Wireless & The IoT

Lab 8: Let's try something ambitious

Introduction

The goal of today's lab is to try to round out the pieces of a production Thread deployment. We're going to build one big thread network with everyone. Work in smaller sub-teams to try to accomplish the sub-goals in parallel. This is a we're-going-to-see-if-we-can-make-it-happen mission, so put on your hacker hat and plan to dig in and debug a bit to get everything working.

The Assignment

The end goal for today will be to build one big Thread network and to have some "sensor" endpoints in this network that are accessible by plain-old HTTP from the Internet at large. We'll have to work on several sub-tasks that should hopefully be parallelizable to get this working:

A. **Create one big thread network.** Choose someone to act as the commissioner for the network (i.e. Step 7 from last week). Disseminate these credentials, or use open Joining, to get several nodes on the network.

B. **[dep A] Create some CoAP endpoints within the network.** Since ~Feb 2020, the default OpenThread device image ships with built-in CoAP support. Using the same interface as last week, you should be able to setup CoAP clients and servers that can talk to one another within the Thread network. Take a look at the variety of options in CoAP. Try 'observing' a resource; how might this be used in production networks?

C. **[dep A] Create (at least one) border router.** We will need to bridge the Thread network to the internet at large, which will require a border router. The easiest / fastest way to do this is to use the official OpenThread Border Router Docker image on a Linux machine (i.e. start from the steps here). You'll need your nrf52840dk to be set up as an RCP for this machine.

- **a.** **First, quietly seethe that UCSD doesn't have IPv6 rolled out.**
- **b.** **Then, configure your laptop to act as a Wi-Fi Hotspot (this is built-in since Gnome 3.36 / ~2020) so that we can have a network with ipv6 support.** Do this last, as you'll lose regular internet.
- **c.** **If we get two border routers running:** Leave one connected to the Internet at large, as OTBR will do 6to4 translation, and allow outbound communication from sensor nodes.

D. **Set up a CoAP Proxy.** Most things don't natively speak CoAP.[1] However, CoAP was designed with interoperability with HTTP in mind (it's basically just compressed HTTP 2.0 over UDP, with fewer options; similar to what 6LoWPAN does for IPv6). There are several implementations of CoAP Proxies online. I have previously used aiocoap successfully.

E. **[dep B, C] Test the border router, CoAP reachability.** Use any tool on a regular computer to try to externally access one of the devices that is acting as a CoAP server from Task B. You will need to connect to the wifi network set up by the border router owner to be able to address endpoints directly, so do any software downloads first. Californium has a nice GUI tool for this. Or you can try using the aiocoap-client to talk to an endpoint (or try to use one of the nrf boards as a client to talk to the aiocoap-fileserver).

F. **[dep D, E] Browse your sensor network!** Using any old browser (that's connected to our IPv6 enabled network…), navigate to a sensor and browse its data!

- **a.** **If you somehow have time:** Try doing an HTTP long poll on an observable endpoint ☺

[1] There was a Firefox plugin once, but you need to install a fairly old version of Firefox to get "browser native" CoAP support: https://github.com/mkovatsc/Copper

