Report on Movie Share P2P Application

Lillian Huang, Haoxue Li, Chenhao Pan, Xuefei Zhou

In this project, we used the PeerBase P2P framework [1] to build the movie share application. The application provides the querying movie and rating movie functionalities. Moreover, we built the chatting message capability for peers to have free chat.

1. PeerBase P2P Framework

The PeerBase P2P framework was built by Nadeem Abdul Hamid. In this framework, the author used the network sockets to build two-way communication between two network nodes. A node discovers and builds peer nodes by probing the peer nodes of its known peers. The probing is limited by a configurable parameter of maximum hops. So, each peer node in the PeerBase network may have a different peer list.

The PeerBase has a stabilizer mechanism in which each node pings its peer node in a configurable time interval. If the peer node does not accept the ping, it is assumed that the peer node is dead and then is removed from the peer list.

2. Movie Share Application

The Movie Share Application is built on the PeerBase P2P framework. A node may have a movie database read capability or read/write capability. There is a configurable maximum number of peers for each node.

The application has three main functionalities: query a movie, rate a movie and send a message.

Query Movie

A node can query movie details by providing some pieces of information about the title of a movie. The application then sends the query to its peer nodes for the information. When a node receives a movie query message, it will query the movie database if it has the database read capability and then send the response back to the sender. Otherwise, it will send the query to its peer nodes for the information until the configurable maximum hops have been reached.

Rate Movie

A node can rate a movie. To rate a movie, a movie ID and a rate must be entered. The application then sends the rating data to its peer nodes for the information. When a node receives a movie rate message, it will update the rating in the movie database if it has the database write capability and then send the new rating back to the sender. Otherwise, it will

send the movie rating message to its peer nodes until the configurable maximum hops have been reached. The new rating is based on the ratings from total number of raters.

Message

A node can send messages to peer nodes. To send a message, enter message text in the input text box. The application then sends the message to its peer nodes. When a node receives a message, it will display the message on the UI. It then sends the messages to its own peer nodes until the configurable maximum hops have been reached.

The Movie Share application has a Java Swing based GUI. It lists peer nodes and two output windows to display incoming data. It also provides text boxes for inputting data.

3. Software Development Lifecycle

For the Software Development Lifecycle, we chose to do scrum since it is used widely in the industry. We used Zenhub to keep track of the user stories and responsibilities of each team member.

Product Backlog Meeting:

We had our first product backlog meeting after we settled down the framework and application theme. We set two main user stories and their related features:

- Build Movie Database
    - Find the public movie resources. We decided to download movie data from a public API (https://www.themoviedb.org/documentation/api)
    - Build a database model for the movie data storage
    - Design the database table
    - Load the movie data into the database.
- Search Movie (a user can search movie by the title information)
    - get movie title from input
    - query the name using wildcard SQL for movie information in the movie database
    - return the table of id, title, director, year, rating of the movies which titles are similar to the movie title input
- Rate Movie (a user can rate movie by the id)
    - Get the movie id from the input (this id can be queried by the search movie function)
    - Get the corresponding input of rating
    - Movie rating is the average rating of all people who rate the movie.
    - Return the updated rating of the movie.

Then we had our second product backlog meeting to add another user story:

- Group Chatting (a user can send and receive messages from all other connected users)

- Once a node is up and builds the peer list by using the PeerBase Framework, the user is able to input messages and hit the send button.
- Once the message is sent, it will be sent to every node in the peer list.

Sprint Planning Meeting:

Sprint 1
In our first sprint planning meeting, we set our goal to finish implementing a P2P system with the function of movie searching and movie rating and group chatting. After implementing, we need to test and debug the whole system to make sure it works well. And, we decided to include the following diagrams/documentation to help with the coding process which include Requirements with Use Case diagrams, GRL diagrams, BPMN diagrams. Also, we should finish the project report and prepare for the final presentation.

The detailed planning is listed below:
Coding part:
04/08 - 04/09 Get movie resources from the API, design the structure of the Movie database
04/10 - 04/11 Build the Movie database and store movie information to the database
04/12 - 04/13 Read the PeerBase P2P framework documents to get familiar with the framework
04/14 - 04/17 Implement a basic P2P system with the PeerBase P2P framework
04/18 - 04/20 Implement the movie search functionality
04/21 - 04/23 Implement the movie rating functionality
04/24 - 04/26 Implement the group chatting function
04/27 - 04/29 Implement the GUI with Java Swing for user interface
04/30 - 05/01 Come up with more test cases for testing and debugging
05/02 - 05/04 Test and debug all three functions

Documentation part:
04/08 - 04/10 Write the first version of Requirements, draw Use Case diagrams
04/11 - 04/13 Draw the first version of GRL diagrams
04/14 - 04/16 Draw the first version of BPMN diagrams
04/17 - 04/26 Revise Requirements, Use Case diagrams, GRL diagrams and BPMN diagrams
04/27 - 05/04 Modify Requirements, Use Case diagrams, GRL diagrams, BPMN diagrams to fit exact implementation
05/05 - 05/07 Write Project Report
05/08 - 05/09 Prepare for the final presentation

Tasks assigned:
Requirements with Use Case diagrams - Chenhao Pan
GRL diagrams - Haoxue Li
BPMN diagrams - Xuefei Zhou
Coding and User Stories - Lillian Huang
Project Report - all four team members

Final Presentation - all four team members

Daily Standup Meeting:

We had our daily standup meeting at least once in a week. We check the project progress and add next-week plans in the meeting.

Standup Meeting 1
Initially we only wanted to make a chat function for our project but after meeting with Professor Franchitti, having only a chat function deemed to be a little bit too easy for our project. Thus, we decided to make a movie rating/querying function to have two use cases.

Progress:
- Write the project proposal
- Basic functionalities: Search Movie and Rate Movie

Plan:
- Write user stories
- Reqview to maintain the Requirements Specifications
- LucidChart to make UML Use Case Diagram, BPMN models
- GRL Diagram using Eclipse JUCM
- MySQL will be used for the database
- Come up with new functionalities/use cases
- Determine a specific P2P framework

Standup Meeting 2
Before the second standup meeting, we had difficulties in coming up with a P2P framework and initially wanted to have a movie rating/querying between peers if peers have neighbor's host and port. With continuous emails back and forth between the Professor and one of our teammates, we decided to build our application on the top of Peerbase framework. Besides the P2P framework, we also had difficulties on knowing how to design the application, in the command terminal or GUI. Therefore, our first draft of the Requirements Specification was not up to par on what some team members had expected about the application. After further communication, the GRL Diagram, BPMN Diagrams, and Requirement specifications were planned to be rewritten according to the updated specifications of the application for the next standup meeting.

Progress:
- Write the first version of Requirement Specifications, GRL diagram and BPMN diagram
- Discuss and modify our project proposal
- Functionalities/use cases to be implemented are movie searching, movie rating and group chatting
- Use PeerBase as the underlying P2P framework

Plan:

- Modify and write the second version of Requirement Specifications, GRL diagram and BPMN diagram to incorporate the new P2P framework and group chatting functionality
- Read the PeerBase P2P framework documents and get familiar with the framework
- Design the movie database table and create database model
- Find movie resources for database

Standup Meeting 3
With all the diagrams finalized, we were able to start on the coding. One tricky part of inserting the movies database records from a json file that was extracted using an API. The team member in charge of coding the database insertion used Jersey, and Jackson framework to map the objects which were being read in by the json file containing the records to be added into the database.

Progress:
- Get movie resources from API (https://www.themoviedb.org/documentation/api)
- Complete the requirement specification document and diagrams
- Designed the movie database table and created database model

Plan:
- Implement a basic P2P system based on the PeerBase P2P framework which can interact with the database
- Learn Java GUI
- Implement the Query Handler, Query Processor, Query Response Handler and Query Button Listener
- Implement the Rate Handler, Rate Processor, Rate Response Handler and Rate Button Listener

Standup Meeting 4
We started to make progress on coding the application and was able to query movies and rate movies successfully. During the process, we still had some difficulties such as the fact that some movie titles have multiple words. The QueryResponseHandler had difficulty splitting the PeerMessage by spaces since the length of the title would vary. Thus, as a result we decided to replace all the spaces in the movie title to be %20 which represents a space in encoded URL. Initially for the peer message type, we had “QUERY” as a type for querying the movie. However, as we took a look closely into the PeerMessage.java file which is implemented by the P2P framework, we noticed that the type of every message should be 4-bytes thus only 4 characters are allowed. We changed the “QUERY” message type to be “QUER” to fix this issue and changed “QUERYRESPONSE” to be “QRES”.

Progress:
- Implement the movie search functionality
- Implement the movie rate functionality

Plan:
- Implement the MessageHandler and MessageButtonListener
- Implement the GUI Display

- Write the report
- Come up with test cases for testing and debugging

Standup Meeting 5
As we finished the coding portion of the project, we discovered that multiple responses can be received. This is due to the fact that multiple peer nodes send results back to the sender. To resolve this issue, we decided to add a duplicate detection mechanism. We introduced a message id to indicate the message sequence number for each message type. At the receiving node, it records the message id it has processed. A message with message id less or equal to the recorded id will be ignored. After implementing this duplicate detection mechanism, we resolved the duplicate message issue greatly. However, in some cases, we still receive duplicate messages. After walking through the PeerBase code and debugging the application, we noticed that this was due to the fact that the PeerBase framework uses multithreading when it handles incoming messages which can cause race conditions when the node receives the same messages from different peers. Such duplicate message problems are difficult to resolve unless we change the PeerBase framework.

Progress:
- Implement the GUI Display
- Implement the Group Chat functionality
- Write the first draft of report
- Finish testing and debugging for all three functionalities

Plan:
- Finalize the report
- Complete the slides
- Prepare for presentation with project demo

Sprint Review Meeting:

Sprint 1
At the end of sprint 1, we had our first sprint review meeting.
What we've done:
- We've implemented a basic P2P system with PeerBase P2P framework
- We've collected movie information for the API and designed and built the Movie database
- We've implemented the two basic functions with the Movie database- Search Movie and Rate Movie
- We've implemented another functionality - Group Chatting
- We've implemented a GUI with Java Swing for user interface
- We've finished all the testing and debugging for the whole system
- We've finished all the documentations - Requirements with use case diagrams, GRL diagrams, BPMN diagrams and Project Report
- We've prepared for the project presentation

- When rating the movie, a race condition may occur. This issue comes from the original PeerBase P2P framework. After checking the PeerBase P2P framework carefully, we successfully solved this problem.

Sprint Retrospective Meeting

What went well in the Sprint:

- Was able to get clarifications from the professor about the project requirements.
- The team was able to adapt the requirement changes.
- Completed implementing and testing the movie share application within planned time.
- Was able to quickly remedy the duplicate message issue by adding the duplicate detection mechanism.
- While testing the MVP, the team was able to catch and understand the race condition.
- The team collaborated well on the project documentation.

What could be improved:

- The timing of Sprint as well
- The communications between team members
    - Difficult to get a hold of team members and have meetings due to the fact that we live in different time zones
    - All have different responsibilities besides this course.
    - Thus, we could only have one standup meeting per week to discuss project details and blockers.

What will we commit to improve in the next Sprint:

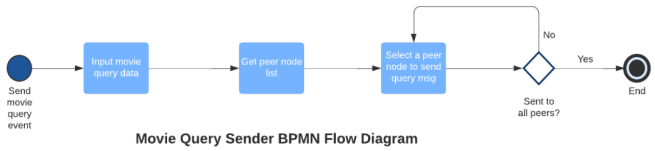
- Improve our communication between team members
- Ask questions if some tasks are unclear to improve our speed in delivering user stories.
- Try to cut our sprints to be a max of 2 weeks per sprint
- Practice better time management.

4. Models

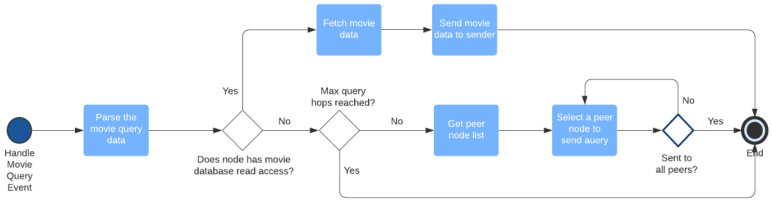
BPMN

Movie Query Sender: When users input movie query data, the movie query sender will get its peer node list and repeat the process of selecting a peer to send the query message until the message is sent to all peers.

Movie Query Receiver: The movie query receiver first parses the movie query data. If the current node has read permission of the movie database, the movie query receiver will fetch movie data and send the movie data to Sender. Otherwise, it will check the maximum query hops. If not reached, it will get its peer node list and repeat the process of selecting a peer to send the query message until the query message is sent to all peers.



Movie Query Sender BPMN Flow Diagram

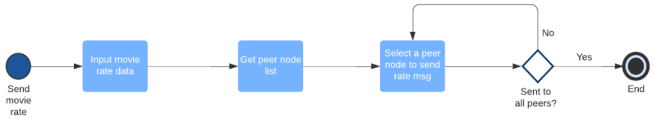


Movie Query Receiver BPMN Flow Diagram

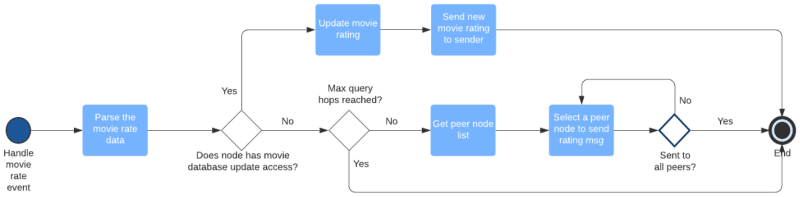
Movie Query Sender / Receiver BPMN Flow Diagram

Movie Rate Sender: When users input movie rate data, the movie rate sender will get its peer node list and repeat the process of selecting a peer to send the rate message until the message is sent to all peers.

Movie Rate Receiver: The movie rate receiver first parses the movie rate data. If the current node has update permission of the movie database, the movie rate receiver will update movie rating and send the new movie rating to Sender. Otherwise, if the current node has no update access to the movie database, it will check the maximum query hops. If not reached, it will get its peer node list and repeat the process of selecting a peer to send the rating message until the rating message is sent to all peers.



Movie Rate Sender BPMN Flow Diagram

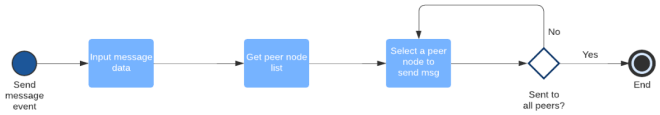


Movie Rate Receiver BPMN Flow Diagram

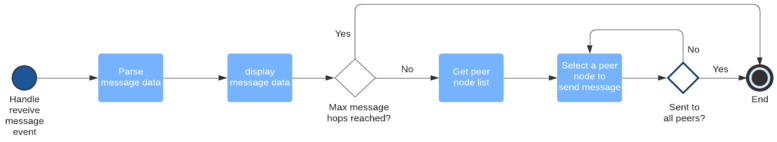
Movie Rate Sender / Receiver BPMN Flow Diagram

Message Sender: When users input message data, the message sender will get its peer node list and repeat the process of selecting a peer to send the message until the message is sent to all peers.

Message Receiver: The message receiver first parses and displays the message data. If the maximum message hops are not reached, it will get its peer node list and repeat the process of selecting a peer to send the message until messages are sent to all peers.



Message Sender BPMN Flow Diagram



Message Receiver BPMN Flow Diagram

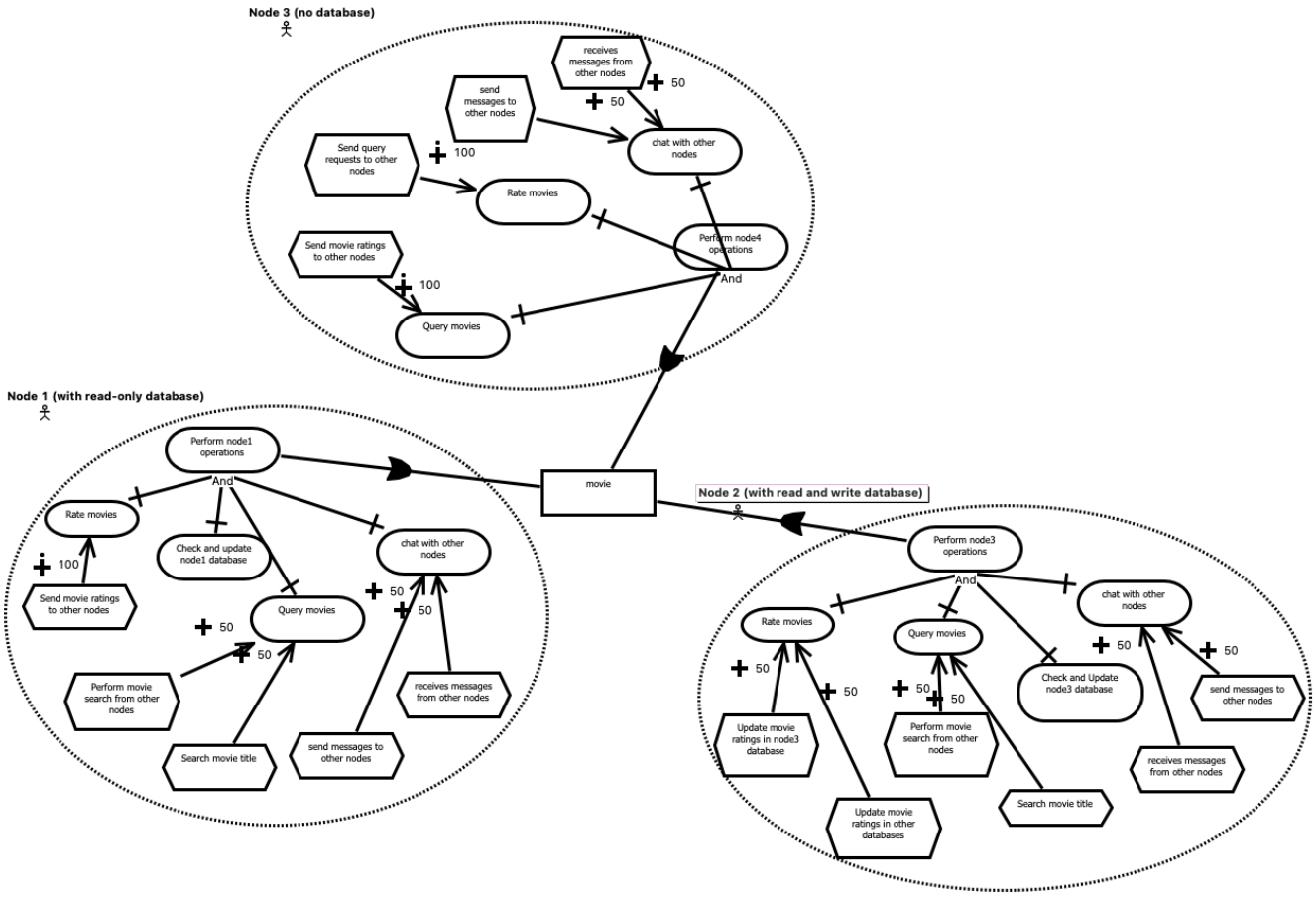
Message Sender / Receiver BPMN Flow Diagram

GRL

We set four nodes as an example for the application. Node 1 has a read-only database. Node 2 has a read-and-write database. Node 3 has no database and if we have other nodes, it will have similar operations with Node 3. For each node, it has three/ four main operation goals: rate movies, query movies, and chat with other nodes.

- rate movies. If this node has a write-available database, this node can handle this request and update movie ratings in its database. Otherwise, it will route its request to other write-available databases on other nodes.
- query movies. If this node has a read-available database, this node can handle this request. Otherwise, it will route its request to other read-available databases on other nodes.
- (node with a database) check and update database.if the node or some other node has performed the rating-movies and querying-movies operations, the node will check the database and update it.
- chat with other nodes. For this request, the node will receive messages from other nodes and send messages to other nodes.

GRL model details are in the following image:



5. Conclusion

Functionality Extension

This application is built for movie sharing features. Ways to extend the application include having a use case of being able to leave your own reviews for the movie as well as show the number of people who rated the movie. There is also a possibility of adding a recommender system where the users can be recommended similar movies to the one they just rated. It can also be extended for other purposes. For example, it can easily be extended to group activity data sharing, social forum activities and file sharing, etc.

Framework Refinements

The PeerBase framework uses multi-threading to handle incoming messages. When a node receives the same message from multiple peer nodes, it could process the same message multiple times. Although we added a duplicate detection mechanism in this implementation. However, due to the multi-threading nature, some race conditions can still happen when the message is processed multiple times. Further enhancement should be investigated and implemented.

Application Scaling

The scaling performance has not been evaluated. We should test the application with a large number of peer nodes. The GUI should also be modified to handle the scale.

References:

[1] Nadeem Abdul Hamid, P2P Programming Framework - Java, http://cs.berry.edu/~nhamid/p2p/framework-java.html