Peer-to-Peer Result
Dissemination in High-Volume Data Filtering

Shariq Rizvi and Paul Burstein
CS 294-4: Peer-to-Peer Systems
P2P: A Delivery Infrastructure

- **Overcast**
  - Application-level multicasting
  - Build data distribution trees
  - Adapt to changing network conditions
  - Inner nodes heavily loaded

- **SplitStream**
  - Load-balancing across all peers
  - Split content into redundant streams
  - Redundancy offers resilience to failures
Our Focus

- Dynamic Application-level Multicast
  - Single source
  - Multiple receivers
  - High-volume data flow (“document streams”)
  - Dynamic: very large number of “groups”
  - IP multicast is bad
    - Rigid to deploy
    - Dynamic groups?
  - “Intelligent” trees on the fly?
Organization

- Motivation
  - Data filtering
  - YFilter@Berkeley
  - Distributed YFilter
- Dynamic multicast
  - Unstructured overlay network
  - Metrics
  - Experiments
- Summary & future work
Data Filtering

- Pub-sub systems
- XML: the “wire format” for data
  - Web services
  - RDF Site Summary (RSS) data feeds
    - News
    - Stock ticks
  - Personalized content delivery
- Message brokers
  - Filtering
  - Transformation
  - Delivery
Y Filter: A Data Filtering Engine

Picture blatantly stolen from “Path Sharing and Predicate Evaluation for High-Performance XML Filtering”, Diao et al., TODS 2003
Y Filter: Some Numbers

- Incoming document flow – 10-20 per second
- Document sizes – 20KB
- Subscribers – Lots!
- Processing bottleneck
  - 50ms per document with 100,000 simple XML path queries
- Dissemination bottleneck
  - Thousands of recipients per document – bandwidth needed ~ GbPS

Solution: Distributed filtering
Content-Based Routing

- Embed filtering logic into the network
  - “XML routers”
- Overlay topologies (e.g. mesh)
  - Parent routers hold disjunction of child routers’ queries
- Streams filtered on the fly
- Problems
  - Low network economy – scalability?
  - Query aggregation challenges
Distributed Hierarchical Filtering

Recurrent theme: dynamic multicast
Peer-to-Peer Result Dissemination
Application-Level Dynamic Multicast

- Each document has a different receiver list
- Exploit “peers” for dissemination
- Build trees on the fly
  - Pass documents wrapped with receiver identities
  - Each peer contributes a fanout
- Possibly high delivery delays
  - Heuristic: Try to minimize tree height
- Application-level approach: high traffic
  - Heuristic: Exploit geographical distribution of clients at source
Possible Evaluation Metrics

- Delivery delay
- Network economy
- Document loss
- Out-of-order delivery
Experimental Setup

- PlanetLab testbed
  - Over 200 nodes
  - 1-10 clients per node

- Document Size: 20KB
- Generation Rate: 1 document/second
- Query Selectivity: 10%

- Filter Fanout: 2
- Filter Host: planetlab1.lcs.mit.edu

- Client Fanout:
  - 1 - 20% - Modem
  - 2 - 40% - DSL
  - 4 - 40% - Cable
Result 1: Distribution of Delays

Delivery Delay Distribution - 200 Clients
Result 2: Scalability

![Delivery Delay Distribution Chart]

- **Delivery Delay Distribution**
- **% Clients**:
  - 200 Clients
  - 400 Clients
  - 1000 Clients
  - 2000 Clients

**Delivery Delay (ms)**:
- 0 to 14,000 ms
Result 3: Bandwidth Requirements

[Graph showing the relationship between Outgoing Bandwidth (KBps) and the percentage of clients for different client counts (200, 400, 1000, 2000 clients).]
Exploiting Geographical Distribution of Clients
Result 4: With the optimization

![Graph showing Regional Optimization with delivery delay in milliseconds on the x-axis and percentage of clients on the y-axis. Two lines represent 2000 Clients and 2000 Clients OP.]
Summary

- Current filtering engines – processing and bandwidth bottlenecks
- A possible scheme for distributed filtering
  - Recurring theme: highly dynamic multicast
- Application-level multicast
  - Peer-to-peer delivery
  - Trees construction on the fly
- PlanetLab is crazy
Future Work

- Reliable, dedicated delivery nodes
- Exploiting query similarity for discovering multicast groups