YaNFD: Yet another NDN Forwarder

Eric Newberry, Xinyu Ma
UCLA

10th NDN Hackathon
October 18, 2020

Overview

- Examine what was done well in and what lessons could be learned from:
  - NFD (evaluation done before hackathon)
  - NDN-DPDK
  - ndn-lite
- Use this information to create “lessons learned”
- Make a number of important design decisions for YaNFD

Lessons Learned from NFD

- NFD started with a quite idealistic design, e.g., high modularity, but has deviated from these principles over time for practical reasons
    - E.g., interdependencies between modules have only increased over time
- Modularity can be useful, but too much can lead to heavyweight and code that is difficult to understand
    - No link services except GenericLinkService -- and features can already be enabled/disabled
- Need to make specific choice whether faces are interfaces (to a broadcast link) or links (to a single remote host)
    - Whether the underlying transport is or not is irrelevant, this concerns how presented to FW
- More than a single forwarding thread is required for an effective forwarder
- Need to keep up with research advances to improve forwarding speed
    - E.g., use character tries in data structures

Lessons Learned from NDN-DPDK

- Data structures for tables
    - Make an abstract interface to support different implementations
    - PIT Sharding
    - Multithreading locks
- Support different strategies
    - Choices: classes, shared-library, eBPF, WASM embedding, standalone program, etc.
- Not requiring the PIT token

Lessons Learned from ndn-lite

- Keep it simple
  - Too much overhead from an elegant design leads to decreased performance
- The environment should determine the architecture
  - Don’t try to force a single forwarder architecture for every environment
  - E.g., use NDN-DPDK for network backbones, ndn-lite for IoT, and YaNFD for desktop/server
- No Nacks
  - The lack of Nacks has significantly reduced the complexity of the forwarding pipeline and consumer APIs

Design Decisions for YaNFD

- Multi-threaded forwarder
    - Multiple forwarding threads, thread for each face, management thread, RIB thread(?)
- Modularity
    - LinkService does not need modularity, since features can be enabled/disabled independently
        - Include in base Face class, like NDN-DPDK
    - Modularity for transport types, strategies, caching algorithms, data structures(?)
- Data structures
    - Readers-writer locking for most data structures
    - However, shard PIT by thread (like NDN-DPDK)
        - Packets sent to forwarding thread based upon hash
        - PIT token does **not** need to be mandatory
        - Hash name to determine FW thread (if no token, hash k prefixes to send Data to k threads)

Design Decisions for YaNFD

- Modular data structures
  - For research and experimentation, use defined API for data structures (e.g., PIT, CS, FIB)
  - Allows implementations of data structures to be swapped in and out
    - Evaluate/use different lookup algorithms
  - → Ease forwarding efficiency research
- Provide better way to include strategies
  - See NDN-DPDK discussion
  - Shouldn’t require additions to forwarder codebase to use new strategies

The Plan from Here

- We have some major design decisions nailed down
- The next step is a more specific design (e.g., UML)
- Then, implementation and evaluation!
  - We have chosen to use Go for our implementation
  - Go is cross-platform and proven for use with NDN via NDN-DPDK
- Timeline:
  - Make significant progress on implementation by end of 2020
  - Finish basic implementation by March 2021
  - Finish evaluations and experimentation by May 2021 to catch Eric’s WQE deadline