Meshed Tree Routing in Folded-Clos Topologies

PETER WILLIS & NIRMALA SHENOY, PH.D.

08/22/2022
Research Motivation

Data centers are growing in size and operational complexity.
- Thousands of servers.
- Variety of use-cases and applications are growing.

The Data Center Network (DCN) connects the servers so they can exchange data.
- DCNs have to be highly reliable and resilient.
- DCNs have to scale and should have high availability.

Innovation in the DCN control plane needs to keep pace with the advancement of other aspects of data centers.
- Trends in modern DCNs lends itself to optimizations.
Folded-Clos Topology

- Folded-Clos topology is popular in DCNs and has several attractive features.
  - High bisection bandwidth.
  - Rearrangably non-blocking.
  - Scales out via commodity hardware.
  - Equal-cost multi-path routing between servers.

Hence we used the folded-Clos topology in our project.
Existing Solutions

Utilize a Link-State Routing Protocol.
- OSPF, IS-IS.
- Flooding is problematic.

Utilize the Border Gateway Protocol (BGP).
- Path-vector based.
- Built for inter-AS routing, modified (retrofitted) for DCNs (RFC 7938).

Utilize a Topology-Aware protocol.
- Routing in Fat Trees (RIFT).
- Link State Vector Routing (LSVR) \(\rightarrow\) Another BGP modification.
The Meshed Tree Algorithm and Protocol

A control and data plane solution designed to use the attributes of folded-Clos topologies.

Uses virtual identifiers (VID) – a simple way to build path vectors in the control plane.

Forwarding decisions in the data plane are made using VIDs, not IP addresses.
MTP Route Establishment

- **VID = Virtual Identifier.**
  Defines a path from a leaf (ToR – the root of the meshed tree) to a spine. (one-way only).

- **Each ToR derives a root VID (can be assigned).**

- **Announcement messages carry the VID information to its neighbor spines.**

- **Spines at the highest tier are configured to stop VID propagation down the topology.**
Root (ToR) VID derivations

- The Root VIDs can be derived from the physical subnet IP address used by servers connecting to ToR.
- Current method utilizes an octet of the server subnet.
  - In this case, all server subnets are /24.
- Other methods can be defined.
# MTP Routing Tables

## Root1 Forwarding Table

<table>
<thead>
<tr>
<th>Spine</th>
<th>Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.1</td>
<td>1</td>
</tr>
<tr>
<td>11.2</td>
<td>2</td>
</tr>
</tbody>
</table>

## S1_1 Forwarding Table

### Upstream

<table>
<thead>
<tr>
<th>Port</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>11.1</td>
<td>1</td>
</tr>
<tr>
<td>12.1</td>
<td>1</td>
</tr>
<tr>
<td>11.2</td>
<td>2</td>
</tr>
<tr>
<td>12.2</td>
<td>2</td>
</tr>
</tbody>
</table>

### Downstream

<table>
<thead>
<tr>
<th>Port</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>11.1</td>
<td>3</td>
</tr>
<tr>
<td>12.1</td>
<td>4</td>
</tr>
</tbody>
</table>

## S2_1 Forwarding Table

### Downstream

<table>
<thead>
<tr>
<th>Port</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>11.1</td>
<td>1</td>
</tr>
<tr>
<td>12.1</td>
<td>1</td>
</tr>
<tr>
<td>13.1</td>
<td>2</td>
</tr>
<tr>
<td>14.1</td>
<td>2</td>
</tr>
</tbody>
</table>
Example

- Source ToR- checks src-dst IP address. Derives the src-dst ToR VIDs – populates MTP header.
- Encapsulates incoming IP packet.
- Sends to any Tier 1 spine – hash algorithm.
- Tier 1 spine checks dst ToR VID in MTP header – forwards by default to Tier 2 spine – hash algorithm.
- Tier 2 spine check dst ToR VID, its VID table, sends on port – that records the dst VID.
- So on...

"ToR Tier1 Spines Tier2 Spines"
# Comparison – Current BGP Implementation

<table>
<thead>
<tr>
<th>MTP</th>
<th>Modified BGP</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Route established with VIDs –</td>
<td>• BGP modified to avoid path hunting.</td>
</tr>
<tr>
<td>no traditional routing</td>
<td>• ASN adjusted.</td>
</tr>
<tr>
<td>protocols, no route discovery.</td>
<td>• Route discovery flooding – periodic.</td>
</tr>
<tr>
<td>• One-way route from ToR to</td>
<td></td>
</tr>
<tr>
<td>Spine.</td>
<td></td>
</tr>
</tbody>
</table>
Current Status

Working MTP code in Python tested on the GENI testbed.
- Has algorithm tested.
- Check functionality.
- Demo available.

Code available on public Github repository.

C Code being developed, will integrate hashing and link aggregation.

Other features – fast failure recover, energy studies.

Seeking industry collaboration.
Thank you!