Poster Abstract: Making Sensor Networks IPv6 Ready

Mathilde Durvy*, Julien Abeillé*, Patrick Wetterwald*,
Colin O'Flynn†, Blake Leverett⋆, Eric Gnoske⋆, Michael Vidales⋆, Geoff Mulligan•,
Nicolas Tsiftes‡, Niclas Finne‡, Adam Dunkels‡
{jabeille,mdurvy,pwetterw}@cisco.com, coflynn@newae.com,
{blake.leverett,eric.gnoske,michael.vidales}@atmel.com, geoff@mulligan.com, {nvt,nfi,adam}@sics.se
* Cisco Systems, † NewAE, ⋆ Atmel Corporation,
• Proto6 LLC, ‡ Swedish Institute of Computer Science

ABSTRACT

With emerging IPv6-based standards such as 6LowPAN and ISA-100a, full IPv6 sensor networks are the next major step. With millions of deployed embedded IPv6 devices, interoperability is of major importance, both within the sensor networks and between the sensors and the Internet hosts. We present uIPv6, the first IPv6 stack for memory-constrained devices that passes all Phase-1 IPv6 Ready certification tests. This is an important step for end-to-end interoperability between IPv6 sensors and any IPv6 capable device. To allow widespread community adoption, we release uIPv6 under a permissive open source license that allows both commercial and non-commercial use.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols

General Terms

Standardization, Design

Keywords

IPv6, Protocol stack, Sensor networks

1. INTRODUCTION

Most of current wireless sensor platforms use IEEE 802.15.4 as a physical and medium access control layer. The upper layers of the communication stack, however, remain either proprietary or specified by exclusive alliances such as Z-Wave or Zigbee. This plethora of solutions renders interoperability between different sensor networks difficult. The diversity of protocols also makes the seamless integration of sensor networks with existing IP networks impossible.

The adoption of IP as the Layer-3 protocol to connect wireless sensors has been slow down by the common belief that IP is too large to fit on a memory constrained device. This belief was shown to be false by existing IPv4 implementations, the most well-known being the uIP stack [5].

Yet, a real *Internet of Things* requires the large address space of IPv6. This extended address space (2^{128} instead of 2^{32}) together with its autoconfiguration capabilities makes IPv6 a suitable protocol for large scale sensor network deployments. Moreover, recent standardization work within the 6LowPan group [10] has reduced the header overhead of IPv6, thereby reducing its power consumption.

SenSys'08, November 5–7, 2008, Raleigh, North Carolina, USA.
ACM 978-1-59593-990-6/08/11.

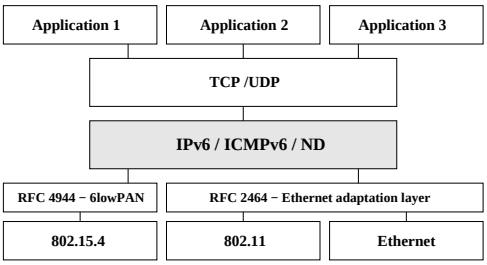

Figure 1: The uIPv6 stack runs over any MAC and link layer, such as 802.15.4/6LowPAN, 802.11, and Ethernet.

We present uIPv6, the smallest IPv6 Ready [13] stack available so far. It has a code size of 11.5 Kbytes and requires less that 2 Kbytes of RAM. As a result, it can fit on the most constrained platforms available today. Our uIPv6 stack is built on top of the Contiki Operating System [6], which also includes the latest version of the uIP stack. uIPv6 will be included in Contiki version 2.3.

2. INTEROPERABILITY IS ESSENTIAL

Adherence to standards is essential to ensure interoperability and to allow large scale deployments. RFC4294 [9] summarizes requirements for IPv6 hosts and routers. In particular, it states that an IPv6 node *must* comply to the following standards: IPv6 Specification [3], IPv6 Addressing Architecture [8], Neighbor Discovery (ND) [11], Internet Control Message Protocol for IPv6 [1], Stateless Address Autoconfiguration [14], Default Address Selection [4] and Multicast Listener Discovery [2, 15].

To ensure compliance with the IPv6 standards, the IPv6 Forum [12] created the IPv6 Ready logo program [13]. The IPv6 Ready program provides conformance and interoperability test specifications and delivers Phase-1 and Phase-2 logos that correspond to two levels of certification. Since its start in 2003, the program has certified over 350 products, ranging from software stacks to hardware devices such as network printers and telephones.

uIPv6 is the first stack for very constrained devices to satisfy all the IPv6 Ready Phase-1 requirements, thereby making it the smallest IPv6 Ready stack in existence today.

3. THE UIPV6 STACK

The uIPv6 stack is implemented in Contiki and uIPv6 integrates into Contiki the same way the uIP IPv4 implementation does. uIPv6

is tightly coupled to the UDP and TCP protocol implementations, and offers the same interface to application developers as uIP does.

uIPv6 does not depend on any particular MAC or link layer. The interface between uIPv6 and the lower layers consists of two wrappers for the link-layer input/output functions, the link-layer address, and a couple of constants. This creates a level of abstraction which enables the easy integration of many different MAC and link layer protocols (see Figure 1).

uIPv6 runs as a Contiki protothread [7]. At system startup, uIPv6 initializes its network interface. The node creates its link-local IPv6 address by combining the $fe80 :: 0/64$ prefix and its 802.15.4 MAC address. It then performs Duplicate Address Detection (DAD) to make sure the address is not already used by another node. In parallel, the node issues router solicitation messages to trigger advertisement from routers on the network. The node uses the information received to configure its global addresses and update its network parameters. The uIPv6 stack uses separate timers to pace the messages sent during the DAD and the router discovery processes. In further operations, events generated by lower-layers trigger the processing of incoming IPv6 packets by the stack. This includes generating the appropriate response and updating the different neighbor discovery and interface data structures. When a node needs to send a packet, it performs next-hop determination to find the neighbor to which the packet should be sent. If the MAC address of this neighbor is not contained in its cache, the node performs address resolution to obtain it.

The uIPv6 stack uses a single global buffer for incoming and outgoing packets. The length of the buffer is the length of the MAC header plus 1280 bytes (i.e., the minimum link MTU [3]). Additional buffers are available to support fragment reassembly and per neighbor packet buffering. The main data structures are the interface address list and the neighbor cache, prefix list, and default router list which are required by ND. The uIPv6 stack uses two periodic timers to manage and remove outdated information from these structures.

4. EVALUATION

As our implementation targets very constrained devices, it is essential that the memory footprint be small. To evaluate our system, we use the Atmel's RAVEN board which is equipped with a Atmega1284P MCU with 128 Kbytes of flash and 16 Kbytes of SRAM. We compile our code with avr-gcc 4.2.2.

The code and memory footprint is given in Table 1. The total IPv6 code size is approximately 11.5 Kbytes and the RAM usage around 1.8 Kbytes. RAM usage has been carefully optimized. The single packet buffer uses already 1296 bytes, the neighbor cache 140 bytes (35 per neighbor stored), the prefix list 69 bytes (23 per prefix), the router list 14 bytes (7 per router) and the interface address list and variables 109 bytes.

We found that on constrained devices, the limited amount of buffer space makes it impossible to support the fragment reassembly of very large packets (as mandated by RFC2460 [3]). It is also unrealistic to expect that a node would be able to buffer one packet per neighbor during the address resolution process (as mandated by RFC4861 [11]).

Moreover, the fact that most ND messages, when they include options, trigger data structure updates creates considerable complexity in the packet processing, as shown by the size of the ND Input/Output file. Other IP design principles, such as the need to accept options and extension headers in any order while nodes usually send them in the recommended order also add complexity to the implementation.

For a complete running system, preliminary results give a total code size of 35K. This includes RAVEN drivers, 802.15.4 PHY and MAC, 6lowPAN with fragmentation and header compression, uIPv6, and the Contiki OS. For a UDP (resp., TCP) transport layer add 1.3KBytes (resp., 4KBytes).

Table 1: Code and memory footprint for the uIPv6 stack, in bytes

Function	ROM	RAM
ND Input/Output	4800	20
ND structures	2128	238
Network interface management	1348	118
Stateless address autoconf	372	16
IPv6 (header processing, etc)	1434	44
Packet buffer	0	1296
ICMPv6	1406	16
Total	11488	1748

5. CONCLUSION

In this work, we show that even severely memory-constrained devices can fulfill the requirements for Phase-1 of the IPv6 Ready program. This is a major step towards full-scale interoperability between IP-based sensor networks and hosts on the wired Internet, and paves the way for the next generation of integrated IP and sensor networks.

6. REFERENCES

[1] A. Conta, S. Deering, and M. Gupta. Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification. RFC 4443, IETF, Mar. 2006.

[2] S. Deering, W. Fenner, and B. Haberman. Multicast Listener Discovery (MLD) for IPv6. RFC 2710, IETF, Oct. 1999.

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

[4] R. Draves. Default Address Selection for Internet Protocol version 6 (IPv6). RFC 3484, IETF, Feb. 2003.

[5] A. Dunkels. Full TCP/IP for 8-bit architectures. In *Proceedings of MobiSys*, May 2003.

[6] A. Dunkels, B. Grönvall, and T. Voigt. Contiki - a lightweight and flexible operating system for tiny networked sensors. In *Proceedings of EmNets I*, Nov. 2004.

[7] A. Dunkels, O. Schmidt, T. Voigt, and M. Ali. Protothreads: Simplifying event-driven programming of memory-constrained embedded systems. In *Proceedings of SenSys 2006*, Nov. 2006.

[8] R. Hinden and S. Deering. IP Version 6 Addressing Architecture. RFC 4291, IETF, Feb. 2006.

[9] J. Loughney. IPv6 Node Requirements. RFC 4294, IETF, Apr. 2006.

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

[11] T. Narten, E. Nordmark, W. Simpson, and H. Soliman. Neighbor Discovery for IP version 6 (IPv6). RFC 4861, IETF, Sept. 2007.

[12] The IPv6 Forum. http://www.ipv6forum.com/.

[13] The IPv6 Ready Logo Program. http://www.ipv6ready.org.

[14] S. Thomson, T. Narten, and T. Jinmei. IPv6 Stateless Address Autoconfiguration. RFC 4862, IETF, Sept. 2007.

[15] R. Vida and L. Costa. Multicast Listener Discovery Version 2 (MLDv2) for IPv6. RFC 3810, IETF, June 2004.