Virtual Layer 2: A Scalable and Flexible Data-Center Network

Microsoft Research

Changhoon Kim
Tenets of Cloud-Service Data Center

• **Agility**: Assign any servers to any services
  – Boosts cloud utilization

• **Scaling out**: Use large pools of commodities
  – Achieves reliability, performance, low cost
What is VL2?

The first DC network that enables agility in a scaled-out fashion

• Why is agility important?
  – Today’s DC network inhibits the deployment of other technical advances toward agility

• With VL2, cloud DCs can enjoy agility in full
Status Quo: Conventional DC Network

**Internet**

**DC-Layer 3**

**DC-Layer 2**

Key
- **CR** = Core Router (L3)
- **AR** = Access Router (L3)
- **S** = Ethernet Switch (L2)
- **A** = Rack of app. servers

~ 1,000 servers/pod == IP subnet

Reference – “Data Center: Load balancing Data Center Services”, Cisco 2004
Conventional DC Network Problems

- Dependence on high-cost proprietary routers
- Extremely limited server-to-server capacity
And More Problems ...

- Resource fragmentation, significantly lowering cloud utilization (and cost-efficiency)
Know Your Cloud DC: Challenges

• Instrumented a large cluster used for data mining and identified distinctive traffic patterns

• Traffic patterns are **highly volatile**
  – A large number of distinctive patterns even in a day

• Traffic patterns are **unpredictable**
  – Correlation between patterns very weak

 Optimization should be done frequently and rapidly
Know Your Cloud DC: Opportunities

• DC controller knows **everything** about **hosts**

• Host OS’s are easily **customizable**

• **Probabilistic** flow distribution would work well enough, because ... 
  – Flows are numerous and not huge – no elephants!
  – Commodity switch-to-switch links are substantially thicker (~ 10x) than the maximum thickness of a flow

DC network can be made simple
All We Need is Just a Huge L2 Switch, or an Abstraction of One
## Specific Objectives and Solutions

<table>
<thead>
<tr>
<th>Objective</th>
<th>Approach</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Layer-2 semantics</td>
<td>Employ flat addressing</td>
<td>Name-location separation &amp; resolution service</td>
</tr>
<tr>
<td>2. Uniform high capacity between servers</td>
<td>Guarantee bandwidth for hose-model traffic</td>
<td>Flow-based random traffic indirection (Valiant LB)</td>
</tr>
<tr>
<td>3. Performance Isolation</td>
<td>Enforce hose model using existing mechanisms only</td>
<td>TCP</td>
</tr>
</tbody>
</table>
Addressing and Routing: Name-Location Separation

Cope with host churns with very little overhead

**VL2** Switches run link-state routing and maintain only switch-level topology

- Allows to use low-cost switches
- Protects network and hosts from host-state churn
- Obviates host and switch reconfiguration

Servers use flat names

Directory Service

Lookup & Response
Example Topology: Clos Network

Offer huge aggr capacity and multi paths at modest cost

<table>
<thead>
<tr>
<th>D (# of 10G ports)</th>
<th>Max DC size (# of Servers)</th>
</tr>
</thead>
<tbody>
<tr>
<td>48</td>
<td>11,520</td>
</tr>
<tr>
<td>96</td>
<td>46,080</td>
</tr>
<tr>
<td>144</td>
<td>103,680</td>
</tr>
</tbody>
</table>

20*($DK/4$) Servers
Traffic Forwarding: Random Indirection

Cope with arbitrary TMs with very little overhead

[ ECMP + IP Anycast ]

- Harness huge bisection bandwidth
- Obviate esoteric traffic engineering or optimization
- Ensure robustness to failures
- Work with switch mechanisms available today

Equal Cost Multi Path Forwarding !

1. Must spread traffic
2. Must ensure dst independence
Does VL2 Ensure Uniform High Capacity?

• How “high” and “uniform” can it get?
  – Performed all-to-all data shuffle tests, then measured aggregate and per-flow goodput

<table>
<thead>
<tr>
<th>Goodput efficiency</th>
<th>94%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fairness $^$ between flows</td>
<td>0.995</td>
</tr>
</tbody>
</table>

$^\$ Jain’s fairness index defined as $(\sum x_i)^2/(n \cdot \sum x_i^2)$

• The cost for flow-based random spreading

![](chart.png)

Fairness of Aggr-to-Int links’ utilization
VL2 Conclusion

• VL2 achieves **agility at scale** via
  1. L2 semantics
  2. Uniform high capacity between servers
  3. Performance isolation between services

**Lessons**

• Randomization can tame volatility
• Add functionality where you have control
• There’s no need to wait!