Evaluating Advanced Routing Algorithms for Content-Based Publish/Subscribe Systems

Instructor: Dr. Yingshu Li
Presented by: Marco Valero
Outline

Publish/Subscribe Systems

Content-Based Publish/Subscribe Systems

Routing Algorithms
  - Routing Tables Sizes
  - Filtering Forwarding Overhead

Conclusions
Publish/Subscribe provides a simple communication abstraction which enables development of large-scale distributed applications. Information consumers (subscribers) indicate their interest in some information via subscriptions. Information producers (publishers) publish their data as events, which in turn are matched to subscriptions and delivered to interested subscribers.

Keywords:
Consumers   Producers   Advertisements   Brokers
In many pub/sub systems, publishers post messages to an intermediary broker and subscribers register subscriptions with that broker, letting the broker perform the filtering. The broker normally performs a store-and-forward function to route messages from publishers to subscribers.
In the pub/sub model, subscribers typically receive only a sub-set of the total messages published.

Filtering: is the process of selecting messages for reception and processing. There are two common forms of filtering: topic-based and content-based.

This paper focuses on content-based filtering through content-based routing which includes the following Routing Algorithms:

- Identity-Based Routing
- Covering-Based Routing
- Merging-Based Routing
Routing Tables: Subscription-based routing tables are used to route notifications from producers to consumers and the advertisement-based routing tables are used to route subscriptions from consumers to producers.

A routing table consist on a set of routing entries with a pair \((F, D)\) a filter \(F\) and a destination \(D\).
Content-Based Publish/Subscribe Systems

sub(F)

sub(G)

Routing tables

(G,B_4)

(F,B_2)

(G,B_3)
Content-Based Publish/Subscribe Systems

\( \text{sub}(F) \)

\( \text{sub}(G) \)

\((F, B_2)\)

\((G, B_3)\)

\((G, B_4)\)

Routing tables

\( n \in N(F) \)

\( n \in N(G) \)
Content-Based Publish/Subscribe Systems

Routing tables

\[ n \in N(F) \]

\[ n \in N(G) \]
Content-Based Publish/Subscribe Systems

Routing tables

\( n \in N(F) \quad n \in N(G) \)
Content-Based Publish/Subscribe Systems

\[\text{sub}(F) \quad \text{sub}(G)\]

\[(F, B_2) \quad (G, B_3)\]

Routing tables

\[n \in N(F)\]
\[n \in N(G)\]
Content-Based Publish/Subscribe Systems

1. \( n \in N(F) \)
2. \( n \notin N(G) \)
3. \( n \notin N(H) \)

Broker \( B_1 \)

Forwarding Engine

Routing Table

- \((F, B_3)\)
- \((G, B_4)\)
- \((H, B_3)\)
Content-Based Publish/Subscribe Systems

![Diagram of Content-Based Publish/Subscribe Systems]

- $n \in N(F)$
- $n \notin N(G)$
- $n \notin N(H)$

1. $n$ flows through $B_2$ to $B_3$ and $B_4$.
2. The routing table contains:
   - (F, B₃)
   - (G, B₄)
   - (H, B₃)
Content-Based Publish/Subscribe Systems

1. \( n \in N(F) \)
2. \( n \notin N(G) \)
3. \( n \notin N(H) \)

Broker \( B_1 \)

Forwarding Engine

Routing Table

- \((F, B_3)\)
- \((G, B_4)\)
- \((H, B_3)\)
Setup of Experiments

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$E$</td>
<td>The set of all edges connecting brokers</td>
</tr>
<tr>
<td>$m$</td>
<td>The number of stocks</td>
</tr>
<tr>
<td>$N_B$</td>
<td>The set of neighbors of a broker $B$</td>
</tr>
<tr>
<td>$S$</td>
<td>The source of notifications</td>
</tr>
<tr>
<td>$\sum</td>
<td>R</td>
</tr>
<tr>
<td>$V$</td>
<td>The set of all brokers</td>
</tr>
<tr>
<td>$x$</td>
<td>The number of active subscriptions</td>
</tr>
</tbody>
</table>

Table 1. Symbols used in this paper.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of consumers per local broker</td>
<td>1 – 200</td>
</tr>
<tr>
<td>Number of subscriptions per consumer</td>
<td>10</td>
</tr>
<tr>
<td>Number of stocks</td>
<td>1000</td>
</tr>
<tr>
<td>Number of notification sources</td>
<td>1</td>
</tr>
<tr>
<td>Number of event routers</td>
<td>40</td>
</tr>
<tr>
<td>Number of local event brokers</td>
<td>67</td>
</tr>
<tr>
<td>Number of neighbor broker</td>
<td>$x$</td>
</tr>
<tr>
<td>Number of hierarchy Levels</td>
<td>5</td>
</tr>
<tr>
<td>Distribution of clients to brokers</td>
<td>$x$</td>
</tr>
<tr>
<td>Distribution of stocks to clients</td>
<td>random</td>
</tr>
<tr>
<td>Degree of locality</td>
<td>none</td>
</tr>
</tbody>
</table>

Table 2. Fixed and varied parameters.
I. Simple Routing

- Each broker has global knowledge about all active subscriptions
- Each subscription stored in every routing table
- Size of routing tables grows linearly in the number of active subscriptions and the number of brokers
- Each routing table affected by a new/canceled subscription
1. Simple Routing

- Each broker has global knowledge about all active subscriptions
- Each subscription stored in every routing table
- Size of routing tables grows linearly in the number of active subscriptions and the number of brokers
- Each routing table affected by a new/canceled subscription

![Diagram showing routing tables and subscriptions]

1. sub(F)
2. sub(G)
1. Simple Routing

- Each broker has global knowledge about all active subscriptions
- Each subscription stored in every routing table
- Size of routing tables grows linearly in the number of active subscriptions and the number of brokers
- Each routing table affected by a new/canceled subscription
I. Simple Routing

- Each broker has global knowledge about all active subscriptions
- Each subscription stored in every routing table
- Size of routing tables grows linearly in the number of active subscriptions and the number of brokers
- Each routing table affected by a new/canceled subscription
1. Simple Routing

- Each broker has global knowledge about all active subscriptions
- Each subscription stored in every routing table
- Size of routing tables grows linearly in the number of active subscriptions and the number of brokers
- Each routing table affected by a new/canceled subscription
I. Simple Routing

- Each broker has global knowledge about all active subscriptions
- Each subscription stored in every routing table
- Size of routing tables grows linearly in the number of active subscriptions and the number of brokers
- Each routing table affected by a new/canceled subscription
1. Simple Routing

- Each broker has global knowledge about all active subscriptions
- Each subscription stored in every routing table
- Size of routing tables grows linearly in the number of active subscriptions and the number of brokers
- Each routing table affected by a new/canceled subscription
1. Simple Routing

- Each broker has global knowledge about all active subscriptions
- Each subscription stored in every routing table
- Size of routing tables grows linearly in the number of active subscriptions and the number of brokers
- Each routing table affected by a new/canceled subscription
1. Simple Routing

Routing Tables Sizes:

\[ \text{RTS} = (|V| - 1) \cdot x \]

Filter Forwarding Overhead:

\[ \text{FFO} = |V| - 1 \]
2. Routing based on Identity

Uses identity tests for routing decisions

Avoids: Forwarding of identical filters and Routing entries with identical filters for the same neighbor broker

\[ F \equiv G \iff N(F) = N(G) \]
2. Routing based on Identity

Uses identity tests for routing decisions

Avoids: Forwarding of identical filters and Routing entries with identical filters for the same neighbor broker

\[ F \equiv G \iff N(F) = N(G) \]
2. Routing based on Identity

Uses identity tests for routing decisions

Avoids: Forwarding of identical filters and Routing entries with identical filters for the same neighbor broker

\[ F \equiv G \iff N(F) = N(G) \]
2. Routing based on Identity

Uses identity tests for routing decisions

Avoids: Forwarding of identical filters and Routing entries with identical filters for the same neighbor broker

\[ F \equiv G \iff N(F) = N(G) \]
2. Routing based on Identity

Uses identity tests for routing decisions

Avoids: Forwarding of identical filters and Routing entries with identical filters for the same neighbor broker

\[ F \equiv G \iff N(F) = N(G) \]
2. Routing based on Identity

Uses identity tests for routing decisions

Avoids: Forwarding of identical filters and Routing entries with identical filters for the same neighbor broker

\[ F \equiv G \iff N(F) = N(G) \]
2. Routing based on Identity

Uses identity tests for routing decisions

Avoids: Forwarding of identical filters and Routing entries with identical filters for the same neighbor broker

\[ F \equiv G \iff N(F) = N(G) \]
2. Routing based on Identity

Uses identity tests for routing decisions

Avoids: Forwarding of identical filters and Routing entries with identical filters for the same neighbor broker

\[ F \equiv G \iff N(F) = N(G) \]
2. Routing based on Identity

**Routing Tables Sizes:**

\[ RTS = 2 \cdot m \cdot (|V| - 1) \]

**Filter Forwarding Overhead:**

FFO depends on the number of active subscriptions
3. Routing based on Covering

F covers G ⇔ N(F) ⊇ N(G)
3. Routing based on Covering

$F$ covers $G \iff N(F) \supseteq N(G)$
3. Routing based on Covering

F covers G ⇔ N(F) ⊇ N(G)
3. Routing based on Covering

F covers G ⇔ N(F) ⊇ N(G)
3. Routing based on Covering

F covers G ⇔ N(F) ⊇ N(G)
3. Routing based on Covering

F covers G ⇔ N(F) ⊇ N(G)
3. Routing based on Covering

F covers G ⇔ N(F) ⊇ N(G)
3. Routing based on Covering

F covers G ⇔ N(F) ⊇ N(G)
3. Routing based on Covering

Advanced Routing Algorithms
3. Routing based on Covering
3. Routing based on Covering

Advanced Routing Algorithms
3. Routing based on Covering
3. Routing based on Covering
3. Routing based on Covering
3. Routing based on Covering

Advanced Routing Algorithms
3. Routing based on Covering

Advanced Routing Algorithms

Routing based on Covering
3. Routing based on Covering
Routing Tables Sizes:

\[ \text{RTS} = m \cdot (|V| - 1) \]

Filter Forwarding Overhead:

FFO starts at the level of simple routing and decreases for larger number of subscriptions
4. Routing based on Merging

\[ N(G) \cup N(H) = N(F) \]

Advanced Routing Algorithms
4. Routing based on Merging

\[ N(G) \cup N(H) = N(F) \]
4. Routing based on Merging

\[ N(G) \cup N(H) = N(F) \]
4. Routing based on Merging

\[ N(G) \cup N(H) = N(F) \]
4. Routing based on Merging

\[ N(G) \cup N(H) = N(F) \]
4. Routing based on Merging

\[ N(G) \cup N(H) = N(F) \]
4. Routing based on Merging

\[ N(G) \cup N(H) = N(F) \]
4. Routing based on Merging

\[ N(G) \cup N(H) = N(F) \]

Advanced Routing Algorithms
4. Routing based on Merging

\[ N(G) \cup N(H) = N(F) \]
4. Routing based on Merging

\[ N(G) \cup N(H) = N(F) \]

![Diagram showing routing based on merging with nodes and edges labeled with subscripts (G, B_1), (G, B_3), (H, B_2), (G, B_3), (H, B_3).]
4. Routing based on Merging

\[ N(G) \cup N(H) = N(F) \]

\[ (H, B_3) \]

1. sub(G)

\[ (G, B_3) \]

2. sub(H)

\[ (F, B_3) \]

\[ (G, B_1) \]

\[ (H, B_2) \]
Routing Tables Sizes:

\[ \text{RTS} < 2 \cdot (|V| - 1) \]

Filter Forwarding Overhead:

FFO is similar to that of covering routing but decreases more slowly.
Advertisements can be used as additional mechanism to further optimize content-based routing. They are filters that are issued by producers to indicate (and revoke) their intention to publish certain kinds of notifications.

\[ N(F) \cap N(G) \neq \emptyset \]
Advertisements can be used as additional mechanism to further optimize content-based routing. They are filters that are issued by producers to indicate (and revoke) their intention to publish certain kinds of notifications.

1. \( \text{adv}(F) \)

\[ N(F) \cap N(G) \neq \emptyset \]
Advertisements can be used as additional mechanism to further optimize content-based routing. They are filters that are issued by producers to indicate (and revoke) their intention to publish certain kinds of notifications.

\[ \text{adv}(F) \]

\[ N(F) \cap N(G) \neq \emptyset \]
Advertisements can be used as additional mechanism to further optimize content-based routing. They are filters that are issued by producers to indicate (and revoke) their intention to publish certain kinds of notifications.

\[
N(F) \cap N(G) \neq \emptyset
\]
Advertisements can be used as additional mechanism to further optimize content-based routing. They are filters that are issued by producers to indicate (and revoke) their intention to publish certain kinds of notifications.

1. adv(F)  2. sub(G)

\[ N(F) \cap N(G) \neq \emptyset \]
Advertisements can be used as additional mechanism to further optimize content-based routing. They are filters that are issued by producers to indicate (and revoke) their intention to publish certain kinds of notifications.

\[ R \]

\[ (B, B_4) \]

\[ B_1 \]

\[ B_3 \]

\[ B_4 \]

\[ B_5 \]

\[ B_6 \]

1. adv(F) 2. sub(G)

\[ N(F) \cap N(G) \neq \emptyset \]
Advertisements can be used as additional mechanism to further optimize content-based routing. They are filters that are issued by producers to indicate (and revoke) their intention to publish certain kinds of notifications.

\[
\begin{align*}
\text{false} & \quad (B, B_4) \\
B_1 & \quad B_2 \\
B_3 & \quad B_4 & \quad B_5 & \quad B_6 \\
1. \text{adv}(F) \quad 2. \text{sub}(G) \\
N(F) \cap N(G) \neq \emptyset
\end{align*}
\]
Simple Routing with Advertisements

Routing Tables Sizes:

\[ \text{RTS} = \text{Pavg} \times x \]

(Pavg is the average path length from a customer to a notification source)

Filter Forwarding Overhead:

\[ \text{FFO} = \left( |V| - 1 \right) / \text{Pavg} = 28.4 \]
Simple Routing vs. Identity Routing (RTS)
Covering Routing vs. Merge Routing (RTS)
Simple Routing vs. Identity Routing (FFO)
Covering Routing vs. Merge Routing (RTS)
Conclusions

1. Advanced routing algorithms can be considered mandatory in large-scale publish/subscribe systems.

2. The use of advertisements considerably improves the scalability.

3. Advanced routing algorithms operate efficiently in more dynamic environments.

4. The good behavior of the algorithms even improves if the interests of the consumers are not evenly distributed, which can be expected in practice.
Q&A
References


Lukasz Opyrchal and Atul Prakash. Content-based publish subscribe systems: scalability and security. ACM 2004


Content-Based Routing http://kbs.cs.tu-berlin.de/ivs/Lehre/WS0405/ASM/08.pps