Adaptive Hierarchical Clustering of Message Flows in a Multicast Data Dissemination System

Yoav Tock, Nir Naaman, Avi Harpaz, Gidon Gershinsky

IBM Haifa Research Lab, Israel
Outline

- Introduction
  - The System – Pub/Sub Messaging for Data Dissemination
  - Multicast Technology
- Multicast Mapping
- Clustering Algorithms
  - Modified K-Means, Hierarchical Clustering
- Real-Life Messaging-Load Model
- Experiments & Results
- An Adaptive System
- Future Directions
- Summary
The Basic Scenario – Pub/Sub for Market Data Dissemination

- Publisher divides data feed into a large number information flows (topics), (~100K) e.g. stock symbols, futures, commodities
- Many stand-alone subscribers (~1K)
- Subscribers display interest heterogeneity - are interested in different yet overlapping subsets of the topics
- Any single topic may be delivered to a large number of subscribers (hot / cold topics)

- Unicast – duplicate transmissions
- Flooding (Broadcast) – receivers burdened by unwanted incoming traffic
Multicast Technology

- IP multicast, Network layer
  - A single packet sent by a transmitter reaches all the hosts that joined a certain Multicast Group
  - Unreliable, no traffic control, no ordering

- Reliable Multicast Transport (RMT) Protocols
  - Reliability, Ordering, Flow & Congestion control
  - “Session” or “Stream” - transport layer entity

- Cannot allocate a group (or stream) per topic
- Limited number of usable multicast groups (NE state problem, receiver resources)
- Limited number of reliable multicast streams

- # Flows >> # RMT Streams >= # IP MC Groups

  => Mapping Flows to Streams
  => Mapping Streams to Groups
Map Structure and Filtering Cost

- Each Topic is mapped to a single RMT stream
- Each RMT Stream is mapped to a single multicast group
- Client filtering is a must
- The cost to the client depends on implementation details

Multicast Mapping

Network layer

Transport layer

Messaging layer
Message Aggregation and Filtering Cost

- Aggregation - multiple messages from the same RMT stream share the same packet
  - At transport layer
    - Some processing for each packet
    - Some processing for each message
    - Amortization of packet-level processing across multiple messages, increases performance
  - At messaging layer – processing per message
    - Depends on implementation

- We estimated the effect of message aggregation and included it in the cost function

![Diagram of cost function](Image)
Example

Cost: $\alpha \sum \text{Excess\_Stream}(n) + \beta \sum \text{Excess\_Topic}(n)$

A two level clustering problem

Multicast Mapping
Algorithm Input - Messaging Statistics

- **Publication**
  - The list of published topics
  - The publication rate of each topic

- **Subscription**
  - The list of topics each client required
  - Client are anonymous

- **Interest Matrix**
  - A binary matrix indicating the interest of the clients

- **Publication Rate Vector**

<table>
<thead>
<tr>
<th>#</th>
<th>Topic</th>
<th>Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>Cars:Toyota/Hilux</td>
<td>10</td>
</tr>
<tr>
<td>#2</td>
<td>Cars:Honda/Civic</td>
<td>20</td>
</tr>
<tr>
<td>#3</td>
<td>Comp:IBM/pSeries</td>
<td>30</td>
</tr>
<tr>
<td>...etc</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Interest Matrix**

**Users’ Interest Vector**

**Topic’s Audience Vector**
Mapping Algorithm

- **Input**
  - interest matrix, topic rate vector

- **Basic insight**
  - Put “similar” topics in the same group
  - “Similar” topics have a similar audience
  - A group with a homogenous audience causes less filtering to the audience

- **Take the rate into account**
  - The cost of putting two topics in the same group
  - The cost of adding a new topic to a group of topics

---

### Interest Matrix

<table>
<thead>
<tr>
<th>Users</th>
<th>Topics</th>
<th>Filtering Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>V</td>
<td>X</td>
</tr>
<tr>
<td>2</td>
<td>V</td>
<td>V</td>
</tr>
<tr>
<td>3</td>
<td>X</td>
<td>V</td>
</tr>
<tr>
<td>4</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

Rk – the rate of topic k
Iterative Clustering Algorithm (K-means)

- **Init:** Topics are assigned into a fixed number of groups
- **Move:** In each step, remove a single topic, and move it to the best group – the one producing the lowest cost
- **Cost:** After each epoch, compute total filtering cost
- **Stop:** time elapsed | cost does not improve | exceeded max number of iterations | number of topics moved

<table>
<thead>
<tr>
<th>Topic group</th>
<th>Group audience vector</th>
<th>Candidate topic 5</th>
<th>The cost of adding topic 5 to topic group {1,2,3}</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2 3</td>
<td>V X V</td>
<td>V</td>
<td>0</td>
</tr>
<tr>
<td>V V X</td>
<td>V V X</td>
<td>V</td>
<td>0</td>
</tr>
<tr>
<td>V V V</td>
<td>V V V</td>
<td>V</td>
<td>0</td>
</tr>
<tr>
<td>X X V</td>
<td>V X V</td>
<td>X</td>
<td>R5</td>
</tr>
<tr>
<td>X X X</td>
<td>X V V</td>
<td>V</td>
<td>R1+R2+R3</td>
</tr>
<tr>
<td>X X X</td>
<td>X X V</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>X X X</td>
<td>X X V</td>
<td>X</td>
<td>R1+R2+R3+R5</td>
</tr>
</tbody>
</table>

The best group for topic K is the group with the lowest cost.
Hierarchical Clustering Algorithms

- **Streams First (SF)**
  - Cluster flows to streams
  - Cluster the resulting streams into groups

- **Group First (GF)**
  - Cluster flows into groups
  - Within each group separately, cluster flows into streams.

- **An Iterative Approach (IT)**
  - Iterative invocation of GF and SF
  - Taking the best map from all the iterations

- **Random Restart with Annealing (RRA)**
  - Random reassignment of a diminishing percentage of flows to streams,
  - Do a GF step
  - Taking the best map from all the iterations

Algorithm

F S G
1 2

SF GF
Randomize
Messing Load Model – Based on Market Research

- **Financial front office**
  - Hundreds of users, requiring stock quotes and financial information from several markets
  - Up-stream action (from brokers to market – buy/sell) is reflected in the down-stream traffic (from market to broker – stock quotes)

- **Topic space structure**
  - Within each market, symbol popularity and rate are exponentially distributed (NYSE market research)
  - Several different markets, with Avg. popularity and size prop. \( \sim 1/m \) (assumption).
Real Life Messaging Load Model

- Based on statistical analysis of NYSE daily trade data
- 20K Topics
- 500 Subscribers
- Avg. ~70 topics / user
- Min 15 topics / user
- Max 115 topics / user
- Avg. message fan out ~10.1 clients

- Multicast - message is transmitted once
- Unicast transmitter data rate is $x10$ of multicast!
The Effect of the Number of Groups and Streams

- Increasing the number of streams and groups always improves performance
- Hierarchical filtering is more efficient than non-hierarchical
- Relative effectiveness depends on the amount of work in each filtering layer

Experiments
Algorithm Comparison

- GF is better than SF
- GF is fastest (not shown)
- Iterative algorithms
  - produce better results
  - take longer to execute (not shown)
- GF / Random = 0.4 - 0.6

Experiments
The Case For Adaptive Mapping

- User interest & message rate change during the day
  - Across markets
  - Within a market
  - In response to world events
  - Trading hours
- Manual management
  - Expensive, intractable
  - Error prone
- The “average” map
  - Of yesterday or a few days back
- Dynamic, Adaptive
  - Adapts to interests and rate
  - Runtime migration mechanism

An Adaptive System
Adaptive Multicast Infrastructure

- **Run**: running a messaging load in a given configuration.
- **Profile**: profiling publications and subscription.
- **Optimization**: the profiling results are fed into the optimization algorithm. The result is a map.
- **Change**: change publisher and subscriber configuration to the new map.

- The optimization starts from a previous map (fast)
- The adaptation time scale can be days, hours, minutes
- Change process is automatic, subject to QoS requirements
- Manual override – process control, map editing, pub/sub profiles
JMS Provider Proof of Concept

Overall View

(with Zvi Har’El)

An Adaptive System
Migration Protocol

General requirements
- Preserve flow message sequencing
- Avoid duplicate transmissions
- Conform with the multicast reliability guarantees
- Fast - reasonable time from start to finish
- Scalable – number of clients / subscriptions

Efficient protocol
- Not “stop the world”, no pipeline drainage
- Messaging activity and throughput is hardly affected

Use existing RMT API with minimal changes

A layered approach
- Isolation of lower level protocols
- Allow for two levels of quality of service
Flow & Stream Migration - Open Loop

- Based on standard reliable multicast transport protocol / infrastructure (e.g. PGM)
- No feedback from receivers, reliability based on timing, receiver detects failures
- Prepare phase – two signals (Change + Beacon)
- Switch phase – two signals (Last + First)
Overall Performance – Extensive Wild Card

- JMS messaging provider POC
- Hierarchic topic-based
- Subscriptions are unique and overlapping, e.g.
  - */b/*/d
  - /a/*/d
- ~7 topics/user @ 10 users
- Unicast – approx. 1/n
- JS1-MC – stream per unique subscription: causes data duplication – performance degradation is almost like unicast

Multicast mapping is scalable, and applicable
Current & Future Work  (With Gregory Chockler, Roie Melamed)

- Distributed Large Scale Pub/Sub
  - A large number of topics  \((x1000)\)
  - A large number of users  \((x10000)\)
  - Correlated user interests  \((x100 / \text{User})\)
  - High churn
  - No IP multicast

- Based on overlay network, P2P
  - That takes into account the user interest

- How do we
  - Define abstract dissemination channels
  - Map topics to abstract dissemination channels
  - Migrate topics between channels

Future
Summary

- Large scale multicast Pub/Sub
  - A huge number of topics
  - A limited number of RMT streams, IP multicast groups
  - Hierarchic approach
- Cost function – hierarchic filtering, message aggregation
- Estimated the relative cost of transport vs. messaging layer filtering
- Iterative clustering algorithm based on K-means
- Several hierarchic clustering algorithms
- Real-life messaging load based on NYSE market research
- Hierarchic filtering is better than flat
- Advantage for efficient filtering at transport layer
- The challenges of an adaptive fully distributed system