Benefits

- Reduced bandwidth usage
- Easier to get data closer to users
- Guaranteed completion times assuming no failures

Data Analytics

- Outstanding load on every edge
- Storage and bandwidth costs on
- Many heuristics available for Steiner tree selection
- Investigation of more policies (e.g. Fair Sharing) in future
- Proactive approaches (leaving spare capacity, backup trees)

Current Solutions and Their Shortcomings

- Separate Unicast Transfers
  - Wastes Bandwidth
  - May increase Completion Times
- Multicasting (Network-Driven)
  - Trees Far From Optimal
  - No attention to resource utilization
  - Trees built gradually with joins/leaves
  - Complex Session Management
  - Example: IP Multicast
- Multicasting (Client-Driven)
  - Limited visibility into network status
  - Limited control over routing
  - Example: Overlay Networks
- Store-and-Forward (SnF)
  - Storage and bandwidth costs on intermediate datacenters
  - Can lead to excessive delays
  - Complexity (running SnF agents, chunking and reassembly, etc.)

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

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
- Applying batching techniques for bursty arrival patterns (e.g. apply SIF 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)

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