The 4th IEEE International Workshop on Practical Issues In Building Sensor Network Applications (SenseApp 2009)
Zürich, Switzerland; 20-23 October 2009

Evaluation of IPv6 over Low-Power Wireless Personal Area Networks Implementations

Kevin Dominik Korte, Iyad Tumar, Jürgen Schönwälder
Computer Science
Jacobs University Bremen
Campus Ring 1, 28759 Bremen
{k.korte, i.tumar, j.schoenwaelder}@jacobs-university.de

***Abstract*—A common radio technology used in wireless sensor networks is the IEEE 802.15.4 standard. Wireless sensor networks have so far used custom, light-weight, and application specific network protocols on top of IEEE 802.15.4, often optimized for resource constrained low power hardware. In order to achieve interoperability at the network layer, the Internet Engineering Task Force published a standard for the transmission of IPv6 packets over IEEE 802.15.4 low-power wireless personal area networks, which provides an additional layer adapting the IPv6 network layer to the low power and resource constrained IEEE 802.15.4 link layer.**

This paper provides an evaluation of several independent implementations of the IPv6 over IEEE 802.15.4 standard. We describe the characteristics of the implementations and the protocol features supported. Our interoperability tests, carried out using two different hardware platforms, show that at least three implementations are to a large extend compatible with each other. At the end of the paper, we summarize recent work in the IETF to improve the IPv6 over IEEE 802.15.4 technology.

I. Introduction

Wireless sensor networks have so far used custom, light-weight, and application specific network protocols. Given the common presence of IEEE 802.15.4 radio interfaces [1]–[3] on wireless sensor network nodes (motes), it becomes interesting to connect motes directly to the global Internet using the IPv6 over Low-power Wireless Personal Area Networks (6LoWPAN) adaptation layer [4].

The 6LoWPAN standard [5] has been introduced to enable direct Internet connectivity to motes and to replace proprietary communication protocols like ZigBee [6], [7], which was developed after the end of the Smart Dust project [8]. The 6LoWPAN technology is also supposed to end the usage of unsuitable technologies such as Bluetooth on wireless sensor nodes. The requirements for 6LoWPAN vary with the application characteristics, ranging from being fast enough for the transporting of audio and video signals in near real time to being flexible enough to allow the construction of widely spread sensor networks that operate unattended over long periods of time.

During the development of the 6LoWPAN standard, challenges imposed by the resource restrictions of motes in terms of power supply, available memory and processing power, and the limitations of the IEEE 802.15.4 radio interface had to be addressed. As a consequence, the standard has to realize the necessary adaptations using very little memory while also limiting the computational complexity. At the same time, the 6LoWPAN standard has to provide a clean layer for interconnecting normal IPv6 networks and 6LoWPAN networks. This allows for the usage of already existing protocols and programs without the need of making these programs 6LoWPAN aware given that they are IPv6 aware [4].

After the publication of the 6LoWPAN standard, several implementations for different hardware platforms and different operating systems have been developed and some of them are already delivered to customers for production deployment. This raises the question whether these implementations meet the specification of the 6LoWPAN standard and whether they inter-operate with each other, the primary goal of the 6LoWPAN standard.

This paper presents an evaluation of several 6LoWPAN implementations based on an analysis of the documentations, an inspection of the source code (where available), and practical interoperability testing by sending 6LoWPAN packets through an IEEE 802.15.4 network constructed out of different hardware platforms. The recommendations as set out in the Internet draft "Interoperability Test for 6LoWPAN" [9] were considered as a first step to evaluate the interoperability. However, our evaluation goes beyond these suggested interoperability tests. We aim to reveal to which extend the tested implementations meet the 6LoWPAN specification and we list which parts of the 6LoWPAN standard are currently not covered by the tested implementations.

The rest of this paper is structured as follows. Section II provides a short introduction into the IEEE 802.15.4 technology and the 6LoWPAN standard. The 6LoWPAN implementations evaluated are described in Section III before Section IV introduces the hardware and the software setup used in our interoperability tests. Our test results are presented in Section V. Section VI reviews current work in the IETF community to enhance the 6LoWPAN standard. Related work is discussed in Section VII before the paper concludes in Section VIII.

II. Background

This section provides some background information on the IEEE 802.15.4 link layer technology and the IPv6 over Low-power Wireless Personal Area Networks (6LoWPAN) adaption layer.

978-1-4244-4487-8/09/$25.00 ©2009 IEEE

A. IEEE 802.15.4

The IEEE 802.15.4 standard is a low data rate, low power consumption, low cost, wireless networking protocol targeted towards automation and remote control applications [1]–[3]. It supports star and peer-to-peer network topologies with a CSMA-CA channel access mechanism with optional guaranteed time slots. The necessary coordination in a Personal Area Network (PAN) is provided by the PAN coordinator. A PAN coordinator is a Full Function Device (FFD), usually mains powered, implementing the full protocol set. Reduced Function Devices (RFDs) are simpler and support only a reduced protocol set.

octets: 2	1	0/2	0/2/8	0/2	0/2/8	variable	2
Frame control	Sequence number	Destination PAN identifier	Destination address	Source PAN identifier	Source address	Frame payload	Frame sequence check

Fig. 1. 802.15.4 frame format

The main characteristics of the IEEE 802.15.4 link layer are a small frame size (127 octets physical layer frame size, see Fig 1), support for 16-bit short and IEEE 64-bit extended MAC addresses and low bandwidth (data rates of 250 kbps, 40 kbps, 20 kbps for each of the physical layers, 2.4 GHz, 915 MHz, and 868 MHz, respectively).

B. IETF 6LoWPAN

The 6LoWPAN adaptation layer defined in RFC 4944 [5] allows to transport IPv6 [10] packets over 802.15.4 links. The 6LoWPAN adaptation layer is placed between the IEEE 802.15.4 link layer and the IPv6 network layer. All 6LoWPAN encapsulated datagrams are prefixed by an encapsulation header stack. Each header in the header stack starts with a header type field followed by zero or more header fields. The header type field starts with a zero bit and a one bit followed by the 6-bit dispatch selector as shown in Table II. Fig 2 shows the 6LoWPAN encoding of a plain IPv6 header and compressed IPv6 header.

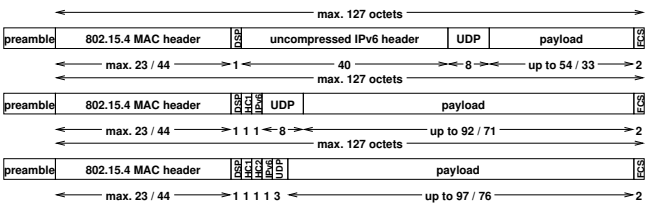


Fig. 2. Uncompressed and compressed 6LoWPAN frames

The dispatch selector is always the first header in a sequence of headers. The selector indicates whether a plain IPv6 datagram is carried in a 6LoWPAN message, whether an HC1 compressed IPv6 datagram is encapsulated and whether mesh routing headers are included. Mesh routing is used for IEEE 802.15.4 devices to increase their range by requiring FFDs to act as relays to transfer frames. The mesh header defines the final destination and the original source while the normal header is used for the current transmission endpoints.

TABLE I
LIST OF 6LOWPAN IMPLEMENTATIONS

Name	OS / License	Hardware	Maintained
Jacobs	TinyOS / 3BSD	Telos B, ...	no
Berkeley IP	TinyOS / 3BSD	Telos B, ...	active
Arch Rock	TinyOS / EULA	Raven, ...	active
SICSlowpan	Contiki / 3BSD	Raven, ...	active
Sensinode	Own / EULA	Sensinode	active
Hitachi	Own / EULA	Renesas	unknown

Apart from the limited range, the small size of IEEE 802.15.4 frames is a crucial issue. A normal IEEE 802.15.4 frame is only 127 octets at the physical layer while the maximum transmission unit of an IPv6 link is required to be at least 1280 bytes. To resolve this discrepancy, the 6LoWPAN standard defines a mechanism of fragmenting large packets into multiple frames.

Another approach for keeping the size of the IPv6 packets small is header compression. The 6LoWPAN standard defines a stateless header compression mechanism for link-local IPv6 addresses. This compression scheme is referred to as HC1 and compression is achieved by simply transferring the onset of the address and constructing the full address from both a node's address and the transferred address part. For higher layer protocols, a stateless compression scheme is defined for UDP (called HC2), which like HC1 reconstructs the UDP header from already known information.

III. 6LOWPAN IMPLEMENTATIONS

This section briefly introduces the 6LoWPAN implementations and their background. Table I provides an overview over the different implementations, the underlying operating system, the license, and the hardware restrictions. It also shows whether these implementation are being actively developed and maintained.

A. TinyOS

TinyOS is an open source real-time operating system developed by the TinyOS Alliance, which started as a cooperation of the University of California, Berkeley and Intel Research. TinyOS is written in the nesC language, a dialect of C and based on a single stack [11]. The non-blocking I/O is based on asynchronous callbacks, which are introduced by the compiler. This architecture requires programmers to write complex functions by fusing together small static event handlers [12]. There are three 6LoWPAN implementations for TinyOS:

1) Jacobs University Implementation: Matúš Harvan et al. [13] have implemented a 6LoWPAN implementation for TinyOS, which was the first TinyOS implementation freely available. This implementation consists of two parts. The first part is a collection of TinyOS event handlers implementing the 6LoWPAN functions for motes not attached to an IPv6 router. These event handlers are written in nesC and adapted to the limited space available on the motes. The second part is a daemon, which connects to a mote acting as a base station attached to a Linux system. The daemon translates 6LoWPAN

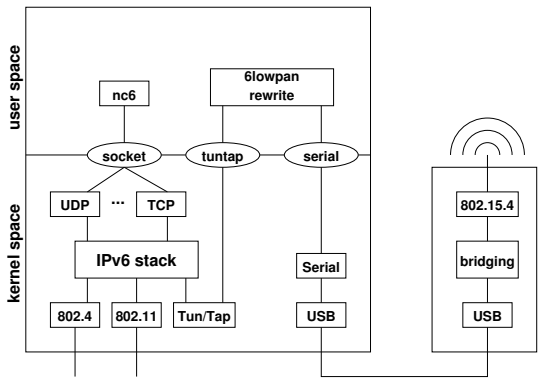


Fig. 3. Edge router with a 6lowpan user space implementation

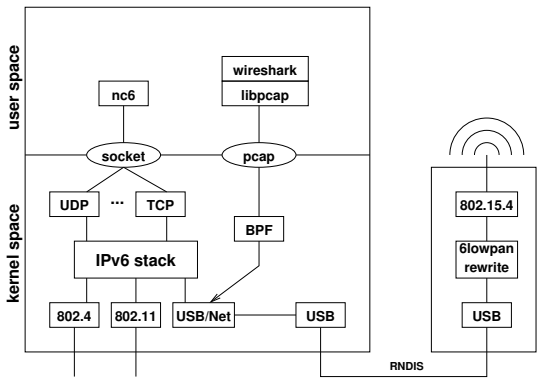


Fig. 4. Edge router with 6lowpan on a USB stick

frames exchanged over the serial interface into IPv6 packets and provides a normal network interface to the Linux kernel. The setup is illustrated in Fig 3.

2) University of California at Berkeley Implementation: The second freely available 6LoWPAN implementation for TinyOS was developed by the University of California at Berkeley and it is called Berkeley IP [14]. Like Jacobs University's implementation, Berkeley IP consists of TinyOS code implementing 6LoWPAN on the motes and a Unix daemon that translates 6LoWPAN packets exchanged via a base station mote connected via a serial interface to IPv6 packets. For IPv6 router advertisement messages, the system is relying on a standard router advertisement daemon (`radvd`).

3) Arch Rock: San Francisco based Arch Rock Corporation has developed a commercial 6LoWPAN implementation [15]. This implementation is shipped, for example, with the Atmel AVR Raven kit and it is provided in both binary and source form. The implementation has been reported to inter-operate with the Sensinode implementation and the Hitachi implementation (see below). The Arch Rock implementation is assumed to be a full implementation of the 6LoWPAN standard. The Arch Rock 6LoWPAN code consists of a 6LoWPAN part for a mote and a base station part for the USB stick. The toolkit also includes a Windows service daemon with a web interface for configuration and mote discovery.

B. Contiki and SICSlowpan

Contiki is a pre-emptive multithreading real-time open-source operating system for embedded systems [16]. Like TinyOS it is available under a BSD license. In contrast to TinyOS, Contiki has multithreading support using protothreads and supports dynamic loading of code. These two features allow the usage of Contiki for bigger programs and allow to load new program code without the need to completely reload the operating system.

The Contiki operating system features an IPv6 stack and a 6LoWPAN implementation. The IPv6 stack is marked with the IPv6 Ready logo and thus it is expected to follow the IPv6 standard closely. The 6LoWPAN code claims to implement the full 6LoWPAN standard. It consists of a program implementing 6LoWPAN on a mote and a program for a USB stick that can be directly attached to a host system and emulates a networking interface. Since the Contiki system in the USB stick emulates a networking interface and not a serial interface, no special daemon is needed on the host system to interface to the 6LoWPAN network. Another benefit of this approach is that it is possible to make 802.15.4 frames accessible to the host's kernel and as a consequence packet sniffers such as `tcpdump` or `wireshark` can be used to inspect and decode 6LoWPAN datagrams. Figure 4 illustrates this setup.

C. Sensinode

Sensinode is a company located in Finland. Their stated mission is "to revolutionize low-power wireless networking with IP technology" [17]. Sensinode provides a complete hardware and software solution, which includes an IPv6 stack that follows the 6LoWPAN standard. Due to the unavailability of the Sensinode 6LoWPAN implementation, it was not possible to test and evaluate it.

D. Hitachi, Ltd.

The Japanese company Hitachi Ltd. has developed its own 6LoWPAN implementation [18]. It is running on Renesas Technology motes [19]. The implementation has reported to be interoperable with the Arch Rock system. This implementation was unavailable as well and thus has not been tested.

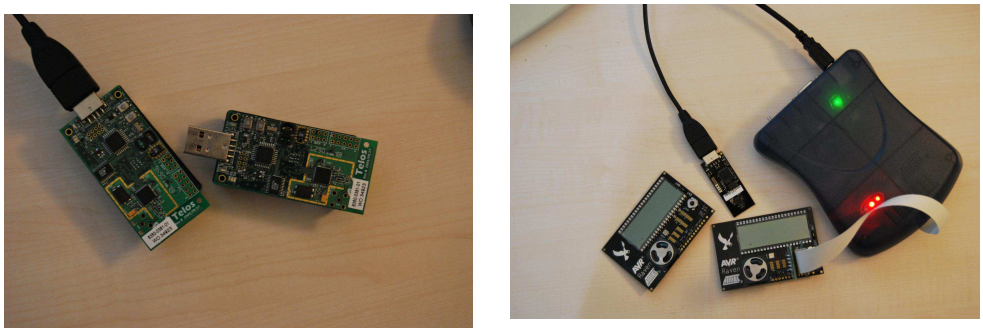
IV. Hardware and Setup

This section describes the setup and the hardware platforms used to evaluate of the 6LoWPAN implementations.

A. Telos B

The first platform that has been used for testing and evaluating the TinyOS implementations is the Telos B hardware [20], shown in Fig 5(a). Telos B is built around the MIPS420 micro-controller and is equipped with 10k of RAM and 48k of flash memory.

Telos B motes can be attached to a host computer via a Universal Serial Bus (USB) port acting as a serial interface through which the motes can be used and programmed. The recognition as a serial interface makes driver programming rather easy but puts most of the load of parsing the data into the host system.



(a) Telos B (b) ATMEL AVR Raven

Fig. 5. Telos B motes and ATMEL Raven development kit

B. Atmel AVR Raven

The AVR Raven produced by Atmel is another IEEE 802.15.4 device that has been used for testing an implementation running on the Contiki operating system. The Raven boards are equipped with two Atmel micro-controllers, a custom segmented LCD display, speakers, a microphone, sensors, and serial ports. The LCD display, which is rather uncommon for sensor motes, is segmented, which means it is addressed as a serial device rather than a frame buffer. This reduces the need for a graphics library to be included when using the display.

The demo kit we used contained two raven boards and a USB stick that is connected to the host operating system. In addition to the motes itself, a programming tool is needed as the Raven boards do not have a USB or serial interface for programming the two micro-controllers. While the motes and the USB stick can be programmed to act platform independent, the programming tool with its proprietary software requires a Microsoft Windows operating system. Fig 5(b) shows the Raven motes, the USB stick, and the programming tool.

C. Host System

The host system for all implementations was running on an VMware Player virtual machine. Using this approach, we were able to test different 6LoWPAN implementations without changing the host system and it was also possible to use multiple implementations at the same time [21].

For testing the TinyOS implementations, a fresh Kubuntu 8.10 virtual machine was used. A TinyOS system was taken directly from the CVS repository to have the latest version. The installation as such was fairly easy and the instructions provided were accurate and understandable.

The Atmel AVR software suite is currently only available for the Microsoft Windows operating system. Due to the Microsoft license conditions and the VMware Player features, an empty virtual machine for the Microsoft Windows operating system had to be created and Microsoft Windows had to be installed onto it. Due to the differences in the service packs and the updates, it was not easy to find a matching virtual machine. However, the installation of the Atmel software suite was fairly easy since the Atmel AVR manual provides a complete overview over the installation process.

For testing the Contiki implementation, some more steps were required since the Contiki operating system has to be compiled from the source code. To achieve this, a Cygwin installation including the GNU auto-tools and the WinAVR compiler are needed.

D. Mote Setup

The motes were flashed according to the instructions provided by the respective manuals, which were sufficiently written for all the implementations.

The connectors to the host system were connected to the respective host operating system and then the network initialized. Due to the small distance among the motes, there was no interference recorded on the incoming channel. The signal in general was sufficiently strong so that the testing could go along without any need of moving the setup or dramatic influences on the results.

V. Evaluation of the Implementations

This section discusses the parts of the 6LoWPAN standard tested and describes the evaluation process. Our evaluation is based on the available documentation and we used the hardware and the setup described in the previous section.

A. Underlying Components

The prerequisite for a standard compliant 6LoWPAN implementation is the usage of an IEEE 802.15.4 data link layer. Our test motes (Telos B, Atmel Raven) use different 802.15.4 transceiver chip-sets but we found that they interoperate with each other. However, it was necessary to pick a suitable channel. This turned out to be time consuming since the channel is not easy to configure on some implementations and not all combinations of channels and mote numbers did work. We ended up using channel 12 for our tests.

The TinyOS implementation developed at Jacobs University sends 6LoWPAN packets using TinyOS Active Messages (AM), also known as T-Frames in the TinyOS community. In other words, the implementation provides 6LoWPAN over AM over IEEE 802.15.4 instead of 6LoWPAN over IEEE 802.15.4.

TABLE II
SELECTED 6LOWPAN DISPATCH CODES [5]

Bit Pattern	Short Code	Description
00 xxxxxx	NALP	Not A LoWPAN Packet
01 000001	IPv6	uncompressed IPv6 addresses
01 000010	LOWPAN_HC1	LOWPAN_HC1 Compressed IPv6 header
01 010000	LOWPAN_BC0	LOWPAN_BC0 Broadcast header
01 111111	ESC	Additional Dispatch byte follows
10 xxxxxx	MESH	Mesh routing header
11 000xxx	FRAG1	Fragmentation header for the first packet
11 100xxx	FRAGN	Fragmentation header for the subsequent headers

Due to this extended header chain, an extra byte is added between the IEEE 802.15.4 MAC header and the 6LoWPAN header. This leads not only to the effect that the packets can not be read by other implementations, but it might also have an effect on other devices in the same channel. However, when the Jacobs University 6LoWPAN code was written, TinyOS did not provide any other way of sending messages [13].

The second TinyOS implementation, Berkeley IP, utilizes a new TinyOS frame type, which does not contain an active message field. However, Berkeley IP does not interoperable with classical TinyOS applications that use the old TinyOS frame formats (T-Frame, I-Frame), which contains the AM field. The Arch Rock implementation utilizes a different custom build networking stack avoiding the AM field. The Contiki operating system follows the IEEE 802.15.4 standard without any problem noticed.

B. Dispatch Selector

The dispatch selector in the 6LoWPAN header type field is used to determine whether an uncompressed IPv6 or a 6LoWPAN specific compressed header follows. This part of our evaluation relies mostly on practical tests since the documentation that was available does not provide any detailed information on how the implementations are reacting on different headers, especially the reaction on the NALP (Not A LoWPAN Packet) bit code. Apart from the NALP bit code, the headers shown in Table II were also tested.

After recording the 6LoWPAN packets and comparing them against the theoretical result, we found that all the implementations support the following headers: the dispatch header, the Not A LoWPAN Packet (NALP) header, the uncompressed IPv6 addresses header (IPv6), the compressed IPv6 header (LOWPAN_HC1), the broadcast header (LOWPAN_BC0), and the fragmentation headers. The mesh routing header is supported by the Berkeley IP, the Arch Rock, and the Contiki implementations. It is not possible to come to a clear conclusion concerning the ESC (Additional Dispatch byte follows) header since there are no programs using additional dispatch bytes at this time. It is not clear whether corresponding packets are discarded because of the first byte or because of the following bytes.

C. Mesh Routing

To evaluate the mesh routing, a practical test was done by distributing motes in different rooms in our college as shown in Fig 6. The distance of 50 meters inside a building is normally sufficient to not allow a packet to be transferred directly and hence a message must use intermediate nodes to reach the final destination. For this test, we did send ICMP echo requests over the network and then we waited for the reply. The destination was a mote running as a base station attached to a host computer so that we could record the incoming packets. To ensure that mesh routing was used, we turned off the intermediate node and we verified that the destination node was not reachable. From our experiment, we found that increasing the number of mesh routing hops by two will increase the drop rate by 20%. In general, our results show that full mesh routing is only supported by the Arch Rock, the Berkeley IP, and the Contiki implementations. While the Contiki system implements mesh routing capabilities completely, it seems to be a bit less reliable in the sense that we experienced a slightly higher loss rate.

Fig. 6. The positions of the motes for the mesh routing setup

D. Multicasting

Multicasting is used for neighborhood discovery, which is used for populating the routing table. The 6LoWPAN standard itself does not define a way how to implement multicasting for mesh routing; it only presents a number of ideas how a mesh routing multicast could be implemented. Some more recent ways of efficient neighborhood discovery are discussed in Section VI. Since the IEEE 802.15.4 standard does not support multicast but only broadcasts, all tested 6LoWPAN implementations did rely on broadcasts for local neighborhood discovery and for mesh neighborhood discovery.

E. Fragmentation

For an IPv6 link the maximum transmission unit (MTU) has to be at least 1280 bytes. The maximum packet size which

can be transferred over the IEEE 802.15.4 link is 127 bytes, excluding the MAC headers and optional encryption overhead. To be compliant with IPv6, the 6LoWPAN standard introduces a fragmentation and a reassembly mechanism to transfer an IPv6 packet in multiple IEEE 802.15.4 frames [5].

We found that all tested 6LoWPAN implementations support fragmentation and reassembly of IPv6 packets. However, all these implementations suffer from packet loss during the transfer process. To determine the packet loss, a number of random bytes that fill two frames has been sent over an IEEE 802.1.5.4 link. The observed packet loss ranges from 20% up to 60% and this value increases significantly for more transmitted frames. The question whether this high amount of packet loss is due to the hardware used or due to a problem of the underlying system could not be answered.

All the tested 6LoWPAN implementations have fragmentation and reassembly sufficiently well implemented to be able to transfer large IPv6 packets and to recombine the received fragments again into one IPv6 packet. However, due to the lack of reliability and the higher energy consumption, we recommend that fragmentation should be avoided whenever possible.

F. Header Compression

The IPv6 addresses contained in the IPv6 header consume most of the IPv6 header space. Hence, compressing the IPv6 address leaves more space for the payload of the IPv6 packet to be utilized. The 6LoWPAN standard currently defines a stateless header compression scheme for link-local addresses. This compression scheme is known as HC1 (LOWPAN_HC1) compression and is supported by all tested implementations. The HC2 compression scheme has been designed to compresses the headers of transport layer protocols (currently only UDP compression is defined). This header compression scheme is discussed further in Section VI.

The Arch Rock implementation supports the compression as defined in the "Stateless IPv6 Header Compression for Globally Routable Packets in 6LoWPAN Subnetwork" Internet-Draft [22], which enables the Arch Rock implementation to communicate using compressed global addresses. The same compression scheme is supported by the Contiki implementation, which achieves interoperability between Contiki and Arch Rock implementations using compressed globally routable packets.

G. ICMPv6

The Internet Control Message Protocol Version 6 (ICMPv6) [23] is used for sending error messages such as destination unreachable messages, and information messages such as the echo request, which is underlying the `ping` or `ping6` command provided by many operating systems.

The echo request and reply messages were tested by simply sending ICMPv6 packets using the `ping6` command on the host machine. As expected, all tested 6LoWPAN implementations support the ICMPv6 echo request and reply messages. Using the LCD and joystick of the AVR Raven board, the Contiki and the Arch Rock implementations were also successfully tested whether they support ICMPv6 echo request and reply messages exchanged directly between motes without the involvement of the host machine.

H. UDP support and HC2

Currently the HC2 compression scheme is only defined for UDP. Other transport protocols have not been tested since they can be sent in the data field, which is unaffected by the 6LoWPAN compression. The HC2 scheme works under the condition that the HC1 scheme defines the next header as a UDP. The compression is applied to the source and the destination ports, and to the length of the UDP segment as well.

From our tests, it seems only the Arch Rock implementation includes support for HC2 compression for UDP.

I. Interoperability Test

The last test of our evaluation was the interoperability test. To perform this test, a network of different implementations was created and IPv6 packets were sent over this network. The TinyOS implementation developed at Jacobs University was not used in this test since it uses a non-interoperable framing as explained in Section V-A. Hence, only the Berkeley IP, Contiki, and Arch Rock implementations have been tested (since Hitachi and Sensinode implementations were not available). We found that the three tested implementations work well together when they are used in the same network. The network worked fine when we were sending both fragmented and non-fragmented packets.

During the interoperability tests, the Contiki and the Arch Rock implementations showed a higher reliability (less packet loss) than the Berkeley TinyOS implementation. The Contiki and Arch Rock implementations both were running on the Atmel AVR Raven boards and the TinyOS Berkeley IP implementation was running on the Telos B motes. As mentioned before, we had to adjust the channels and the 6LoWPAN IDs and it took significant amounts of time to figure out which channels we have to use to run this test and which constraints exist on the 6LoWPAN IDs in various implementations. The Arch Rock implementation seems to dislike motes with an ID larger than the base station.

The full network was also reachable from the wired Ethernet network, which shows that the translation from 6LoWPAN compressed packets to regular IPv6 packets works flawless with the base station applications of all three tested implementations. Unfortunately, this test also showed a significant increase of the size of the 6LoWPAN packets due to the fact that global IPv6 addresses are not compressed. Solutions to this problems are discussed in Section VI-A.

J. Summary

Table III summarizes the features that are supported by the four different 6LoWPAN implementations we have tested. The table indicates whether a feature has been practically verified in our evaluation work or whether a feature seems to by supported but we could not test it.

TABLE III
IMPLEMENTED FEATURES: + MEANS SUPPORTED AND TESTED, O MEANS SUPPORTED BUT NOT TESTED, - MEANS NOT SUPPORTED

Feature	Jacobs	Berkeley	Contiki	Arch Rock
Dispatch Header	+	+	+	+
Dispatch Type	+	+	+	+
Mesh Header	-	+	+	+
Mesh Routing	-	see V-C	see V-C	+
Multicasting Header	-	+	+	+
Multicasting	+	+	+	+
Fragmentation	see V-E	see V-E	see V-E	see V-E
HC1	+	+	+	+
HC2 for UDP	-	-	-	+
HC1g	-	-	o	o
ICMPv6 Echo	+	+	+	+

VI. 6LoWPAN Improvements

In this section we summarize ongoing discussion in the IETF community on how to further develop and enhance the 6LoWPAN standard. Note that the 6LoWPAN work has seen a significant push in the IETF due to some large projects such as the Smart Grid initiative looking for protocols to support the large scale deployment of metering networks.

A. Compression Schemes

The current 6LoWPAN standard defines a stateless header compression scheme (HC1). In order to reduce the size of the 6LoWPAN header, the link-local address has to be compressed before sending the header over the IEEE 802.15.4 network. This compression scheme increases the space available for data in 6LoWPAN datagrams.

There is, however, currently no standard method for shortening a global address. Therefore, motes have to send complete IPv6 global addresses, which increases the header size significantly, may cause fragmentation, and reduces the effective use of the low bandwidth link. Furthermore, when using HC1 and HC2 together with a global address, the order of the headers (HC1, HC2, global address) will not be logical from a layering perspective. To overcome these problems, new compression forms have been proposed to enhance the 6LoWPAN standard.

The "Compression Format for IPv6 Datagrams in 6LoWPAN Networks" Internet-Draft [24] describes a method to compress a full header by introducing shared state between the sending mote and the receiving node that needs to expand the 6LoWPAN header to a full IPv6 header. The draft defines a method of shortening the global address and the messages needed to establish and maintain the shared state between the nodes. The draft also introduces a shortened form for the maximum number of hops that a packet is allowed to travel, which further reduces the size of the 6LoWPAN header in common cases. The ordering of the compressed headers is also addressed by moving the UDP header behind the global address prefix (HC1, global address, HC2) and thus creating an order that follows the protocol layers. However, while this document addresses one of the biggest shortcomings of the 6LoWPAN standard, the question remains whether the size of a message always justifies the establishment of common shared state between two nodes.

In contrast to the previous work, the "Stateless IPv6 Header Compression for Globally Routable Packets in 6LoWPAN Subnetwork" Internet-Draft [22] describes a stateless compression scheme for both, the IPv6 header and the higher protocols such as UDP, TCP, or ICMP. The draft follows a similar method as outlined in the 6LoWPAN standard for HC1. It enabled the compression of global IPv6 addresses into a smaller form without requiring shared state between the sending and receiving notes. However, a state-full compression can be more efficient in terms of header size reduction and the used processing cycles required, especially if multiple small packets are sent to the same destination.

B. Neighborhood Discovery

Neighboring discovery is one of the most energy consuming processes for nodes in 6LoWPAN networks, typically achieved by sending broadcast messages. To reduce the amount of broadcast messages needed for neighbor discovery, the "Neighbor Discovery for 6LoWPAN" Internet-Draft [25] allows 6LoWPAN routers to maintain a list of nodes and it defines a way to send this list to an individual node in order to update its list as well.

C. Mobility Support

The mobility support for IPv6 [26], [27] allows to move a running IPv6 devices from one network location to another without the need of remote systems to change their destination IPv6 address. However, this is only an option for devices with sufficient resources. Currently, the IPv6 mobility extension is not an option for battery powered 6LoWPAN enabled devices. However, one can consider to include network mobility support in the router connecting a regular IPv6 network to an IEEE 802.15.4 based network [28].

VII. Related Work

Pinedo-Frausto et al. [29] present a paper that intends to point out the features and the capabilities of Zigbee. They performed several experiments using commercially available Zigbee software and hardware in order to analyze and evaluate Zigbee networks and to show where Zigbee can be applied and where it is not suited for. The evaluation was based on the Zigbee Alliance interoperability testing specifications that meet the requirements of home automation applications and intend to achieve interoperability between devices from different vendors.

According to Arch Rock Corporation, the first interoperability test of the IETF 6LoWPAN standard had been done at Arch Rock's San Francisco headquarters. This test showed two independent 6LoWPAN implementations (Arch Rock and Sensinode) communicating with each other over a low-power wireless network [30]. Arch Rock, Hitachi, and Renesas Technology Corporation demonstrated the interoperability of their 6LoWPAN implementations at the 70th IETF meeting. According to Arch Rock, this test showed that the three 6LoWPAN implementations communicate with each other on the same local 6LoWPAN network and over the Internet with

other non-sensor IPv6 devices such as laptops through Ethernet or WiFi [31].

These previous 6LoWPAN interoperability tests are not documented in detail and were conducted by the implementers of the protocol stacks themselves. Our contribution is an independent evaluation with a more detailed description of the protocol features supported and the insights gained.

VIII. Conclusions

The results of our work show that three 6LoWPAN implementations (Arch Rock, Contiki, and the Berkeley IP) meet the 6LoWPAN standard specifications and pass the interoperability test. The implementation developed at Jacobs University could not be tested due to a different framing format that made it not interoperable with the other implementations.

All the 6LoWPAN implementations support fragmentation and reassembly. However, using fragmentation is not recommended because a lost frame causes the whole IP packet to be lost, requiring a full retransmission of the fragments making up the IP packet. Therefore, transport and application layer protocols should fragment the data because this allows to resend a lost small IPv6 packet while by using the 6LoWPAN fragmentation, it would only be possible to resend all fragments of a larger IPv6 packet.

Mesh routing works for Berkeley IP, Contiki, and Arch Rock. In most cases the packet arrives at the right destination within an acceptable time frame. However, the reliability of the system decreases when the number of hops increases. In general the suggestions by the "Routing Over Low Power and Lossy Networks" (Roll) working group of the IETF should be considered when looking for a possibility to improve mesh routing.

The evaluation of the multicasting and neighborhood discovery shows that the current implementations of neighborhood discovery for Arch Rock, Berkeley IP, and Contiki works fine when not considering energy consumption. Some more efficient and reliable methods for neighbor discovery are presented in Section VI.

This paper is a first step to investigate the fast evolving 6LoWPAN collection of protocols. Future studies may focus on performance aspects of 6LoWPAN implementation techniques, something we left out in this study. Another issue to look at in future studies is the usage and impact of link-layer security mechanisms versus the usage of end-to-end security mechanisms at the application layer. Finally, we need to study how existing Internet application protocols can be adapted to 6LoWPAN environments where energy conservation is a major aspect to consider for large scale deployments.

IX. Acknowledgement

The work reported in this paper is supported by the EC IST-EMANICS Network of Excellence (#26854).

References

[1] IEEE, "Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)," IEEE Computer Society, Tech. Rep., Sept. 2006.

[2] E. Callaway, P. Gorday, L. Hester, J. A. Gutierrez, M. Naeve, B. Heile, and V. Bahl, "Home Networking with IEEE 802.15.4: A Developing Standard for Low-Rate Wireless Personal Area Networks," *IEEE Communications Magazine*, vol. 40, no. 8, pp. 70–77, Aug. 2002.

[3] L. D. Nardis and M.-G. D. Benedetto, "Overview of the IEEE 802.15.4/4a standards for low data rate Wireless Personal Data Networks," in *Proc. of the 4th IEEE Workshop on Positioning, Navigation and Communication 2007 (WPNC'07)*. Hannover: IEEE, Mar. 2007.

[4] N. Kushalnagar, G. Montenegro, and C. Schumacher, "IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals," RFC 4919, Aug. 2007.

[5] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks," RFC 4944, Sept. 2007.

[6] H. Labiod, H. Afifi, and C. D. Santis, *Wi-Fi, Bluetooth, Zigbee and WiMax*. Springer Netherlands, 2003.

[7] S. Farahani, *ZigBee Wireless Networks and Transceivers*. Newnes, 2008.

[8] B. Warneke, M. Last, B. Liebowitz, and K. S. J. Pister, "Smart dust: Communicating with a cubic-millimeter computer," *Computer*, vol. 34, no. 1, pp. 44–51, Jan. 2001.

[9] J. Hui, Z. Shelby, and D. Culler, "Interoperability Test for 6LoWPAN," Arch Rock Corporation, Sensinode, Internet Draft <draft-hui-6lowpan-interop-00.txt>, July 2007.

[10] S. Deering and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification," RFC 2460, Dec. 1998.

[11] D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, and D. Culler, "The nesC Language: A Holistic Approach to Networked Embedded Systems," in *Proc. of the ACM Conference on Programming Language Design and Implementation (PLDI03)*. ACM, June 2003.

[12] L. Beniniand, M. Kandemir, and J. Ramanujam, *Compilers and Operating Systems for Low Power*. Springer, 2003.

[13] M. Harvan and J. Schönwälder, "TinyOS Motes on the Internet: IPv6 over 802.15.4 (6lowpan)," *PIK - Praxis der Informationsverarbeitung und Kommunikation*, vol. 31, no. 4, pp. 244–251, 2008.

[14] "Blib," http://smote.cs.berkeley.edu:8000/tracenv/wiki/blip, last access in May 2009.

[15] "Arch Rock," http://www.archrock.com/, last access in May 2009.

[16] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki – A Lightweight and Flexible Operating System for Tiny Networked Sensors," in *Proc. 29th IEEE International Conference on Local Computer Networks (LCN'04)*, 2004.

[17] "Sensinode," http://www.sensinode.com/, last access in May 2009.

[18] "Hitachi Global," http://www.hitachi.com/, last access in May 2009.

[19] "Renesas Technology Europe," http://eu.renesas.com/, last access in May 2009.

[20] J. Polastre, R. Szewczyk, and D. Culler, "Telos: Enabling Ultra-Low Power Wireless Research," in *Proc. Information Processing in Sensor Networks (IPSN 2005)*. IEEE, 2005, pp. 364–369.

[21] "VMware," http://www.vmware.com/, last access in May 2009.

[22] J. Hui and D. Culler, "Stateless IPv6 Header Compression for Globally Routable Packets in 6LoWPAN Subnetworks," Arch Rock Corporation, Internet Draft <draft-hui-6lowpan-hc1g-00>, June 2007.

[23] A. Conta and S. Deering, "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification," RFC 2463, Dec. 1998.

[24] J. Hui and P. Thubert, "Compression Format for IPv6 Datagrams in 6LoWPAN Networks," Arch Rock Corporation, Cisco, Internet Draft <draft-ietf-6lowpan-hc-04.txt>, Dec. 2008.

[25] Z. Shelby, P. Thubert, J. Hui, S. Chakrabarti, and E. Nordmark, "6LoWPAN Neighbor Discovery," Sensinode, Cisco, Arch Rock, IP Infusion, Sun, Internet Draft <draft-ietf-6lowpan-nd-04.txt>, July 2009.

[26] D. Johnson, C. Perkins, and J. Arkko, "Mobility Support in IPv6," RFC 3775, June 2004.

[27] Y. Mun and H. K. Lee, *Understanding IPv6*. Springer US, 2005.

[28] V. Devarapalli, R. Wakikawa, A. Petrescu, and P. Thubert, "Network Mobility (NEMO) Basic Support Protocol," RFC 3963, Jan. 2005.

[29] E. D. Pinedo-Frausto and J. A. Garcia-Macias, "An Experimental Analysis of Zigbee Networks," in *Proceeding of the 33rd IEEE Conference on Local Computer Networks (LCN)*, Oct. 2008, pp. 723–729.

[30] "Arch Rock, Sensinode Conduct First Interoperability Test of IETF 6LoWPAN Standard," http://www.industrial-embedded.com/news/db/?7426, last access in May 2009.

[31] "Arch Rock and Hitachi Demo 6LoWPAN Interoperability with Renesas at December IETF Meeting," http://www.industrial-embedded.com/news/db/?9560, last access in May 2009.