Day 17
Challenges in Data Centers (and Clouds)

• You want low latency (Tail of Latency Paper, VL2)
  – Hard because of queues, background jobs, e.t.c

• You want high-Throughput (DCTCP, VL2)
  – Hard to get high throughput cause S/W are expensive

• You want VM Migration (VL2)
VM Migration

• Very easy to do:
  – If you change IP addresses, but why is this bad?

• IP addresses are tied to locations (hierarchical)
  – Must make address flat!!

• Layer 2 has flat address.....
  – But you need to use Broadcast to discover location
  – Why is broadcast bad?
• Can you have a network that support High throughput and VM Migration?
VL2’s Methods

• Flat Addressing:
  Allow service instances (ex. virtual machines) to be placed anywhere in the network.

• Valiant Load Balancing:
  (Randomly) Spread network traffic uniformly across network paths.

• End-system based address resolution:
  To scale to large server pools, without introducing complexity to the network control plane.
How Does VL2 Work

• Each VM has to addresses
  – LA: location address
  – AA: Application address

• VM only understands AA
  – This is what you use to configure MySqL and the Apache webserver. This is what you try to PING

• The Network only understand LA.
  – So you need to convert before you send a packet

• How do you get an LA?
• At Start time, a VM registers with the central server using the AA
  – The server creates a mapping of LA to AA
What are different Ways to Create a LA?

Approach 1: Create special LA (like 3rd floor in LSRC)
What are different Ways to Create a LA?

Approach 1: Create special LA (like 3\textsuperscript{rd} floor in LSRC)

Using a hierarchical naming scheme: means the network can automatically determine how to get packets to the local Addresses.
What are different Ways to Create a LA?

Approach 2: The LA is the address of the TOR (like making the UPS store your address)
What are different Ways to Create a LA?

Approach 2: The LA is the address of the TOR (like making the UPS store your address)

The network needs a routing algorithm to learn how to get to all the different TOR switches.
# How to Send a Packet: PT 1

- **Do an ARP**
  - ARP is sent to central server ₹ this returns LA

<table>
<thead>
<tr>
<th>End-Host Interception:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add code to the End-host</td>
</tr>
<tr>
<td>It converts ARP broadcast to unicast</td>
</tr>
<tr>
<td>Sends to the central server</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Design Issue:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Only a small number of VM share cache</td>
</tr>
<tr>
<td># of VMs</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ToR Interception:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add code to the H/W Switch</td>
</tr>
<tr>
<td>It converts ARP broadcast to unicast</td>
</tr>
<tr>
<td>Sends to the central server</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Design Issue:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Must change H/W Switch</td>
</tr>
<tr>
<td>Many more VM can share cache</td>
</tr>
<tr>
<td># of VMs</td>
</tr>
</tbody>
</table>
How to Send a Packet: PT 1

- **Do an ARP**
  - ARP is sent to central server → this returns LA

### End-Host Interception:
- Add code to the end-host
- It converts ARP broadcast to unicast
- Sends to the central server

### ToR Interception:
- Add code to the H/W Switch
- It converts ARP broadcast to unicast
- Sends to the central server

### Design Issue:
- Only a small number of VM share cache
- # of VMs

### Many more VM can share cache
- 40*# of VM per server

Going to a central for each ARP is problematic. How do you get around this problem?
How to Send a Packet: PT1

• Change AA to LA because Network doesn’t know AA.

Re-write Addresses:
• No packet over-head
• No control over path
  • Trust network to route correctly

Encapsulate addresses:
• Encapsulation header requires space → smaller packets
• You can control the path
  • No Trust in Network routing

A_M: B_M
A_IP: B_IP
Payload

A_M: LA
A_IP: B_IP
Payload

MAC: MAC
A_M: B_M
A_IP: B_IP
Payload

MAC: MAC
A_IP: TOR_B
A_IP: IA
Payload

A_IP: B_IP
Payload
How to Send a Packet: PT 3

• Network Routes the packet

Network Layer 3 Routing Protocol:
• Network is AWESOME at OSPF
• ECMP allows for multi-path

ToR Interception:
• Add code to the H/W Switch
• It converts ARP broadcast to unicast
• Sends to the central server
What happens after a migration?
What happens after a migration?

• Need to update all Central Servers

• Need to update all caches
Problems with this Type of Approaches

• Normal Broadcast Traffic
  – Why is this a problem?
  – Use IP Multicast: how?

• Internet Traffic (20% of all Data center traffic)
  – LA->AA works well with in the data center
  – External traffic doesn’t care about LA
  – Only advertises AA in BGP. So external traffic uses AA
  – Create a Gateway device that converts form LA to AA
    • Why do you need this?
Issues with a Central Server

• Scaling issues
  – Use paxos

• Use cache to reduce overhead
  – During failure must update all caches.
VL2

• Focuses on High throughput

• Applying Valiant Load balancing at the flow level leads to problems
  – Only 80% as effective
  – Problem becomes worse when there are elephant flows!!

• DCTCP: LOTs of ELEPHANT FLOWS!!!!!!
Data Center Workloads

Partition/Aggregate

(Query)

(Coordination, Control state)

(Data update)

Delay-sensitive

Delay-sensitive

Throughput-sensitive
Recall From Last Week

To improve latency, send the request to two servers. This helps avoid waiting queues!!!
The Network Also has Queues.....

Worker 1

Worker 2

Worker 3

Worker 4

TCP timeout

Aggregator
When will MPTCP Help?
Why Have Queues in the Network?

• Queues Add Latency!
• Why have them?
Why Have Queues in the Network?

- Queues Add Latency!
- Why have them?

For Regular TCP:
- You need a certain amount of Buffers for TCP to have high utilization (high throughput)
- Congestion control leads to oscillation in utilization
- Queues help smoothen out the oscillations.
Data Center Workloads

Partition/Aggregate

(Query)

(Coordination, Control state)

Delay-sensitive

Delay-sensitive

We need queues for these jobs

(Data update)

Throughput-sensitive
A Solution From Last Week.

• Use priority queues. When are these ineffective?
Incast

• Synchronized mice collide.
  ➢ Caused by Partition/Aggregate.

Worker 1

Worker 2

Worker 3

Worker 4

TCP timeout

RTO_{min} = 300 ms
Scalability of Netflix

Other implication of Network Limits

Number of machines grow.

Communication rates decrease.

Today's transport protocols: Deadline agnostic and strive for fairness

Application SLAs

Cascading SLAs...
INCAST Impact of Partition Aggregate
Potential Solutions

• Add More buffers:

• Make the RTO much shorter!!!

• Reduce the number of packets in response
Incast

- Synchronized mice collide. ➢ Caused by Partition/Aggregate.

- If you double the queue → problem happens when a job has 8 workers (double)
- If you make the response 1 instead of 2 → problem happens when a job has 8 workers (double)
- If you make RTO smaller then all flows with dropped packets will send again at the same time → potential synchronization
More Solutions?

• How about the Gentle TCP paper from Google?
  – Proactive Solution Send each packet twice.
  – Reactive Solution, use FEC code
More Solutions?

- How about the Gentle TCP paper from Google?
  - Proactive Solution Send each packet twice.
    - MORE CONGESTION $\rightarrow$ HORRIBLE
  - Reactive Solution, use FEC code
    - Depends, potentially no.
Even More Solutions

• How about sending request to two different servers?
Even More Solutions

• How about sending request to two different servers?
  – For INCAST, the problem is in the response this doesn’t help
Data Center Transport Requirements

1. High Burst Tolerance
   Incast due to Partition/Aggregate is common.

2. Low Latency
   Short flows, queries

3. High Throughput
   Continuous data updates, large file transfers

The challenge is to achieve these three together.
Ideal Approach: TeXCP

• The Switch tells each person how much to send
  – Keeps buffers low
  – Stops bursts !!
  – Keeps throughput high
Ideal Approach: TeXCP

• The Switch tells each person how much to send
  – Keeps buffers low
  – Stops bursts !!
  – Keeps throughput high

• Problems with TeXCP?
  – If in slowpath then very slow!!
  – If in fast path, then requires changes to the Hardware switches
Review: The TCP/ECN Control Loop

ECN = Explicit Congestion Notification

ECN Mark (1 bit)
Two Key Ideas

1. React in proportion to the **extent** of congestion, not its **presence**.
   - Reduces **variance** in sending rates, lowering queuing requirements.

<table>
<thead>
<tr>
<th>ECN Marks</th>
<th>TCP</th>
<th>DCTCP</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 0 1 1 1 1 0 1 1 1</td>
<td>Cut window by 50%</td>
<td>Cut window by 40%</td>
</tr>
<tr>
<td>0 0 0 0 0 0 0 0 0 1</td>
<td>Cut window by 50%</td>
<td>Cut window by 5%</td>
</tr>
</tbody>
</table>

2. Mark based on **instantaneous** queue length.
   - Fast feedback to better deal with bursts.
Data Center TCP Algorithm

**Switch side:**
- Mark packets when Queue Length > K.

**Sender side:**

- Maintain running average of fraction of packets marked ($\alpha$).

In each RTT:

- Adaptive window decreases:

$$F = \frac{\text{# of marked ACKs}}{\text{Total # of ACKs}}$$

$$\alpha \leftarrow (1 - g)\alpha + gF$$

$$\text{Cwnd} \leftarrow (1 - \frac{\alpha}{2})\text{Cwnd}$$
Why it Works

1. High Burst Tolerance
   - Large buffer headroom → bursts fit.
   - Aggressive marking → sources react before packets are dropped.

2. Low Latency
   - Small buffer occupancies → low queuing delay.

3. High Throughput
   - ECN averaging (Better Algorithm) → smooth rate adjustments
Concluding Remarks.