NDN-DPDK Forwarder (ICNRG 2020 April Interim)

Junxiao Shi, Davide Pesavento, Lotfi Benmohamed
Advanced Network Technologies Division
National Institute of Standards and Technology

https://github.com/usnistgov/ndn-dpdk
NDN-DPDK High-Speed Forwarder

- NDN forwarding over Ethernet.
  - Goal: line speed on commodity hardware.
  - Achieved: 106Gbps between two ports.
    - CPU: Xeon Gold 6240; NIC: Mellanox ConnectX-5 100Gbps.
    - Benchmarking is underway. Preliminary results shown.

- Design highlights:
  - Parallel architecture.
  - Efficient data structures in pre-allocated memory pools.
  - User-space PCI drivers with hardware offloads.
  - Disk-based caching on NVMe drives.
  - FPGA acceleration.
Forwarder Architecture

input0
input1
Name Dispatch Table

fwd0
PIT+CS FIB

fwd1
PIT+CS FIB
crypto helper
disk service

output0
output1
FIB mgmt

input stage forwarding stage output stage
FIB Design

• FIB lookup: 2-stage LPM algorithm.
  • Inspired by "Named Data Networking on a Router: Fast and DoS-resistant Forwarding with Hash Tables".

• Management updates FIB via RCU.

• FIB entry has pointer to forwarding strategy.

• Strategy can store measurements on FIB entry.
  • Measurements granularity is same as FIB entry.

• Strategy updates FIB measurements without RCU.
  • Each forwarding thread has its own FIB partition, to avoid thread safety issues in measurements updates.
PIT Sharding & Interest Dispatching

- Each forwarding thread has a private PIT.
  - Hash table.
  - Non-thread-safe.

- Interest dispatching requirements:
  - Interests with same name must go to same PIT: required for Interest aggregation.
  - Interests with same name prefix should go to same PIT: needed for effective strategy decisions.

- Solution: dispatch Interests by hash of first two name components.
Dispatch Interest by Name

• Name Dispatch Table (NDT)
  • Map: hash of name prefix => fwd-thread ID
  • Thread safe: NDT is an array of atomic_int.
  • Many name prefixes share the same entry.

• In input threads:
  1. Compute hash of first two name components (configurable)
  2. Choose NDT bucket: hash % NDT.size()
  3. Dispatch to fwd-thread

```
<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

fwd0 PIT+CS
fwd1 PIT+CS
```
PIT Sharding & Data Dispatching

• Data/Nack must go to the same forwarding thread that forwarded the Interest.
  • Other PITs do not know about the Interest.

• Name dispatching works most of the time, except:
  • Interest /A CanBePrefix=1 goes to NDT[H(/A)].
  • Data /A/B/1 goes to NDT[H(/A/B)].

• Solution: use PIT token to associate Interest and Data.

• PIT token is an opaque token that encodes:
  a) forwarding thread ID.
  b) PIT entry index.
Dispatch Data/Nack by PIT Token

- Every outgoing Interest carries a PIT token.
  - 64-bit field in link layer header. Hop by hop.
- Data/Nack must carry the same PIT token.
  a) Packet goes to the same fwd-thread that sent Interest.
  b) PIT entry index is for accelerating PIT lookup.
Prefix Match in CS

- In-Network Name Discovery:
  - Interests should be able to use incomplete names to retrieve Data packets.
- CS is a hash table, which only supports exact match.
- Solution: indirect entries.

This is the Interest name. We assume that consumers use the same Interest name for name discovery.

This is the Data name, so that an Interest with exact name can also match the cached Data.

/A/B indirect entry

/A/B/1 direct entry with Data packet

An indirect entry must be evicted before the direct entry.

A direct entry may associate with multiple indirect entries.
Hardware Offloads

• Most NICs today support RSS rules matching on Ethernet/IP header fields.
  • It's being used to support multiple faces on the same NIC, distinguished by MAC addresses.

• Preliminary benchmark insights:
  • Input thread is bottleneck with ≥8 forwarding threads.
  • NUMA crossing reduces throughput by 12~25%.

• Need more powerful RSS rules to alleviate these bottlenecks.

• eBPF and FPGA might help, but:
  • Limited products.
  • High development effort.
Wish List for NIC Vendors

• Match an offset into Ethernet frame. Useful for:
  • Distinguish Interest vs Data.
  • Read first octet of PIT token (i.e. forwarding thread ID), so that Data is DMA'ed to the NUMA socket of the forwarding thread.
  • Note: may need changes to hop-by-hop header format.

• Randomly dispatch to multiple queues. Useful for:
  • More than one input threads can decode and process Interests from the same NIC.

• Ultimately: NDN-NIC.
Implicit Digest

- Challenge for a high-speed forwarder:
  - Digest computation is slower than regular forwarding.
  - One slow packet => hundreds of other packets dropped.

- Solution: crypto helper thread.

1. Fwd-thread determines digest is required to process a packet.
2. Pass the packet to a crypto helper thread.
3. Compute digest in crypto device (e.g. OpenSSL, Intel QuickAssist).
4. Re-dispatch packet to forwarding thread.
5. Re-process packet with the computed digest.
Forwarding Hint Support

• Forwarding hint is a routing scalability solution.
  • Interest /ndnsim/frontpage <= unrouteable name
    FH: /edu/ucla, /telia/terabits <= routable names

• To process Interest with forwarding hint:
  1. Lookup FIB with forwarding hint delegation names.
  2. The first delegation name found in FIB is called the chosen forwarding hint.
  3. Interests with different chosen forwarding hints cannot be aggregated in PIT.

• Data is matched to PIT entry via PIT token.
  • Content Store for each chosen forwarding hint is logically isolated to prevent cache poisoning.
If I'm to Implement Reflexive Forwarding

• Updating FIB from forwarding threads via RCU is too slow.
  • Skip the FIB. Use PIT instead.

• Reflexive Interests contain PIT token of original Interest in the forwarding hint.
  • FH will change hop-by-hop.

• Reflexive Interests/Data can have normal names.
  • No T_REFLEXIVE_NAME component.
  • No dependency on consumer's randomness quality.
  • No need to encapsulate Data.
Dispatch and Forward Reflexive Interests

- **Input thread:** dispatch with PIT token in FH.
- **Forwarding thread:** (ID=0x01 in this example)
  - Locate PIT entry of original Interest by PIT token in FH.
  - Verify that current Interest matches reflexive-name in original Interest, and there's no T_RETURN_PROHIBITED.

---

### PIT table examples:

**PIT table 1**

- **Thread:** Input thread
- **Interest Entry:**
  - PIT token 0x08B00003
  - Interest: /command/f44c6682
  - reflexive-name: /upload/4a13615c

**PIT table 2**

- **Thread:** Forwarding thread (ID=0x01)
- **Interest Entry:**
  - PIT token 0x01C00003
  - Interest: /upload/4a13615c
  - FH: /reflexive=08B00003

**PIT table 3**

- **Thread:** Input thread
- **Data Entry:**
  - PIT token 0x08B00003
  - Data: /command/f44c6682

**PIT table 4**

- **Thread:** Forwarding thread (ID=0x01)
- **Data Entry:**
  - PIT token 0x01C00003
  - Data: /upload/4a13615c
References

• W. So, A. Narayanan and D. Oran, "Named data networking on a router: Fast and DoS-resistant forwarding with hash tables," ANCS 2013

• Junxiao Shi, "Named Data Networking in Local Area Networks," University of Arizona PhD Dissertation
  • Chapter 3: NDN forwarding behavior

  • Forwarding Hint

• David R. Oran and Dirk Kutscher, "Reflexive Forwarding for CCNx and NDN Protocols," Internet-Draft 2020