Interconnecting Smart Objects with the
Internet Workshop

Prague, Friday, 25th March 2011

M. Durvy
M. Valente
Cisco
{mdurvy,marcvale}@cisco.com

# Making Smart Objects IPv6 Ready: Where are we?

## Abstract

In 2008, the uIPv6 stack demonstrated that it was possible to run IPv6 on very constrained devices. The success of uIPv6 was due to the fact that it was lightweight, open-source, and IPv6 Ready. This paper reviews our experience in developing and certifying the uIPv6 stack. In addition, it discusses both the need and the challenges of interoperability and conformance testing for IPv6 Smart Objects. Are we really ready to connect IPv6 Smart-Objects to the Internet?

## 1. uIPv6 Stack: Development

Three years ago, the concept of an IPv6 Smart Object did not exist. People did not believe that IPv6 could run on constrained devices. To weigh beliefs against facts, we decided to implement uIPv6 (i.e. micro-IPv6), a tiny IPv6 host stack targeting constrained platforms such as sensors and actuators. To achieve this goal we selected the [Contiki] Operating System. Contiki is a well-known, highly portable, multitasking OS for low-memory networked embedded systems. Our choice was driven by the fact that Contiki is a very active open-source project, the code is written in C, and at that time, Contiki had already a stable IPv4 stack. We used [RFC-4294] as a starting point. Our initial implementation covered all the MUST in [RFC-2460], [RFC-4291], [RFC-4861], [RFC-4862], [RFC-4443], and [RFC-3484]. Based on this first code, we realized we could even afford to support most of the SHOULD in the aforementioned RFCs. However, given their limited use in the context of Smart Objects, we decided not to implement PMTU discovery, Redirects, and MLD.

The total code size of uIPv6 is approximately 11 Kbytes and the RAM usage is around 2 Kbytes when compiled for Atmel’s RAVEN target. The RAVEN board is equipped with an Atmega1284P MCU featuring 128 Kbytes of flash and 16 Kbytes of SRAM, as well as with a 2.4 GHz [IEEE 802.15.4] transceiver.

Thanks to its open-source release, uIPv6 demonstrated to the world that it was possible to run a fully compliant IPv6 stack on SO. Yet, our implementation also pointed out a few challenges. We found that the limited amount of buffer space on constrained devices makes it difficult to support the reassembly of large packets' fragments (as mandated by [RFC-2460]). For the same reason, it is also unrealistic to expect that a node should be able to buffer one packet per neighbor during the address resolution process (as mandated by [RFC-4861]). 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 ND input/output code size of approx 5 Kbytes. 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.

## 2. uIPv6 Stack: Certification

To ensure compliance with the IPv6 standards, the [IPv6 Forum] created the [IPv6 Ready] Logo program. The IPv6 Ready Logo 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 such as network printers and telephones.

The process is rather simple. It is sufficient to download the free self-testing tools (another possibility is to go to a testing lab), install them on a FreeBSD machine, and run them against the target. This process generates automatically log-files with detailed test results which are extremely useful for debugging purposes. This part of the testing verifies the target's conformance to the different IPv6 standards, and it includes tests for all the MUST features. The number of SHOULD features tested varies between Phase-1 and Phase-2. In addition to conformance testing, basic interoperability testing needs to be run with products from at least 4 different vendors. Finally, all test results must be submitted via an on-line application to the IPv6 Ready Logo Program, which examines the submission and delivers (or not) the logo.

uIPv6 was the first stack for very constrained devices to satisfy all the IPv6 Ready Phase-1 requirements. This certification was a key step towards the interconnection of IPv6 SO with the Internet.

## 3. Interoperability and Conformance Testing for IPv6 Smart Objects

Thanks to the IPv6 Ready Program, today it is possible to certify an IPv6 software stack for SOs. However, due to the lack of extensive media support in the current test tools, this is only true if the SO stack runs

on top of Ethernet. Building this support is not a trivial task. Take for example IPv6 over IEEE 802.15.4. 802.15.4 is a wireless link with very different characteristics than Ethernet. Therefore, the IETF 6lowpan WG has defined an adaptation layer to run IPv6 on top of 802.15.4. This adaptation layer is responsible for fragmenting IPv6 packets into smaller chunks (max 127 bytes), compressing the IPv6 header to reduce the number of bytes that need to be sent, and specifying neighbor discovery. As a consequence, new test specifications (for 6lowpan standards) and test tools (supporting 802.15.4) are needed to test IPv6 on top of 802.15.4. In March 2010, the IP for Smart Objects (IPSO) Alliance partnered with the IPv6 Forum to address the testing of IPv6 SOs. This could be another major step for the integration of SOs into the Internet. Finding the right balance between performance, flexibility, and interoperability will be the next challenge. For each SO protocol that we standardize, we should ensure that the mandated set of features is minimal (one cannot afford wasted resources on SOs) but still guarantees a working, reasonably performing, system. Indeed, while optional features bring flexibility, the mandated features are the ones that guarantee interoperability. Are we there yet?

## 4. References


[uIPv6] Poster Abstract: Making Sensor Networks IPv6 Ready Sensys, Raleigh, USA, November 2008

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

[IPv6 Ready] http://www.ipv6ready.org/

[Contiki] http://www.sics.se/contiki/

[IEEE 802.15.4] IEEE Computer Society, "IEEE Std. 802.15.4-2006", October 2006.

[RFC-4294] J. Loughney, Ed., “IPv6 Node Requirements”, RFC 4294, April 2006

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

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

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

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

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

[RFC-4443] 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.