NDN-RIOT Package Report

Tianyuan Yu
royu29@ucla.edu

Notes

Some parts of package are not well-written because I'm not a coding expert, so any suggestions about coding or design you can provide are highly helpful! I'm keep tidying up code and revising neccessary documentation. Apology again if my not so good coding style confuses you.

Package History/Overview

NDN-RIOT package is based on Wentao Shang's same named work in 2015, you can find the original paper work via NDN's publication list, or resort to link `https://named-data.net/wp-content/uploads/2015/01/design_implementation_ndn_protocol.pdf`.
Wentao's work provide NDN protocol stack on RIOT OS, but limited on basic Interest/Data exchanges. Original package creates a `ndn` thread aside from the user's main thread, serving networking things. Whereas, IoT scenario need built-in app-layer protocols (e.g., bootstrapping, service discovery) to facilitate development. The new package additionally create a `ndn-helper` thread to interact with core `ndn` thread, register faces, fib entries, etc. User can call `ndn-helper` function to retrieve issued certificate, neighbour identities and available services, and allocated access keys.

Environment Setting

Source

Use RIOT OS from `https://github.com/named-data-iot/RIOT` (not the official RIOT OS)
Use NDN-RIOT package from `https://github.com/Zhiyi-Zhang/ndn-riot`

Package Makefile

RIOT cloned from address above have already equipped with NDN-RIOT package (new old version of Wentao's work in 2015). To re-configure, go to folder `../RIOT/pkg/ndn-riot`, redirect the makefile here to a local source folder, or to remote github link (be sure of using the newest commit version number in makefile).
For example, if you'd like to replace original package, to `../RIOT/pkg/ndn-riot`, find `Makefile` and replace the package configuration with
`PKG-SOURCE-LOCAL ?=` $(RIOTBASE)/ndn-riot$
$PKG-BUILDDIR? =$`(PKGDIRBASE)/ndn-riot`

Project Makefile

Each new project's which use this package should have makefile with

```
USEPKG += micro-ecc                               //dealing with ECDSA signature
USEPKG += ndn-riot
USEMODULE += crypto                              //dealing with crypto operation
USEMODULE += cipher-modes        //dealing with AES-128 cipher block chain mode
```

and CFLAGS to enable RIOT's crypto module

```
CFLAGS += -DCRYPTO-AES
CFLAGS += -DCRYPTO-THREEDES
```

Test and Examples

Basic APIs inherited from Wentao's original library can be found in example folder `https://github.com/Zhiyi-Zhang/ndn-riot-tests/examples`. But tests for each module are still missing. Protocols design can be found in the same repo's wiki page. Bootstrapping protocol is little complicated since we optimized it many times for speed issue.

Tips

Boards vs. Native

If you are using samr21-xpro, it can't run discovery and access control thread together for limited RAM. Currently you can try the combination bootstrap + discovery or bootstrap + access control. If you try as a native MacOS/Linux Process, memory won't bother us.

Debugging

Basically, most issues happen after one side receive the packet and begin processing. If one doesn't receive any packets, perhaps the reasons lie in the networking configuration or hardware modules imported.

Bootstrapping

With `ndn-helper`, native MacOS/Linux process or boards can perform a bootstrapping client role, but not the server part. Bootstrapping controller need configuration manually. Source code for bootstrapping controller can be seen at example folder. Such consideration is because we plan to re-implement the bootstrapping controller part over Android, where device are powerful enough to generate key pairs with enough security level. The similar situation also exist in access control module.

Service Discovery

Neighbour Table is only used in Service Discovery, to automatically collect available identities and services under these prefixes. Table will only be initiated once when you initiate the discovery thread. You can manually add/remove entries of the table if you need.

Access Control

Like bootstrapping, `ndn-helper` can only delegates the identity applying for access control or access keys. Access controller in the network need configuration manually. Source code in the example folder.

Core Helper APIs

Listed APIs can be found in `helper-app.h`

1. `ndn-helper-init()`
   Create and Initialize the `ndn-helper` thread

2. `ndn-helper-bootstrap-start()`
   Passing a key pair structure to the function and start bootstrap process with a pre-configured controller. This function will create a thread `ndn-helper-bootstrap`.To notice, current NDN-RIOT uses ECDSA curve secp160r1. Please make sure the input key pair are the correct type

3. `ndn-helper-bootstrap-info()`
   Retriece the issued certificate from `ndn-helper`

4. `ndn-helper-discovery-init()`
   Create and Initialize the thread `ndn-helper-discovery`, as well as the neighbour table

5. `ndn-helper-discovery-register-prefix()`
   Register a subprefix to `ndn-helper-discovery`, this function is called before `ndn-helper-discovery-ini`
   and `ndn-helper-discovery-start()`. All subprefix should be registered before `ndn-helper-discovery-`

6. `ndn-helper-discovery-start()`
   Start periodically broadcast the registered services and automatically collect the listened neighbour identities and service names to neighbour table. User can check neighbour table via APIs in `neighbour-table.h`

7. `ndn-helper-access-init()`
   Create and Initialize the `ndn-helper-access` thread

8. `ndn-helper-access-terminate()`
   Terminate `ndn-helper-access`

9. `ndn-helper-access-producer()`
   Apply for access control as a producer, return a pointer of allocated key for content encrytion. This operation need a Access Controller in the network to negotiate symmetric key for applicants.

10. `ndn-helper-access-consumer()`
    Apply for certain producer identity's access, return a pointer of producer's encrytion key.