Routing over eXplicit Path in Data Centers: Design and Applications

Kai Chen
The Sing Group, HKUST
@SJUT, July 24
In real life, humans are smart enough to make the right choice.
Background and Motivation

But, in DCNs, ECMP is not smart enough to make right choice

We need explicit path control over the multiple paths
The Case for Explicit Path Control - 1

If 300Mbps bandwidth required between X and Y, e.g., for network virtualization or provisioned IOPS storage (Amazon EBS), we should leverage explicit path control to choose P1.

Remaining bandwidth on P1, P2, P3: 300Mbps, 100Mbps, 50Mbps.

Explicit path control makes end-to-end bandwidth provisioning easy to enforce.
Explicit path control makes network updating (e.g., ZUpdate [Sigcomm’13]) easy to conduct.
Explicit path control can be leveraged to arrange non-interfering parallel paths to reduce shuffle time.
Cont., Many Other Application Cases

• Traffic engineering, e.g., MicroTE [CoNEXT’11], B4, SWAN [SIGCOMM’13]
• Flow scheduling, e.g., Hedera [NSDI’10]
• Power saving, e.g., ElasticTree [NSDI’10]
• Multi-path transport, e.g., MPTCP [SIGCOMM’11]
• Network diagnosis/failure handling, e.g., NetPilot [SIGCOMM’12]
• …

All of them require or benefit from explicit path control.
State-of-the-art Implementations

- **OpenFlow**: scalability issue, 1-2K forwarding entries in most today’s commodity switches; 4K entries in the next generation Broadcom Trident-II chipset.
  - Simply, a TE over 10s nodes require >20K entries;
  - Existing work (e.g., SWAN, ZUpdate): dynamically sharing limited entries by identifying working sets of paths, complicated and heavy-weight;

- **MPLS**: the # of tunnels that existing MPLS routers support is quite limited and insufficient for data center scale.

- **Source routing**: usually implemented in software and slow paths, most existing commodity switches do not support source routing.
  - ...

Our RoX Idea

• Key observation motivating RoX
  – IP table in commodity switches becoming very large (144K).
    • E.g., Arista 7050QX (144K), or Broadcom Strata XGS Trident-II (128K)

• Natural idea of RoX
  – Harness IP table (cheap but much large) instead of OpenFlow forwarding table (expensive and scarce) to implement explicit path control.

• One sentence describing RoX
  – Explicitly identify a path with a path ID and pre-install all these IDs using IP table.
RoX’s Challenges and Opportunities

• What paths to consider?
  – By no means enumerate *all possible* paths, exponential
  – Our key observation: DCNs have *desired* paths by design
    • E.g., k-port Fattree: \(k^2/4\) paths between any two ToRs;
    • n-layer Bcube: \((n+1)\) paths between any two servers.
  – Our first step: consider all these desired paths.

• Challenges:
  – Even desired paths set is very large
    • E.g., Fattree(64) has over \(2^{32}\) desired paths between ToRs, even a 32-bit IP address cannot express!

• Opportunities:
  – DCNs under centralized control, allowing sophisticated ID compression
Fattree topology

VL2 (Sigcomm’09)

BCube (Sigcomm’09)

HyperX (SC’09)
RoX’s Challenges and Opportunities

• What paths to consider?
  – By no means enumerate all possible paths, exponential
  – Our key observation: DCNs have desired paths by design
    • E.g., k-port Fattree: $k^2/4$ paths b/t any two ToRs;
    • n-layer Bcube: $(n+1)$ paths b/t any two servers.
  – Our first step: consider all these desired paths.

• Challenges:
  – Even desired paths set is very large
    • E.g., Fattree(64) has over $2^{32}$ desired paths between ToRs, even a 32-bit IP address cannot express!

• Opportunities:
  – DCNs, centralized control, allows sophisticated ID compression
RoX’s Two-step Compression Algorithm

Paths → Path sets → IP prefix entries

Reduce unique IDs
Reduce prefixes

Basically, two ways to aggregate paths into path sets:
- Convergent paths → path set;
- Disjoint paths → path set;

<table>
<thead>
<tr>
<th>Path set</th>
<th>Outgoing port</th>
<th>ID encoding (bad)</th>
<th>ID encoding (good)</th>
</tr>
</thead>
<tbody>
<tr>
<td>pathset₀</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>pathset₁</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>pathset₂</td>
<td>2</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>pathset₃</td>
<td>0</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>pathset₄</td>
<td>1</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>pathset₅</td>
<td>2</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>Prefix</th>
<th>Outgoing port</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>001</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>010</td>
<td>2</td>
</tr>
<tr>
<td>0,3</td>
<td>0**</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>101</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>Prefix</th>
<th>Outgoing port</th>
</tr>
</thead>
<tbody>
<tr>
<td>0,1</td>
<td>00*</td>
<td>0</td>
</tr>
<tr>
<td>2,3</td>
<td>01*</td>
<td>1</td>
</tr>
<tr>
<td>4,5</td>
<td>1**</td>
<td>2</td>
</tr>
</tbody>
</table>
## Evaluation on RoX’s Scalability

<table>
<thead>
<tr>
<th>DCNs</th>
<th>Nodes #</th>
<th>Links #</th>
<th>Original paths #</th>
<th>Max. entries #</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fattree(4)</td>
<td>36</td>
<td>48</td>
<td>224</td>
<td>14</td>
</tr>
<tr>
<td>Fattree(8)</td>
<td>208</td>
<td>384</td>
<td>15,872</td>
<td>116</td>
</tr>
<tr>
<td>Fattree(16)</td>
<td>1,344</td>
<td>3,072</td>
<td>1,040,384</td>
<td>968</td>
</tr>
<tr>
<td>Fattree(32)</td>
<td>9,472</td>
<td>24,576</td>
<td>66,977,792</td>
<td>7,952</td>
</tr>
<tr>
<td>Fattree(64)</td>
<td>70,656</td>
<td>196,608</td>
<td>4,292,870,144</td>
<td>64,544</td>
</tr>
<tr>
<td>BCube(4, 1)</td>
<td>24</td>
<td>32</td>
<td>480</td>
<td>9</td>
</tr>
<tr>
<td>BCube(4, 2)</td>
<td>112</td>
<td>192</td>
<td>12,096</td>
<td>108</td>
</tr>
<tr>
<td>BCube(8, 2)</td>
<td>704</td>
<td>1,536</td>
<td>784,896</td>
<td>522</td>
</tr>
<tr>
<td>BCube(8, 3)</td>
<td>6,144</td>
<td>16,384</td>
<td>67,092,480</td>
<td>4,989</td>
</tr>
<tr>
<td>BCube(8, 4)</td>
<td>53,248</td>
<td>163,840</td>
<td>5,368,545,280</td>
<td>47,731</td>
</tr>
<tr>
<td>VL2(10, 4, 20)</td>
<td>219</td>
<td>240</td>
<td>900</td>
<td>30</td>
</tr>
<tr>
<td>VL2(20, 8, 40)</td>
<td>1,658</td>
<td>1,760</td>
<td>31,200</td>
<td>310</td>
</tr>
<tr>
<td>VL2(40, 16, 60)</td>
<td>9,796</td>
<td>10,240</td>
<td>1,017,600</td>
<td>2,820</td>
</tr>
<tr>
<td>VL2(80, 64, 80)</td>
<td>103,784</td>
<td>107,520</td>
<td>130,969,600</td>
<td>49,640</td>
</tr>
<tr>
<td>VL2(100, 96, 100)</td>
<td>242,546</td>
<td>249,600</td>
<td>575,760,000</td>
<td>117,550</td>
</tr>
<tr>
<td>HyperX(1, 4, 20)</td>
<td>84</td>
<td>86</td>
<td>12</td>
<td>3</td>
</tr>
<tr>
<td>HyperX(2, 4, 40)</td>
<td>656</td>
<td>688</td>
<td>480</td>
<td>20</td>
</tr>
<tr>
<td>HyperX(3, 4, 60)</td>
<td>3,904</td>
<td>4,128</td>
<td>12,096</td>
<td>107</td>
</tr>
<tr>
<td>HyperX(4, 10, 80)</td>
<td>810,000</td>
<td>980,000</td>
<td>399,960,000</td>
<td>8,732</td>
</tr>
<tr>
<td>HyperX(4, 16, 100)</td>
<td>6,619,136</td>
<td>8,519,680</td>
<td>17,179,607,040</td>
<td>36,164</td>
</tr>
</tbody>
</table>
RoX Application Showcase #1: Provisioned IOPS

We leveraged RoX to make necessary I/O bandwidth (required for Amazon EBS’s provisioned IOPS) easier to implement.
RoX Application Showcase #2: Zero-loss network update

Path P1: T1 -> A1 -> T3;
Path P2: T1 -> A2 -> T3;
Path P3: T1 -> A3 -> T3.

Time $t_1$: move $f_3$ from P2 to P3;
Time $t_2$: move $f_1$ from P1 to P2;
Time $t_3$: move $f_1$ from P2 to P1;
Time $t_4$: move $f_3$ from P3 to P2.

We leveraged RoX to assist ZUpdate (SIGCOMM’13) to accomplish data center network update at zero loss.
We leveraged RoX to accurately enforce the VDC with bandwidth guarantees (e.g., SecondNet [CoNEXT’10], Oktopus [SIGCOMM’11], TIVC [SIGCOMM’12]).
RoX Application Showcase #4: Mapreduce data shuffle

We leveraged RoX to select non-conflict parallel paths to speed up the many-to-many Map-reduce data shuffle.
Summary

- **Design:** at its very core, RoX builds on
  - A simple concept of path ID to express an end-to-end path,
  - A bold idea of pre-installing all desired paths into IP tables,
  - A deep, effective algorithm of paths to IP prefix entries compression that translates the idea into practice.

- **Application**
  - Scalable, very easy to implement,
  - Practical, no modification on existing commodity switches,
  - Can be incorporated into many applications and directly benefit them,
  - Our other projects heavily rely on RoX for routing.

- **Next step**
  - RoX implemented on Windows/Linux platforms, we will make RoX Linux code publicly available with the hope to benefit the community.
Thanks, Q&A