Querying XML Peers

EMIRAN CURTMOLA, ALIN DEUTSCH, YANNIS PAPAKONSTANTINOU
University of California San Diego
{ecurtmola, deutsch, yannis}@cs.ucsd.edu

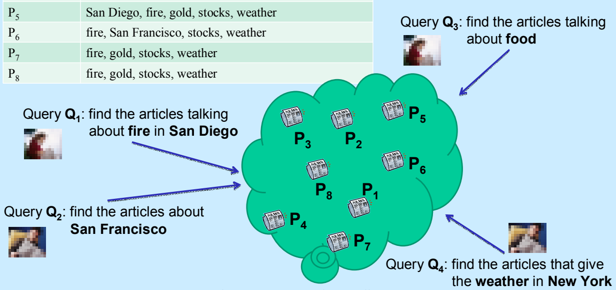
K. K. RAMAKRISHNAN, DIVESH SRIVASTAVA
AT&T Research Labs
{kkrama, divesh}@research.att.com

- **Motivation**
  - Democratization of data creation on the web
    - easy to create and publish data
  - Self-organization in online communities
    - easy to form online communities in an ad-hoc fashion
    - members create, publish and share data items
  - Need to query the overall *community data collection* (union of all published data)

- **Problem**
  - Consumers can query the community data collection on the actual published content
  - The system sends the queries to the relevant publishers
  - Data resides with the publisher
  - Publishers can maintain complete control over who accesses and who is interested in their data

- **Example scenario**: "The virtual newspaper"

Newspapers	Advertised data items about the publisher's articles
P_1	San Diego, San Francisco, stocks, food, weather
P_2	San Diego, gold, New York, food
P_3	San Diego, San Francisco, gold, New York, weather
P_4	San Diego, fire, gold, stocks, weather
P_5	San Diego, fire, gold, stocks, weather
P_6	fire, San Francisco, stocks, weather
P_7	fire, gold, stocks, weather
P_8	fire, gold, stocks, weather



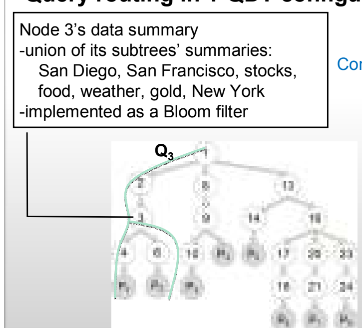
The community data collection

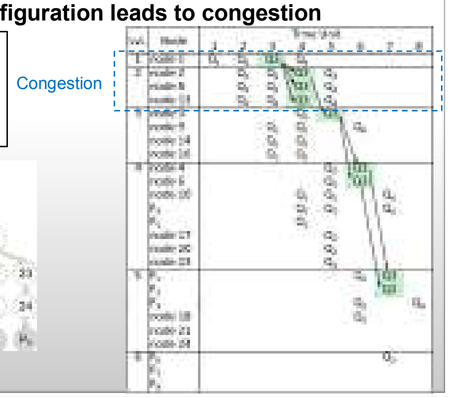
- **Current approaches**
  - Centralized (search engines, hosted online communities)
    - limitations: disintermediate publishers from consumers via a centralized authority
  - Decentralized: content and query dissemination
    - limitations: distributed content-based querying does not focus on efficiency

- **Challenges: query dissemination**
  - Distributed nature of the data among publishers
    - Data is not materialized globally but it resides with publisher
  - Large number of decentralized publishers and consumers
    - Consumers: "whom to ask" + Publishers: "whom to tell"

- **Our solution: query dissemination**
  - Build an overlay network to act as distributed index structure
    - union of *query dissemination trees* (QDTs)
    - each QDT summarizes data items advertized by the publishers in its subtrees
  - Efficient algorithms for query dissemination over QDTs
  - Effective techniques for load balancing and throughput maximization
    - relieve traffic congestion (when only one QDT) by overlaying multiple QDTs
    - query selectivity estimation by maintaining low state about the low selective (popular) data items

Query routing in 1-QDT configuration leads to congestion

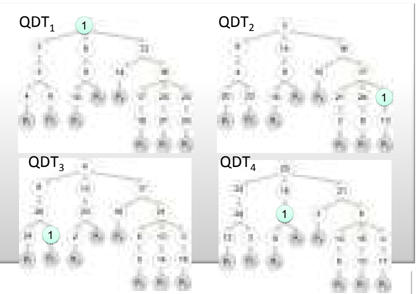




Relieve the congestion

The congestion in the upper levels calls for load balancing techniques

- overlay multiple logical QDTs (e.g., QDT_1 … QDT_4)
- a node belongs to multiple QDTs but at different tree levels
  - organize the nodes into QDTs such that the distribution of tree levels for a node is uniform across the QDTs (e.g., cyclical permutation of nodes on the tree levels)

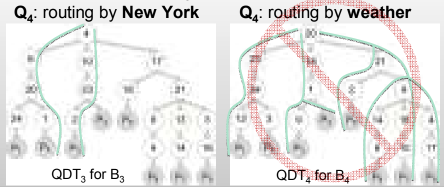


Informed query routing in 4-QDT configuration

- partitioning the data space into blocks B_1 … B_4
- QDT_i groups all publishers with data items in B_i

Block	Data items	
B_1	San Diego, fire	→ QDT_1
B_2	San Francisco, gold	→ QDT_2
B_3	**New York**, stocks	→ QDT_3
B_4	food, **weather**	→ QDT_4

- route queries on the corresponding QDT
- routing Q_4 on QDT_3 or QDT_4 ?



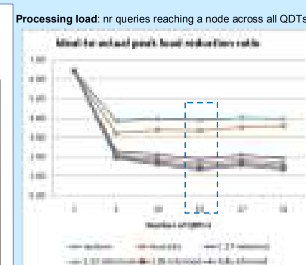
- ideally, use the most selective data item to route with
- in practice, this is not possible → use *informed routing*: avoid routing with low selective (popular) data items

- **Experimental setup**
  - 10,000-node overlay network simulator
    - 9,400 publishers and 600 routers
  - XML Wikipedia dump of 1.1M articles (8.6GB)
  - 50,000 conjunctive queries
    - each query has 1..10 conjunctive data items with at least one match in the global collection

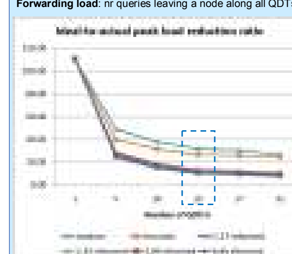
- **Measuring the throughput**
  - processing and forwarding load at each node
    - peak load ↘ results in throughput ↗
  - ideal-to-actual peak load reduction ratio
    = (peak load for k QDTs) / (average load for 1 QDT)

- **Results**
  - bring actual peak load very close to the ideal load
    - near-optimum peak load reduction at 15 QDTs for Scribe generated topologies
  - query selectivity estimation
    - for only 1-3% state, we get 65-75% of the routing benefit

Processing load: nr queries reaching a node across all QDTs



Forwarding load: nr queries leaving a node along all QDTs



- fanout-balanced trees are closest to optimal throughput