Term Project

Publisher-Subscriber Application

Shraddha Atrawalkar **Kumar Abhishek**

(ssa2923@g.rit.edu) **(kxa7954@g.rit.edu)**

I. Introduction:

The Publish-Subscribe system is a messaging system/mechanism where entities are loosely coupled and the sender does not have to always know who its receivers are.

The 2 entities involved in the messaging pattern are:

- "Publishers": Who publish messages/feed for a particular "topic".
- "Subscribers": Who receive messages by subscribing to particular topics.

There is a middle-ware that communicates between the publisher and subscriber and forwards messages according to the topics subscribed to appropriate subscribers.

This messaging system provides the user to take advantage of the dynamic nature of the network since any number of receivers can be added to or removed from the network at any given time.
(In our demo, Subscribers leave the network by unsubscribing the topics)

II. **Motivation**:

In a normal messaging system, the sender and the receiver have to aware of each other's identity. Also, they had to be connected to each other directly in order to exhibit efficient message transfer.
Now, imagine such a system to be scaled to thousands of more senders and receivers. This system would not only get complicated but it would involve a complex mechanism of forwarding only those messages that a receiver wants to receive.

To overcome this issue, we have implemented the publisher-subscriber messaging pattern which not only allows senders and receivers to be loosely coupled, be unaware of each other's identity but also, allows senders to receive only those messages that he "subscribes" to.

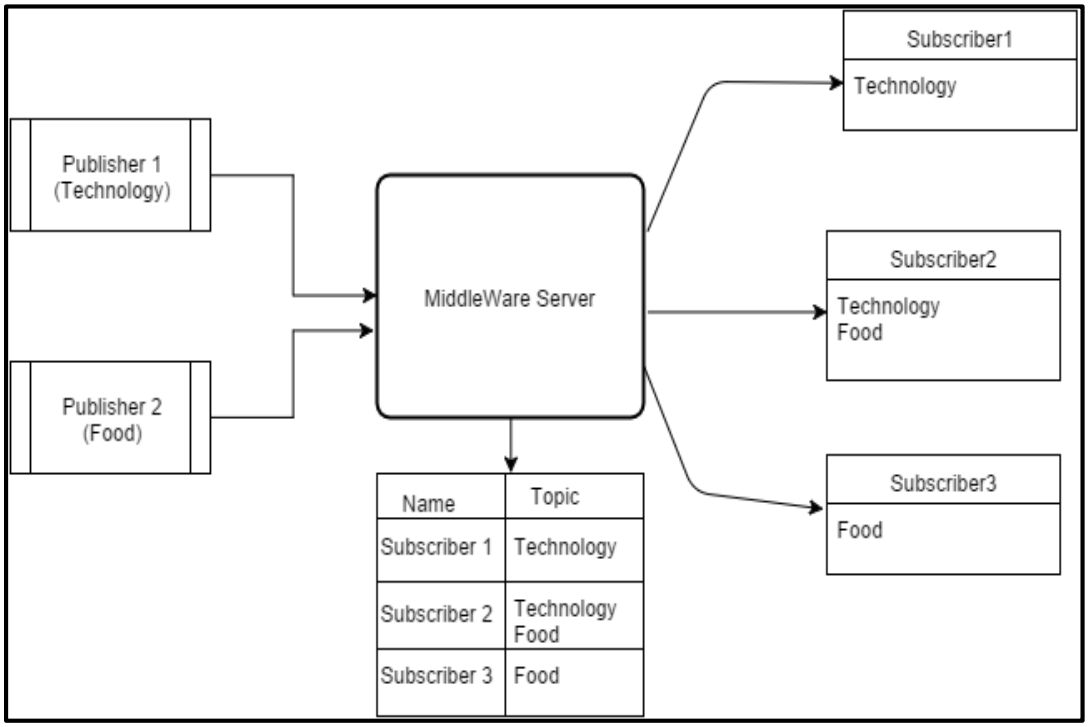
III. Types of Message filtering mechanism:

There are 2 types of message filtering mechanisms:

1. **Topic-based filtering (Implemented in our application):**
   When a subscriber subscribes to certain topics that a publisher publishes messages for; the subscriber receives only those messages that were published against subscribed topics by the publisher.

2. **Content-based filtering:** Subscribers subscribe to certain kinds of content for which they want to receive messages for. The middleware has to implement different algorithms (Say, text mining etc.) to determine the type of content published by the publisher and accordingly forward the message.

IV. System Architecture:



As it can be seen from the figure above,
There can be multiple publishers for particular topics or a publisher with multiple topics. (*We have implemented the latter one*)

The middleware maintains a map for the Subscriber and the topics he is subscribed to. This makes it easy for the middleware to forward messages from the publisher to the appropriate subscriber.

One point to note here is that, neither the publisher knows who the messages are being forwarded to, not does the subscriber know where he is receiving the messages from. This particular property is called being loosely coupled.

V. System Implementation:

In our implementation of this system, we have ***NOT*** used any system provided libraries and have implemented the whole mechanism from scratch.
The implementation included an efficient use of socket programming to communicate between the publisher-middleware and the middleware-subscriber.

"*As mentioned during our presentation and discussed with Professor, the Publisher-Subscriber system in general does not provide guaranteed delivery. But in our implementation, with the use of TCP protocol, we have tried to achieve guaranteed delivery between the entities"*

The implementation is as follows:
Publisher: Since this is a Single Publisher-Multiple Topics publisher, the publisher is initiated at a given port. The publisher adds a list of topics for which he will later publish feeds.

Middleware: The middleware communicated with the Publisher using **TCP protocol** and maintains a list of Subscribers and topics they are subscribed to in a **forwarding table**.

Subscriber: Whenever a subscriber joins the network, he has to register itself with the middleware. The Subscriber-Middleware also use TCP for communication.

The entire mechanism has to maintain ***Synchronization*** between them to ensure efficient delivery of messages.

Also, when the subscriber decides to unsubscribe to a topic, the middleware removes it from its forwarding table assuming the subscribe is no longer interested in receiving messages for a particular topic.

VI. Challenges during implementation:

1. UI based: The major issue we faced while implementing the application is the synchronization between the UI and the backend application. But, we have that solved now.

2. Forwarding table based: There were certain instances during the testing of the application that, when multiple subscribers were added to the system, the system experienced loss of messages. This issue would prevail in any message mechanism system that involves a large number of senders or receivers. Another point to note is the publish/subscribe mechanism does not involve resending of messages, so it in itself does not handle message loss.

VII. Advantages/Disadvantages:

Advantages	Disadvantages
Scalable	No Guaranteed Delivery
Message distribution can be controlled	Instability due to load
Abstraction layer for Sender(Publisher) and Receiver(Subscriber)	Secured Communication not provided

Loose Coupling	Mediator needed, message format & unknown data structures

VIII. Application:

The working of our application is included in the video attached.

The application supports any number of topics for any number of subscribers. For Simplicity, our demo shows 2 subscribers and 2 topics. It can be scaled further for more subscribers.