Implementation and evaluation of Multi-hop routing in 6LoWPAN

Alessandro Ludovici, Anna Calveras

Wireless Networks Group (WNG), Universitat Politècnica de Catalunya, C/ Jordi Girona, 1-3, Mòdul C3 – Campus Nord.
08034 Barcelona, Spain
alessandro.ludovici@entel.upc.edu,

***Abstract*- 6LoWPAN enables the transmission of IPv6 packets over LoWPAN networks. In order to make it possible, 6LoWPAN introduces an adaptation layer between network and link layers. This layer allows IPv6 packets to be adapted to the lower layers constraints. It provides fragmentation and reassembling of packets and header compression. It also can be involved in routing decisions. Depending on which layer is responsible of routing decisions 6LoWPAN divides routing in two categories: mesh under if the interested layer is the adaptation layer, route over if it is the network one. In this paper we compare the two routing solutions evaluating their performances in terms of end-to-end delay and round-trip time. All the performance evaluation has been realized in a real implementation of 6LoWPAN.**

***Key Words*- 6LoWPAN, *route over*, *mesh under*, blip, sensors networks**

INTRODUCTION

Wireless Sensor Networks (WSN) represents a low-cost and distributed solution for network applications in environments where wired networks cannot be applied or their deployment is not feasible. WSN are composed by many devices (sensors/actuators) usually embedded in a physical environment to monitor its conditions (e.g. temperature, pollution).

WSN are typically composed by a number of low-power devices having a short communication range. Depending on the application environment, physical obstacles can mask the wireless link between them. This scenario suggests the adoption of multi-hop routing solutions. Furthermore, multi-hop routing helps to broaden the communication range and to use alternative links.

Although a WSN can be based on various protocol stacks, it would be preferable to adopt a standard solution as the one provided for physical and data link layers by the IEEE 802.15.4 standard [1]. Concerning the upper layers, it is possible to find a number of protocols developed as a proprietary solution. As consequence, the integration of WSN with external networks or the interoperability of different WSN become more difficult. WSN would result as stand-alone networks while it would be useful to integrate them in external networks like IP based networks. In this sense, the adoption of IP as network layer protocol would give a chance to integrate embedded wireless network with Internet. In particular, the adoption of IPv6 will provide enough space to address large WSN. The specification on how to enable IPv6 over WSN based on IEEE 802.15.4 are carried out by a specific IETF WG called 6LoWPAN [8] (IPv6 over Low power Wireless Personal Area Networks). Its aim is to provide mechanisms and architectural solutions to ease the integration of the IPv6 in constrained networks environments specified as Low power Personal Area Networks (LoWPAN).

IEEE 802.15.4

LoWPAN are defined by the IEEE 802.15.4 standard. Following the OSI reference model, it specifies the physical and the Medium Access Control (MAC) sub-layer of the data link layer. LoWPAN are characterized to be resource constrained. Devices forming them typically have an 8-bit or 16-bit CPU, 4 KB to 8 KB of RAM and 48 KB to 128 KB of ROM. Due to the limitation in processing and in storage capabilities, the software components developed for these networks have to be very simple and light-weight. Since devices are mainly battery powered, it has to be considered the energy consumption as key aspect when developing such constrained networks. Usually, these devices alternate between being awake for a short amount of time and then entering a sleep mode in which they consume less energy.

A LoWPAN network has also limitation in the available bandwidth. Depending on the frequency band it operates the resulting data-rate are of 250 Kbps (2.4 GHz), 100 Kbps, 40 Kbps or 20 Kbps. Furthermore, LoWPAN networks are self-healing and self-organizing meaning that a node is able to recover from errors and to join a network without external intervention.

IEEE 802.15.4 defines the use of 16-bit short or 64-bit extended link layer addresses and a MTU of 127 bytes. Short addresses are preferable in order to reduce overhead in the MAC frame.

6LoWPAN

The transmission of IPv6 packets over IEEE 802.15.4 links introduces several challenges. Besides the low capabilities in terms of processing, memory and bandwidth of 802.15.4 devices, the IPv6 adoption as network layer protocol does not fit with its MTU (Maximum Transferable Unit) specifications. IPv6 requires at least a MTU of 1280 bytes that is ten times the one specified for 802.15.4 networks. The 40-bytes length IPv6 header imply a huge overhead that, considering the presence of transport layer header (8 bytes for UDP), MAC header (25 bytes) and link-layer security (21 bytes), would leave only 33 bytes available for application data payload. Both problems are addressed in the 6LoWPAN specification [2]. In order to satisfy the MTU requirements of IPv6 and reduce the overhead, 6LoWPAN implements an adaptation layer placed between network and data link layers. This layer provides a mechanism for packet fragmentation, header compression and support for data link layer forwarding of IP packets [3].

IPv6 header compression in 6LoWPAN is possible since the information contained in the header can be inferred from lower layers or by assuming a shared context between network nodes. It can be supposed that a number of 6LoWPAN application scenarios would require packet size considerable smaller than the IPv6 MTU. Consequently, the application payload in addition with the IPv6 compressed header would fit with the MTU requirements of 802.15.4. However, the introduction of further overhead from higher layer protocols or routing headers would require fragmentation of IPv6 packets into multiple MAC frames.

The adaptation layer can also be responsible to take routing and forwarding decisions instead of the network layer. Depending on which layer is in charge of routing and packet forwarding, 6LoWPAN divides routing in two schemes: *mesh under* if routing is done at the adaptation layer and *route over* if done at the network layer.
In fig. 1 and 2 are showed the 6LoWPAN protocol stacks for *mesh under* and *route over*.

The network topology of LoWPAN is expected to be a star or either a mesh topology. However, even if a mesh routing protocol is expected to run over LoWPAN, the 802.15.4 specification does not define such capability [2].

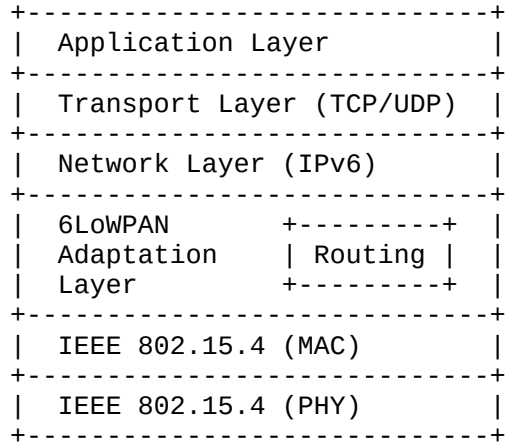

Fig.1 6LoWPAN *mesh under* protocol stack [4]

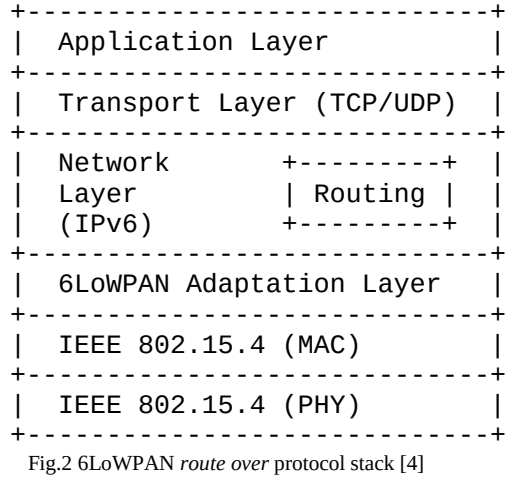

Fig.2 6LoWPAN *route over* protocol stack [4]

The work presented in this paper is to analyse the characteristics of both routing schemes and test their performances in a real 6LoWPAN implementation. We will show how *mesh under* and *route over* performs in terms of latency in a multi-hop 6LoWPAN network.

The rest of the paper is organized as follow: In the next section we present previous research and discussions on *mesh under* and *route over*. Then we will explain how the two routing schemes work and differs. Subsequently, we will introduce the implementation and the test-bed used for their performance evaluation. Finally, we will present the main results of our research and give guidelines for future works.

Related Work

Discussions on *mesh under* and *route over* are presented in [3] and [4]. In [4] there are specified a series of guiding principle for 6LoWPAN routing including both *mesh under* and *route over* solutions. In [3] an extended explanation of the adaptation layer and issues of *mesh under* and *route over* are given.

To our best knowledge, there are no performance analyses of *mesh under* and *route over* but an analytical analysis on a comparison of both routing schemes [5]. Chowdury et al. show in their work how *mesh under* and *route over* differs in terms of packet arrival probability, retransmission policies and latency. A probabilistic model has been used to compare the routing schemes. Considering a transmission of IP fragmented packets in a multi-hop 6LoWPAN network, they demonstrate how *route over* has a higher fragment arrival probability than *mesh under*. In fact, in *route over* the packet is reassembled at each hop before being fragmented again and forwarded. If a fragment gets lost, the previous hop resends the whole IP packet. In *mesh under* the packet is reassembled at destination and each fragments can be forwarded trough a different path. If a packet gets lost the source resend the whole packet. Furthermore, they have shown how in *route over* there is the possibility of buffer overflow when a node is reconstructing a packet. Analysis on latency has demonstrated that it is higher in *route over* due to the time spent in reassembling and fragmenting the packet at each hop. In the next section we explain in more detail the 6LoWPAN routing schemes.

6LowPAN Routing

Since in *route over* decisions are taken at network layer, the addresses carried in the IPv6 headers are used to forward the packet to next hop. IPv6 source address represents the node that has initiated the communication while the destination address is the final node. When a node receives a packet, it unpacks the IPv6 header and uses the destination address to determine whether or not the packet is for itself. If not, it decrements by one unit the hop limit field and, if not zero, it checks its routing table to determinate the next hop. Intermediates nodes addresses are specified as link layer addresses. In *route over* each hop is considered as an IP hop and each node acts as an IPv6 router as well as the edge router. An example of network topology in route over is reported in Fig.6, while in Fig.7 it is showed an example for *mesh under*.

Route over protocols could use routing headers included in the IPv6 payload as an IP extension header. The header chain for *route over* is reported at Fig. 3.

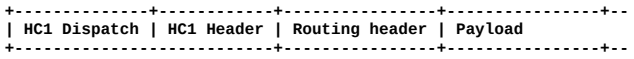

Fig.3 Header encapsulation in *route over*

In *mesh under*, packets are routed at adaptation layer. As well as for *route over*, addresses of intermediates hops are specified as link layer addresses. To realize *mesh under* it is defined the use of a mesh header. In Fig.4 it is showed the mesh header pattern and in Fig.5 how it is encapsulated in the 6LoWPAN frame.

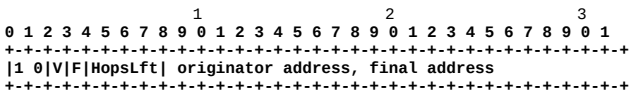

Fig.4 mesh header

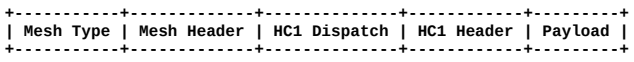

Fig.5 header encapsulation in *mesh under*

The mesh type is specified by the first two bits settled to 1 and 0. The length of originator and final addresses are specified respectively by the V and F bits. If they have the value of 0, the addresses are IEEE extended 64-bit addresses; if 1 they are short 16-bit addresses. The originator and final addresses are in that order the address of the node starting the communication and its destination. Four bits of the first octet are used to specify the number of hops. It can be defined up to 14 hops. An extra byte can be added to define a number of hops greater than 14 by setting the Hops Left to 0xF.

When a node receives a packet with a mesh header, it checks the final address to decide whether or not the packet is destined to itself and update the hops left field in case of forwarding.

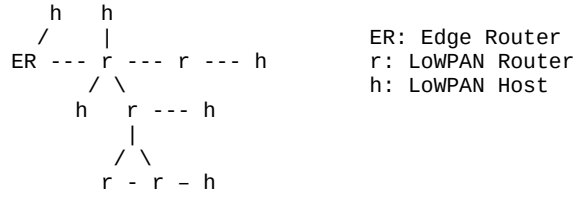

Fig.6 *route over* topology [4]

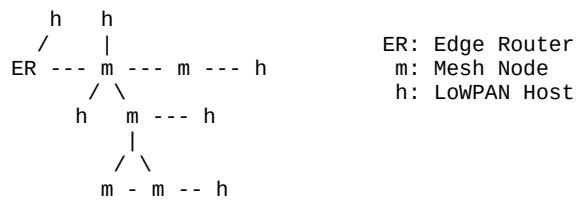

Fig.7 *mesh under* topology [4]

The use of mesh header introduces further overhead. If using IEEE 16-bit short addresses and a number of hops lower than 14, the mesh header length will be of 5 bytes. However, using a mesh header it is possible to compress the IPv6 addresses down to 0 bytes and to elide the hop limit field of the IPv6 header. In that way, the presence of mesh header does not increase the overhead.

IMPLEMENTATION

In this section we describe the implementation of *mesh under* and *route over* in a real 6LoWPAN network environment. We start presenting the protocol stack and the hardware used for our studies. Finally, we will give details on how tests have been done and the network topology used to realize them.

Protocol Stack

We adopted an open-source TinyOS based 6LoWPAN implementation developed by the University of California at Berkeley. It is called Berkeley IP (blip) [4]. Blip consists of TinyOS and ANSI-C code that implements the 6LoWPAN stack on the motes. A Unix daemon translates 6LoWPAN packets reaching the base station mote to IPv6 packets. IPv6 router advertisement messages are generated by standard daemon RADVD. As IPv6 header compression, it uses the 6LoWPAN standard compression HC1 [2].

Blip implements an hybrid routing protocol called HYDRO [5]. Routing decisions as well as packets forwarding are taken at network layer. However, fragmented packets are forwarded trough the destination without the hop-by-hop reassembling required by *route over*.

Blip has libraries to add *mesh under* header but it is not implemented any mesh handler routine. In this work we have developed proper code and modified some of the existing to give to blip the necessary *mesh under* and *route over* capabilities.

Hardware Platform

The Crossbow's TelosB mote is the hardware platform we used four our experiments [9]. It is an open source, low-power wireless sensor module. TelosB motes have a 16-bit RISC MCU at 8 MHz and 16 registers. The platform offers 10 kB of RAM, 48kB of flash memory and 16 kB of EEPROM. Requiring at least 1.8 V, it draws 1.8 mA in the active mode and 5.1 μA in the sleep mode. The MCU has an internal voltage reference and a temperature sensor. Further sensors available on the platform are a visible light sensor (Hamamatsu S1087), a visible to IR light sensor (Hamamatsu S1087-01) and a combined humidity and temperature sensor (Sensirion SHT11).

TelosB motes can be plugged via a USB port to a computer through which the motes can be programmed.

Test-bed

A performance analysis of *route over* and *mesh under* has been done taking into account the average end-to-end delay in packets transmission within the sensor network and the average round-trip delay time. Both tests have been done in a multi-hop network topology. The number of hops for end-to-end delay measurements ranges from a minimum of 2 to a maximum of 14. Regarding RTT, measurements have been done with a number of hop ranging from 2 to 5. The network topology was composed by a number of nodes varying according to the number of hops. In Fig.9 it is showed the topology for a 2 hops network. The network was composed by the followings elements:

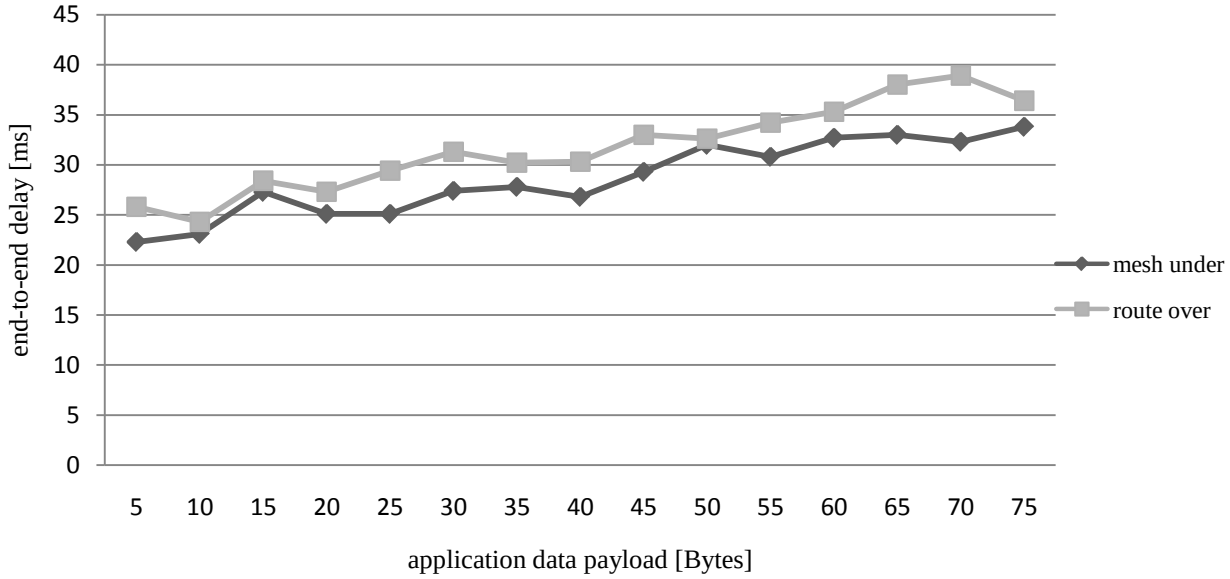

Fig.8 end-to-end delay variation according to application data payload

1) A border router called IP Base Station acting as destination node in end-to-end tests and source in case of round-trip time.
2) A sensor node that transmits UDP packets to the IP Base Station. This is considered the source node in end-to-end tests while it is the destination in case of round-trip time.
3) A variable number of relay nodes acting as packets forwarders.

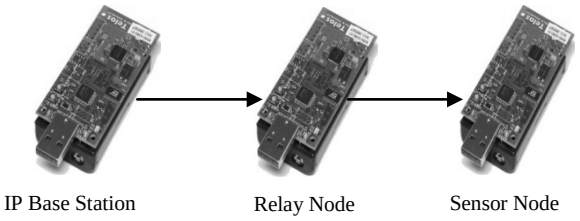

Fig.9 Network topology for a 2 hop scenario

RESULTS

End-to-end delay has been analysed considering two different situations. Firstly, we was interested in evaluating delay for *mesh under* and *route over* according to the application data payload while the number of hops was kept constant. Then, we have observed the delay evolution according to the number of hops keeping the application data payload fixed to a constant value.

In Fig.8 it is showed the end-to-end delay time (in ms) for a 6LoWPAN communication in a network with 4 hops and an application data payload ranging from 5 to 75 bytes, reaching the limit to do not fragment the packet. Nodes were at a constant distance of 25 cm from each other. This distance results enough to avoid direct connectivity between the base station and the destination node. To avoid that the blip routing protocol has influence on results, we used static routes to forward packets trough the network.

For each different application data payload, it has been taken 10 observations of nodes processing and propagation time. This number results to provide a reliable evaluation since we noticed that propagation and processing had no significant variations. The sum of the obtained times gives the delay for the observation. The mean value of the 10 observation taken for each payload size is the resulting end-to-end delay time.

As shown in Fig.8, *mesh under* improves the end-to-end delay trend. It can be explained considering that the processing time of the nodes is lower because the hop-by-hop decompression and compression of the IPv6 header is avoided. The transmission time has no influence on the enhancement of performances since it has the same value in both routing schemes. Considering the difference between the average end-to-end delay times obtained for *mesh under* and *route over*, the first one has an average improvement of 3,1 ms with a peak of 6,6 ms for 70 bytes and a minimum of 0,6 ms for 50 bytes of application data payload.

Fig.10 shows the average end-to-end delay evolution according to the number of hops between source and destination. The application data payload has been fixed to 75 bytes, which is the maximum we can have to avoid fragmentation. The average values from 2 to 5 hops have been obtained in the same way used for the previous delay computation through a real 6LoWPAN communication. The average end-to-end delay obtained from 6 to 14 hops has been obtained simulating a 6LoWPAN communication. In fact, we observed that the delay trend for both *mesh under* and *route over* followed a linear evolution:

$$y = n \times x \tag{1}$$

Considering that the transmission time was constant in each hop and the node processing time was independent from the number of hops, we can calculated the mean value of the processing time for each node. The function expressing the end-to-end delay is:

$$T = n \times (t_t + t_p) \tag{2}$$

Where T is the end-to-end delay, n the number of hops, t_t and t_p stands respectively for the transmission time and average node processing time.

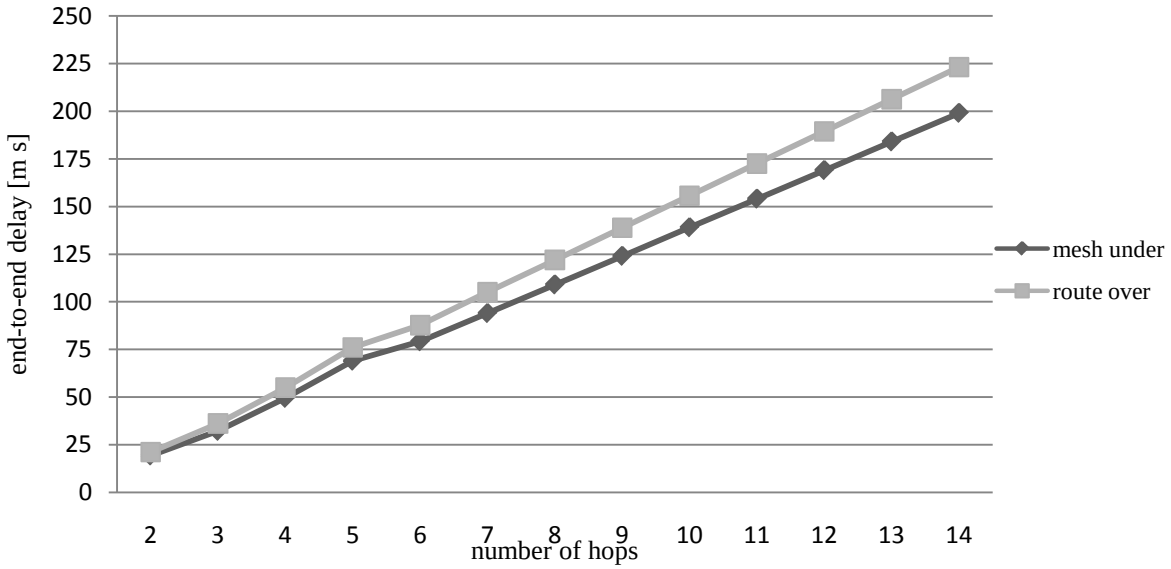

Fig.10 Throughput variation according to the number of hops

Mesh under outperforms *route over* also in this case. Increasing the number of hops the differences between them becomes bigger. For a 2 hops network, delay for *mesh under* is 1,9 ms lower respect to *route over*, considering 14 hops the difference between them is approximated to 24,05 ms. It can be estimated that the introduction of a new hop augment by 1,84 ms the differences between end-to-end delay for *mesh under* and *route over*. Differences in terms of delay between *mesh under* and *route over* are big as we could expect.

A further interesting analysis is in the differences in the time spent by a node to process and forward a packet. It has to be noticed that *mesh under* forwards packets from the adaptation layer avoiding the decompression and compression of the IPv6 packets done in *route over*. For *mesh under* the mean value of node processing time is 11 ms with a standard deviation of 2,1 ms. Regarding *route over* the node processing time is 12,85 ms with a standard deviation of 2,3 ms. The difference between the mean value of processing time of *mesh under* and *route over* gives an estimation of the time spent in both compression and decompression routines, that is 1,85 ms.

It has to be noticed that transmission and node processing time includes DIFS and SIFS time interval defined in the CSMA\CA mechanism used by LoWPAN. In [1] are defined minimum DIFS and SIFS period expressed in number of symbols. According to [10] for the 2.4 GHz band 2 symbols are correspondent to 32µs. DIFS is fixed to 40 symbols and it is equivalent to 640 µs while SIFS to 192 µs since it is set to 12 symbols. DIFS period has to be included in the node processing time while SIFS in the transmission time.

A further analysis has been done taking into account the round-trip time delay. Table 1 shows the results obtained for *mesh under* while Table 2 for *route over*. The round-trip time delay has been measured using the ping6 command. For each number of hops it has been sent 1000 packets with a payload size of 60 bytes. However, the performances obtained for end-to-end delay and round-trip-time are not comparable. In fact, end-to-end delay take in account only the time spent by the packet to reach the destination node without taking into account the processing time of the final node. Instead, in round-trip-time results it is included the time needed to process ping response and presents the results.

Number of hops	Average RTT (ms)	Max RTT (ms)	Min RTT (ms)	Standard deviation (ms)
2	117,72	131,01	103,53	5,78
3	148,83	168,07	129,51	7,03
4	175,68	192,07	152,09	7,83
5	202,51	224,1	181,12	8,41

Table 1: Round-Trip Time (RTT) statistics for mesh under

Number of hops	Average RTT (ms)	Max RTT (ms)	Min RTT (ms)	Standard deviation (ms)
2	118,47	133,03	104,9	6,03
3	149,39	184,09	131,03	7,53
4	178,82	245,04	164,06	8,34
5	213,91	252,12	190,12	9,37

Table 2: Round-Trip Time (RTT) statistics for route over

Comparing round-trip time statistics for *mesh under* and *route over*, it can be appreciated how *mesh under* outperforms *route over*. The lower time spent to forward the packet in a *mesh under* node respect to a *route over* one, is the key to keep latency low.

In a multi-hop scenario, the use of *mesh under* would decrease the node occupancy time having an important reflection in the node's energy consumption. In fact, less time a communication lasts and more energy can be saved by nodes.

CONCLUSION

The contribution of this paper has been the examination of *mesh under* and *route over* routing schemes in 6LoWPAN. Our attention has been focused on the end-to-end delay and round-trip time performance evaluation. All the tests have been done on a real 6LoWPAN network. A TinyOS based open-source implementation of 6LoWPAN protocol stack named blip has been used for our studies. We adapted it in order to implements correctly the mesh under capabilities and installed in TelosB motes. For our tests, we have considered only 6LoWPAN communications not requiring packet fragmentation. As expected, to forward packets from the adaptation layer instead of the network one has reduced the time spent in packet processing. This allowed *mesh under* to have better performance in both end-to-end delay and round-trip time. We have tested both routing techniques in a multi-hop network according to different size of application data payload and number of hops.

FUTURE WORK

The tests made in this work have been done in absence of packet fragmentation. We will include in future work the evaluation of both *mesh under* and *route over* in 6LoWPAN communications with fragmented packets. Besides the repetition of end-to-end delay and round-trip time performance evaluation, we will take in account also to test the energy consumption of both routing schemes. Future work will be based on the same 6LoWPAN protocol stack implementation and hardware used in this work.

ACKNOWLEDGEMENTS

This work has been supported by Spanish Government through project TEC2009-11453, and by the Catalan Government (Comissionat per a Universitat i Recerca del DIUE) and the Social European Budget.

REFERENCES

[1] IEEE Computer Society. IEEE Standard 802.15.4-2006. Part 15.4. Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs). 2006.

[2] Montenegro, G.; Kushalnagar, N.; Culler, D.E. Transmission of IPv6 Packets over IEEE 802.15.4 Networks. IETF RFC 4944. 2007

[3] Hui, J.W.; Culler, D.E. Extending IP to Low Power, Wireless Personal Area Networks. IEEE Internet Computing 2008, vol.12, no.4, 37-45.

[4] Kim, E.; Kaspar, D.; Gomez, C.; Bormann, C. Problem Statement and Requirements for 6LoWPAN Routing. draft-ietf-6lowpan-routing-requirements-04. July 2009.

[5] Chowdhury, A. H. et al. ”Route-over vs Mesh-under Routing in 6LoWPAN”, IWCMC '09, June 2009.

[6] Blip. Available online: http://smote.cs.berkeley.edu:8000/tracenv/wiki/blip (accessed on 30 June 2010).

[7] Tavakoli, A.; Dawson-Haggerty, S.; Hui, J.; Culler, D. HYDRO: A Hybrid Routing Protocol for Lossy and Low Power Networks. draft-tavakoli-hydro-01.

[8] 6loWPAN IETF Working Group. Web page: http://datatracker.ietf.org/wg/6lowpan/charter/ (accessed on 30 June 2010).

[9] Crossbow Technology Inc. TelosB Datasheet: http://www.willow.co.uk/TelosB_Datasheet.pdf (accessed on 30 June 2010).

[10] Scheers, B.; Mess, W.; Lauwens, B. Developments on an IEEE 802.15.4-based wireless sensor network. Journal of Telecommunications and Information Technology, pp. 46-53, 2008.