**Global Placement of Datacenters and Inter-Datacenter Networks**

- Datacenters spread globally (Microsoft Azure)
- Inter-datacenter networks

**Benefits**
- Applying the Fair Sharing policy (rather than FCFS)
- Slotted timeline with fixed rates during timeslots

**According to FCFS:**
- Simple, no rate

**Multicasting (Network-Driven):**
- Improving Mean TCT
- Lower RTT to users (Higher Avg Throughput)

**May increase Completion (request volume)**
- Complexity (running SnF agents)
- No attention to resource utilization
- Parallel trees to same subsets of receivers (increasing throughput)

**Example:** IP Multicast

- Load balancing / Reducing completion times
- Store chosen by a controller with global view of network status

**Easier to get data closer to users**
- Every edge
- Simple weight assignment to edges

**Investigation of more policies (e.g., Fair Sharing) in future**

**Future Work**
- Applying batching techniques for bursty arrival patterns (e.g., apply SJF policy to batches)
- Applying the Fair Sharing policy (rather than FCFS)
- Handling Failures
- Proactive approaches (leaving spare capacity, backup trees)
- Reactive approaches (rescheduling affected transfers, local activation)

**Current Solutions and Their Shortcomings**

- **Separate Unicast Transfers**
  - Wastes Bandwidth
  - May increase Completion Times

- **Multicasting (Network-Driven):**
  - Trees Far From Optimal
  - Limited visibility into network status
  - Limited control over routing
  - No attention to resource utilization
  - Trees built gradually with joins/leaves
  - Complex Session Management
  - Example: IP Multicast

- **Multicasting (Client-Driven):**
  - Storage and bandwidth costs on intermediate datacenters
  - Can lead to excessive delays
  - Complexity (running SnF agents, chunking and reassembly, etc.)

- **Store-and-Forward (SnF):**
  - Guaranteed completion times assuming no failures
  - Single Source
  - Known and fixed receiver set

**Our Solution: DCCast**

- For every P2MP transfer, send traffic to all receivers over a single Forwarding Tree
  - Reduced bandwidth usage
  - Forwarding Tree Selection (at controller)
  - Chosen by a controller with global view of network status
  - Simple weight assignment to edges
  - Minimum weight Steiner tree selection
  - Load balancing / Reducing completion times
  - Rate-Allocation (controller) and Rate-Limiting (senders)
  - Slotted timeline with fixed rates during timeslots

**Tree Selection and Rate-Allocation: Upon Request Arrival**

- **Input:** \( L_n \) (outstanding load on every edge) and \( V_n \) (request volume)
- **Every edge** \( e \) gets a weight of \( W_e = V_e + L_e \)
- **Minimum weight Steiner tree** → Forwarding Tree of \( R \)
- **Many heuristics available for Steiner tree selection**

- **According to FCFS:** Simple, no rate-recalculations
- **Guaranteed completion times assuming no failures**
- **Investigation of more policies (e.g., Fair Sharing) in future**

**Future Work**

- Improving Mean TCT
  - Multiple trees each connected to a subset of receivers (addressing slow receivers)
  - Parallel trees to same subsets of receivers (increasing throughput)
  - Applying SRPT with only BW preemption (trees selected upon request arrivals)
  - Combining forwarding trees with store-and-forward

**Evaluations**

- **Tree Selection Techniques**
- **DCCast vs. Separate Unicast Transfers**

**Source Code Available on GitHub:**
https://github.com/noormoha/DCCast

---

**Efficient Point to Multipoint Transfers Across Datacenters**

M. Noormohammadpour, C. S. Raghavendra, S. Rao, S. Kandula

**Point to Multipoint (P2MP) Transfers: An Abstraction Model**

- Many Applications Cast (Copy) Objects to Multiple Locations
- Reason for delivery to multiple datacenters
- Getting closer to users
- Making backup copies
- Synchronization of state
- Global load balancing
- Input for next processing stage

**P2MP Abstraction Model**

- Single Source
- Known and fixed receiver set

---

**Global Load Balancing**

- Better communication across datacenters
- Increased reliability
- Load balancing
- Easier to get data closer to users
- Lower RTT to users (Higher Avg Throughput)
- Less hops to users (Bandwidth Savings)