Project Topic - Publish-Subscribe system for news feed updates

Team

- Sahil Mhatre (018174091)
- Shivank Singh Thakur (018186064)
- Dheeraj Kumar Alla (017929977)

Problem Description

A distributed Publish-Subscribe (Pub/Sub) system is needed to facilitate reliable distribution of news feed updates across a broad user base, ensuring that subscribers receive relevant information as soon as it is published. This system enables publishers to push news articles, alerts, and updates to specific topics (like politics, sports, or technology), while subscribers can choose to receive updates on topics of interest. We plan to build this system to efficiently handle a high volume of updates from multiple publishers while maintaining the consistency and reliability of the news content delivered to subscribers. Key challenges include managing concurrent updates, ensuring the correct order of news delivery, and providing fault tolerance in case of network failures or node outages.

Motivation

With the increasing demand for personalized and real-time information delivery, a Pub/Sub system for news feed updates is crucial for media platforms, social networks, and mobile applications. Users expect to receive breaking news and topic-specific updates instantly, especially during live events or critical situations. Implementing a Pub/Sub system using decentralized algorithms like Gossip Protocol, mutual exclusion, timestamps and replication ensures that the system scales effectively, avoids central bottlenecks, and remains resilient against node failures. Additionally, such a system promotes a dynamic and user-centric news distribution model where subscribers have control over the topics they follow, enhancing user engagement and satisfaction in consuming relevant and timely content. Users get a more personalized and engaging news experience tailored to their interests, while publishers can connect with their audiences more effectively. Additionally, timely access to accurate information is crucial during crises, enhancing public awareness and safety. Ultimately, a well-functioning Pub/Sub system can empower people to stay informed, fostering a more connected and knowledgeable society.

Previous Work

Recent studies in distributed computing have focused on gossip algorithms that help nodes share information in networks with limited resources. These algorithms have shown how their efficiency is linked to specific mathematical properties, which helps in finding better ways to use them. Researchers have also examined the strengths and weaknesses of gossip protocols, making it clear when they work best and when alternatives might be needed.

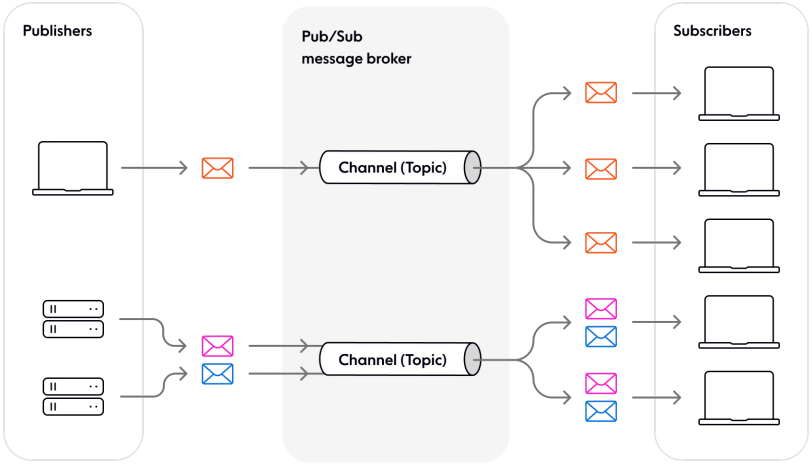
In the area of mutual exclusion, which ensures that multiple processes can access shared resources without conflict, two main types of algorithms have been explored: assertion-based and token-based. The assertion-based methods reduce message exchanges, while token-based methods offer a simpler approach but struggle with managing tokens properly. These studies help improve how resources are accessed in distributed systems.

Additionally, research on distributed databases looks at how different ways to manage data access can impact performance. It has been found that replicating data can improve response times under certain conditions. Lastly, the development of vector timestamps helps clarify the order of events in message-passing programs, which is essential for avoiding issues like race conditions. Together, these studies provide practical solutions to common problems in distributed computing.

Technical Approach

High Level Architecture

Below is a brief flow diagram for the working of publish-subscribe system:



Implementation Overview

1. **System Architecture:**
    - **Decentralized Brokers**: We will set up a cluster of broker nodes on AWS, using EC2 instances to host these brokers. Each broker will manage subscriptions and deliver updates to subscribers.
    - **Publishers and Subscribers**: Publishers will send updates to their nearest broker, and subscribers will receive updates based on their topic preferences.
2. **Core Algorithms**:
    - **Gossip Protocol**: Brokers will employ a Gossip Protocol to efficiently propagate updates among themselves, ensuring that even if some brokers fail, the updates can still reach all nodes over time.
    - **Timestamps**: Each news update will be assigned a logical timestamp to maintain order, using methods like Lamport timestamps to ensure that updates are processed in the correct sequence.
    - **Mutual Exclusion**: We will implement a distributed mutual exclusion algorithm to manage concurrent updates to shared topics, ensuring data integrity and preventing conflicts.
    - **Replication and Consistency**: Using primary-backup replication or quorum-based protocols (e.g., Paxos or Raft), we will ensure that updates are consistently replicated across brokers.
3. **Handling Updates**:
    - When a publisher sends an update to its local broker, the broker will check for mutual exclusion before assigning a timestamp and propagating the update using the Gossip Protocol. The update will be replicated across brokers to maintain consistency.
4. **Fault Tolerance**:
    - To ensure reliability, we will take periodic distributed snapshots of the system state, allowing us to recover from failures effectively. Brokers will also gossip state information to synchronize with one another.

Deployment Strategy

The deployment of the distributed Pub/Sub system will be done on AWS. Amazon EC2 will host the broker instances, while Amazon RDS will manage relational data like user subscriptions. News articles will be stored in Amazon S3 for scalable object storage.

Containerization will be implemented using Docker to package the broker application, which simplifies deployment and scaling. By encapsulating the application and its dependencies in containers, it ensures consistent environments across development, testing, and production.

Timeline

Task	**Deadline**
High Level Architecture Design	Oct 8, 2024
DB Schema Design	Oct 20, 2024
Low Level Design	Oct 25, 2024
Gossip Protocol Integration	Nov 10, 2024
Consistency Protocol Integration	Nov 17, 2024
Timestamps Integration	Nov 24, 2024
UI Interface Development	Nov 27, 2024
Deployment	Nov 30, 2024
Testing and Bug Fixes	Dec 2, 2024

Resources

- Frameworks and Libraries: Node, Express and React JS
- Databases: NoSQL
- Cloud Platforms: AWS
- Containerization: Docker
- Development Tools: VSCode, Postman, GitHub, Trello

References

[1] S. Boyd, A. Ghosh, B. Prabhakar and D. Shah, "Gossip algorithms: design, analysis and applications," Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies., Miami, FL, USA, 2005, pp. 1653-1664 vol. 3, doi: 10.1109/INFCOM.2005.1498447.

[2] Ken Birman. 2007. The promise, and limitations, of gossip protocols. SIGOPS Oper. Syst. Rev. 41, 5 (October 2007), 8–13. doi: https://doi.org/10.1145/1317379.1317382.

[3] Ratan K. Ghosh; Hiranmay Ghosh, "Mutual Exclusion," in Distributed Systems: Theory and Applications , IEEE, 2023, pp.189-217, doi: 10.1002/9781119825968.ch8.

[4] Taubenfeld, G. (2008). Concurrent Programming, Mutual Exclusion. In: Kao, MY. (eds) Encyclopedia of Algorithms, doi: https://doi.org/10.1007/978-0-387-30162-4_88.

[5]B. Ciciani, D. M. Dias and P. S. Yu, "Analysis of replication in distributed database systems," in IEEE Transactions on Knowledge and Data Engineering, vol. 2, no. 2, pp. 247-261, June 1990, doi: 10.1109/69.54723.

[6]A. Bechini and K. . -C. Tai, "Timestamps for programs using messages and shared variables," Proceedings. 18th International Conference on Distributed Computing Systems (Cat. No.98CB36183), Amsterdam, Netherlands, 1998, pp. 266-273, doi: 10.1109/ICDCS.1998.679522.