Lab 4 - Thread

The long awaited Thread lab finally exists!

Goals

- Get a Thread network running!
- Enable end devices to communicate with various protocols.

Equipment

- Computer with lab toolchain setup
- Three nRF52840DK and cables

Lab Steps

1. Update your repository

- Pull in updates from https://github.com/brghena/nu-wirelessiot-base
  - cd into the base of your forked repo
  - `git pull https://github.com/brghena/nu-wirelessiot-base.git main`
  - Then merge and push to your own forked repo.

2. Fix your virtual machine

If you are on **VirtualBox on native MacOS** or **VMware Player on native Windows** or **native Linux** already, you can skip this and go straight to the next step.

If you are on **native MacOS without a VM**, you'll need to install a virtual machine in order to get Docker working with USB support, which is necessary for a Thread border router. Oddly, VirtualBox seems to work fine for this on MacOS. Go install a virtual machine. It only takes half an hour or so, and most of that is waiting for things to load.

- https://docs.google.com/document/d/1-Hour3iuty7jU8w-sl47Cp10fJ34dYFxXIF8oHNlvzI/edit#
- After you're done you can skip to the next step

If you are on **native Windows with VirtualBox**, unfortunately we're going to have to switch you to VMWare Player to get this lab working. I don't know why, but USB modems seem to just not work in VirtualBox. If you want to try your luck, you can skip this step for now and try to get the Thread border router working first. If it doesn't work you can come back without any harm.

1. Export your VirtualBox VM
    - Completely shut down your VM.
    - In the main VirtualBox window, go to the tab on the left labeled "Tools"
    - Select Export, pick your VM, the default settings are fine, "Export"
    - This will take 5-15 minutes, go to the next step while you're waiting
2. Install VMWare Player
    - Download and install VMWare Workstation Player
    - https://www.vmware.com/products/workstation-player/workstation-player-evaluation.html
3. Import your VM
    - Once the export is complete, close VirtualBox and open VMWare Player
    - Open a new VM and choose the .ova file you exported. It's probably in "Documents/"
    - Choose Import
        - A pop-up might appear telling you that "Import failed" and will have a "Retry" button. Click "Retry" and don't worry about it.
    - This will take another 5-10 minutes

4. Start it and make sure things are working. It should boot and log in and all of your files should still be there.

5. Fix virtual machine extensions. These are the things making window resizing and shared clipboard work.
   - First remove virtualbox guest additions:
     ```
     sudo vbox-uninstall-guest-additions
     ```
   - Next install VM tools:
     ```
     sudo apt install open-vm-tools-desktop open-vm-tools
     ```

6. Make sure you can still program an nRF. Just load the blink example on a board and confirm it can actually program it. You’ll have to attach the USB device to the VM, like with VirtualBox.

3. Flash the Thread border router

The border router consists of two components: a Radio Co-Processor (this is 802.15.4 firmware that runs on an nRF52840DK) and a Docker container. When both are working, a Thread network is automatically created with Internet connectivity.

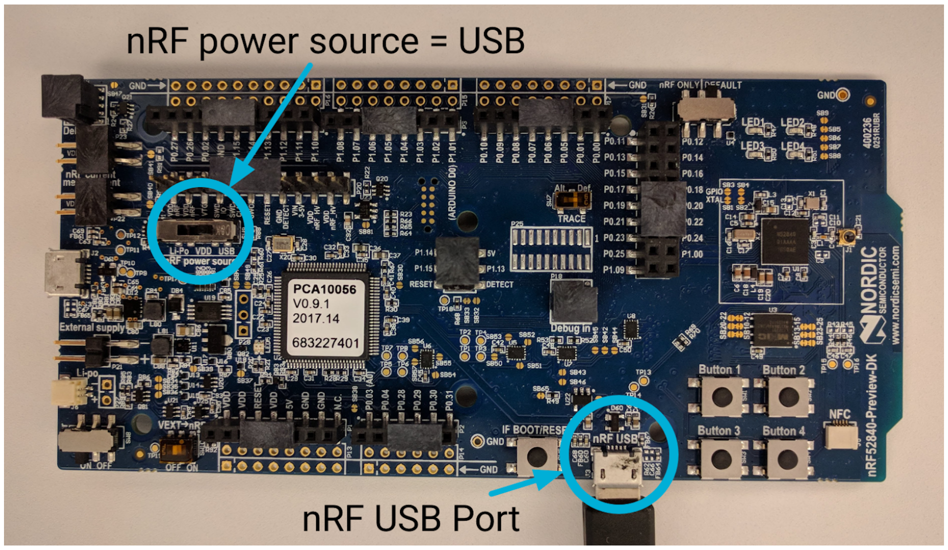
- cd software/apps/thread_border_router/
    - The README.md file has these instructions as well.

- Flash an nRF58240DK
    - JLinkExe -device nrf52840_xxaa -if swd -speed 4000 flash.jlink

- Connect the nRF52840DK
    - Unplug USB from the top, and connect it to the “nRF USB Port” on the left side

    - You also need to set the “nRF Power Source” switch to USB
        - No LEDs will turn on when it is attached. This is normal.

    - The device should now appear as “OpenThread Device”
      (rather than “Segger” as it did previously)



WARNING: to get your board back to a normal state later, you will have to remember to set the “nRF Power Source” back to VDD. If you plug into the top USB and don’t see the green led come on, remember this!!

4. Create the Thread border router

Warning: I had some serious bugs getting this working myself. So be careful with these directions, and let me know when you run into problems. Most of this should be solved by being on VMWare Player.

- Install more stuff
  ```
  sudo apt install curl libffi-dev libssl-dev python3 python3-pip
  ```

- Install docker and set up permissions
  ```
  curl -sSL https://get.docker.com | sh
  sudo usermod -aG docker YOUR_USERNAME_HERE
  ```

- Install docker-compose tool
  ```
  sudo pip3 install docker-compose
  ```

Important: be gentle with docker because it's fussy. "Ctrl-C" to stop it, it should respond ""Gracefully stopping", then wait for 30 seconds until it stops on its own. After, run "sudo docker-compose down" or else future calls to "sudo docker-compose up" won't work.

- The first step is to unplug all USB devices.

- Run `ls -lah /dev/` and confirm that there is no /dev/ttyACM0 listed right now
  - I twice had a problem where there would be a ttyACM0 even with no USB devices connected that was owned by group root. A reboot of my VM fixed it.

- Connect your Thread border router to USB and the VM,
  wait 10 seconds, then `ls -lah /dev/` again.
  - There should now be a /dev/ttyACM0 and it should be of group dialout

- cd into apps/thread_border_router/

- sudo docker-compose up
  - This will load a bunch of services then delay for about 30 seconds
  - After that it should say "otbr_1 | Done" a dozen times. See README for exact print statements. That means everything is good!!
    - "connect session failed: No such file or directory" is bad. It means it cannot find the USB device. Maybe it's not connected or you haven't flashed it?
    - This is that bad failure case. `sudo docker-compose down` then reboot the VM and try this whole set of steps again. Ask Branden for help if it still doesn't work.
  - If it starts sending Advertisements over IPv6 UDP messages, everything is good. Keep it up and running for the rest of the lab to make your life easier.

5. Create a Thread end device

This code is in “apps/thread_end_device/”. You program this onto boards just like any normal application (and don’t have to do weird USB things like with the border router). `make flash` and then `make rtt` will open up a debug window which should show the state of your device becoming “Child”. The border router should show increased traffic as the end device joins the network.

Most of the calls in this app use Simple Thread, which is our own library for simplifying Thread setup. The code is located in “nrf52x-base/lib/simple_thread/”. That library calls to the OpenThread API, a reference for which can be found at: https://openthread.io/reference

Various OpenThread commands can return errors. A list of error codes can be found at: https://openthread.io/reference/group/api-error

6. Thread and NTP

This code is in “apps/thread_ntp/”. Build and upload the code and run an RTT window to see DNS retrieve an IPv6 address for the NTP server and then request epoch timestamps.

https://www.epochconverter.com/ can display the current epoch timestamp so you can double-check it.

The Simple Thread library has wrappers for DNS and NTP that this is using in “nrf52x-base/lib/simple_thread/”.

7. Thread and CoAP

This code is in “apps/thread_coap”. First you need to get the CoAP server working.

- Install the requirements
  ```
  sudo apt install nodejs npm
  ```

- In the “apps/thread_coap/coap/” folder, run `npm install`

- Start the CoAP server and leave it open and running: `./coap-server.js`

- In another window, test that it works with the client and your local IPv4 address

    - `ip addr` lists your IP addresses (ifconfig is the way of the past)
    - Find one under the heading “ens33:”. Should be something like 192.168.xxx.xxx or maybe 10.0.xxx.xxx
    - Modify “coap-client.js” to use this IPv4 address
    - Run `./coap-client.js` and confirm that the CoAP server received the message

- In “apps/thread_coap/main.c”, modify the “COAP_SERVER_ADDR”. It’s a IPv6 to IPv4 translation address, so your local IPv4 address should be the last 32-bits.

- Make and flash it onto a board. Open an RTT window to see that it’s running. The server should start receiving messages.

    - This app doesn’t seem to receive responses, and I’m not sure why. After about 30 seconds they start timing out (error code 28). But the request does make it to the CoAP server.

The Simple Thread library has wrappers for CoAP that this is using in “nrf52x-base/lib/simple_thread/”. The OpenThread CoAP API can be found at: https://openthread.io/reference/group/api-coap

Lab Submission

The hard part today was getting anything working. You could certainly combine CoAP with buttons and LEDs. Request IPv6 addresses for other domains over DNS. And use the sense of time in an interesting way. I’m not going to make you do so, but feel free to experiment on your own.

1. Demonstrate that you did the lab. Probably with some screenshots of terminal windows.

    - If you couldn’t get a Thread border router working at all, demonstrate instead that you tried really hard and that you talked with Branden about it at least once.