μNDN: NDN for Constrained IoT Platform

NDN Retreat 2016
Level of “constrainedness”

- **Raspberry Pi 3 Model B**
  - 1.2GHz quad-core ARMv8 (64-bit), plus GPU
  - 1GB RAM, external SD card, 802.11n & Bluetooth

- **Atmel SAM R21**
  - ARM Cortex-M0+ (32-bit), up to 48MHz
  - 32KB SRAM, 256KB Flash, 802.15.4 radio

- **Arduino Mega**
  - ATmega2560 microcontroller (8-bit, 16MHz)
  - 8KB SRAM, 256KB Flash
Progress by the NDN team

• Cross-compiled NFD for less-constrained platforms
  – Raspberry Pi
  – DD-WRT

• NDN for very constrained platforms
  – NDN APP over Arduino
  – NDN over RIOT-OS
μNDN concept

• Target the “middle-class” of devices
  – Enough resources to do something interesting
  – Capable of running a forwarder along the apps
• Small code size and memory usage
• Small packet and low data rate
Example: NDN for RIOT-OS

Software architecture of NDN on RIOT-OS
**Example: NDN for RIOT-OS**

<table>
<thead>
<tr>
<th>text</th>
<th>data</th>
<th>bss</th>
<th>dec</th>
<th>hex</th>
<th>file name</th>
</tr>
</thead>
<tbody>
<tr>
<td>39636</td>
<td>228</td>
<td>11204</td>
<td>51068</td>
<td>c77c</td>
<td>ndn_ping.elf</td>
</tr>
</tbody>
</table>

Code size & static memory usage (compiled for SAM R21 IoT board)
Discussion

• System design
  – What functionality to put in the micro-forwarder?
    • Routing protocols? Forwarding strategies? Caching policies?
• Link layer
  – Efficient forwarding over L2; energy efficiency;
• Security
  – Solutions using (mostly) symmetric crypto?
• Auto-configuration
  – How to bootstrap trust?
  – How to configure L3 & L2(?) connectivity?