Internet of (Named) Things:
NDN Protocol Stack for RIOT-OS

Wentao Shang, Alex Afanasyev, Lixia Zhang

Presented by Alex Afanasyev

July 21, 2016
ICNRG Meeting, Berlin, Germany
ICN/NDN “Edge” for IoT

- Forget about hassle with managing IP addresses

- Bring IoT semantics to the network layer
  - Name the “things” and operations on “things”
    - “temperature in the room”, “humidity on the second floor”
    - “blood pressure”, “body temperature”
    - “max/min/avg pH of soil in specific point of US soil grid”
  - Focus on data associated with things, not devices
    - status information or actuation commands
  - Secure data directly

W. Shang et. al, "Named Data Networking of Things,” in proc. of IoTDI’2016
IoT at the Edge

- Ultra low cost, longevity
  - constrained battery, low-power networking, limited memory, low CPU
  - SAMR21-PRO: 32-bit ARM, 48 MHz, 32KB RAM, 256KB flash

- RIOT-OS: multi-platform light-weight OS
  - [https://www.riot-os.org/](https://www.riot-os.org/)
  - C and C++ programming environment
  - micro-kernel for multi-threading, priority scheduling, interrupt handling, IPC
  - standard build tools (gcc, make)
  - simulator for testing on Linux PCs
  - gaining a lot of momentum

Other platforms
- Contiki
- ARMmbed
  - [https://www.mbed.com/](https://www.mbed.com/)
- tinyOS
  - [http://tinyos.net/](http://tinyos.net/)
NDN-RIOT: NDN For RIOT-OS

- Optimized for IoT apps
- Memory efficient packet encoding & decoding
- Data-centric security support
- Basic stateful NDN packet forwarding
- Support for 802.15.4 and Ethernet
- Application API

- A few basic examples

Open source, contributions welcome
https://github.com/named-data-iot/ndn-riot

ICNRG Meeting, Berlin, Germany
NDN-RIOT Architecture

Threads
- NDN app
- NDN-RIOT module
- Net Device Driver

RIOT-OS Core
- Sched
- IPC
- Interrupt Handler

Hardware
- CPU
- Timer
- NIC
- Peripherals
Memory-Optimized Packet Decoding

• Shared memory block structure to move packets
  – avoid memory copy in most cases
• On-demand packet field extraction
  – avoid memory for decoded meta data
Security Support

• ECDSA
  – secp256r1 curve with 64-byte signatures
  – deterministic signing (RFC 6979) given lack of good entropy on many current devices
    • keys need to be generated outside the device

• no RSA
  – too much overhead and too expensive to produce signatures

• HMAC
  – RIOT-OS built-in APIs
Packet Forwarding

• PIT
  – exact match for interest
  – “any” prefix match for data (all interests that are prefix of the data)

• FIB
  – longest prefix match for interest names
  – static compile-time prefix registration
  – IPC-based run-time prefix registration (for local apps)

• CS
  – “any” match for interests (a data for which interest is a prefix)
  – compile-time adjustable size (~24KB default settings)
  – FIFO policy

• Work in progress
  – Extendable / adaptive interest forwarding strategy
  – Support for basic Interest selectors
  – Extend dynamic prefix registration and maintenance
L2 Communication

• Run directly over layer 2 interfaces
  – IEEE 802.15.4
    • send packets to FF:FF (broadcast)
  – Ethernet (e.g., debugging on native platform)
    • send packets to FF:FF:FF:FF:FF:FF:FF (broadcast)

• Simple hop-by-hop fragmentation if needed

```
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1|0|M|   SEQ   |         Identification        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
Application API

ICNRG Meeting, Berlin, Germany
The IPC channel is used for:

- NDN protocol is implemented as a kernel thread.
- HMAC-SHA256 data signing and verification
- Support for Ethernet and 802.15.4
- Sending configuration commands (e.g., add faces, device driver threads)
- Register prefixes
- Memory efficient packet encoding & decoding
- Passing NDN packets from & to APP and network
- Register prefixes
- Configure LORACoT support to the constrained IoT devices with great potential in supporting network applications.

Hardware

- CPU
- Device Driver
- Timer
- Object Interface
- Net
- Device
- Driver
- Net
- Object Interface
- Net Device
- App Handle
- Multi-threading + IPC
- Standard build tools (gcc, make)
- Simulator for testing on Linux PCs

The NDN code on RIOT-OS is C99-compatible.

List of API for NDN APP on RIOT-OS

- ndn_app_express_interest,
- ndn_app_create, ndn_app_run,
- ndn_app_destroy, ndn_app_schedule,
- ndn_app_register_prefix,
- ndn_app_unregister_prefix,
- ndn_app_register_face,
- ndn_app_unregister_face,
- ndn_app_get_interest_salt,
- ndn_app_get_interest_lifetime,
- ndn_app_get_interest_nonce,
- ndn_app_get_interest_ssid,
- ndn_app_get_name_from_event,
- ndn_app_get_name_from_uri,
- ndn_app_get_name_from_face,
- ndn_app_get_name_from_block,
- ndn_app_get_name_from_text,
- ndn_app_release_prefix,
- ndn_app_release_face,
- ndn_app_release Interest,
- ndn_app_release Data,
- ndn_app_release Face,
- ndn_app_release Face,
- ndn_app_release Face,
- ndn_app_release Face,
- ndn_app_release Face,
- ndn_app_release Face,
- ndn_app_release Face,
- ndn_app_release Face,
- ndn_app_release Face,
- ndn_app_release Face,
- ndn_app_release Face,
- ndn_app_release Face,
- ndn_app_release Face,
- ndn_app_release Face,
- ndn_app_release Face,
- ndn_app_release Face,
- ndn_app_release Face,
- ndn_app_release Face,
- ndn_app_release Face,
- ndn_app_release Face,
- ndn_app_release Face,
- ndn_app_release Face,
- ndn_app_release Face,
- ndn_app_release Face,
- ndn_app_release Face,
- ndn_app_release Face,
- ndn_app_release Face,
- ndn_app_release Face,
- ndn_app_release Face,
- ndn_app_release Face,
- ndn_app_release Face,
- ndn_app_release Face,
- ndn_app_release Face,
- ndn_app_release Face,
- ndn_app_release Face,
To gauge the run-time performance of the system, we first analyze the execution speed of individual APIs through a set of benchmarks, and then show the application-level RTT as an indicator for the performance of an integrated NDN-IoT design.

Table III shows the output from the GNU command to measure the memory usage of the NDN-RIOT APIs. The static data in the .data and .bss sections includes fixed-size stack for each thread and other pre-allocated global variables.

<table>
<thead>
<tr>
<th>Function Name</th>
<th>ARMv6-M</th>
<th>ARMv7-M</th>
</tr>
</thead>
<tbody>
<tr>
<td>ndn_name_from_uri</td>
<td>420</td>
<td>408</td>
</tr>
<tr>
<td>ndn_name_append</td>
<td>232</td>
<td>232</td>
</tr>
<tr>
<td>ndn_name_get_size_from_block</td>
<td>124</td>
<td>124</td>
</tr>
<tr>
<td>ndn_name_get_component_from_block</td>
<td>152</td>
<td>164</td>
</tr>
<tr>
<td>ndn_interest_create</td>
<td>196</td>
<td>192</td>
</tr>
<tr>
<td>ndn_interest_get_name</td>
<td>92</td>
<td>94</td>
</tr>
<tr>
<td>ndn_data_create</td>
<td>668</td>
<td>692</td>
</tr>
<tr>
<td>ndn_data_get_name</td>
<td>98</td>
<td>100</td>
</tr>
<tr>
<td>ndn_data_get_content</td>
<td>160</td>
<td>168</td>
</tr>
<tr>
<td>ndn_data_verify_signature</td>
<td>450</td>
<td>502</td>
</tr>
<tr>
<td>ndn_app_run</td>
<td>612</td>
<td>596</td>
</tr>
<tr>
<td>ndn_app_schedule</td>
<td>96</td>
<td>88</td>
</tr>
<tr>
<td>ndn_app_express_interest</td>
<td>160</td>
<td>168</td>
</tr>
<tr>
<td>ndn_app_register_prefix</td>
<td>180</td>
<td>180</td>
</tr>
<tr>
<td>ndn_app_put_data</td>
<td>60</td>
<td>56</td>
</tr>
</tbody>
</table>

Table IV: Execution time of the NDN-RIOT APIs

<table>
<thead>
<tr>
<th>Test Case</th>
<th>Data Size</th>
<th>URI to Name</th>
<th>Create Data (ECDSA)</th>
<th>Verify Data (ECDSA)</th>
<th>Create Data (HMAC)</th>
<th>Verify Data (HMAC)</th>
<th>Create Data (ECDSA)</th>
<th>Verify Data (ECDSA)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>100 bytes</td>
<td>184</td>
<td>164</td>
<td>384</td>
<td>232</td>
<td>232</td>
<td>196</td>
<td>192</td>
</tr>
<tr>
<td></td>
<td>192 bytes</td>
<td>282</td>
<td>280</td>
<td>860</td>
<td>411</td>
<td>411</td>
<td>596</td>
<td>596</td>
</tr>
<tr>
<td></td>
<td>196 bytes</td>
<td>294</td>
<td>292</td>
<td>864</td>
<td>416</td>
<td>416</td>
<td>640</td>
<td>640</td>
</tr>
</tbody>
</table>

Our final evaluation measures the RTT of Interest-Data exchange between two RIOT-OS devices. The experiments are carried out on the FIT IoTLab platform.

Newly created data, the RTT is dominated by the ECDSA signing operation. SAMR21-XPRO can create and verify about 2 Data packets with ECDSA signatures per second. On IoTLab-M3 (72 MHz MCU) the performance is about 150 times slower than using HMAC.

To summarize, on SAMR21-XPRO with a 48 MHz MCU, although it is still an order of magnitude slower than on SAMR21-XPRO, the system is still able to handle a reasonable amount of traffic with acceptable RTTs. The results are in line with our expectations based on the memory footprint and execution speed of the APIs.
## Performance Numbers

<table>
<thead>
<tr>
<th>Test Case</th>
<th>SAMR21-XPRO</th>
<th></th>
<th>IoTLab-M3</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time (μs)</td>
<td>Cycles</td>
<td>Time (μs)</td>
<td>Cycles</td>
</tr>
<tr>
<td>URI to Name</td>
<td>184</td>
<td>8,832</td>
<td>282</td>
<td>20,304</td>
</tr>
<tr>
<td>Get Name size</td>
<td>13</td>
<td>624</td>
<td>11</td>
<td>792</td>
</tr>
<tr>
<td>Get Name component</td>
<td>8</td>
<td>384</td>
<td>7</td>
<td>504</td>
</tr>
<tr>
<td>Append to Name</td>
<td>28</td>
<td>1,344</td>
<td>29</td>
<td>2,088</td>
</tr>
<tr>
<td>Create Interest</td>
<td>25</td>
<td>1,200</td>
<td>23</td>
<td>1,656</td>
</tr>
<tr>
<td>Get Interest Name</td>
<td>2</td>
<td>96</td>
<td>2</td>
<td>144</td>
</tr>
<tr>
<td>Create Data (HMAC)</td>
<td>1,806</td>
<td>86,688</td>
<td>1,333</td>
<td>95,976</td>
</tr>
<tr>
<td>Create Data (ECDSA)</td>
<td>451,215</td>
<td>21,658,320</td>
<td>269,314</td>
<td>19,390,608</td>
</tr>
<tr>
<td>Verify Data (ECDSA)</td>
<td>500,115</td>
<td>24,005,520</td>
<td>294,225</td>
<td>21,184,200</td>
</tr>
<tr>
<td>Get Data Name</td>
<td>3</td>
<td>144</td>
<td>2</td>
<td>144</td>
</tr>
<tr>
<td>Get Data Content</td>
<td>4</td>
<td>192</td>
<td>4</td>
<td>288</td>
</tr>
</tbody>
</table>

### Table II: Object code size of the NDN-RIOT API (in bytes)

<table>
<thead>
<tr>
<th>API</th>
<th>SAMR21-XPRO</th>
<th>ARMv7-M</th>
<th>ARMv6-M</th>
</tr>
</thead>
<tbody>
<tr>
<td>ndn_app_express_interest</td>
<td>192</td>
<td>160</td>
<td>96</td>
</tr>
<tr>
<td>ndn_app_schedule</td>
<td>192</td>
<td>160</td>
<td>96</td>
</tr>
<tr>
<td>ndn_data_get_content</td>
<td>451,215</td>
<td>21,658,320</td>
<td>269,314</td>
</tr>
<tr>
<td>ndn_data_create</td>
<td>451,215</td>
<td>21,658,320</td>
<td>269,314</td>
</tr>
<tr>
<td>ndn_interest_get_name</td>
<td>192</td>
<td>160</td>
<td>96</td>
</tr>
<tr>
<td>ndn_interest_create</td>
<td>192</td>
<td>160</td>
<td>96</td>
</tr>
<tr>
<td>ndn_name_get_size_from_block</td>
<td>192</td>
<td>160</td>
<td>96</td>
</tr>
<tr>
<td>ndn_name_append</td>
<td>192</td>
<td>160</td>
<td>96</td>
</tr>
</tbody>
</table>

### Table III: Overall memory usage of the skeleton NDN-RIOT application (in bytes)

<table>
<thead>
<tr>
<th>Consumer</th>
<th>Producer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flash</td>
<td>Flash</td>
</tr>
<tr>
<td>BSS</td>
<td>BSS</td>
</tr>
<tr>
<td>Data</td>
<td>Data</td>
</tr>
<tr>
<td>Code</td>
<td>Code</td>
</tr>
</tbody>
</table>

### Table IV: Execution time of the NDN-RIOT APIs

<table>
<thead>
<tr>
<th>Test Case</th>
<th>SAMR21-XPRO</th>
<th>IoTLab-M3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time (μs)</td>
<td>Cycles</td>
<td>Time (μs)</td>
</tr>
<tr>
<td>URI to Name</td>
<td>184</td>
<td>8,832</td>
</tr>
<tr>
<td>Get Name size</td>
<td>13</td>
<td>624</td>
</tr>
<tr>
<td>Get Name component</td>
<td>8</td>
<td>384</td>
</tr>
<tr>
<td>Append to Name</td>
<td>28</td>
<td>1,344</td>
</tr>
<tr>
<td>Create Interest</td>
<td>25</td>
<td>1,200</td>
</tr>
<tr>
<td>Get Interest Name</td>
<td>2</td>
<td>96</td>
</tr>
<tr>
<td>Create Data (HMAC)</td>
<td>1,806</td>
<td>86,688</td>
</tr>
<tr>
<td>Create Data (ECDSA)</td>
<td>451,215</td>
<td>21,658,320</td>
</tr>
<tr>
<td>Verify Data (ECDSA)</td>
<td>500,115</td>
<td>24,005,520</td>
</tr>
<tr>
<td>Get Data Name</td>
<td>3</td>
<td>144</td>
</tr>
<tr>
<td>Get Data Content</td>
<td>4</td>
<td>192</td>
</tr>
</tbody>
</table>

### Data Size

<table>
<thead>
<tr>
<th>Data Size</th>
<th>Cached?</th>
<th>Fragmented?</th>
<th>RTT (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>100 bytes</td>
<td>No</td>
<td>No</td>
<td>280</td>
</tr>
<tr>
<td></td>
<td>Remote</td>
<td>No</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>Local</td>
<td>No</td>
<td>&lt;1</td>
</tr>
<tr>
<td>196 bytes</td>
<td>No</td>
<td>Yes</td>
<td>286</td>
</tr>
<tr>
<td></td>
<td>Remote</td>
<td>Yes</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>Local</td>
<td>No</td>
<td>&lt;1</td>
</tr>
</tbody>
</table>
Work in Progress

• Energy consumption evaluation / optimizations
• Advanced forwarding strategy supportData discovery
• Nearby data discovery
• Pub-sub API on top of Interest/Data exchange

Help welcome!
Use Cases and Other IoT-Related NDN Efforts

• **NDN-BMS**: encryption-based access control

• **NDN-ACE**: authorization framework for actuation apps

• **NDN-IoT**: toolkit for NDN dev on Raspberry Pi
  – [https://github.com/remap/ndn-pi](https://github.com/remap/ndn-pi)

• **NDN on Arduino**: minimal app for Arduino
  – [https://github.com/ndncomm/ndn-btle](https://github.com/ndncomm/ndn-btle)