YaNFD: Yet another Named Data Networking Forwarding Daemon

Eric Newberry, Xinyu Ma, Lixia Zhang

Forwarding in NDN

- Forwarding in IP is simple:
  - Find longest-prefix matching route in FIB → send packet on indicated NIC
- Forwarding in NDN is more complex:
  - Two pipelines: Interests and Data
  - In-network state: Pending Interest Table (PIT), Dead Nonce List (DNL)
  - In-network caching: Content Store (CS)
  - Intelligent forwarding decisions: Forwarding strategies

Existing Forwarders for NDN

- Numerous developed over the years
- Different target environments:
  - E.g., Desktop/laptop, mobile, embedded, network core
- Different goals:
  - Performance, extensibility, ease of use
- However, in existing forwarders:
  - High performance forwarders have high requirements/limited compatibility
  - Forwarders for general purpose computing generally unwieldy or complex
    - Often difficult to extend and use in novel research/development scenarios

Existing Forwarders for NDN: NFD

- Research prototype developed by the NDN team
- De-facto “official” forwarder for NDN
- C++, 156k lines of code (ndn-cxx library + NFD), ~412 MB compiled
- Strengths:
  - Extensively documented
  - Good overall design
- Limitations:
  - Single-threaded
  - Complex codebase and false modularity

Existing Forwarders for NDN: ndn-lite

- Low-overhead forwarder for embedded/constrained environments
- Integrated with a single application thread
  - Instead of being a separate process
- Written in C
- Simplifies portions of the NDN pipeline
  - Notably, removes negative acknowledgements (Nacks)
- Strengths
  - Simplified forwarder design compared to NFD
- Limitations
  - Unable to support beyond a single NDN application at once

Existing Forwarders for NDN: MW-NFD

- Fork of NFD that integrates multi-threading
- Forwarding pipeline split into *n* “worker” threads
- Interests dispatched to threads by hashing first *k* name components
- Data packets directed to threads using “PIT tokens”
- Strengths
  - Multi-threaded
  - High performance
- Limitations
  - 100% CPU polling used on input threads

Existing Forwarders for NDN: NDN-DPDK

- Extremely high performance forwarder for backbone networks
  - Up to 100 Gbps forwarding rates observed
- Uses DPDK to bypass the OS network stack
- Data plane in C, control plane in Go
- Dispatches packets via similar methods to MW-NFD
- Strengths
  - Shards data structures across threads to avoid locking
  - High performance
- Weaknesses
  - Only supports a subset of NIC hardware
  - Uses 100% CPU polling to process incoming traffic quickly

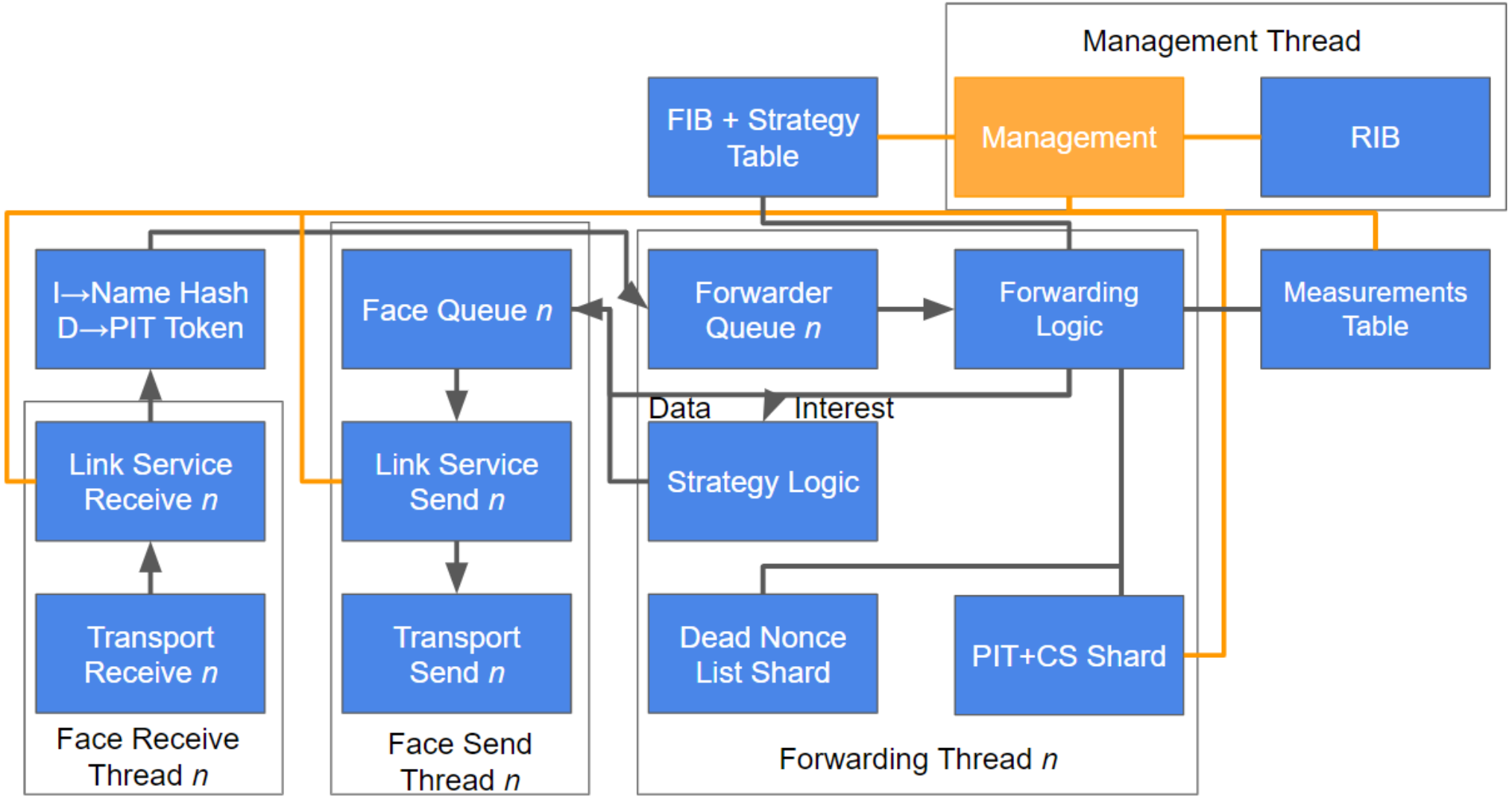
Designing a New Forwarder

- What do we want in a new forwarder?
  - High performance
  - Simple codebase to assist innovation
  - Compatibility with NDN wire format and existing applications
  - Applies “lessons learned” from existing forwarder designs
- Developed YaNFD to achieve these goals
  - Multi-threaded with sharded data structures
  - Simplified design and codebase
  - Reuses existing application and forwarder interface standards
  - Design influenced by strengths/limitations of existing NDN forwarders

Parallelizing NDN Pipelines

- Interest pipeline
  - PIT: Interests w/ different names update different entries
  - All other data structure accesses are read-only
- Data pipeline
  - Write accesses to many data structures
- Solution
  - Only read-only: Use global instances
  - Need to write: Shard data structures across threads

YaNFD Design Overview



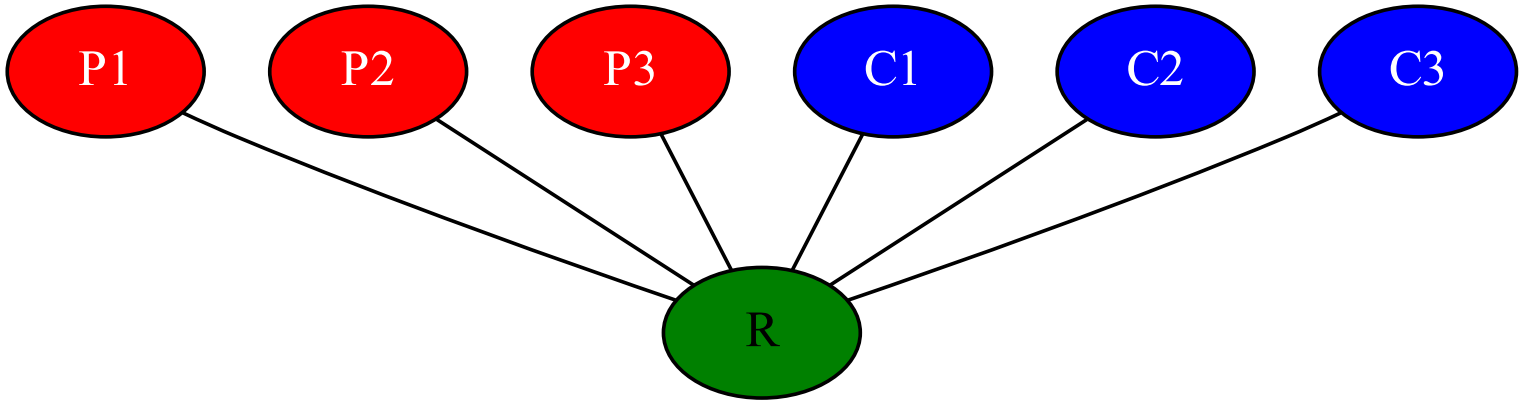
Data Structures: Sharded PIT/CS

- Pending Interest Table
  - Interest pipeline uses exact lookups
  - Data pipeline uses longest prefix matching
- Content Store
  - Uses longest prefix matching
  - However, use of CS is entirely opportunistic
- PIT and CS both use a tree of names
  - Combine into a single data structure (like NFD)
- Can shard PIT/CS across threads

Data Structures: Global FIB/RIB

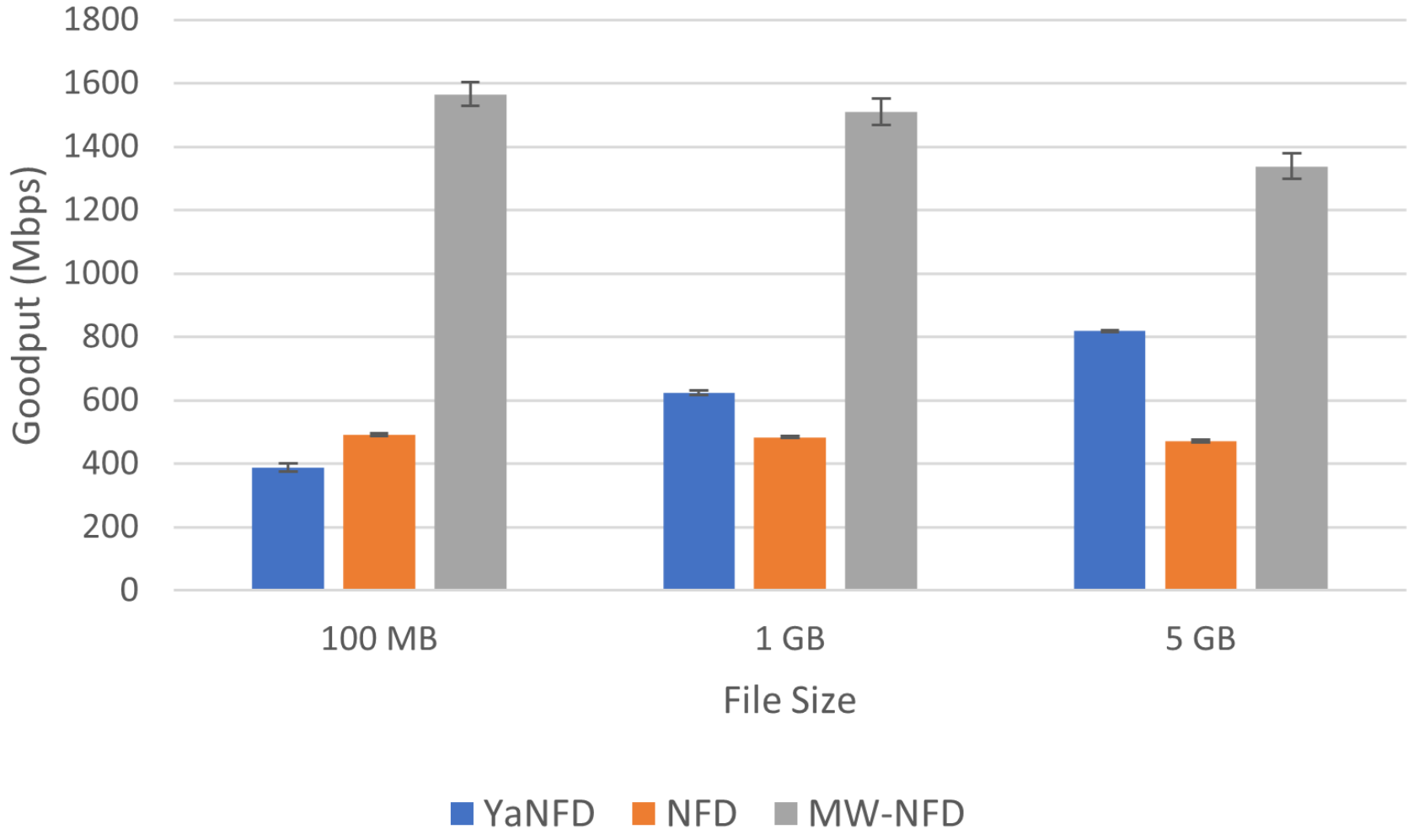
- In the forwarding pipelines, all accesses to FIB are read-only
- RIB is kept separate and periodically “flattened” into FIB
- Therefore, can have a global FIB with simultaneous access
  - Can be locked when being flattened from RIB

Evaluation: Scenario

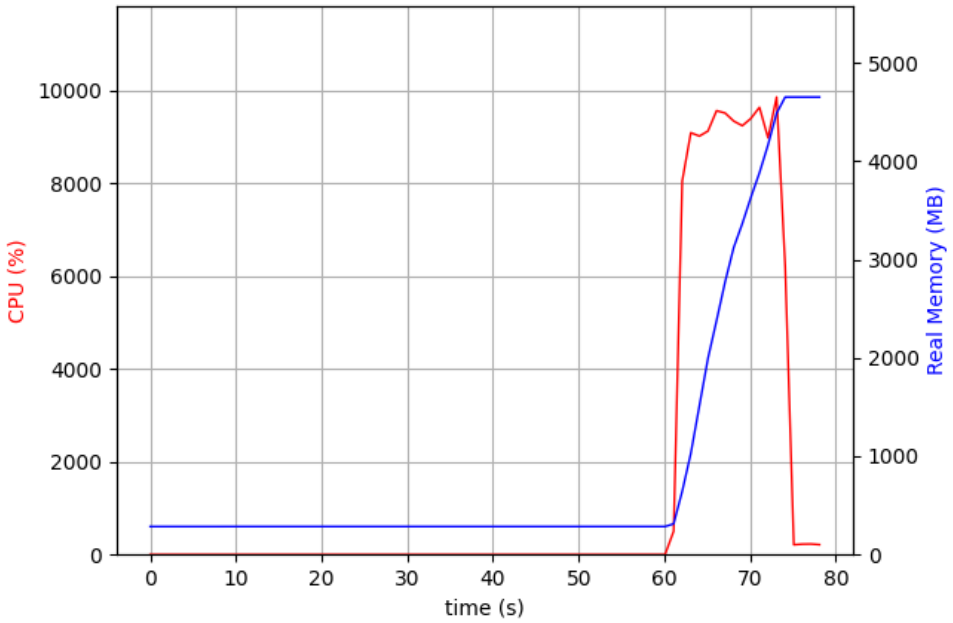


- Simultaneously transferring three large files via ndnchunks
- Each consumer requested content from a respective producer
- Producers, consumers, and forwarder running on a single node

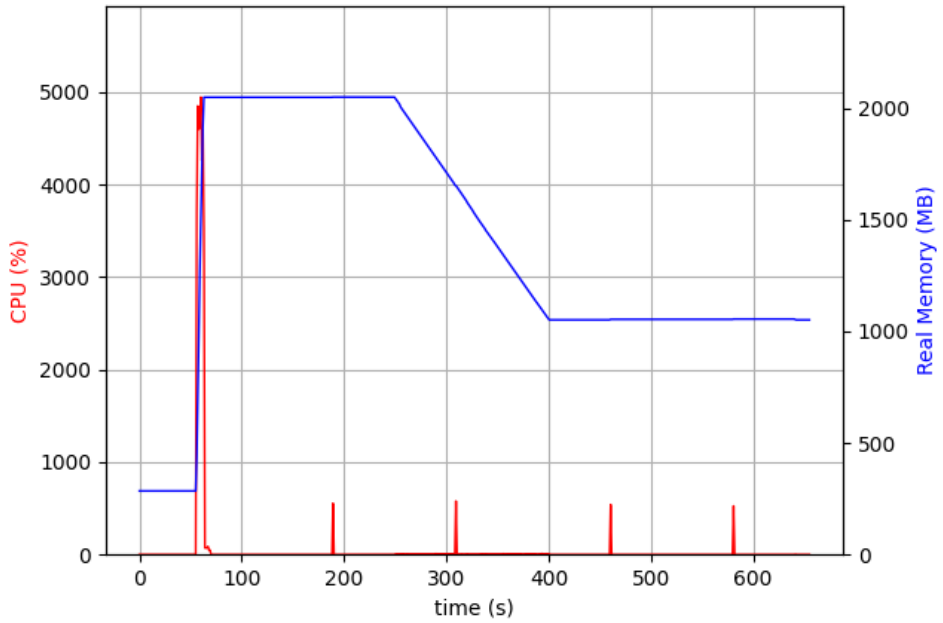
Evaluation: Performance



Evaluation: Overhead

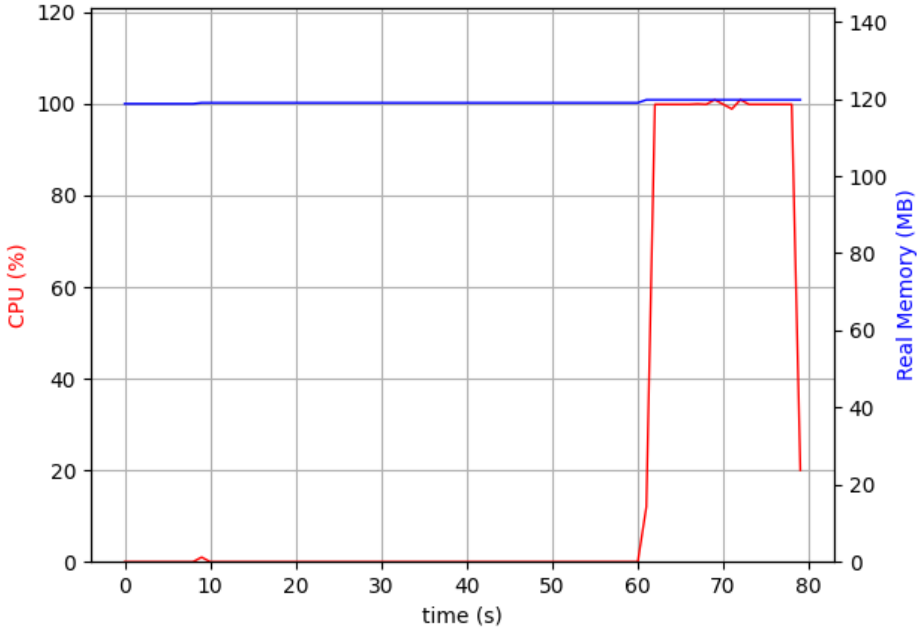


YaNFD

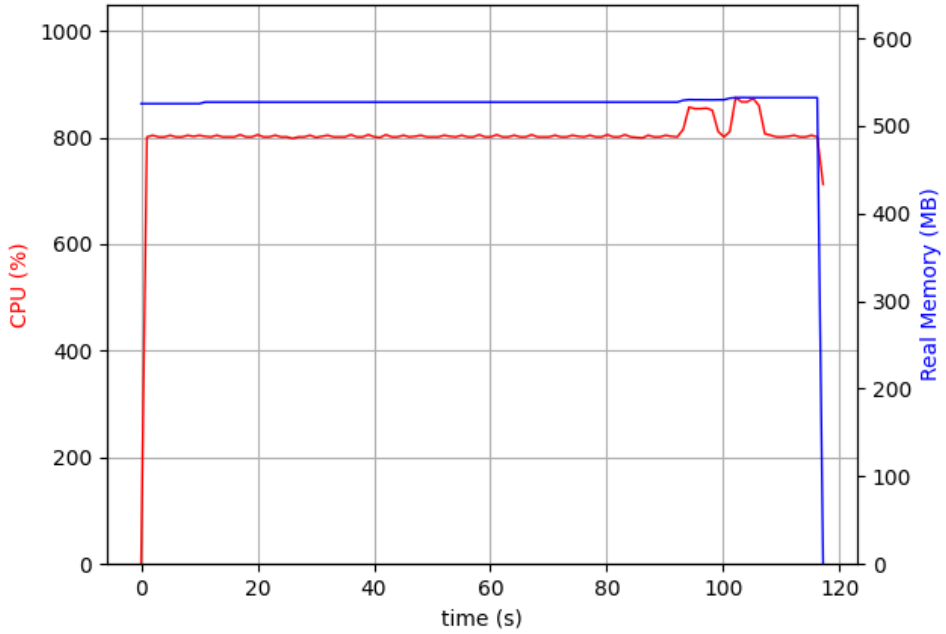


YaNFD (long term)

Evaluation: Overhead



NFD



MW-NFD

Evaluation: Overhead

- Software package size
  - NFD: 189 MB + 223 MB for ndn-cxx (~412 MB total)
  - MW-NFD: 217 MB + 233 MB for ndn-cxx (~450 MB total)
  - YaNFD: 6.7 MB
- Lines of code
  - NFD: ~82,000 + ~74,000 for ndn-cxx (~156,000 lines total)
  - YaNFD: ~13,000 lines

Lessons Learned from YaNFD

- Multi-threaded forwarding has inherent benefits to NDN throughput
- Garbage collection overhead can be significant
- Must consider impact of NDN forwarding on other applications
  - Particularly on multi-purpose edge devices that we target
- Multi-threading doesn’t always increase forwarding performance

Future Work

- Allow forwarding strategies to be dynamically loaded at runtime
- Add support for NLSR router package for NDN
  - NLSR requires NDN management protocol features not yet implemented
- Devise way to utilize PIT tokens in producer applications
  - Avoid prefix dispatching of Data packets with missing PIT tokens

Summary

- YaNFD is able to outperform NFD with a much simpler design
- Multi-threaded forwarding in NDN is feasible and desirable
- NDN forwarders can be both lightweight and effective
- Open source! https://github.com/named-data/YaNFD