6LoWPAN Forwarding Techniques for IoT

Kevin Mc Gee  
The Entwine Centre  
Dublin City University  
Ireland  
kevin.mcgee7@mail.dcu.ie

Martin Collier  
The Entwine Centre  
Dublin City University  
Ireland  
martin.collier@mail.dcu.ie

Abstract—Recently, there has been significant interest in the 6LoWPAN standard and its applicability to IoT. This paper is concerned with 6LoWPAN using unslotted CSMA-CA IEEE802.15.4 and compares two forwarding approaches described in RFC 6606 for 6LoWPAN routing, namely “route-over” and “mesh-under”. It extends previous work by evaluating both approaches on a large testbed. Results were obtained for both for low-noise and noisy channels. The results show that, although performance is affected by multiple factors, including transmission power, network topology and channel noise, the use of route-over forwarding generally results in a more scalable and robust 6LoWPAN network.

Keywords—6LoWPAN, Forwarding Techniques, IoT

I. INTRODUCTION

With the emergence of new wireless technologies for IoT applications, the 6LoWPAN standard is gaining significant interest both in academia and in industry. It has similarities to the proprietary protocol; ZigBee [1], but incorporates the advantages of low power mesh networking with IPv6 compliance. Two incompatible networking approaches satisfy the 6LoWPAN standard. These approaches are referred to as the route-over approach and the mesh-under approach [2]. The former approach treats each mesh hop as an IPv6 hop and thus the 6LoWPAN fragments are reassembled and decompressed at each hop before forwarding. The latter approach uses link-layer forwarding and thus fragments are simply forwarded at intermediate nodes.

Prior work in this area indicated that the mesh-under forwarding approach is more susceptible to noise but typically has shorter latencies [3, 4]. However, these studies used either analytical methods or a hardware testbed of limited scale. Whether these results hold for larger and more complex topologies is an open question.

This paper involves a comparison of both forwarding approaches in NS-3. Both approaches are simulated in a variety of network topologies; including a large topology with significant levels of noise present. A small-scale hardware testbed is used to tune the simulation parameters.

A brief overview of the design of the hardware and simulator testbeds is presented in Section III. The results obtained and their analysis are discussed in Sections IV and V respectively. Our conclusions regarding the respective merits of the route-over and mesh-under approaches are given in Section VI.

II. PRIOR RESEARCH INTO FORWARDING APPROACHES

An analytical comparison of the two forwarding approaches concluded that the route-over approach is far more suitable for noisy channels but that it results in greater latency [3]. The CSMA-CA performance of a 7 node 6LoWPAN network was analyzed using a Markov Chain model [1]. It was found that route-over forwarding outperformed mesh-under for routes with multiple hops, and in the presence of channel interference.

RPL-based [5] implementations of the route-over approach have been evaluated by computer simulation in [6, 7]. No such research has included a simulation of the mesh-under approach and their route-over NS-3 module implementations are not publicly available.

A. Ludovici et. al implement and test both approaches on an actual hardware testbed in [4]. This implementation involved building the mesh-under approach into the same OS as that used for the route-over approach. They test both approaches in a line topology. Their results show that the mesh-under approach has lower latencies for all the ping tests carried out, regardless of payload size. They also found that the route-over approach consumes more power.

The 6LoWPAN performance of OpenThread (using mesh-under forwarding) and the Contiki OS(using route-over) was compared by Ellmer on a hardware testbed in [8]. It was found that the OpenThread stack exhibits higher latencies, which is attributed to implementation issues in the OpenThread stack. The tests were performed in a clean environment, which may not be representative of an industrial setting. A custom network topology was used in the testbed that would be extremely hard to duplicate.

TECHNICAL DESCRIPTION

A. Hardware Testbed

1) Hardware Used

The forwarding approaches are evaluated on a testbed whose nodes utilize the following hardware:

- A SAMR21-XPRO microcontroller [9]
- One INA219 current measurement sensor [10]
A Raspberry Pi board [11] Route-over forwarding was implemented through the use of the GNRC network stack present in the RIOT OS [12] and the Mesh-under forwarding approach was implemented using the OpenThread stack, which executes as a standalone process in RIOT OS [13]. This process is given the highest possible priority to ensure a fair comparison of the two approaches.

2) Data Collection
To compare the performance of the two forwarding approaches, various pieces of information are required from each of the hardware nodes. Table 1 describes the information collected at each node to compare the performance of the two forwarding approaches and how it is measured.

<table>
<thead>
<tr>
<th>Element</th>
<th>Collection Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current Consumption</td>
<td>A current sensor(INA219) and microcontroller logs the data, whilst ping tests of various sizes are performed.</td>
</tr>
<tr>
<td>Round Trip Time &amp; Jitter</td>
<td>The output of the Ping utility is visible via UART connection and is logged to a Raspberry Pi. The interval between successive ping tests was set between 1 and 4 seconds, for payload sizes ranging between 50 and 1200 bytes.</td>
</tr>
<tr>
<td>Throughput</td>
<td>The stacks were modified to repeatedly send large amounts of UDP messages at a specific frequency and the Rx counts at Layers 2, 3 &amp; 4 of the destination node were monitored.</td>
</tr>
</tbody>
</table>

Whilst these tests are being performed, there is a need to record information regarding the current state of the network stacks executing on the various nodes in the network. In any mesh network implementation there are source, destination and forwarding nodes. The relevant information for all the node types include factors like Tx/Rx counts and errors at layers 2 and 3, along with buffer information. This data is sent out once every second to the UART port of the SAM-R21 in a CSV format, or in the case of the source node; is presented along with the test results in CSV format.

3) Data Logging
To record the data from the INA219 sensor and from the SAM-R21 UART port, a Raspberry Pi is used. This device can be used as the I2C master device for retrieving the current measurement data from the INA219 and can log this data to a file. This application is written in C and it samples the current at around 1300 times per second and writes the average, minimum and maximum current value to a file. The screen utility is used on the Embedded Linux device to log incoming UART data from the SAM-R21 to a file.

4) Test Design
Various topologies are tested, first in a clean environment and again in an industrial environment. To characterize the environments, a modified SAMR21 spectrum scanner application was used [14]. Fig. 1 and Fig. 2 describe the spectrum scanner test results for the Industrial (Factory), Wi-Fi-challenged (RPI) and Clean (Field) environments.

An isolated rural field was chosen as the clean environment, as it is largely free of RF signals in the 2.4GHz ISM band. The field was recently harvested and the nodes were mounted on wooden posts and therefore sit 1m above the ground. The noise levels in the IEEE802.15.4 2.4GHz channels were measured using the spectrum scanner application prior to testing, to verify that this environment is clean. From all of the spectrum tests, no power was measured on any IEEE802.15.4 channel in this environment.

A small machine-shop was chosen as the industrial environment, as it has large amounts of metal, concrete and similar obstacles. Furthermore, there is also an active WiFi Access Point in the factory and large amounts of industrial machinery in operation. It should be noted that even in this environment, the nodes were in a Line-Of-Sight configuration, albeit somewhat cluttered.
B. Simulator Testbed

1) NS3 Overview

NS3 is used to simulate the 6LoWPAN forwarding approaches. NS3 was chosen because it is a well-known simulator and already has libraries that allow simulation of basic 6LoWPAN systems and has plenty of other libraries related to routing. There is also great support for PCAP file logging at all layers of the network stack.

No 6LoWPAN routing protocol currently exists in NS3 and there is no support at all for mesh-under 6LoWPAN simulations. Such routing features as mesh formation and healing are not required in our simulations as they reflect the performance of the IEEE802.15.4 standard and not of the 6LoWPAN layers above it. Thus, static topologies/routing are used, as the control plane aspects of dynamic routing protocols do not need to be simulated.

The hardware testbed results provide a means of verifying the performance of the simulator. Results presented in [4,8] would suggest that the performance of both forwarding approaches is highly dependent on implementation. Although the simulator implementation will be modified slightly to match the hardware results, extra delays that are the result of shortcomings in the design of the RIOT GNRC and OpenThread stacks are not included in the simulator design.

2) Route-Over Implementation

Using the many examples of IPv6 and 6LoWPAN simulations already in existence, the IPv6 static routing library [15] is used to link the nodes together into specific topologies.

3) Mesh-Under Implementation

To implement link-layer forwarding in the existing NS3 6LoWPAN stack, a static lookup table was added to the MAC layer of the stack. This lookup table is a simple static array and is populated at compile time.

4) Small-scale Simulation Modelling

The evaluations of various network topologies using the hardware testbed are replicated in the simulator to validate it. Smaller-scale simulations are also performed to assess the scalability of both forwarding approaches. The measure of scalability chosen is the amount by which network performance degrades with increasing hop count. These small-scale tests involve using line-based topologies of between three and seven nodes.

5) Industrial Simulation Model

To simulate a large 6LoWPAN implementation, a life-like factory model containing over seventy Nodes is used as the test topology. The factory model has dimensions relevant to a large factory setting (111m x 86m) and can be in Fig. 5. The model was designed to represent a large plastic extrusion factory with twelve blown-film extrusion lines and thus the topology consists of twelve clusters of six sensor nodes that are messaging a common central node (seen in green), intended to represent an ideally located border router.

Two different test procedures are used to evaluate the forwarding approaches. The first approach involves all sensors in a single cluster sending one message concurrently to the central node. Then twenty seconds later, this procedure is repeated for an adjacent cluster, and so on for each of the remaining ten clusters. The alternative procedure involves the sending of ten messages consecutively by each node in the cluster before the adjacent cluster becomes active. Tests like these place a balanced load on the network, so as to minimize the probability of packet loss.
IV. RESULTS OBTAINED

A. Hardware Test Results

Table 2 describes the average (Avg) and standard deviation (Std) in Round-Trip Time (RTT) results of the 3-Node line topology tests. Overall, both approaches had approximately the same levels of timeouts.

<table>
<thead>
<tr>
<th>Payload (Bytes)</th>
<th>50</th>
<th>100</th>
<th>200</th>
<th>400</th>
<th>800</th>
<th>1200</th>
</tr>
</thead>
<tbody>
<tr>
<td>Std(ms)</td>
<td>1.5</td>
<td>1.8</td>
<td>2.6</td>
<td>3.4</td>
<td>4.8</td>
<td>5.1</td>
</tr>
</tbody>
</table>

Table 3 describes the average current measurement across all the above hardware tests. The source node is defined as the node from which the ping request originates and the root node is defined as the intermediate node that sits between the source and destination.

<table>
<thead>
<tr>
<th>Field Setting</th>
<th>OpenThread - Field Setting</th>
<th>OpenThread - Factory Setting</th>
<th>RIOT - Factory Setting</th>
<th>RIOT - Field Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Payload (Bytes)</td>
<td>50</td>
<td>100</td>
<td>200</td>
<td>400</td>
</tr>
<tr>
<td>Avg(ms)</td>
<td>45.4</td>
<td>65.2</td>
<td>92.1</td>
<td>156.1</td>
</tr>
<tr>
<td>Std(ms)</td>
<td>1.7</td>
<td>2.2</td>
<td>3.4</td>
<td>4.3</td>
</tr>
</tbody>
</table>

Table 3: Overall Hardware Current Consumption

Other interesting hardware results include:

- OpenThread had a lower average UDP throughput of 39.517kb/s in comparison to 49.787kb/s with RIOT.
- 1200 byte 3-Node Line OpenThread ping timeouts increased by a factor of 6.75 between environments.
- Incorrectly tuned transmission power leads to greater levels of power consumption and more frequent control plane messaging due to the nature of the trickle algorithm [16].
- The twin-source topology exhibited similar average RTT results to the Line topology, but the standard deviation increased significantly. This topology exposed shortcomings in both stacks; the RIOT stack frequently exhibited buffer overflow during the 1200 byte tests and the OpenThread timeouts in certain tests grew by as much as 160% between environments.
- All timeouts experienced were caused by exceeding the CSMA-CA retry limit and not by error-based effects.

- Changing the minimum CSMA-CA back off limit down from the default 3ms to 1ms resulted in as much as a 16% decrease in average RTT for both stacks.

B. Small Topology Simulation Results

After a small amount of simulator tuning, the performance of the simulated 3-Node Line topology was found to be that described in Fig. 6. This figure also shows the RTT simulation results for the same topology with a more accurately tuned transmission power (Low Power(LP) tests).

With the transmission power adjusted optimally, the scalability of both approaches was investigated. Table 4 shows the average (Avg) and standard deviation (Std) in RTT results for a variety of simulated line topologies.

Table 4: Simulated Line RTT Test Results

C. Large Topology Simulation Results

Fig. 7 shows the number of ping requests received by the central node with $P_{drop}$ equal to zero (CLEAN) and 0.4 (DIFFICULT) respectively, in the simulated bulk messaging test.

Table 2: 3-node Line RTT Hardware Results

<table>
<thead>
<tr>
<th>NODE</th>
<th>Std(ms) 1.5</th>
<th>Std(ms) 1.8</th>
<th>Std(ms) 2.6</th>
<th>Std(ms) 3.4</th>
<th>Std(ms) 4.8</th>
<th>Std(ms) 5.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROOT NODE</td>
<td>17.91mA</td>
<td>17.92mA</td>
<td>18.01mA</td>
<td>18.01mA</td>
<td>18.01mA</td>
<td>18.01mA</td>
</tr>
<tr>
<td>SOURCE NODE</td>
<td>17.74mA</td>
<td>17.73mA</td>
<td>17.73mA</td>
<td>17.9mA</td>
<td>17.9mA</td>
<td>17.9mA</td>
</tr>
</tbody>
</table>

Fig. 6: Simulated 3-Node Line Ping Results

Fig. 7: Ping Requests Received by ROOT Node
V. INVESTIGATION AND ANALYSIS

A. Hardware Test Analysis

Both approaches consumed more power in the industrial environment. The probable cause is that more Clear Channel Assessments (CCAs) are performed in a more challenging environment. The evidence for this claim is that all the losses experienced in both environments were caused by exceeding the CSMA-CA limit. From the results, it appears that the RIOT stack consumes more power. The fact that the RIOT application required a larger increase in power between environments than OpenThread, would suggest that OpenThread is more scalable in this regard. Further analysis of the RIOT OS revealed that operations such as header compression and decompression take approximately 28µs and 140µs respectively. This would suggest that the delays in the route-over forwarding node are driven by the design and execution characteristics of the route lookup and 6LoWPAN fragmentation/reassembly code.

The hardware results were then used to adjust the simulator accordingly. The average RTT performance of both forwarding approaches was better in the simulation results than those of hardware. Also, deviations in RTT and ICMP Ping losses were similar, if not smaller in the simulator results. One small issue in the comparison was that mesh under simulation test results exhibited slightly higher losses and RTT deviation in the 1200 Byte tests.

B. Small Topology Simulation Performance

From the 3-Node hardware and simulation results presented above, it is apparent that the mesh-under approach has shorter round-trip times for payloads less than 100 bytes. This phenomenon is not repeated for larger payloads. During the multi-hop count tests, the route-over approach exhibits no timeouts but Fig. 8 shows that the simulated mesh-under approach does, even with UDP tests.

The time interval needed to transmit and receive all the 6LoWPAN fragments in the above 3-node UDP tests, can be seen in Fig. 9. The standard deviation of these parameters also increased in a similar fashion to those of the mean. In the Figure, both intervals are much greater for the mesh-under tests and therefore some effect must be interfering with the transmissions and causing the large reception interval.

Although it is possible that after the first fragment transmission, the random CSMA-CA back off in N1 may beat N2 to channel access, eventually N3 will be transmitting and is likely to be doing so when N1 is still transmitting its other fragments. The route-over approach does not suffer from this problem because fragment transmissions are performed between an exclusive pair of nodes, i.e. N1 sends all of its fragments to N2 before N2 sends all of the fragments to N3. To prove this hypothesis, the maximum back off interval was increased from 5ms to 8ms, so that it is less likely for collisions to occur between nodes N1 and N3. Table 6 shows the effect of this change on UDP datagram loss in the simulated 5-Node mesh-under test.

Alternative solutions include introducing an inter-fragment delay, which will not affect network latency in deep networks to the same degree as the above solution. However, this solution relies on the inter-fragment delay remaining constant, which will not hold true if two transmission paths in the network share a common node or are in radio range of each other. OpenThread would appear to exhibit this timing characteristic; tests involving two hardware nodes running the standalone OpenThread stack revealed a 14ms delay between the reception of successive fragments. The same test in the simulator resulted in an inter-fragment delay of only 7ms. Although this may help to reduce the effects “hidden node” problem in LINE topologies, it does so at the cost of latency and buffer usage time. The use of larger inter-fragment delays will also spread out the channel usage time of each node and
would therefore increase the possibility of “hidden node” based collisions in multi-source topologies. It will also result in larger duty cycles in reduced function nodes and thus increase their power consumption.

A price paid for the benefits of the route-over approach is its memory usage. Fig. 10 shows the rate of change in reception count against transmission count, for the root node of a simulated 3 node UDP test. From the figure, it can be concluded that the fluctuations in memory usage with time have a larger variance in the route-over approach.

Several avenues of investigation were not explored as part of this paper. This research did not utilize duty-cycling of source nodes as its effects may be highly implementation-dependent. Profiling of the OpenThread implementation of 6LoWPAN will be required, as the OpenThread stack exhibited large variations in performance across several hardware test scenarios. In contrast, the degree of performance variation in the mesh-under simulations was not as large.

VI. CONCLUSIONS

The two possible forwarding approaches permitted within the 6LoWPAN standard have been compared in this paper. From this investigation, it appears that the route-over approach is more suitable for use in large industrial applications. This conclusion is supported by the results found in an industrial hardware testbed and by simulation studies. Furthermore, the effects of poorly tuned node transmission power will also have a serious effect on network performance. The mesh-under approach has superior power consumption and memory usage characteristics. However, route-over has the advantage of simplicity and is less susceptible to the “hidden node” problem. General conclusions about the respective performance of the two approaches are difficult to draw, as the performance of any 6LoWPAN network will be highly topology-dependent. Variations in performance may also be attributable to implementation issues rather than to the features of the protocol being implemented. The worst-case scenario for the mesh-under approach would be that the CSMA-CA retry limit is being exceeded to such an extent that no complete IPv6 packets arrive at the destination node. Instead of this, the route-over approach is more likely to exhibit massive levels of buffer overflow in nodes that are relied on too heavily for forwarding purposes.

REFERENCES


