Roadmap to an IP Stack in TinyOS

TEP:	136
Group:	Network Protocol Working Group
Type:	Informational
Status:	Draft
TinyOS-Version:	> 2.1
Author:	Stephen Dawson-Haggerty, Matus Harvan, and Omprakash Gnawali

Abstract

Recently, there has been renewed interest in the applicability of Internet Protocol-based networking for low power, embedded devices. This interest is being driven by several sources. One, emerging standards from the IETF are beginning to make it possible to design efficient, compatible implementations of IP for this class of resource-constrained device. Two, there has been an emergence of a significant number of both open and closed IP embedded IP stacks which demonstrate the applicability of this model of networking. Third, a set of recent academic papers have made the case that this network architecture is a significant research enabler and shown in more detail the structure of a full stack.

In this TEP, we briefly explain how IP mechanisms map onto the well-studied problems of the sensor network community like collection routing and local aggregation. Next, we discuss the current "state of the standards." Finally, we propose a path for the adoption of IP within TinyOS.

1. IP Requirements and Mechanisms

There are two versions of IP: IPv4 (RFCXXX) and IPv6 (RFCXXX). Previous studies have indicated that IPv6 is more applicable to the low-power, embedded space then IPv4 for various reasons; most standardization efforts have focused on adapting IPv6 to these devices. Therefore, we consider only IPv6.

1.1 IPv6 Routing

A central question for implementing IPv6 in sensor networks is what has become know as "route over" vs. "mesh under" in the IETF. In mesh under networking, routing is done on layer two addresses, and every host is one hop from every other. Although this is the most compatible with existing assumptions about subnet design, it leads to significant redundancies and inefficiencies in this space. The alternative, so called route-over exposes the radio topology at the IP layer. While not strictly compatible with some IPv6 mechanisms, this is becomming the favored approach since a single

set of tools can be used to debug. For the rest of this TEP we assume that route over is the model.

There are a number of existing routing protocols for IPv6, some targeted at wireless links. However, IPv6 itself does not require any particular routing protocol to be used with a domain; common choices in wired networks are OSPF and IS-IS. Recent study in the IETF has indicated that existing protocols are probably not appropriate for this space [draft-ietf-roll-protocols-02], and so further work is needed on this point. Existing protocols with TinyOS implementations such as DYMO or S4 may be appropriate for this task; collection and dissemination protocols like CTP or Drip are probably not directly applicable since they are address-free, although the insight gained from their implementations may be transferable.

1.2 Addressing

The most well-known property of IPv6 is probably it's address length. An IPv6 address is 128 bits long. Within this space, IPv6 defines unicast, anycast, and multicast address ranges; each of these ranges further have properties of their own. For a full explaination of IPv6 addressing we refer the reader to, for example, [OReilly, RFC], we cover briefly some of the important details.

1.2.1 Unicast Addressing

Unicast addresses in IPv6 consist of two parts: the network identifier (the first 64 bits), and the interface identifier (the second). The interface identifier is a flat space, and may be derived from the interface's MAC address, a random number, or other mechanism. IPv6 contains mechanisms to ensure that the interface ID is unique within the subnet. The network may be assigned using many different mechanisms, but some network identifiers are special.

Unlike IPv4, each interface in IPv6 is multihomed: it is expected to have multiple IPv6 addresses. When an interface is brought up, IPv6 contains mechanisms to configure the interface with a locally unique, non-routable address known as the link-local address. This address has the network identifier fe80::, and can be used for communication between hosts in the same subnet.

In a TinyOS IPv6 stack, this address range might be used to allow TinyOS nodes to communicate locally without routing, for instance to enable local aggregation. If the TinyOS hosts can assign themselves link-local addresses derived from their IEEE802.15.4 16-bit short id, or full 64-bit EUID. For instance, a node with short ID 16 might assign the address fe80::10 to its 802.15.4 interface. These addresses would not be routed; an IP stack would send directly to the short id 0x10 contained in the address.

IPv6 also contains several mechanisms to allow a host to obtain a publicly-routable network identifier. TinyOS hosts communicating with these addresses could contact nodes in other sensor networks or on the internet; the fact that they are multihomed allows them to use both public and link-local addresses simultaneously.

1.2.2 Multicast Addressing

IPv6 contains a multicast address range; addresses beginning with the byte containing all ones (0xff) are multicast addresses. Following this byte are four bits of flags and four bits of "scope". For instance, scope 1 is node-local, and scope 2 is link local. IPv6 defines many well-known multicast groups [http://www.iana.org/assignments/ipv6-multicast-addresses];

of most interest here are the "link-local all nodes" and "link local all-routers" addresses: ff02::1 and ff02::2, respectively. Depending on weather TinyOS IP hosts are also IP routers, these addresses are effecitvely link-local broadcast addresses which might be mapped into the layer 2 broadcast address (0xffff). Thus IPv6 contains mechanisms for local broadcast.

There is also a site-local scope defined in IPv6 (5) with a similar ff05::2 address. "Site local" is an administratively defined scope in IPv6, and might naturally consist of an entire sensor network. Thus, sending to this address would correspond to disseminating a value to each node in the network and could be implemented with a trickle-based algorithm.

Futhermore, the TinyOS community could develop additional conventions to provide and address for scoped flooding or delivery to nodes with particular capabilities. A full IP multicast implementation within the sensor network could be used for many things including control applications or publish-subscribe network layers which have previously been special purpose.

1.2.3 Anycast Addressing

Anycast addresses indicate that the packet should be delivered to at least one host with the anycast address. This seems initially similar to the model for collection routing such as MultiHopLQI or CTP, in that a collection report is delivered to a single root. However, it is more likeley that those "collection roots" in an IP-based infrastructure are not actually the final destination for the data; they are likely to only be intermediate routers who send the data on to a final destination. Therefore, while there may be applications for anycast addressing, we do not believe this addressing mode to be appropriate in the common case.

1.3 IPv6 Configuration Mechanisms

As alluded to earlier, IPv6 contains two mechanisms to allow internet hosts to become associated with a public network identifier. These methods are stateless autoconfiguration and DHCPv6. Stateless autoconfiguration defines Router Solicitations and Router Advertisements. A host joining a network sends router solicitations to the link-local all-routers address (ff02::2) using his link-local address as the source address. Routers respond with a Router Advertisement containing, among other things, a public network identifier which the host may use.

In a TinyOS IP implementation, router solicitations and advertisements might be used for default route selection on the hosts, as well as neighbor discovery.

1.4 Extension Mechanisms

A common idiom in TinyOS is to provide "stacked" headers by implementing a series of components, all of which implement the Packet interface. IPv6 supports this a more flexible way with options headers. These headers fall into one of two categories: hop-by-hop options headers and destination option headers. These headers are chained together with small, two-octet common header fields which identifiy the header type of the next header, and the length of that options header. This allows hosts to process a header chain even if they do not know how to interpret all of the options headers.

These headers are commonly used for protocol options, or to piggyback data on outgoing packets. For instance, a link estimator component could add an extra "link

options" header to occasional outgoing packets as an options header, while avoiding the overhead when it is not necessary to do so. This header architecture is significantly more flexible then the existing TinyOS packet architecture, although it does come at the slight cost of complexity.

2. The State of the Standards

All of the previously defined mechanisms are well defined for IPv6 in various RFCs. However, most nodes running TinyOS are significantly more resource-constrained then typical IPv6 hosts. Thus, there is ongoing work on adapting these mechanisms to the characteristics of embedded devices.

2.1 Header Compression

The first issue which must be addressed is the sheer size of the IPv6 header: 40 octets. Of this, 32 octets are the source and destination addresses of the packet. The IETF has published RFC4944 which describes a header compression technique known as HC1. However, this scheme has significant problems, most importantly the inability to take advantage of 16-bit short identifiers when available. There have been a sequence of internet drafts proposing improved techinques such as HC1g, and HC. The most recent version of these drafts is draft-hui-6lowpan-hc-02, and there are strong indications that the working group will depreciate HC1 from RFC4944 in the future.

2.2 MTU

IPv6 requires a minimum link MTU of 1280 octets in RFCXXX. Clearly, this is much larger then the IEEE802.15.4 link MTU of 127 octets. RFC4944 defines "layer 2.5" fragmentation which allow packets up to this MTU to be transfered over small MTU links. While there are some issues have been raised with this scheme, it seems likely to remain relatively unaltered.

2.3 Autoconfiguration

IPv6 stateless autoconfiguration as originally defined has some problems, expecially once a "route over" network topology is established; for instance, Duplicate Address Detection is likely to require some changes. A subcomittee of the 6lowpan working group is currently investigating these issues is is likely to propose adaptation mechanisms but they are currently in flux.

2.4 Routing

It is currently not possible to develop a standards-compliant routing layer, as the relevant standards do not exist. Work in this direction is occuring in the Roll working group, but a full specification is likely some way off.

3. The TinyOS Way

As the previous sections have shown, many sensor network abstractions map well into IPv6 mechanisms, with a significant gain in clarity of abstraction. However, standards

are currently unavailable to specifiy exactly how this should be accomplished. Therefor, our position is that TinyOS should move quickly to incorporate existing techinques and standards into an IPv6 stack where possible, while taking liberties with the standards when doing so improves performance or simplifies implementation. Given that the standards themselves are in a state of flux, having an open implementation which demonstrates challenges and feasible mechaninisms is likely to be of significant value.

The most important places where it may be necessary to move ahead of the standards are the routing; an IPv6 stack with no routing will not be as useful as one with it. One open question is weather we choose a routing algorithm which functions in a completly decentralized fashion like AODV, OLSR, DYMO, S4, or many existing protocols or choose one which imposes more hierarchy on deployments. We do note that existing standards like Zigbee and WirelessHART both contain multi-tier architectures with devices of differing capabilities. This has also been a common deployment model in many applications, but it limits the utility to places where this is possible. The correct long-term answer is probably to support both types of routing.

4. Conclusion

This document is meant to introduce readers to the ways in which IPv6 mechanisms can be used in a TinyOS-based sensor network deployment, and how they relate to previous work. It does not address any implementation issues, which will be presented in a later TEP.

5. Authors

Stephen Dawson-Haggerty
Computer Science Division
University of California, Berkeley
410 Soda Hall
Berkeley, CA 94701

stevedh@eecs.berkeley.edu

Matus Harvan
Information Security
IFW C 48.1
ETH Zentrum
Haldeneggsteig 4
8092 Zurich
Switzerland

phone - +41 44 63 26876
email - mharvan@inf.ethz.ch

Omprakash Gnawali
Ronald Tutor Hall (RTH) 418
3710 S. McClintock Avenue
Los Angeles, CA 90089

phone - +1 213 821-5627
email - gnawali@usc.edu

6. References