delivery rate. By extrapolating with data from X-MAC, we can assume that more nodes successfully connected using NullMAC. A connected node is a node that printed its parent identifier during the execution.

The packet delivery rate is definitely affected by the parent choice strategy in use. By using RSSI, sensor nodes choose a parent that has a better connection quality. This leads to less communication problems when sending sensor data to the sink. The influence of both configuration options on the average hop-count is insignificant. However, the choice of hop-count vs. RSSI has an influence on the physical average length of individual links between nodes. With RSSI, the nodes tend to choose parents that are physically closer.

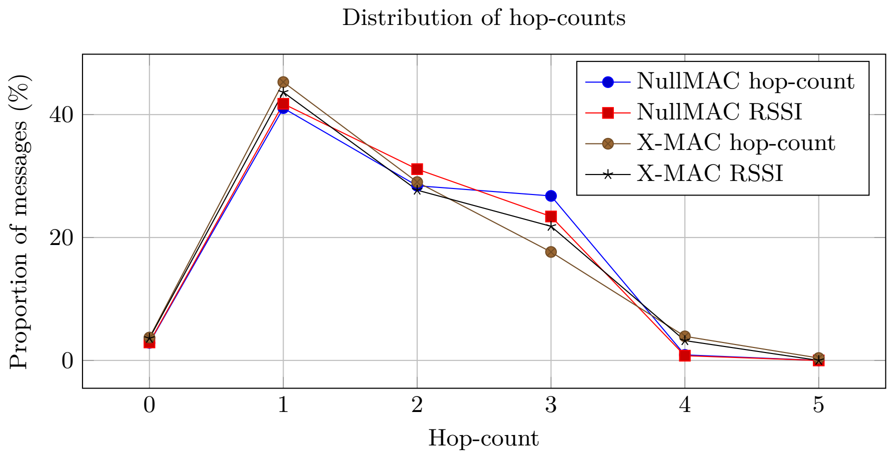

Figure 1. Distribution of hop-counts with different strategies

Looking at the distribution of received messages hop-counts (in Figure 1) tells us that the majority of nodes in the system were located within 3 hops of the sink. The distribution of hop-counts is only slightly affected by configuration.

In Figure 2, a 3D representation of the parent relations within nodes is shown. Each arrow represents a child-parent relation. The sink node is obvious to locate, it is the node with the most children. We can see that even remote nodes were able to connect to the system. Some paths are quite surprising and really show that radio propagation in a building is hard to predict. Nodes that are not connected are sometimes adjacent to connected nodes. This means that the disconnected node did not receive any discovery packet while its neighbor did.

5 Conclusion

Our implementation of the protocol is functional and adheres to the specification given by the instructors. It is able to create a tree structure that nodes can use to send data towards the root node. As specified, we implemented multiple variations of the protocol, differing in the MAC layer and in the evaluation strategy.

Our findings show that NullMAC performs better than X-MAC as far as the global delivery rate is concerned. This means that the transmission of sensor data is more reliable. However, NullMAC is known for being power hungry and unable to cope efficiently with congestion. Moreover, when we used NullMAC, important logging messages were lost. We were therefore not able to compute certain types of statistics.

In the comparison of the two parent-choice strategies, RSSI offers better reliability due to the choice of more reliable links between nodes. We thought that choosing the hop-count strategy would yield overall lower hop-counts, but it is not the case. Therefore, RSSI is the best strategy to choose in our setup.

The results presented in this report come from executions on a single topology which is that of the TARWIS testbed of the University of Bern. It is important to note that any conclusion drawn in this report only applies to this specific case only.

References

1. Dunkels, A., Gronvall, B., Voigt, T.: Contiki - a lightweight and flexible operating system for tiny networked sensors. In: Local Computer Networks, 2004. 29th Annual IEEE International Conference on, pp. 455–462 (2004)
2. 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)
3. Wikipedia, “Received signal strength indication.” `https://en.wikipedia.org/w/index.php?title=Received_signal_strength_indication&oldid=689425904`

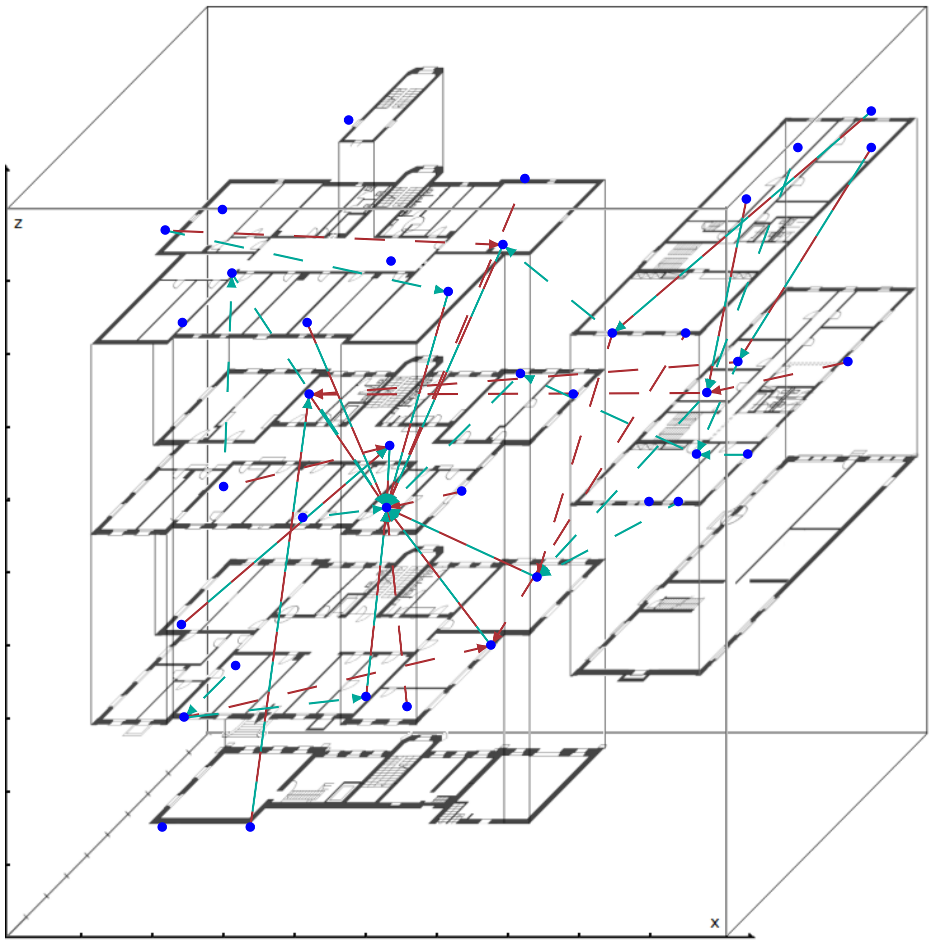

Figure 2. Parent relations with X-MAC, using the hop-count or RSSI strategies