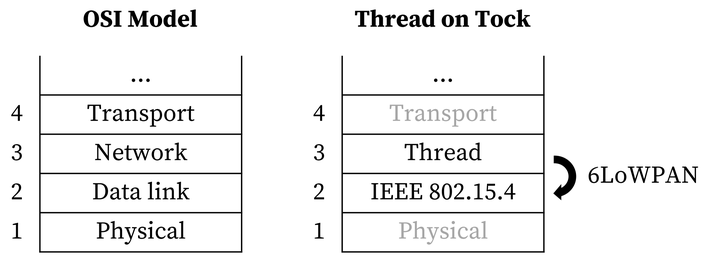
Bringing IP Networking to the Internet of Things

Mateo Garcia, Hubert Teo, Paul Crews
CURIS 2017

Motivation

The goal of our CURIS project this summer was to add low-power wireless networking support to the Tock operating system. Tock is a secure embedded OS designed for the future of the Internet of Things (IoT), when embedded devices will run multiple concurrent, mutually distrustful applications on a single embedded device.

We were tasked with enabling IPv6 packet transmission over a low-rate (and hence, low-power) wireless personal area network (WPAN), the type of network (Layer 3) used to connect IoT devices. The most widely adopted standard in the IoT space is IEEE 802.15.4, which specifies the physical (Layer 1) and data link (Layer 2) layers that support a WPAN. A number of network (Layer 3) protocols have been defined on top of IEEE 802.15.4, but because the IoT is still young, a de facto L3 standard has not yet been set. So far, only one such network protocol has been defined with Internet Protocol version 6 (IPv6) in mind: Thread, created by Nest Labs. The Thread protocol makes use of 6LoWPAN, a compression scheme defined by the Internet Engineering Task Force (IETF) that allows IPv6 packets to be fragmented and sent over an IEEE 802.15.4 based network.



Related Work

- IEEE 802.15.4
- 6LoWPAN, originally defined in RFC 4944
- Thread, a networking protocol for low-power wireless personal area network (WPANs) defined by Nest Labs in the Thread 1.1.1 Specification.
- OpenThread, an open-sourced implementation of the Thread networking protocol, written in C++
- Rust, a systems programming language that lends itself to secure systems programming with semantics like type and memory safety enforced at compile time

Background on Mesh Link Establishment (MLE)

Thread is designed to enable mesh network routing (Layer 3) over the static point-to-point and star topologies of the data links provided by IEEE 802.15.4 (Layer 2). For this mesh network to work effectively, links must be identified, configured, and secured automatically as the network's membership and physical environment change. In Thread this is handled using mesh link establishment (MLE). MLE comprises a set of commands that a node in a Thread network can use to determine the presence and quality of radio links to its neighbors.

MLE for network attaching, for example, comprises a four-step handshake that works as follows:

1. A child device multicasts a Parent Request MLE command.
2. Each potential parent device on the network unicasts a Parent Response MLE command.
3. The child device selects a parent based on a hierarchy of connectivity metrics and unicasts a Child ID Request MLE command.
4. The selected parent unicasts a Child ID Response MLE command.

MLE messages consist of a command type and a series of Type-Length-Value (TLV) parameters. TLVs are used to serialize the configuration values exchanged during MLE, including link-layer (Layer 2) addresses, transmit and receive modes, and security parameters.

Findings on MLE Using Type-Length-Value (TLV) Structures

Certain types of TLVs have value fields that can vary in length, and certain such value fields comprise a list of sub-TLVs, some of which have value fields that can in turn vary in length (see below diagram). This means that using TLVs it is possible to construct a network packet that exceeds, for example, the IPv6 maximum transition unit (MTU).

Unbounded data length is problematic in the world of embedded systems, where there is no dynamic memory allocation and dealing with variable-length data can be difficult. The Thread 1.1.1 Specification does not specify a maximum length for an MLE message — according to the authors of the specification, the implied limit is the IPv6 MTU.