OF-CCN: CCN over OpenFlow

NDN hands-on Workshop
Junho Suh
(jhsuh@mmlab.snu.ac.kr)
Contents

- Motivation
- Problem Definition
- Design Overview
- Conclusion
Motivation (1/2)

• Communication paradigm in Internet has been changed
  – Users no longer want to know “where”
  – Some overlay approaches
    • CDN: indirect DNS query
    • P2P: tracks all the hosts with content
  – Cannot be a fundamental solution
    • Patch-on solutions
    • Retain current IP problems
Motivation (2/2)

• Content-centric Networking (CCN) is one of the candidate for future Internet
  – Efficient content publish/subscribe
    • Route-by-name approach
    • In-network caches
    • Requests would be guided toward nearby content
  – Resolve/mitigate current IP problems
    • Traffic explosion
    • Mobility, multicast/anycast, multipath/multisource
    • Security
Problem Definition

• Problems in deployment of CCN
  – Clean-slate approach
    • Alternative to IP
      – New naming schemes
      – Difficult for network operators to deploy
  – Overlay approach
    • E.g., CCNx over IP
      – Cannot fully take advantages of CCN benefits
Objective

• Provide CCN functionalities by using OpenFlow programmability
  – Realize CCN node model over OpenFlow
    • Content naming
    • In-networking caching
    • Route setup
    • Network monitoring
    • Security
Question 1

Why is SDN/OpenFlow a candidate to realize CCN?

- Programmability in network
  - Clean slate approach can be directly applied to the network through minimal modifications
- Currently many vendors support OpenFlow
  - HP, IBM, NetGear, Juniper, Cisco, NEC, Broadcom
- Manageability
  - Centralized controller
    - Fine-grained network control
    - Near-optimal throughput
  - But no consideration on monitoring CCN nodes
Question 2

• How can SDN/OpenFlow realize CCN?
  – R1) Naming
    • Content names instead of IP address
      – Ex) youtube.com/my_video.avi/s1
    • Named data interpretation in OpenFlow nodes
  – R2) In-network caching
    • Content caching/retrieval in line-card
  – R3) Decentralized CS/PIT updates
    • Currently, tables are updated only by the centralized controller
  – R4) Network monitoring
    • Traffic, caching
Contributions

- To the best of our knowledge, this is the first trial for realizing CCN over OpenFlow
  - Validate feasibility of CCN
  - Use Off-the-shelf devices
- Extending OpenFlow to realize CCN node model
  - In-network caching
  - Extended action(s)
- Extend CCN to add monitoring node
  - Traffic-aware content delivery
OF-CCN: Overall Framework

- OF Controller
- Monitoring Node
- OF-Switch
- Legacy IP Network
- OF API
- Cache Manager
  - Routing
  - Topology
  - OF API
- OF Switch
  - Face 1
  - Face N
  - Flow Table
  - CS → PIT → FIB
Monitoring Node

- Two components
  - Cache manager
    - Manages all in-network caches in charge
    - Ex) coordination mechanism across the network
  - Traffic monitor
    - Monitors on-going flows
    - Ex) load balancing, QoS
OF-Switch

• In-network caching
  – Extending OF-switches to equip with storage such as HDD or SSD
    • Extending OpenFlow protocol to manage in-network caches
    • Matching name inside OF-switches flow table
  – Caching policy
    • Autonomous policy such as LRU or LFU
    • Managed policy in cache manager
Interest Packet

- **Msg type**
  - Interest
- **Hashed URL**
  - E.g., snu.ac.kr/mmlab/doctors/jhsuh/video.avi
- **Seg#**
  - Integer numbers

*H: hash function*
Data Packet

- **Msg type**
  - Data
- **Hashed URL**
  - E.g., snu.ac.kr/mmlab/doctors/jhsuh/video.avi
- **Seg#**
  - Integer numbers

```
<table>
<thead>
<tr>
<th>D</th>
<th>H(URL)</th>
<th>Seg#</th>
<th>Data Payload</th>
</tr>
</thead>
<tbody>
<tr>
<td>8bit</td>
<td>128bit</td>
<td>64bit</td>
<td></td>
</tr>
</tbody>
</table>
```

*H: hash function*
Matching in CS

• Table structure in an OF-switch

- CS
- PIT
- FIB

Switch Port | MAC src | MAC dst | Eth type | Type | HASH (URL) | Seg# | Actions | Statistics #packets+#bytes

• Extended actions
  – Store/Load content
  – Add an entry
Matching in PIT

• Table structure in an OF-switch

  HW Flow Tables in a OF-switch

  CS → PIT → FIB

  Entry

  Switch Port | MAC src | MAC dst | Eth type | Type | HASH (URL) | Seg# | Actions | Statistics #packets+#bytes

• Extended actions
  – Add an entry when a packet is unmatched
Matching in FIB

- Table structure in an OF-switch

- FIBs are generated by the routing component in OF-CCN
- Only type and Hashed URL are matched
Content Advertisements

1: Advertising a new content
2: Switch -> NOX
4: Constructing FIBs
6: Advertising FIBs
Conclusion

• Realize CCN functionalities over OpenFlow
• Use off-the-shelf OpenFlow switches
  – Need to upgrade firmware of OF-switches
• Add monitoring node
  – Traffic-aware content delivery