2009 Third International Conference on Sensor Technologies and Applications

# Comparisons of 6LoWPAN Implementations on Wireless Sensor Networks

Yannis Mazzer, Bernard Tourancheau
*LIP UMR 5668 of CNRS-ENS-INRIA-Université Lyon1*
*Lyon, France*
*Firstname.Lastname@ENS-lyon.fr*

## Abstract

*This paper introduces our work on the communication stack of wireless sensor networks. We present the IPv6 approach for wireless sensor networks called 6LoWPAN in its IETF charter. We then compare the different implementations of 6LoWPAN subsets for several sensor nodes platforms. We present our approach for the 6LoWPAN implementation which aims to preserve the advantages of modularity while keeping a small memory footprint and a good efficiency.*

## Index Terms

*Wireless Sensor Network, Communication Stack for Sensor, 6LoWPAN, IPv6.*

## 1. Introduction

Sensor network is a growing technology which aims to monitor the environment and possibly interact with remote device in the field. The distributed data sampling can be realized over a large area by spreading sensors. During the last few years the wireless technology has been the new communication mean of this kind of network. However in order to comply with the current and future standards and to ensure the accessibility of each node of a sensor network, the use of IP and especially IPv6 seems unavoidable. Since a node has to be economical, the power calculation and the storage are limited. For long, this limitation has been said to be too much for classical IP stacks. In order to overcome this, a new protocol definition, called 6LoWPAN (IPv6 over Low-Power Wireless Personal Area Networks), has been designed. This protocol enables all the capabilities of IPv6 on a very constraint node and thus opens the gate to the Internet of things.

This paper introduces different aspects of this networking problematic on wireless sensor networks (WSN). After the short presentation of 6LoWPAN, we introduce the sensor node typical hardware. We then describe a widely used operating system especially designed for sensor nodes and compare it to its concurrents. The associated 6LoWPAN implementations are comparatively introduced with emphasis on the main implementation issues. We finally discuss our approach choices for the design of a modular 6LoWPAN stack for sensor network.

## 2. Presentation of 6LoWPAN

6LoWPAN [1][2] enables all the capabilities of IPv6 on WSN. This standard access to the IP world opens the usage of WSN in the field of, for instance, distributed computation with the classical Message Passing Interfaces [3], the implementation of sensing device on-line control [4] and tools for the user interfaces [5].

The adaption of IPv6 to constrained devices starts by compressing the long IPv6 headers to 6 bytes, taking into account the link-local informations. The routing is adapted to the hop-by-hop "meshed" point of view of the WSN. The routing main capabilities are placed at the border routers for each node to have limited routing tables.

## 3. Hardware

Nowadays, different platforms have been developed from little micro-controllers (based on Atmel chips) to almost full-featured computers (ARM based). The Figure 2 describes well-known platforms. The Mica mote is based on the Atmel Atmega chips with an optional external antenna and very limited capabilities. The Telos is based on the Texas Instrument MSP430 chipset which integrates a 802.15.4 Chicon radio, it has more resources than the Mica node. The ARM based LiveNode was designed for medical purposes with dedicated sensors. Finally, the Sun Microsystems SunSPOT is the most powerful node, running a Java Runtime Environment subset.

The main issue with wireless sensor network nodes is the fact that they are running on battery or limited power sources. This means that they have a limited lifetime, and that any energy loss is critical. For instance, the wireless communications have to be treated carefully in order to avoid overpowering the radio or use all duty cycles.

## 4. Presentation of TinyOS and tools

TinyOS is an open-source event-driven operating system developed at UC Berkeley [11] for very low power devices

978-0-7695-3669-9/09 $25.00 © 2009 IEEE
DOI 10.1109/SENSORCOMM.2009.111

| | Mica [6] | Telos [7] | SunSPOT [8] |
|---|---|---|---|
| Controller | AVR atmega88 | TI MSP430 | ARM 920T |
| Bus | 8-bit | 16-bit | 32-bit |
| Memory | 128k | 10k | 512k |
| Flash | 512k | 48k | 4M |
| Radio Chip | variable | TI CC2420 | TI CC2420 |
| Sensors | | humidity, temperature, luminosity | accelerometer, temperature, luminosity |
| Connectivity | | 16 ports | 13 ports |
| Antenna | external | onboard | external |

Figure 1. Comparison of existing hardware

| | Sensinode [9] | LiveNode [10] |
|---|---|---|
| Controller | TI 8051 | ARM 7 |
| Bus | 8-bit | 32-bit |
| Memory | 8k | 64k |
| Flash | 128k | 256k |
| Radio Chip | TI CC2420 | Xbee Pro |
| Sensors | temp, luminosity | n.c. |
| Connectivity | 21 ports | 11 ports |
| Antenna | onboard | external |

Figure 2. Comparison of existing hardware (continuing)

with limited battery power supply. It has been ported on several hardware platforms and the latest version supports the PAN 802.15.4 MAC protocol[12]. Furthermore, it is component-based so that the compiled BLOB (Binary Large OBject) remains as small as possible, whatever the host architecture is. One of the particularities of this operating system is the Active Messages based networking which gives a sort of RPC (Remote Procedure Call) view of the communications.

TinyOS was designed for WSN, but several other Operating Systems (OS) exist for microcontrolers which give different approaches of the sensor networking issue. The Figure 4 briefly compares those OSes with their respective options and characteristics. The last column tells if there is any available 6LoWPAN implementation.

| | TinyOS [11] | Contiki [13] | FreeRTOS [14] | Mantis [15] |
|---|---|---|---|---|
| Licence | BSD | BSD | mod GPL | BSD |
| Schedul. | run2comp FIFO ev preemp | run2comp threads | RT | t-slot r-rob prio opt sleep m-thread |
| Footprint | 0.4ko | 20ko | 256ko | 0.5ko |
| 6lowpan | Yes | Yes | Yes | No |

Figure 3. Comparison of existing OSes

| | BTnut [16] | SOS [17] | NanoRK [9] | Dream [10] |
|---|---|---|---|---|
| License | BSD | mod BSD | Qt-like | n.c. |
| Schedul. | m-thread | kern mod hot-plug | RT | ev-driv m-thread |
| Footprint | n.c. | 1.5ko | 2ko | 3.5ko |
| 6lowpan | No | No | No | No |

Figure 4. Comparison of existing OSes (continuing)

## 5. 6LoWPAN for WSN

The hardware limitations of WSN nodes led to the design of new PHY and MAC protocol layers, see [12]. However, these limitations did not preclude IPv6 developments as it was shown in [18]. Most of the proposed solutions for IPv6 were settled down in the 6LoWPAN RFCs.

### 5.1. 6LoWPAN stacks

The Figure 5 describes the enabled capabilities of each existing open-source 6LoWPAN stack. The first part of the table describes general characteristics, such as supporting OS, License and state of development. The second part lists the implemented protocols, such as UDP (User Datagram Protocol), TCP (Transmission Control Protocol), AM (Active Message) and ICMP (Internet Control Message Protocol). The main issue with all existing 6LoWPAN stacks is their monolithic approach which may lead to efficiency but does not exploit all the capabilities of their respective hosts.

| Concept | Matus [19] | blip [20] | $\eta$stack [9] | $\mu$IPv6 [21] |
|---|---|---|---|---|
| State | Unachiev | Compl. | Compl. | In dev. |
| Support OS | TinyOS AM | TinyOS | FreeRTOS | Contiki |
| License | | LGPL | GPL | BSD |
| AM | X | | | |
| UDP | X | X | X | X |
| TCP | | | | X |
| ICMP | X | X | X | X |
| Mesh-Rout. | | X | X | |
| Frag. | X | X | X | X |
| Compress. | X | X | X | X |
| Broad. BC0 | X | X | X | X |
| Neigh. Disc. | | X | | X |
| Radio chip | CC2420 | CC2430 | CC2430 | CC2430 |
| Unix tools | X | X | X | X |
| Monolithic | X | X | X | X |
| Start | 2007 | 2000-08 | 2006 | 2008 |

Figure 5. Comparison of open-source 6LoWPAN stacks

### 5.2. Our 6LoWPAN implementation aim

As explained in Section 5.1, the existing 6LoWPAN stacks are monolithic softwares and this does not allow for an easy modification of the networking strategies. In order to exploit all the capabilities of the TinyOS operating system and to be able to evolve easily in our networking choices, we developed a modular stack.

Notice that programming microcontrollers at the system level is not an easy task. The implementation challenges are numerous because our aims are somehow conflicting. The modular approach increases the code size, may slow down execution and impact the power supply lifetime.

**5.2.1. The dispatch function.** In order to provide a treatable frame to the third ISO layer, the hardware automatically unpacks and stores in a memory register the raw packet. From this point on, the treatment should be first to determine what is the content of the frame, for instance: a packet, an ack, a fragment, ...and what to do with it, for instance: forward, route, accept locally, drop, .... This selection operation is generally called the dispatching. It needs some access to the header part of the frame content in order to process the corresponding information and determine the selection.

**5.2.2. The dispatch discovery.** The main module of this stack is the dispatch discovery module. According to the corresponding RFC4944[2], dispatches are one octet long IDs which allow to set any of the IPv6 parameters. However this only points out the fact that the option is, or is not, enabled. Furthermore dispatches are associated with headers which describe how the enabled features are carried in the transmitted packet.

This RFC[2] defines the order of the different dispatches, which implies a linear complexity of the discovery algorithm according to the number of dispatches in the packet. That is why this algorithm is a simple sequence of "if", respecting the order above. In each "case", the corresponding header treatment modules are called by their public interfaces. The returned values are used to fill a structure which aims to reflect the IPv6 packet one. So, if there is any update in one or more modules, there is no need to modify the main part of the stack. Furthermore, if a module hot-plug implementation is considered in a near future in TinyOS, it would be possible to load specialized modules of the 6LoWPAN implementation on the fly.

Notice that not all the cases sequences are valid as shown in Figure 6. Thus optimized treatment and error discovery are possible in our modular approach.

**5.2.3. The header treatment.** Linked with the dispatch the header defines how features are carried. For instance, when the Mesh-Routing dispatch is enabled, the header next to it, tells how the hop limit is encoded in the package. Another example is, the source address and destination address encoding, those can be carried inline, like in a standard IPv6 packet, shortened or compressed. The shortening or compressing method is still under discussion in the IETF charters of 6LoWPAN. Generally, each important dispatch, i.e. not the discarding ones, are associated with a header to clearly define the packet structure.

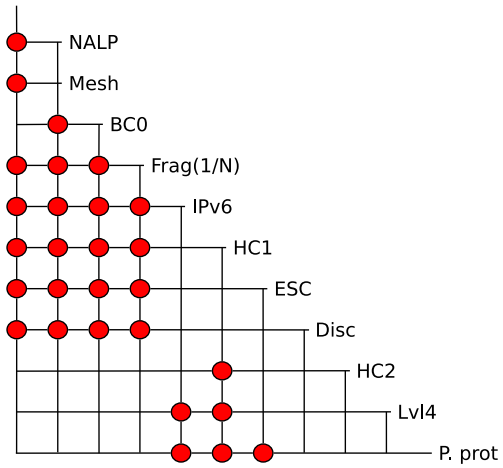


Figure 6. Diagram of the possible encapsulations for 6LoWPAN messages dispatching

## Conclusion and Future Work

IP for WSN is a hot topic because of its implications for the WSN usages, tools and applications. The modularity of our 6LoWPAN implementation will allow to replace any part of the stack with another one that has the same interfaces. This will improve research tests, development productivity, and opens new aims for the OS, such as module hot-plugging. This last point may become of major importance with application-based policies for WSN. For instance enforcing a special routing strategy for fire-alarm usage of the WSN, using another routing module for regular temperature recording and another dedicated stack for distributed computation of transfer functions.

## References

[1] G. Montenegro, N. Kushalnagar, J. Hui, and D.Culler, "IPv6 over low-power wireless personal area networks (6LoWPANs): Overview, assumptions, problem statement, and goals," RFC 4919, 2007.

[2] ——, "Transmission of IPv6 packets over IEEE 802.15.4 networks," RFC 4944, 2007.

[3] Y. Mazzer and B. Tourancheau, "MPI in wireless sensor networks," in *Recent Advances in Parallel Virtual Machine and Message Passing Interface*. Springer, 2008.

[4] ——, “Calibration study of wirelessly networked temperature sensors,” in *International Building Performance Simulation Association (IBPSA)*, 2008.

[5] ——, “Low consumption embedding sensors microcontrollers networks,” INRIA, Tech. Rep. 00256210, janvier 2008, (in french).

[6] Crossbox Inc., “Crossbow technology,” 2009. [Online]. Available: www.xbow.com

[7] J. Polastre, R. Szewczyk, and D. Culler, “Telos: Enabling ultra-low power wireless research,” in *Information Processing in Sensor Networks*. IEEE press, 2005.

[8] Sun Microsystem, “Sunspotworld,” 2009. [Online]. Available: www.sunspotworld.com

[9] Sensinode Ltd., “Nanostack by sensinode,” 2007-2009. [Online]. Available: sensinode.com

[10] J. P. H.Y. Zhou, K.M. Hou, “Wireless sensor networks dedicated to remote continuous real-time cardiac arrhythmias detection and diagnosis,” *Global Mobil Congress 05 (GMC05)*, 2005.

[11] J. L. Hill, “System architecture for wireless sensor networks,” Ph.D. dissertation, University of California, Berkeley, 2003, adviser-David E. Culler.

[12] *Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (LR-WPANs)*, IEEE Computer Society, 2006.

[13] A. Dunkels, B. Grönvall, and T. Voigt, “Contiki - a lightweight and flexible operating system for tiny networked sensors,” in *EmNetS*. IEEE, 2004.

[14] FreeRTOS, “Freertos,” 2009. [Online]. Available: www.freertos.org

[15] H. D. S. Bhatti, J. Carlson, “MANTIS OS: An embedded multithreaded operating system for wireless micro sensor platforms,” *Mobile Networks & Applications (MONET), Special Issue on Wireless Sensor Networks*, vol. 10, no. 4, pp. 563–579, 08 2005.

[16] BTNut, “Btnut system software,” 2009. [Online]. Available: www.btnode.ethz.ch

[17] R. Balani, Z. Charbiwala, and C.-C. Han, “Sos 2.x,” 2008. [Online]. Available: nesl.ee.ucla.edu/projects/SOS

[18] J. W. Hui and D. E. Culler, “IP is dead, long live IP for wireless sensor networks,” in *SenSys ’08: Proceedings of the 6th ACM conference on Embedded network sensor systems*. ACM, 2008, pp. 15–28.

[19] M. Harvan, “Connecting wireless sensor networks to the internet, a 6lowpan implementation for tinyos 2.x,” Master’s thesis, Jacobs University, May 2007.

[20] University of California at Berkeley, “BLIP,” 2009. [Online]. Available: smote.cs.berkeley.edu:8000/tracenv/wiki/blip

[21] SICS, “Contiki operating system,” 2009. [Online]. Available: www.sics.se/contiki