FeedEx: Collaborative Exchange of News Feeds

Seung Jun and Mustaque Ahamad
Georgia Tech Information Security Center
Georgia Institute of Technology
Motivation

• RSS/Atom feeds have become increasingly popular
  – Published by most traditional media and blogs

• Scalability of feed servers
  – Frequent pull requests create high load
  – Infrequent requests increase latency and may lead to missed items

• Our Approach
  – Use resources at peer nodes to deliver feed items
  – Scalable growth in resources with service demand

• Challenges
  – Peers may not fully cooperate and execute the agreed protocols
FeedEx Overview

• Feeds have different update and usage patterns.
  – A new hybrid transport mechanism
  – Pull from servers
  – Push among peer nodes
• Peers in FeedEx
  – Form a distribution mesh,
  – Fetch feeds from web servers occasionally, and
  – Exchange new entries among each other
  – Peer incentives for exchanging entries
RSS/Atom Primer

• Feed format

<feed>
  <title>NYT Technology</title>
  <!-- other elements -->
  <entry>
    <title>Basics: Going Wireless on ...</title>
    <link>http://www.nytimes.com/2006/05/18/...</link>
    <summary>Wi-Fi has revolutionized the...</summary>
    <!-- other elements -->
  </entry>
  <!-- more entries -->
</feed>

• Current way of reading feeds
  – Stand-alone applications (e.g., Mozilla Thunderbird)
  – Web-based service (e.g., Bloglines and My Yahoo!)
Analysis of Feed Publishing

• Purpose
  - Interesting by itself and helpful in designing FeedEx

• Methodology
  - 245 popular feeds monitored for 10 days
  - Feeds fetched every 2 minutes
Publishing Rate by Rank

Entries published per day (log scale) vs. Rank (log scale)

- Reuters
- CNN
- Yahoo(T)
- Yahoo(E)
- BBC(U)
- Fark.com
- Yahoo(M)
- ABC
- BBC(W)

Rank (log scale)

Entries published per day (log scale)
Entry Count

Mean of entry count

Range of entry count

Rotten Tomatoes
MSDN
EurekAlert
Techbargains.com
Slate
MacInTouch
Washington Post
MSNBC
EurekAlert
Techbargains.com
Publishing Rate by Time

Entries published per hour

- Reuters
- Yahoo(M)
- Motley Fool
- NPR

Time (day)
Entry Lifetime

Cumulative probability vs. Lifetime (hours)

- CNN
- FOX News
- Beta News
- Techbargains.com
Architecture of FeedEx

To News Feed Servers

To Neighbors

Neighbor Server

RPC

From Neighbors

To Neighbors

Feed Fetch Scheduler

To List Server

Connector
Bootstrapping

• Obtain a list of peers
  – Dedicated list server (Gnutella and BitTorrent)
  – Embedding (Pseudoserving [Kong and Ghosal 1999] and CoopNet [Padmanabhan and Sripandkulchai 2002])
  – Local cache

• Connect to peers
  1. Establish connection
  2. Exchange subscription sets: {(url,hop),…}
Neighbor Selection

• Metrics for good neighbors
  - Subscription set match
    \[ u(Q) = \sum_{i \in (S_P \cap S'_Q)} w_i d^{-h_i} \]
  - Topological proximity
  - Duration of relationship
Adaptive Fetching from Servers

• Coordinated fetching by peers
  – High coordination overhead
  – Lots of nodes with high churn rate

• Solution: Adaptive fetching
  – Freshness rate $f$: Fraction of new entries in a fetched document
  – Set a target freshness rate $f_t$
  – Fetching interval is doubled or halved, bounded by $T_{\text{min}}$ and $T_{\text{max}}$
Entry Exchange Among Peers

• New entries obtained
  – By fetching from web servers
  – From neighbors

• Entry bundle
  – A set of new entries
  – Document identifier (did): Assigned by SHA-1 digest
  – Flooded to matching neighbors

• Two-phase flooding
  – `check_did(did)` call: 344 bytes including HTTP request header
  – `put_entries(bundle)`
Incentive Mechanism

• Pairwise fairness is simple and effective
  – Uses local information only
  – Easy to implement and enforce the mechanism

• Contribution metric $c_{j,i}$:
  \[ c_{j,i} += w_f \cdot h_f \]

• Deficit of contribution $d_{i,j}$:
  \[ d_{i,j} = c_{i,j} \cdot c_{j,i} \]

• Node $i$ ensures $d_{i,j} < D$ for every neighbor $j$ and a parameter $D$. 
Prototype Implementation

- **Python**: python.org
- **XML-RPC**: xmlrpc.com/spec
- **Twisted**: twistedmatrix.com
- **SQLite**: sqlite.org
- **Universal Feed Parser**: feedparser.org
Experimental Setup

• Two modes
  – Stand-alone applications: sln
  – FeedEx: xch

• Metrics
  – Time lag
  – Missing entries
  – Communication cost

• Experiments
  – Use 189 PlanetLab nodes
  – Run 22 hours on a weekday
  – Primary factor: 6 fetching intervals
  – Let each node subscribe 20 out of 70 feeds
Results: Time Lag

![Graph showing the relationship between fetching interval (hours) and time lag (hours). The graph indicates a linear increase in time lag with increasing fetching interval, with SLN and XCH showing distinct trends.]
Results: Missing Entries

- XCH miss
- XCH gain
- SLN miss
- SLN gain

Missing entries (%) vs. Fetching interval (hours)
Results: Communication Cost

![Graph showing received calls per minute against fetching interval (hours). The graph compares check_did and put_entries for different intervals.]
Advantages

• Server scalability
• Archivability
• Controllability
• Filtering and recommendation
• Privacy
Related Work

- News feed delivery
  - Corona (Cornell)
  - FeedTree (Rice)


Conclusions

• A new transport mechanism for news feeds
  – Pull by and exchange among peers
• FeedEx encourages cooperation by enforcing pair-wise fairness, while achieving
  – Reduced feed server load
  – Low latency
  – High coverage
  – Low communication overhead