On Efficient Data Transfers Across Geographically Dispersed Datacenters

Ph.D. Defense Presentation
By
Max Noormohammadpour

Committee: Prof. Ellis Horowitz, Prof. Viktor K. Prasanna, Prof. Cauligi S. Raghavendra (Chair)
Outline

• Introduction
• Research Contributions
• Background
• P2MP Traffic Engineering (DCCast)
• P2MP Traffic Engineering (QuickCast)
• P2MP Traffic Engineering (Iris)
• P2MP Traffic Engineering (Parallel Forwarding Trees)
Datacenters

• Compute/storage/networking Infrastructure for distributed services
• From 100s to 100,000s of servers
Datacenter Services

• Run across many geographical locations in different datacenters
  • Closer to regional users
  • Improved performance
  • Increased availability
Example: Cache Services

- Netflix cache locations (Green), Internet Exchange Points (Orange)
Inter-DC Networks

- Shared/Dedicated links connecting multiple datacenters
  - Earlier shared WAN links
  - Increasingly dedicated (owned or leased) links
  - Relatively small network with very high capacity links

Microsoft’s Backbone for Azure Services

Google B4 (GScale)
Inter-DC Transfers

• Generated by datacenter services
• Go over inter-DC networks
• Two types:
  • Point to Point (P2P)
  • Point to Multipoint (P2MP)
Point to Point Transfers

• Deliver data from one datacenter to another datacenter

• **Example:** An application running on west coast that backs up data to a second datacenter in east coast
Point to Multipoint (P2MP) Transfers

- Deliver same data from one datacenter to multiple datacenters
- **Example:** Popular video content pushed to regional datacenters for high throughput/low latency user access
What Applications Generate P2MP Transfers?

<table>
<thead>
<tr>
<th>Application</th>
<th>Reason for delivery to multiple datacenters</th>
</tr>
</thead>
<tbody>
<tr>
<td>CDN, Web</td>
<td>Getting closer to users</td>
</tr>
<tr>
<td>Data Recovery</td>
<td>Making backup copies</td>
</tr>
<tr>
<td>Search</td>
<td>Synchronization of state</td>
</tr>
<tr>
<td>Recommendation, Ads</td>
<td>Global load balancing</td>
</tr>
<tr>
<td>Databases</td>
<td>Input for next processing stages</td>
</tr>
<tr>
<td>Geo-Distributed Data Analytics (broadcast-join)</td>
<td></td>
</tr>
</tbody>
</table>
Inter-DC Traffic Classes

• User-generated Traffic (To Internet)
  • Result of direct user interactions
  • Highest priority

• Internal (i.e., bulk) Traffic
  • Majority of inter-DC traffic
  • Result of internal business operations

Facebook’s Express Backbone:
“bandwidth demand for cross-data center replication of rich content like photos and video has been increasing rapidly”
Inter-DC Traffic Engineering

- **Coordinator** with global knowledge of
  - Transfers
  - Topology

- **Centralized** transmission coordination (i.e., rate-control and routing)
  - Improves performance and efficiency
  - Reduces provisioning costs
Inter-DC Traffic Engineering

• Feasible
  • Tens to hundreds of locations
  • Focus on inter-DC transfers
    • Latency resilient (not in the critical path of user experience)

• Effective
  • Majority of traffic
Outline

• Introduction
• Research Contributions
• Background
• P2MP Traffic Engineering (DCCast)
• P2MP Traffic Engineering (QuickCast)
• P2MP Traffic Engineering (Iris)
• P2MP Traffic Engineering (Parallel Forwarding Trees)
Research Contribution 1

• Bulk transfers from source DC to destination DC
• Best Worst-case Routing (BWR) of inter-DC P2P transfers
  • Objective: minimizing the completion times of transfers
  • Constraints: link capacities
• Summary of results
  • Improving the average completion times of transfers by up to $3.5 \times$ compared to popular routing approaches used today
  • Improving the tail completion times of transfers by up to $2 \times$ compared to popular routing approaches used today
Research Contribution 2

• P2P transfers can arrive with a deadline
• Demand maybe higher than available capacity some may have to be rejected
• Admission control for inter-DC P2P transfers with deadlines
  • Objective: maximize the number of admitted transfers
  • Constraints: transfers deadlines, and link capacities

• Summary of results
  • Admits almost identical number of transfers (+ no packet reordering) compared to state-of-the-art [Eurosys’15] and general linear programming
  • Speeds up admission control by over $1000 \times$ compared to optimization formulations based on linear programming
  • Can speed up admission control by over $60 \times$ compared to state-of-the-art admission control that uses approximations [Eurosys’15]
Research Contribution 3

• Transfers from one sender DC to multiple receiver DCs
• Fast and efficient inter-DC P2MP transfers
  • Objective: minimizing the completion times of transfers/receivers
  • Constraints: link capacities, all receivers on a tree complete together
  • We developed multiple solutions which will be discussed in this presentation
    • DCCast, QuickCast, Iris.
• Summary of results (for QuickCast)
  • Up to $3.64 \times$ reduction in average receiver completion times while at the same time using $0.71 \times$ the bandwidth compared to performing P2MP transfers as separate unicast transfers
  • Top $50\%$ of receivers complete between $3 \times$ to $35 \times$ faster on average compared with when a single forwarding multicast tree is used
Outline

• Introduction
• Research Contributions
• Background
• P2MP Traffic Engineering (DCCast)
• P2MP Traffic Engineering (QuickCast)
• P2MP Traffic Engineering (Iris)
• P2MP Traffic Engineering (Parallel Forwarding Trees)
Inter-DC Network: A Graph

- Datacenters $\rightarrow$ Nodes ($V$)
  - Host the end-points of transfers
- Links $\rightarrow$ Edges ($E$)
  - Have a capacity associated with them

$G(V, E)$
Properties of Inter-DC (Bulk) Transfers

• A (bulk) transfer, i.e., $R$
• A source datacenter, i.e., $S_R$, where it is originating
• One or more receivers, i.e., set $D_R$
  • Depending on whether a P2P or a P2MP request
• The total volume of the transfer, i.e., $V_R$
Traffic Engineering Problem

• Transfers arrive at the network in an online manner

• Routing Problem
  • What paths/trees should be used by each transfer for data delivery

• Rate-allocation Problem
  • At what rate each transfer should transmit over different timeslots

• Objective
  • Optimize one or more system-wide performance metrics
System Model

• A central Traffic Engineering Server (TES) for bandwidth allocation
  • Receives transfer requests (for inter-DC transfers)
  • Is responsible for coordination
• Uses global knowledge of network status and transfer properties to perform
  • Routing
  • Rate-allocation
Assumptions

• Routes assigned to transfers cannot be changed after initial assignment
  • Reduces potential network forwarding state inconsistencies
  • Limits transient congestion due to traffic diversion
Slotted Timeline

- Bandwidth allocations are calculated on a per timeslot basis
- Allows us to formulate mathematical models for computation of rates
  - Every transfer is transmitted at a constant rate (i.e., $r_{e,t_i}$) over a timeslot over some edges
  - New transfers are allocated starting next timeslot
Rate-allocation Problem Formulation
(Example Scenario with one tree $T(R)$ per transfer $R$)

- Set of requests $R$ with many requests $R(S_R, D_R, V_R)$ where parameters determine volume, source, and set of destinations of P2P/P2MP transfer request $R$
- Assume one path/tree from $S_R$ to all nodes in $D_R$ for every request $R \in R$ which we write as $T(R)$
- Variables $r_{R,t}$ determine the transmission rate of request $R \in R$ at timeslot $t$ over its tree $T(R)$

**Capacity Constraints:**
- $\sum_{\{R \mid e \in T(R)\}} r_{R,t} \leq C_e$ (capacity of edge $e$) (per timeslot, per edge)

**Flow Constraints:**
- Traffic flows with the same rate across all edges of a path/tree (per timeslot, per transfer)

**Demand Constraints:**
- $\sum_t r_{R,t} = V_R$ (per transfer)
Rate-allocation Policies

• A policy based on which transmission rates are computed per transfer and per timeslot

• Popular policies
  • First Come First Serve (FCFS)
    • Bandwidth allocated in the order of arrival
  • Shortest Remaining Processing Time (SRPT)
    • Bandwidth allocated in the order of remaining size (smallest first)
  • Fair Sharing, e.g., Max-Min Fairness (MMF)
    • Bandwidth allocated according to how transfers share links
Performance Metrics (1)

- Mean (average) completion times
- Tail (worst-case) completion times
  - Usually computed as the 99th percentile
- Median (50th percentile) completion times
- Can be per receiver or per transfer
  - Transfer completion time: The time its last receiver completes
Performance Metrics (2)

• Total bandwidth (capacity) consumption
  • Sum of the bandwidth used over network edges to complete a given set of inter-DC transfers
  • Increases if the number of edges over which a transfer is sent increases

Total BW Used: 2X

Total BW Used: X
Transfer Size Distributions

• Light-tailed and heavy-tailed transfer size distributions
  • Light-tailed is based on Exponential dist.
  • Heavy-tailed is based on Pareto dist.
  • Covers the two ends of traffic size distribution spectrum

• Hadoop and cache-follower transfer size distributions
  • From Facebook inter-DC traffic traces [1]

Wide Area Network (WAN) Topologies

• From Topology Zoo (http://www.topology-zoo.org/) and [1]
  • Small (< 25 nodes): GScale, AGIS, ANS
  • Medium (25 nodes ≤, < 100 nodes): AT&T, GEANT, UNINETT
  • Large (> 100 nodes): Cogent

Outline

• Introduction
• Research Contributions
• Background
• P2MP Traffic Engineering (DCCCast)
• P2MP Traffic Engineering (QuickCast)
• P2MP Traffic Engineering (Iris)
• P2MP Traffic Engineering (Parallel Forwarding Trees)
Point to Multipoint (P2MP) Transfers

• A P2MP transfer
  • Is an abstraction model
  • To deliver the same data from one location to multiple locations
  • Set of receivers known apriori
  • Set of receivers is fixed

• Delivery over a tree to save bandwidth
  • At most one copy of $X$ traverses any link
Efficiently Performing P2MP Transfers

<table>
<thead>
<tr>
<th>Completion Time</th>
<th>T</th>
<th>T</th>
<th>T</th>
<th>2T</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total BW</td>
<td>3X</td>
<td>4X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

X: Object Size, R: Link Throughput, T: X/R
P2MP Routing Problem

• Given an inter-DC network $G(V, E)$ with link capacities of $C_e$ for $e \in E$, and a set of transfers $R = \{R_1, \ldots, R_n\}$ already going on in the network with residual demands of $\{V_{R_1}^r \ldots V_{R_n}^r\}$ and arrival of new transfer $R_{n+1}$ with demand $V_{R_{n+1}}$, which tree should be selected for $R_{n+1}$?

• Objectives:
  • Minimizing the mean/tail completion times of transfers

• Constraints:
  • Link capacities, all receivers on a tree complete together
General Approaches to P2MP Transfers

• Usually performed as separate P2P transfers
  • Simple/Requires no modifications
  • The fact that same data is delivered is not considered
  • Wastes bandwidth and can increase completion times

• Multicasting
  • **Network-driven** (e.g. IP Multicast)
    • Locally and gradually built trees far from optimal
    • No load distribution management
    • Complex session management protocols
  • **Client-driven** (e.g. Overlay Networks)
    • Limited visibility into network status
    • Limited control over routing
Minimum Weight Steiner Trees

• Select a minimum weight Steiner tree
• Edge weight assigned using the global network view
  • Client-side information (transfer sizes and routes)
  • Inter-DC network information (topology, link capacities)
Edge “Load”

• We define a new edge parameter called “load”

\[ L_e = \sum_{\{\forall R \mid e \in P_R\}} V_R^r \]

• \( L_e \) is the total number of bytes allocated on a link \( e \) starting now

---

**Diagram:**
- **Rate (link \( e \))**
- **Capacity \( e \)**
- **\( t_0, t_1, t_2, t_3, t_4, t_n \)**
- **\( t_{\text{now}} \)**
- **\( L_e = \sum \text{blue areas} \)**
Selection of Forwarding Trees

• For a request $R_{new}$ with size of $V_{R_{new}}$, every edge $e$ is assigned a weight of:

$$W_e = \frac{L_e + V_{R_{new}}}{C_e} \quad \text{Eq. 1}$$

• A minimum weight Steiner tree is selected that connects the source of $R_{new}$ to all of its receivers (using a heuristic algorithm)

• Adding $V_{R_{new}}$ to edge weights allows selection of smaller trees for very large transfers that saves bandwidth
DCCast

• Uses minimum weight Steiner tree selection with weight of $Eq. 1$

• Uses the FCFS rate-allocation policy
  • Predictable completion times for transfers

• Goals
  • Save bandwidth compared to performing P2MP transfers as multiple separate P2P transfers
  • Avoid creating network hotspots (highly loaded links)
Compared Schemes

• Performing every P2MP transfer as multiple separate P2P transfers

• Based on $K$-Shortest paths (for every P2P transfer)
  • Frequently used in literature [e.g., Eurosys’15, Sigcomm’14]
  • We used the $K$-Minimum Hop paths

• Considered both SRPT and FCFS policies
  • Once do SRPT for the best mean transfer completion times
  • Once do FCFS for the best tail transfer completion times
Evaluation (DCCast)

- Similar gains in bandwidth consumption and tail TCT with FCFS policy
Outline

• Introduction
• Research Contributions
• Background
• P2MP Traffic Engineering (DCCast)
• P2MP Traffic Engineering (QuickCast)
• P2MP Traffic Engineering (Iris)
• P2MP Traffic Engineering (Parallel Forwarding Trees)
Properties of DCCast

• Per P2MP transfer:
  • All receivers download the data from the original sender
  • All receivers are attached to the sender using one tree
  • All receivers finish at the same time
Improving on DCCast

• Assuming that we can relax the constraint that “all receivers should finish together”

• We can prevent slow receivers from slowing down others

• Our Approach
  • Use multiple trees each attached to a subset of receivers
  • We call each subset of receivers a partition
  • **Pro:** This allows many receivers to complete faster
  • **Con:** This may increase the total network capacity usage
P2MP Routing Problem: Two Sub-problems

• Partitioning
  • How to break receivers of a new transfer into multiple partitions
  • How many partitions?
  • Which receivers should be grouped per partition?

• Tree selection (per partition)
  • Given partitions, select a forwarding tree per partition
  • Minimum weight Steiner trees
  • Same weights as DCCast
Partitioning

• Goals:
  • Reduce the effect of slow receivers on the rest of receivers
  • Minimally increase the total bandwidth consumption doing so

• Two techniques:
  • *Partitioning by proximity*: Group closer receivers together (QuickCast)
  • *Partitioning by speed*: Isolate slow receivers (Iris)
QuickCast

• Group receivers by mutual distance
  • Pairwise distance \((a, b)\) is the hop count on the min hop path from \(a\) to \(b\)

• **Pros:** Keeps extra bandwidth usage negligible on average (compared to using a single partition)

• **Cons:** Some fast receivers may still get bundled with slow receivers and suffer

• Implemented using *hierarchical agglomerative clustering*
Building a Partitioning Hierarchy

1) Each layer presents a possible partitioning combination.
2) This will tell us which receivers should be grouped together given a solution.
Selecting a Partitioning Solution

• Use tree weights to decide

• For every layer $l$ in the hierarchy
  • Compute a minimum weight Steiner tree to every partition in the layer
  • Sum the weights of all forwarding trees for this layer to obtain $W_l$

• Select the layer with maximum number of partitions for which:
  $$W_l \leq p_f \times W_{1\text{-partition}}$$

• Partitioning factor $p_f \geq 1$
Discussion and Example

- **Pro:** Straightforward and fast technique
- **Con:** Requires us to configure a new parameter $p_f$

We can also limit the number of partitions to minimize bandwidth consumption (up to 2 here)

- Compute $W_1$, weight of a single tree to all receivers
- Compute $W_{P_1} + W_{P_2}$, sum of tree weights to two partitions

Partition iff $(W_{P_1} + W_{P_2}) \leq 1.1 \times W_1$ (assuming that $p_f = 1.1$)
Rate-allocation Policies

- Focus on fair sharing
  - All transfers will make constant progress
  - Over larger trees, fair sharing offers higher average throughput
Evaluations (QuickCast)

System Metrics
Evaluations
(QuickCast)
by Receiver Rank
Evaluations (QuickCast)

Effect of $p_f$

System Metrics

4 receivers per transfer
Outline

• Introduction
• Research Contributions
• Background
• P2MP Traffic Engineering (DCCast)
• P2MP Traffic Engineering (QuickCast)
• P2MP Traffic Engineering (Iris)
• P2MP Traffic Engineering (Parallel Forwarding Trees)
Partitioning to Min. Mean Completion Times
(Assuming Fair Sharing)

• A complex problem on a general topology (i.e., affected by routing)
• Simplified topology (no bottlenecks in network core, one down/uplink)
• Uplink speed at the sender is $r_s$
• Receivers sorted by their downlink speeds $r_i, 1 \leq i \leq n$
Partitioning to Min. Mean Completion Times

• Two main techniques
  • Consecutive receivers need to be grouped
  • One partition with a number of fast receivers and all slow receivers as separate partitions (slow receivers isolated)

• Example of partitioning hierarchy for 4 receivers →
Iris

- Group receivers according to their download speeds
  - Uses the fair sharing policy
  - Slow receivers are kept isolated
  - Fast receivers are grouped to save bandwidth

- **Pros**: Sizeable gains in receiver completion times
- **Cons**: Increased bandwidth consumption compared to QuickCast
Computing Receiver/Partition Speeds

• Computing exact speeds is challenging
  • Affected by routing
  • Subject to change as new transfers arrive or existing transfers finish
  • The computation needs to be fast

• We use an approximation
  • Select minimum weight Steiner trees with the weight of Eq. 1
  • Use the best-case completion time of every partition
  • **Pros:** Quick computation, best-case completion times do not change
  • **Cons:** Unbounded optimality gap (though empirically works well)
Selecting the “Right” Partitioning Solution

• Per partitioning solution (layer) in the hierarchy
  • Compute the average of best-case completion times, \( \tau_l \)
  • \( P_l \) is the set of partitions for layer \( l \)
  • \( \tau_P \) is the best-case completion time of partition \( P \)

\[
\tau_l = \sum_{P \in P_l} \frac{|P|\tau_P}{|D_{R_{new}}|}
\]

• Select the layer with minimum number of partitions that offers the minimum \( \tau_l \)
Iris’s Pipeline

Traffic Engineering Server

- Estimate Minimum Completion Times
- Rank Receivers by Mn. Comp. Times
- Compute Receiver Partitions
- Compute Forwarding Trees

Existing Transfers’ Information
(Receiver Partitions, Remaining Volumes, Forwarding Trees)

Forwading State

Multicast Transfer Request

Bulk Multicast Sender
Intermediate Node (Datacenter, IXP, PoP, etc.)
Bulk Multicast Receiver
System Metrics

Evaluations (Iris)
Evaluations (Iris) by Receiver Rank
Outline

• Introduction
• Research Contributions
• Background
• P2MP Traffic Engineering (DCCast)
• P2MP Traffic Engineering (QuickCast)
• P2MP Traffic Engineering (Iris)
• P2MP Traffic Engineering (Parallel Forwarding Trees)
Application of Parallel Forwarding Trees

• Parallel trees connected to the same receivers
  • All receivers in every partition are connected to the same trees

• Use edge-disjoint forwarding trees per partition
  • No direct contention among trees of the same transfer

• Selection process for a partition $P$
  • Select a minimum weight Steiner tree using edge weights of $Eq. 1$
  • Remove all the edges of this tree from the inter-DC graph
  • Repeat... (up to $K$ times)
Evaluations (Parallel Trees)  
GEANT Topology
Evaluations (Parallel Trees), GEANT Topology
K = 2 / K = 1
Evaluations (Parallel Trees)

K = 2 / K = 1
Thank you for your time!

Q&A
Future Directions #1

• Adding the storage capability to intermediate datacenters
  • This will help in case network utilization follows diurnal patterns
  • Also when some network edges have heavy-hitters
  • Referred to as “store-and-forward”
Future Directions #2

• Exploring custom rate-allocation policies besides FCFS, SRPT, and fair sharing
  • Policies that maximize average network throughput
  • One candidate is $\alpha$-Fairness with varying value of $\alpha$
    • $\alpha \to \infty$ is max-min fairness which we used