Integration of Wireless Sensor Networks to the Internet of Things using a 6LoWPAN Gateway

L.F. Schrickte, C. Montez and R. de Oliveira
Univ. Federal de Santa Catarina (UFSC)
Pós-grad. em Eng. de Automação e Sistemas (PGEAS)
lnando@gmail.com, (montez, romulo)@das.ufsc.br

Alex R. Pinto
Univ. Estadual Paulista Júlio de Mesquita Filho (UNESP)
Inst. de Bioc., Letras e Ciências Exatas (IBILCE)
arpinto@ibilce.unesp.br

***Abstract*—The Internet of Things is a new paradigm where smart embedded devices and systems are connected to the Internet. In this context, Wireless Sensor Networks (WSN) are becoming an important alternative for sensing and actuating critical applications like industrial automation, remote patient monitoring and domotics. The IEEE 802.15.4 protocol has been adopted as a standard for WSN and the 6LoWPAN protocol has been proposed to overcome the challenges of integrating WSN and Internet protocols. In this paper, the mechanisms of header compression and fragmentation of IPv6 datagrams proposed in the 6LoWPAN standard were evaluated through field experiments using a gateway prototype and IEEE 802.15.4 nodes.**

I. Introduction

The IEEE 802.15.4 standard defines the lower layers (PHY and MAC) of low-speed wireless local area networks (*Low Rate, Low Power Wireless Personal Area Networks* – LR-WPAN). ZigBee, Wireless HART and 6LoWPAN are some products that adopt this standard. Several topologies are supported – including cluster tree, star and mesh – connecting the network devices with a speed of up to 250 kbps. Due to these characteristics and the low-power consumption, the IEEE 802.15.4 can be considered a *de facto* standard for WSNs.

WSNs are composed of tiny nodes with processor, memory, sensors and battery, whose main purposes are sensing and transmitting data. Several kinds of devices can be used in a WSN like: robots, IP cameras and unmanned vehicles. However, in order to achieve an effective integration with the Internet, it is necessary the use of IP addressing.

The IPv4 has been gradually replaced by the IPv6. The main motivation for this change is the increasing need for a wider range of IP addresses, that are essential for the IoT. This way, 6LoWPAN has some mechanisms that can simplify its use in critical applications (for instance, the control and supervision of industrial processes), like QoS management, datagram fragmentation and header compression.

In this paper, prototypes of IEEE 802.15.4 nodes and a 6LoWPAN gateway were designed and implemented in an embedded system based on Contiki operating system. Several field experiments were performed and the results present evidence that the 6LoWPAN protocol can be used in resource constrained networks (e.g. WSN). The remaining of this paper is divided in the following sections. The IPv6 and 6LoWPAN are presented in Section 2. The prototype used in the tests is presented in Section 3. Finally, results and final remarks are discussed in sections 4 and 5.

II. IPv6 and 6LoWPAN

WSNs can be really useful if they are effectively integrated in the Internet. Scalars from the environment can be sensed and made available in timely fashion for decision making together with Internet-distributed data. However, IEEE 802.15.4 resource constraints make the IPv6 utilization difficult because:

- The IPv6 minimum MTU (Maximum Transmission Unit) size of 1280 bytes is much bigger than the maximum frame size of 127 bytes defined in IEEE 802.15.4 MAC specification.
- The excess of data in IPv6 header is a significant overhead for WSN nodes.
- IPv6 addresses use 128 bits while IEEE 802.15.4 addresses use 64 bits for full addresses and 16 bits for short addresses.
- The mechanisms of IPv6 require lots of processing which is not suitable for devices with severe constraints like WSN nodes.

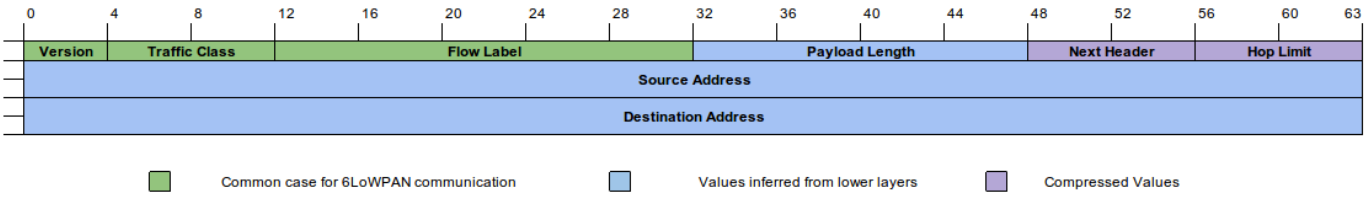

Fig. 1. IPv6 Header and 6LoWPAN compression.

- Routing approaches for mesh topologies are not defined in the IPv6 protocol.

The 6LoWPAN standard [2] addresses these issues defining objectives for the use of IPv6 over low power wireless networks (LoWPANs). The 6LoWPAN proposes an adaptation layer between the network and data link layers of OSI model, thus it makes possible the use of IPv6 in WSN nodes. The main advantage of this approach is that IEEE 802.15.4 nodes can be integrated with any IP network and Internet.

The adaptation layer that is specified in [1] proposes the following mechanisms:

- IPv6 header compression that reduces the header size of datagrams from 40 bytes up to 2 bytes. The data reduction is achieved by removing fields with fixed information on 6LoWPAN networks, by inferring data from lower layers and by narrowing some fields ranges.
- Header compression of higher layers (TCP, UDP and ICMP).
- IPv6 datagram fragmentation in order to send small frames compatible with IEEE 802.15.4 MAC.
- Inclusion of header and information that optimize IEEE 802.15.4 mesh and star topologies

III. Conversion between IPv6 and 6LoWPAN

The operations that convert IPv6 datagrams into 6LoWPAN are mainly based on the following services: (i) header compression, (ii) datagram fragmentation and (iii) stateless address autoconfiguration. This paper is focused on the first two services, which have direct influence over the gateway performance and its capacity of attending time critical requirements. Thus, these two services will be described.

A. Header compression

RFC4944 [1] defines the header compression algorithms LOWPAN_HC1 and LOWPAN_HC2. However, these algorithms became obsolete in RFC6282 [3], that specifies algorithms called LOWPAN_IPHC and LOWPAN_NHC. The headers defined in RFC6282 use a byte named *dispatch* that identifies the kind of compression used in the frame. Frames which are not compressed use a specific *dispatch byte* to identify it. For instance, the compression in HC1 takes this form:

```
+------------+----------+--------+
|HC1 Dispatch|HC1 Header|Payload |
+------------+----------+--------+
```

When the header is not compressed:

```
+-------------+-----------+--------+
|IPv6 Dispatch|IPv6 Header|Payload |
+-------------+-----------+--------+
```

In this paper, the compression modes specified in RFC6282 were used. Figure 1 shows the standard IPv6 header and the changes that can be promoted when the LOWPAN_IPHC compression of the adaptation layer of 6LoWPAN is used. Some parameters can be omitted and/or compressed, however the standard keeps the option of mantaining some of the parameters in their full form.

Figure 1 shows the best case, when the maximum compression is obtained. The IPHC compression uses the *dispatch byte* to identify the compression type. Moreover, the LOWPAN_IPHC compression can reduce the IPv6 header to just 2 bytes: one dispatch byte and another one with the compressed header information. The compression changes of the 6LoWPAN adaptation layer are presented in Table I.

The LOWPAN_NHC compression allows higher layer headers to be compressed too [3]. For instance, the ports

TABLE I. 6LOWPAN COMPRESSION.

Header field	IPv6 size	Changes when 6LoWPAN is used
Version	4 bits	Version is always 6
Traffic class	8 bits	Class is always 0
Flow label	20 bits	This value is always 0
Payload length	16 bits	Inferred from the data link layer or from the fragmentation header
Next header	8 bits	Compressed using LOWPAN_NHC
Hop limit	8 bits	This value is fixed when there are no intermediate nodes
Source address	128 bits	Inferred from the data link layer
Destination address	128 bits	Inferred from the data link layer

and checksum of UDP segments can be compressed, this way its header is reduced from 8 bytes to just 1 byte.

B. Fragmentation

The IPv6 minimum MTU size is 1280 bytes; that is, all links in the network must handle a datagram size of at least 1280 bytes. On the other hand, IEEE 802.15.4 networks present frame sizes up to 127 bytes. Therefore, in order to maintain the IPv6 compatibility, a gateway must be able to fragment IPv6 datagrams and to concatenate the packets received from a IEEE 802.15.4 network that are addressed to an IPv6 network.

Since an IPv6 datagram must often be fragmented in the data link layer [1], each data link frame has a fragmentation header. Thus, all fragments of the same datagram have parameters to specify the total size of fragment, an identification tag and an offset, which identifies the fragment position in the original datagram.

IV. PROTOTYPE DESCRIPTION

Following the Internet of Things paradigm, in this work we developed a system comprised of wireless sensor network devices connected to the Internet. In this sense, a prototype was specified and implemented with IEEE 802.15.4 nodes and a 6LoWPAN gateway that connects these nodes to the Internet through an Ethernet interface. Both nodes and gateway have a radio module in conformance with IEEE 802.15.4. The radio module that was developed is based on the Atmel AVR-Atmega128RFA1 component that has a 8 bits RISC microprocessor and a 2.4 GHz transceiver. This component was chosen due to its desirable characteristics [5]: low power consumption, reduced size and open source operating system availability.

Developed nodes average current consumption is 18.6 mA when transmitting and 4mA when idle. Its frequency clock is 16MHz, but it can be adjusted in order to achieve lower power consumption levels (1μA current level can be achieved when not transmitting).

The open source operating system Contiki, developed by the Computer Science Institute of Sweden [4], was used. Contiki has a low overhead and optimized implementation that allows the development of multi-task applications and enable the access to device drivers with few kbytes of memory. The reduced power consumption of ATmega128RFA1 and Contiki optimized code allow the use of low-power devices which can use wireless TCP/IP. Energy savings are achieved through, among other ways, low-power consumption modes and the possibility of using radio duty cycling mechanisms implemented in Contiki.

Contiki has some key features necessary for the gateway implementation, such as TCP, UDP and ICMP support and an implementation of the 6LoWPAN adaptation layer (including the header compression, address conversion and fragmentation algorithms). The TCP/IP stack of Contiki uses the uIP (or microIP) layer – the smaller TCP/IP stack validated by Cisco [6].

uIP is a complete TCP/IP implementation which is optimized for devices with constrained processors and small memory. The uIP version for IPv6, called uIPv6, was developed by Atmel, Cisco and Swedish Institute of Computer Science and it is considered the smallest implementation of the IPv6 protocol [6].

The usual protocols of WPANs, such as WirelessHART and Zigbee, have a proprietary code, which hinders interoperability with other devices and even the connection to devices on Ethernet networks. The uIPv6 stack in Contiki is an important asset to allow the aplicability of the Internet of Things concept, where any device can be part of an IPv6 network and therefore be connected to the Internet. In order to fulfill the idea behind the IoT concept, nodes developed must have a way to communicate with the Internet. In this sense, a gateway was developed by connecting one IEEE 802.15.4 node to an Ethernet enabled device.

Figure 2 shows the overview of the hardware developed for the gateway. It is out of scope of this paper to go into full detail on the hardware and software development. However, very briefly, it should be noted that the *Ethernet-IPv6 Interface Device* hardware was developed using a BeagleBone board (a low-power and low-cost open-source single-board hardware computer) and the software of the *WPAN Device* component was based on the Contiki's source-code named *Border-Router*.

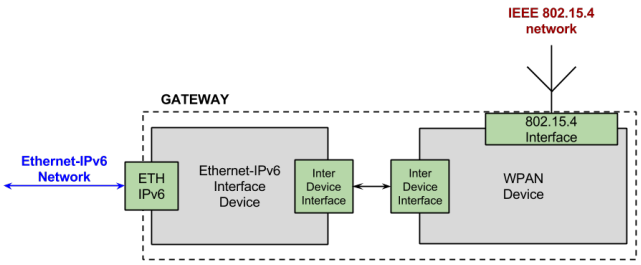

Fig. 2. Diagram of the gateway developed in this work.

The gateway hardware requires a more complex design than conventional nodes because IPv6 requires a larger amount of RAM flash memory for the code and further processing. In addition, the Ethernet physical interface itself also results in higher energy consumption.

V. Evaluation

Several experiments were conducted to evaluate the performance of the gateway. In this paper we present only results that reflect, directly or indirectly, the performance regarding the compression and the fragmentation obtained by the gateway. The main objective is to measure the performance improvement obtained by the use of 6LoWPAN compared with the use of the pure IPv6 specification.

In the first experiments we evaluated the impact of compression in IPHC communication, without participation of the gateway, between two of the developed nodes. Two IEEE 802.15.4 nodes were programmed to act as a client and a server communicating via UDP. The nodes were placed 3 feet away from each other. Periodically, the client node sends a packet to the server, which responds immediately with the same application data. The client node sends another package once it receives the response for the first message. If there was a timeout, the test was considered invalid (and this event stored for later error statistics).

The experiments were performed with different sizes of UDP segments, ranging from 20 bytes to 520 bytes of payload. We analyzed the load of each layer – MAC, IP, UDP and the payload in application layer – and the impact of the exchanged data, considering the sum of the transmission and reception of packets. Test results for uncompressed and compressed HC06 were compared.

In the context of this paper, no compression means an IPv6 datagram being sent with full header. It is important to notice, however, that fragmentation by the 6LoWPAN continues, because otherwise datagrams exceeding 56 bytes could not be sent. The difference between experiments with and without compression resides only in the way the data of each layer was sent in their respective headers.

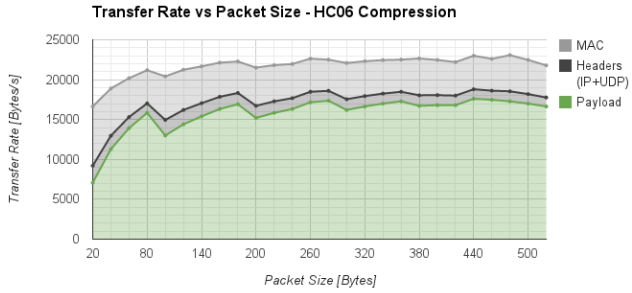

Fig. 3. Amount of data by layer – HC06 compression.

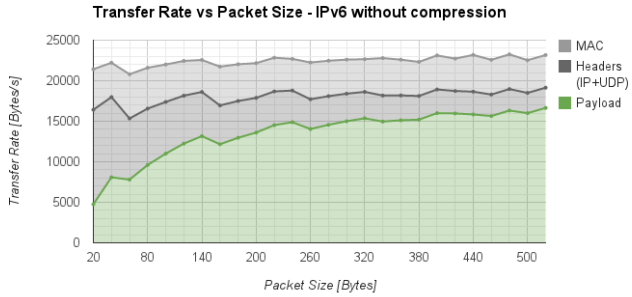

Fig. 4. Amount of data by layer – IPv6 compression.

As expected, the results in terms of total number of bytes were similar in both cases (Figures 3 and 4). The payload sent when using compression exceeds the amount of useful data sent when there is no compression. The overhead of headers in the absence of compression becomes less significant as the package size increases. This is because the compression happens in the main header of that package, which is only sent in the first fragment in the case of fragmentation. The remaining fragments have a constant size header, which are smaller than the full header of the first fragment and identical in both cases with and without compression.

Observing these results we can conclude that, in low speed networks and packets with only a few bytes of payload, performance is improved significantly with the use of header compression. In the transmission of 80-byte packets, for example, the speed of payload transmission with compression was approximately 15 kbps, whereas

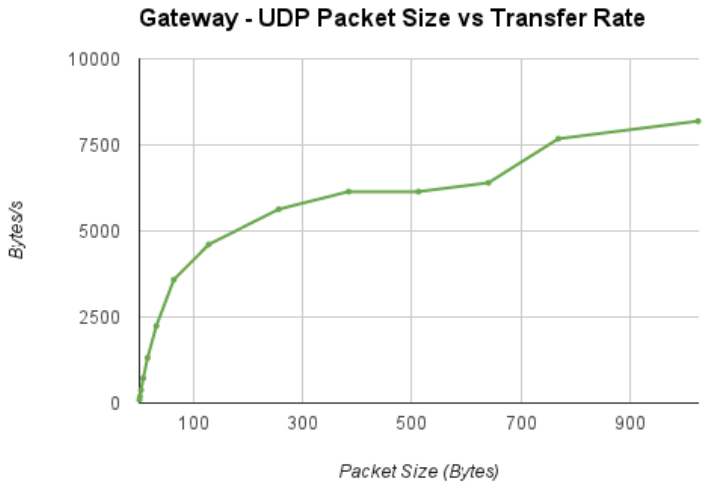

Fig. 5. Speed, considering the sum of transmission and reception data.

uncompressed speed was about 9.5 kbps. This observation corroborates the motivation for using 6LoWPAN over IEEE 802.15.4 networks.

There is another observation that shows we have a better performance when using compression with respect to fragmentation. The limit amount of payload when the package starts needing fragmentation in the MAC sublayer is substantially different when there is compression. This is because the header becomes much smaller, making this limit greater as the compression increases. The data obtained in the experiments show jumps in the number of bytes of the MAC header, which corresponds to the sending of an additional fragment. It may be noticed that, when compression is used, increases in the amount of fragments are performed with larger packages.

Another feature that can be noticed in the graphs is that the higher the amount of data per fragment, the higher the total byte transmission rate. This value tends to stabilize at 180 kbps, despite the IEEE 802.15.4 specifying transmission rates up to 250 kbps. This result is similar to that obtained by [7], who also used Contiki and justified the lower transmission rate in the channel saturation. The transmission rate of 250 kbps does not consider the use of CSMA/CA adopted in the tests. The dynamics of the Contiki operating system, as well as the implementation of the uIPv6 stack, the task management, among other factors, also influence the maximum rate achieved.

Figure 5 displays the results of the experiments designed to measure speed and transmission rate through the gateway developed. It is observed that the transmission rate increases as the size of the package increases (lower cost for transmission of the lower-layers headers). The speed stabilizes with packets larger than 1 KiB. Moreover, packet sizes greater than 1220 bytes are not allowed. That is because the limited resources of RAM, which restricts the size of receiving and transmitting buffers.

The aim of the last experiment was to evaluate the response times and compare the overheads with and without compression HC06, assuming a HTTP protocol at the application layer. Figure 6 presents the results obtained.

Small pages imply small payload resulting in greater effectiveness of compression on 6LoWPAN, as the overhead of IP header compression is great. In the absence of compression, much of the data is dedicated only to the IP header: more specifically 39.5% of the total data sent against 25% in the case of compression. For larger pages, the overhead difference is reduced. When data is compressed, about 7% of all bytes sent corresponds to the IP header; against 9.5% in case of no compression. In absolute terms, the amount of bytes exchanged when the page is 16 bytes long in 1221 bytes with compression and 1507 bytes without compression (a difference of 286 bytes). For a 100 KiB page, there are 145316 bytes exchanged when compression is used and 149579 bytes when compression is not used. That is about 4 KiB less when using compression.

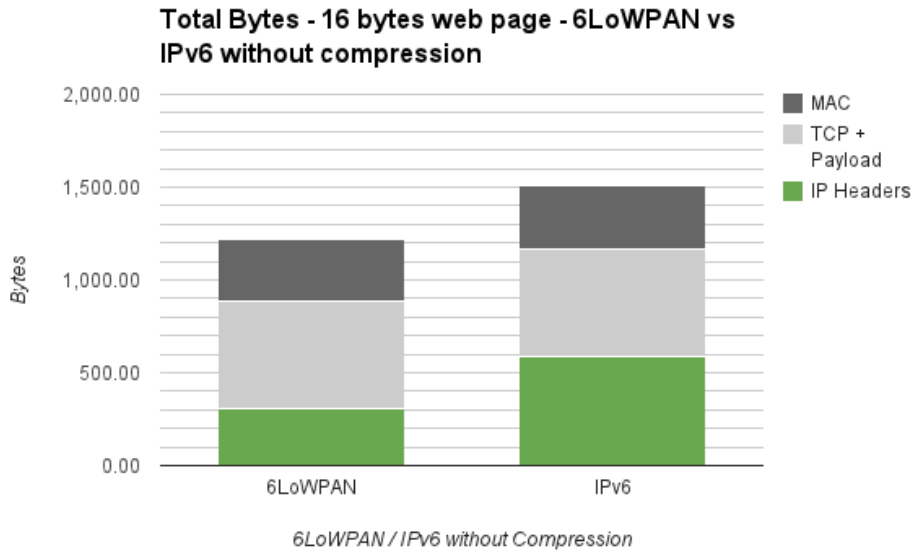

Fig. 6. Absolute amount in bytes for 16-bytes page being served.

In relative terms there is a higher gain when smaller pages are used – which can become an important gain in absolute terms if the requests for that page are frequent. Despite the lower impact with larger pages, saving the transmission of 4 KiB in the case of the payload of 102400 bytes is significant and justifies the use of 6LoWPAN. Pages with 100 KiB are impractical on devices with few resources. In order to run the experiment that page was divided into blocks of 1 KiB. In a real situation pages with such size could be, for example, generated with data collected in real time (once the largest buffer in RAM can not exceed 2 KiB).

VI. Conclusions

This paper presented the implementation of a 6LoWPAN nodes and gateway compatible with IEEE 802.15.4 standard. The prototypes were used on the analysis and evaluation of the 6LoWPAN specification effectiveness. The main purpose of the evaluation was to assess the performance of header compression and fragmentation algorithms of IPv6 datagrams proposed by the specifications of this new standard.

In terms of network performance, we observed the importance of header compression. Moreover, the results obtained can be used for the design and implementation of soft real-time applications with soft deadlines, integrated with the Internet and running on IEEE 802.15.4 WSN. We believe this kind of application will be ever growing with the increasing adoption of the concept of the Internet of Things.

Acknowledgments: The authors acknowledge the support received from their institutions, CNPq and INCT-SEC (CNPq 573963/2008-8 and FAPESP 08/57870-926). The authors would like to thank Adrián Fritz from Boreste Sistemas Embarcados for the support in the prototypes development and for his helpful suggestions.

References

[1] G. Montenegro and N. Kushalnagar and J. Hui and D. Culler, *Transmission of IPv6 Packets over IEEE 802.15.4 Networks*, Internet Engineering Task Force, Set. 2007.

[2] N. Kushalnagar and G. Montenegro and C. Schumacher, *IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement and Goals*, Internet Engineering Task Force, Internet Standard, Ago. 2007.

[3] J. Hui and P. Thubert, *Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks*, Internet Engineering Task Force, 2011.

[4] The Contiki Project and its contributors, *Contiki: The Open Source Operating System for the Internet*, http://www.contiki-os.org/.

[5] ATMEL, *ATmega128RFA1*, http://www.atmel.com/devices/atmega128rfa1.aspx.

[6] Adam Dunkels et al., *Making Sensor Networks IPv6 Ready*, SenSys'08, 2008.

[7] Muhammad Omer Farooq and Thomas Kunz, *Contiki-based IEEE 802.15.4 Nodeś Throughput and Wireless Channel Utilization Analysis*, Wireless Days (WD), IFIP, 2012, pp. 1-3.