Brute Force Routing for Delay Optimizations in Data Center Networks

Ali Fattaholmanan
USI
Traffic Characteristics in DCNs

Scatter-gather traffic pattern

- Every request can trigger 40 different sub-requests, up to 10 layers

Image source: Talk on “Speeding up Distributed Request-Response Workflows” by Virajith Jalaparti at ACM SIGCOMM'13
Traffic Characteristics in DCNs

Huge Volume

- Google: It’s doubling every two years!

Figure 1: Aggregate server traffic in our datacenter fleet.

Traffic Characteristics in DCNs

Web servers and cache hosts have 100s to 1000s of concurrent connections

- Median inter-arrival time of flows is 2ms

Traffic Characteristics in DCNs

Mostly very small flows!

- Median flow size < 1KB

Image source: Benson, et al.. "Network traffic characteristics of data centers in the wild." SIGCOMM'10.
Data Center Networks (DCNs)

- What makes it Special?
  - Single Administration; full control over
    - hosts
    - middle boxes
  - Known topology
Common Topology
Common Topology
Routing in DCNs

- Layer 2
  - STP
    - It uses only a subset of links, to avoid loops

- Layer 3
  - OSPF
    - Not scalable because of flooding messages
  - BGP
    - Slow convergent time. Too complex to tune!
Multi-Path Routing

- Equal Cost Multi-Path (ECMP)
  - Randomly distribute traffic among all available paths,
  - Randomly per packet?
    - TCP reordering
  - Randomly per flow
    - Output port ~ hash(TCP header)
Multi-Path Routing

Local decision, without considering global view

10.2.0.3:80  port 1
10.2.0.3:25  port 2
CONGA: edge based

Median inter-arrival times of approximately 2 ms, (i.e., more than 500 flows per second)

<table>
<thead>
<tr>
<th>Path</th>
<th>Dest. switch A</th>
<th>Path 1</th>
<th>Path 2</th>
<th>Path 3</th>
<th>Path 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path 1</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Path 2</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Path 3</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Path 4</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Brute Force Routing

- Brute Force all possible routes!
  - Simply duplicate packet when going up!
  - Is it TCP friendly?
    - Drop duplicate packets in the last ToR switch
    - Less congested path is not approximated, it is been chosen already!
- How it works?
  - Fastest packet go through the ToR switches, other delayed packets will be dropped
  - Totally transparent for end-hosts
Brute Force Routing

One Objection!
It may also increase the number of congested links

Simply duplicate packet when it goes up!
Brute Force Routing

- Flooding Risk?
  - Median flow size < 1KB
  - Can be applied only on delay-sensitive application
  - Common approach in application level

\[
\frac{50\% \times 1000}{50\% \times 1000 + 5\% \times 100000} < 10\%
\]
P4 Implementation

definition action set_drop_flag (switchID_t id) {
    bit<32> key = (bit<32>) hdr.tcp.checksum;
    bit<16> tmp_counter = 0;
    bit<48> tmp_timer = 0;
    bit<48> timelapse = 0;
    last_seen_flag.read(tmp_counter, key);
    last_seen_flag.write(key, tmp_counter + 1);
    last_seen_time.read(tmp_timer, key);
    last_seen_time.write(key, standard_metadata.ingress_global_timestamp);
    timelapse = (standard_metadata.ingress_global_timestamp - tmp_timer) >> 12;
    meta.my_drop_metadata.flag = 0;
    if(tmp_counter != 0 && timelapse == 0){
        meta.my_drop_metadata.flag = 1;
    }
}

definition table dcn_tcp_dup {
    actions = {
        tcp_broadcast;
        NoAction;
    }
    key = {
        hdr.tcp.dstPort: exact;
    }
    size = 512;
    default_action = NoAction();
}
# Experiment Results

## Experiment Config

<table>
<thead>
<tr>
<th></th>
<th>16</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Number of Servers</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Number of Switches</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Target**
- bmv2

**Emulator**
- Mininet

## Improvement

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>STDev</strong></td>
<td>27.47%</td>
</tr>
<tr>
<td><strong>Quartile 1</strong></td>
<td>59.69%</td>
</tr>
<tr>
<td><strong>Quartile 2</strong></td>
<td>54.62%</td>
</tr>
<tr>
<td><strong>Quartile 3</strong></td>
<td>51.42%</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td>47.33%</td>
</tr>
</tbody>
</table>

**Delay improvement over >300 probes**
Future Work

- Keep track of the best path in the IP header, so as to use the same path again
- To use network coding to have higher granularity
- Do experiments over real networks
Question?