Deadline-Aware Scheduling and Routing for Inter-Datacenter Multicast Transfers

Siqi Ji*, Shuhao Liu, Baochun Li
Inter-Datacenter Traffic

- Interactive
  - Highly sensitive to loss and delay
  - Should be delivered instantly with strictly higher priority
- Elastic
  - Requires timely delivery— prior to a deadline
- Background
  - No explicit deadline or a long deadline
Why we need to consider deadlines?

- Total demand for inter-DC transfers typically far exceeds the available capacity.

- Cloud providers set different data replication SLAs (or deadlines) based on delay tolerance.

- Customers are willing to pay more for guaranteed deadlines.
Multicast Transfers

- Deliver data from one datacenter to multiple datacenters
  - Fault tolerance, availability and high service quality.
- Examples: data replication, database synchronization…
- Most of them have deadlines.
The Problem?

› Scheduling and allocating bandwidth for multiple inter-datacenter multicast transfers.

› Meet deadline requirements.

› Maximize throughput.
Motivation Example

<table>
<thead>
<tr>
<th>Requests</th>
<th>Source</th>
<th>Destinations</th>
<th>Volume (MB)</th>
<th>Deadlines (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_1$</td>
<td>1</td>
<td>3, 4</td>
<td>200</td>
<td>40</td>
</tr>
<tr>
<td>$R_2$</td>
<td>4</td>
<td>2, 3</td>
<td>200</td>
<td>40</td>
</tr>
</tbody>
</table>

Link 1-2 is saturated, no more available bandwidth for request 2!
Previous Work

- Unicast transfers:
  - SWAN [sigcomm’13], B4 [sigcomm’13], BwE [sigcomm’15], Tempus [sigcomm’14], Amoeba [eurosys’15]
  - DCCast [hotcloud’17] and DDCCast [tech report]:
    - Did not maximize throughput
    - Not effective for requests that require high bandwidth
Deadline Transfers

- Considering there are n transfers, a transfer request i can be specified as a tuple \{S^i, R^i, Q^i, D^i\}:

  - \(S^i\): source datacenter of request i
  - \(R^i\): destination datacenters of request i
  - \(Q^i\) and \(D^i\): data volume and deadline requirements of request i

- Objective: Maximize throughput for all transfers with the consideration of meeting deadlines.
Linear Program

\[ T^i = \{ t \mid t \text{ is a Steiner tree (or multicast tree) from } S^i \text{ to } R^i \}. \]

\[
\begin{align*}
\text{maximize} & \quad \chi \\
\text{subject to} & \quad \chi \leq \sum_{t \in T^i} x^i(t), \forall i = 1, \ldots, n, \\
& \quad \sum_{i=1}^{n} \sum_{t \in T^i} x^i(t) \phi(t, e) \leq C(e), \forall e \in E, \\
& \quad D^i \sum_{t \in T^i} x^i(t) \geq Q^i, \forall i = 1, \ldots, n, \\
& \quad x^i(t) \geq 0, \chi \geq 0, \forall t \in T^i, \forall i = 1, \ldots, n.
\end{align*}
\]

where \( \phi \) is defined as:

\[
\phi(t, e) = \begin{cases} 
1, & \text{if } e \in t, \\
0, & \text{otherwise.}
\end{cases}
\]
Sparse Solution

- Reduce splitting and packet reordering overhead

- We add a penalty function at the objective to get a sparse solution

\[
\text{maximize} \quad \chi - \mu \sum_{i=1}^{n} \sum_{t \in T^i} g(x^i(t)),
\]

\[
g(x^i(t)) = \begin{cases} 
0, & \text{if } x^i(t) = 0, \\
1, & \text{if } x^i(t) > 0.
\end{cases}
\]
Sparse Solution

- We can linearize the penalty function by using a l1-norm weighted heuristic.

\[
\text{maximize} \quad \chi - \mu \sum_{i=1}^{n} \sum_{t \in T^i} (W^i(t) \cdot x^i(t)),
\]

- In each iteration we recalculate the weight function \( W^i \) where:

\[
W^i(t) = \frac{1}{(x^i(t))^k + \delta}.
\]
Sparse Solution

- Upon convergence, \((x^i(t))^k \approx (x^i(t))^{k+1} = (x^i(t))^*\), for \(i = 1, \ldots, n, t \in T^i\)

\[
W^i(t) \cdot (x^i(t))^* = \frac{(x^i(t))^*}{(x^i(t))^k + \delta} = \begin{cases} 
0, & \text{if } (x^i(t))^* = 0, \\
1, & \text{if } (x^i(t))^* > 0.
\end{cases}
\]

- Eventually, the transformed problem approaches the original problem and yield a sparse solution
An example of the optimal solution obtained by our linear program:

- Steiner Trees for Request 1
- Steiner Trees for Request 2

### Assumptions
- Assume all link capacities are 15MB/s
- If we use only one tree, the shortest completion time is 20s, all requests will miss their deadlines
- Maximize throughput, request R2 can even finish the transfer before its deadline.

### Table

<table>
<thead>
<tr>
<th>Requests</th>
<th>Source</th>
<th>Destinations</th>
<th>Volume (MB)</th>
<th>Deadline (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_1$</td>
<td>2</td>
<td>1, 4</td>
<td>300</td>
<td>8</td>
</tr>
<tr>
<td>$R_2$</td>
<td>5</td>
<td>1, 3</td>
<td>300</td>
<td>18</td>
</tr>
</tbody>
</table>

### Table Data

<table>
<thead>
<tr>
<th>Request 1</th>
<th>Request 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trees</td>
<td>Rate</td>
</tr>
<tr>
<td>2 ➔ 1 ➔ 4</td>
<td>15</td>
</tr>
<tr>
<td>2 ➔ 4 ➔ 1</td>
<td>12.06</td>
</tr>
<tr>
<td>2 ➔ 5 ➔ 1</td>
<td>10.44</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Implementation

- We have completed a real-world implementation in a software-defined overlay network testbed at the application layer.
How a transfer is routed and completed through our application-layer SDN testbed?

- Destinations: subscribe to a specific channel

- Source: publish its data, destinations and deadline requirements to the channel

- Aggregator: consult the controller for routing rules

- Controller: routing rules — next hop and sending rate to each datapath node
Experiment

- Google Cloud Platform
- Six Virtual Machines (VM) instances located in six different datacenters, and one of the VMs is also launched as the central controller.
Experiment

- We use file replication as inter-datacenter traffic
  - The volume of each file is set to be 300MB
  - Deadlines: generate from a uniform distribution between \([T, \alpha T]\)
    - \(\alpha\) represents the tightness of deadlines for generated transfers
Performance Evaluation

- Comparison of different solutions as the tightness factor increases:

![Graph showing performance evaluation with different solutions vs tightness factor](image-url)
Performance Evaluation

- Comparison of different solutions as the number of destinations increases:
Performance Evaluation

- Throughput comparison of different solutions:
Performance Evaluation

- Comparison of different solutions as the number of requests increases:
Conclusion

- Our solution performs better in maximizing throughput and meeting transfer deadlines than related work.

- Future work:
  - Dynamic resources
  - Different request arrival rates
Thank you!

Q&A