DumbNet: A Smart Data Center Network Fabric with Dumb Switches

Yiran Li¹ Da Wei¹ Xiaoqi Chen² Ziheng Song⁴
Ruihan Wu¹ Yuxing Li¹ Xin Jin³ Wei Xu¹

¹Tsinghua University ²Princeton University ³Johns Hopkins University ⁴Beijing University of Posts and Telecommunications
Function Division Between Network & Software

- Packet routing
- Network failure handling
- Name service
- ......
- Dynamic routing
- Coflow
- Optical network
- ......
DCN with Internet Technologies

- Internet protocols are too complicated for DCN
- Applications hard to take advantage of network
Supercomputer Network Adopts Source Routing

- Only supports fixed topology
- Limited failure-tolerance
SDN Uses Centralized Controller

- State limits port number and hard to manage
- State consistency problem
DumbNet Moves Logics and States into Hosts

- Switch has fixed simple logics for performance
Challenges

• Topology discovery
  • Host-based topology discovery using probe packets

• Topology change
  • Two-stage topology maintenance protocol
**DumbNet Tag-Based Routing**

1. Host queries controller for destination
2. Host agent adds tags according to query result
3. Tags are popped at each hop
Topology Discovery Solution: Host-Based Probe

Breadth-first search

Queue 1

Probe 1, 2, 3, ..., n
Topology Change Solution: Update Notification

1. Limited range hardware broadcast by switches
2. Notification flood by hosts
Topology Change Solution: Quick Fix Patch

- Controller generates and sends patch

Controller sends Patch 1-4-2-5 to A and B
Improve Host Agent Tag Insertion by Caching

- Controller returns subgraph with multiple paths
- K-shortest paths cached in PathTable
Flowlet Implements Easily on **DumbNet**

- **Supporting flowlet in router**
  - Remember available paths
  - Track flow states
  - Extra logics

- **DumbNet**: adds timestamp to paths in PathTable
  - Use previous path if it is recently used
  - Otherwise randomly select from k-shortest paths
DumbNet Helps Integrate Network & Software

Application

$\text{tagged}$

DumbNet Agent

$g(pkt)$

Checker

Network

Application

$\text{tagged}$

$g(pkt)$

$\text{tagged}$

Network
DumbNet Implementation is Simple

• C/C++ implementation

<table>
<thead>
<tr>
<th>Agent</th>
<th>Discovery</th>
<th>Maintenance</th>
<th>Graph</th>
<th>Total</th>
<th>+Flowlet</th>
</tr>
</thead>
<tbody>
<tr>
<td>5000</td>
<td>600</td>
<td>200</td>
<td>1700</td>
<td>7500</td>
<td>100</td>
</tr>
</tbody>
</table>

• FPGA implementation
  • 90% less resource consumption of OpenFlow demo (4-port)
Performance Evaluation Setup

• Emulation
  • 24-core Intel CPU

• Experiment
  • 7 × Arista 7050, 64-port, 10GE
  • 27 × servers, 6-core Xeon E5, 128GB, 10GE
  • DPDK + MPLS
Topology Discovery Scales Well

Limited by CPU

Time / sec

0 100 200 300 400 500

Number of switches in the network

Cube-corner, Cube-center, FatTree

Limited by CPU
Topology Maintenance Execution Time

CDF (Delay)

0 2 4 6 8

Notification delay / ms

0.2 0.4 0.6 0.8 1

Link Failure Msg

Topology Patch Msg

0 100 200 300

Rate / Mbits/sec

Time / ms

Link Failure
DumbNet Based Flowlet Easily Optimizes Real World Application Performance

![Graph comparing DumbNet with Flowlet, DumbNet, and No-op DPDK]

**Better**
Conclusion

• How to divide responsibilities between switches and hosts?
• How to better integrate network with software?

• What else must we add back to the dumb switches?
Thanks!
Q&A