# A 6lowpan Implementation for TinyOS 2.0

Matúš Harvan
Computer Science
Jacobs University Bremen
Campus Ring 1
28759 Bremen, Germany
m.harvan@jacobs-university.de

Jürgen Schönwälder
Computer Science
Jacobs University Bremen
Campus Ring 1
28759 Bremen, Germany
j.schoenwaelder@jacobs-university.de

## 1. INTRODUCTION

Traditionally, wireless sensor networks have used custom, light-weight network protocols such as Active Messages. However, given the common presence of an 802.15.4 radio interface [4] on the motes and the 6lowpan adaptation layer [6] allowing the exchange of IPv6 packets over 802.15.4 links, enabling IPv6 connectivity in wireless sensor networks and connecting them to the global Internet becomes feasible. By natively supporting the IPv6 protocol, these devices would become first-class Internet citizens capable of communication with any other IPv6-enabled host and benefit from the standardized and already estabilished technology and a plethora of readily available applications. To this end a 6lowpan/IPv6 stack [3] has been implemented for TinyOS 2.0 [2], an embedded operating system commonly used in wireless sensor networks.

The rest of this document is structured as follows. The 6lowpan adaptation layer is briefly introduced in Section 2, the implementation is discussed in Section 3, related work is described in Section 4 and the document concludes in Section 5.

## 2. 6LOWPAN

The 6lowpan adaptation layer allows to transport IPv6 packets over 802.15.4 links. To meet the IPv6-required MTU of at least 1280 bytes with the 802.15.4 layer offering at most 102 bytes of payload per frame, a fragmentation mechanism below the IP layer is specified using an optional Fragmentation Header before the actual IPv6 header. Support for mesh networking is provided by the optional Mesh Addressing and Broadcast Headers. The former carries the Originator and Final Destination link-layer addresses while the latter contains a sequence number used to detect duplicated frames. Both are carried at the beginning of the 802.15.4 payload. Furthermore, mechanisms for compressing the IPv6 header from 40 down to 2 bytes and the UDP header from 8 down to 4 bytes, in the ideal case, are specified. To distinguish between a compressed and uncompressed header, a 1-byte dispatch value is prepended before the header. The optional 6lowpan headers mentioned earlier also start with a dispatch value allowing the recipient to determine what types of headers are present. The format of the 6lowpan adaptation layer is specified in [6].

## 3. IMPLEMENTATION

The goal for the implementation was to support replying to an ICMP echo request message (ping) and exchanging of UDP datagrams. Only the bare minimum necessary for meeting that goal was implemented.

The main restriction of the implementation was the amount of RAM available on the target platform, i.e., 4 KB on the MicaZ. Although aiming for an embedded implementation, easily readable and maintainable code was preferred over optimizing to squeeze into the least possible amount of memory at the cost of hard to understand programming constructs, hacks or munging of code into a few large functions for saving space on the stack. Each network layer and protocol are handled by a separate function. This allows to easily add more functionality in the future.

### 3.1 6lowpan for Linux

Most PCs today do not have an 802.15.4 interface and common operating systems such as Linux or the BSDs do not include a 6lowpan implementation. To allow for exchanging packets between the motes and a Linux PC, a tunneling daemon has been developed to use a mote as an 802.15.4 interface. The scenario is shown in Figure 1.

The mote runs the TinyOS sample application *BaseStationCC2420* forwarding traffic between the 802.15.4 and the USB interface of a mote. This mote is connected to the PC via the USB interface.

The translating daemon on the PC is a C program exchanging packets between the USB interface and a tun interface. The latter is a virtual network interface allowing a user space process to read and write packets to it. The daemon decapsulates the 6lowpan-encapsulated IPv6 packets received from the mote and 6lowpan-encapsulates the plain IPv6 packets received on the tun interface. This allows to use standard IPv6 applications on Linux for communication with the motes without modifying the Linux kernel. Furthermore, by enabling IPv6 forwarding on the PC, the motes can be connected to the global Internet.

### 3.2 Evaluation

The implementation was tested using a scenario as shown in Figure 1. A TelosB mote with the BaseStationCC2420 application was connected to a Linux PC running the translating daemon. Two other TelosB motes and a MicaZ mote were flashed with the 6lowpan implementation. The motes were

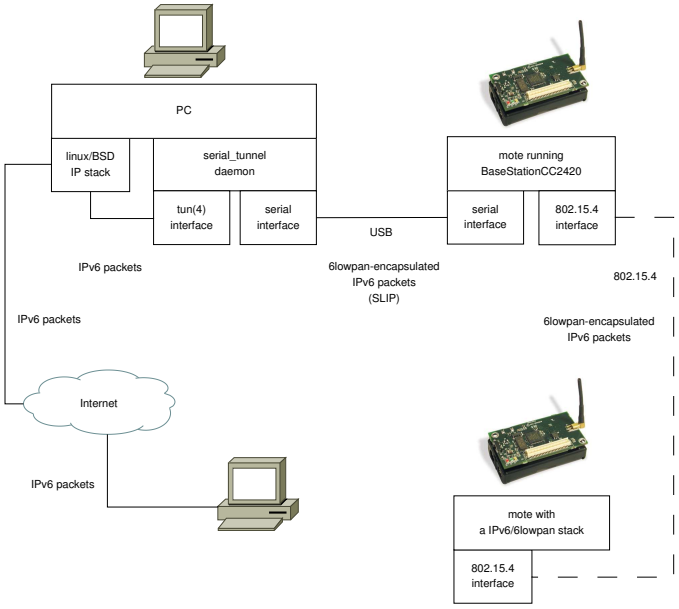


**Figure 1: The motes and the Internet**

successfully replying to ICMP echo requests initiated from the PC as well as exchanging UDP datagrams. Both short unfragmented packets as well as large, fragmented packets of size up to 1280 bytes were successfully exchanged.

The main limitation to interoperability with other 6lowpan implementations is the absence of a proper 802.15.4 stack in TinyOS 2.0. Although the implementation supports the ICMP echo mechanism and the UDP protocol, many features required for IPv6 implementations are missing. Among others, the Neighbor Discovery has not been implemented and packets are broadcasted on the link-layer, IPv6 extensions headers are not processed, IPv6 fragmentation is not supported and ICMP error messages are not generated. While many of these could be added, it is unclear whether they make sense in an embedded system. For example, is one willing to trade decreased battery life for regular neighbor advertisements or neighbor unreachability detection? Or if an error is encountered while processing a received packet, should a 1280 bytes long ICMP error message be sent back? Should one be sent back at all?

# 4. RELATED WORK

Several 6lowpan implementations for wireless sensor networks have been announced while this project was in progress.

The *Arch Rock* company has announced a commercial 6lowpan implementation *Primer Pack/IP* in March 2007. As this is a commercial implementation, technical information is scarce.

The *Sensinode* company has released a GPL-licensed 6lowpan implementation called *NanoStack v0.9.4* in April 2007. It is claimed to be up to date with version 12 of the 6lowpan format draft and to include IEEE 802.15.4 Beacon-mode support. The source code, however, does not seem to include 6lowpan fragmentation support and UDP checksumming.

uIP[1] is a TCP/IPv4 stack written by Adam Dunkels. It runs on 8-bit controllers with a few hundred bytes of RAM. It has been ported to TinyOS 1.1 by Andrew Christian from the Hewlett-Packard Company. It is available in the `tinyos-1.x/contrib/handhelds/tos/lib/UIP/` directory of the TinyOS 1.1 distribution.

While an IP stack can be implemented on the motes, it is also possible to use a proxy-based scheme. In this case a special proxy server is employed as a gateway separating the sensor network and the IP network. This allows to freely choose the communication protocol used within the sensor network. Although limited to IPv4, the Sensor Internet Protocol (SIP) [5] is an example of such a proxy scheme.

# 5. CONCLUSION

A 6lowpan/IPv6 stack has been implemented for the TinyOS 2.0 operating system and was tested on the TelosB and MicaZ hardware platforms. Using the translating daemon on the PC and a mote as the base station, it is possible to exchange IPv6 packets between the motes and a PC without an 802.15.4 interface. In case IP forwarding is set up on the PC and a properly assigned and routable global IPv6 prefix is used, the motes can be connected to the global Internet. More information about the implementation can be found in [3] and it can be downloaded from [7]. Furthermore, discussions have started to include it in the TinyOS distribution.

As future work it would be useful to add a proper 802.15.4 stack. Instead of broadcasting on the link-layer, Neighbor Discovery could be implemented. Various mesh routing algorithms could be investigated and the Mesh Addressing and Broadcast Headers could be used for mesh networking. Finally, SNMP could be implemented on top of the 6lowpan stack for collecting sensor values.

# 6. REFERENCES

[1] A. Dunkels. Full TCP/IP for 8-bit architectures. In *Proceedings of The First International Conference on Mobile Systems, Applications, and Services (MOBISYS '03)*, May 2003.

[2] 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 *PLDI03*. ACM, June 2003.

[3] M. Harvan. Connecting Wireless Sensor Networks to the Internet - a 6lowpan Implementation for TinyOS 2.0. Master's thesis, School of Engineering and Science, Jacobs University Bremen, May 2007.

[4] IEEE. IEEE Std. 802.15.4-2003, Oct. 2003.

[5] X. Luo, K. Zheng, Y. Pan, and Z. Wu. A TCP/IP implementation for wireless sensor networks. In *IEEE International Conference on Systems, Man, and Cybernetics*, 2004.

[6] G. Montenegro, N. Kushalnagar, D. E. Culler, and J. W. Hui. Transmission of IPv6 Packets over IEEE 802.15.4 Networks. Internet-Draft Version 13, IETF, April 2007. Work in progress.

[7] `http://www.eecs.iu-bremen.de/users/harvan/files/6lowpan.tar.gz`.