FACULTY OF SCIENCE AND TECHNOLOGY

COURSEWORK FOR THE BSC (HONS) INFORMATION TECHNOLOGY; BSC (HONS) COMPUTER SCIENCE; YEAR 3

ACADEMIC SESSION 2020; SEMESTER 8

NET3204: Distributed System

Project 2020 **DEADLINE: 13th December**

INSTRUCTIONS TO CANDIDATES

- **This project will contribute 40% to your final grade.**
- **This Project has two component, one individual assignment 10 % and a group project 30%.**

IMPORTANT

The University requires students to adhere to submission deadlines for any form of assessment. Penalties are applied in relation to unauthorized late submission of work.

- **Coursework submitted after the deadline but within 1 week will be accepted for a maximum mark of 8%.**
- **Work handed in following the extension of 1 week after the original deadline will be regarded as a non-submission and marked zero.**

Lecturer's Remark (Use additional sheet if required)

I..... (Name) .....std. ID received the assignment and read the comments..... (Signature/date)

Academic Honesty Acknowledgement

"I Darryl Tan Zhe Liang, Cherlynn Chew Hui Fen, Yohannes Luke Koh, Len Wei Xin, Vincent Seaw.(student name). verify that this paper contains entirely my own work. I have not consulted with any outside person or materials other than what was specified (an interviewee, for example) in the assignment or the syllabus requirements. Further, I have not copied or inadvertently copied ideas, sentences, or paragraphs from another student. I realize the penalties *(refer to page 16, 5.5, Appendix 2, page 44 of the student handbook diploma and undergraduate programme)* for any kind of copying or collaboration on any assignment."

Darryl Tan, Cherlynn, Yohannes, Len Wei Xin, Vincent, (11/12/20) (Student's signature / Date)

Member contribution percentage:

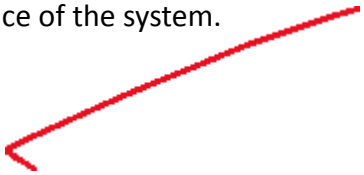
Name:	Student ID:	Contribution Percentage:
Darryl Tan Zhe Liang	17023326	100%
Cherlynn Chew Hui Fen	16103988	100%
Yohannes Luke Koh	17018342	100%
Len Wei Xin	17039322	100%
Vincent Seaw	17028754	100%

Contents

Introduction

Tetris is a tile-matching video game that allows 2 players to match and play together through internet. Tetris uses AKKA as the framework to design a scalable and reliable distributed system. AKKA is a set of libraries that is based upon the actor model, which is used to built high scalable and trustable distributed system. The implication of AKKA system consist of many components such as the Actor System that enables each actor to execute its own isolated context by giving every actor its own mailbox and communicates with other actor via message passing; Cluster which provides fault-tolerant distributed cluster of actor system. These components will be further explained in the later section to breakdown how

these components are implemented in Tetris to achieve scalability and reliability. Moreover, to better understand the system design and communication flow, the physical and interactive model of the system will be shown in the next section. Furthermore, a series of test cases were performed and recorded to ensure the performance of the system.



Physical and interactive model of proposed system:

System architecture diagram:

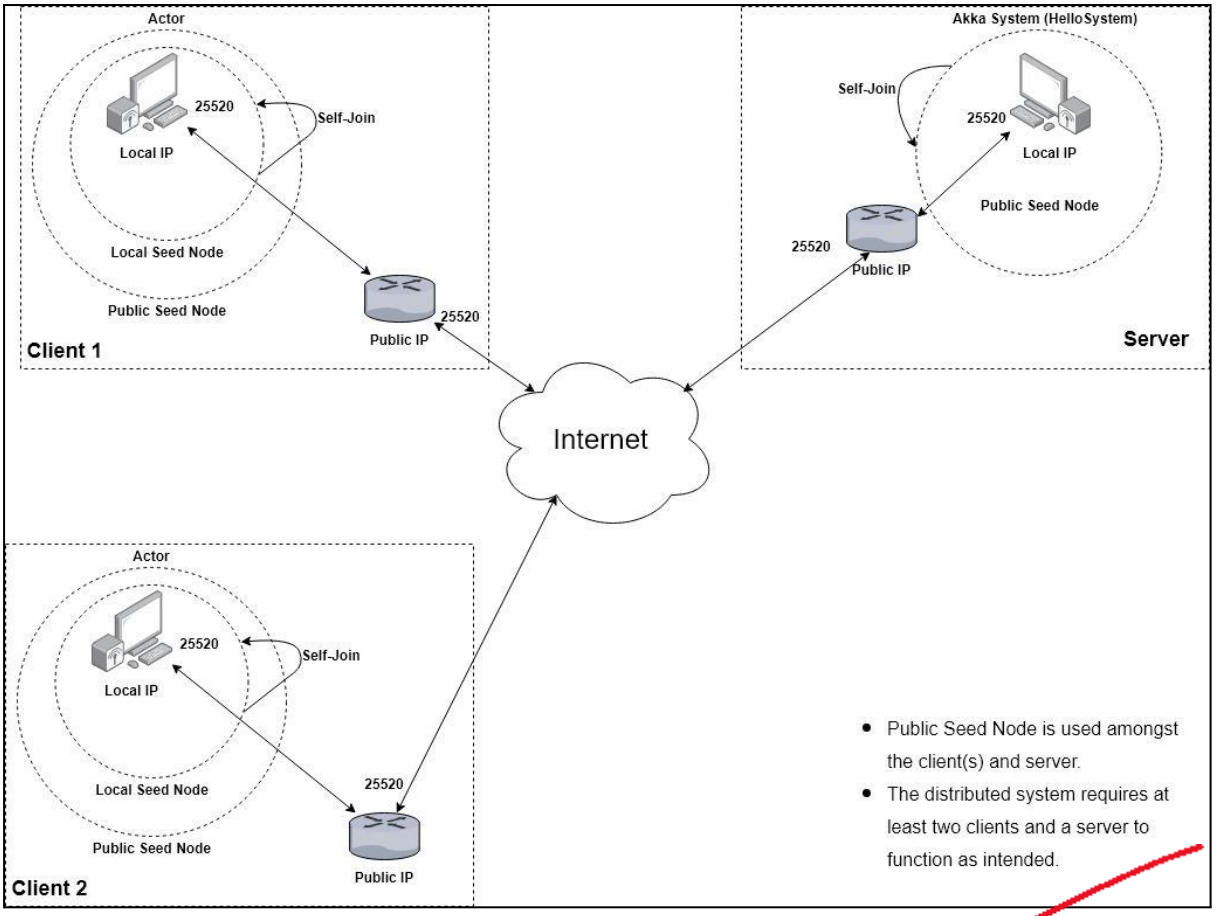


Figure 1.0

Scalability and reliability of the system:

Scalability:

The scalability of the program can be explained using akka clustering methods. The akka clustering method essentially creates a peer to peer communication instead of a client-server communication, it means that there is no single point of failure. In our program, there are 3 nodes that are seed nodes themselves and each of them has their own clusters. 1 seed node is the server side and the other 2 seed nodes is the players. None of them knows about each other but with clustering, 2 players connect to the seed node of the server seed node through the internet, where the seed node is hardcoded on the players. After establishing connection, player A and player B now joins the cluster alongside with the server. However, player A and player B does not know of each other, this is where the scalability of akka cluster takes place. There is a term "gossip" used in this system where player A will somehow heard that there is a player B and that is how they will know about each other in the cluster. When the two players know about each other's existence, even if the seed node dies which is the server during the middle of the game, the two players can still communicate with each other which means they are still able to play with each other. This is the reason why akka clustering was used. However, for this program after the game ends, the two nodes could not play against each other because the game lobby is run by the server which means if the server dies, lobby cannot be loaded. This indicates that the program is not peer to peer at all, which is more likely a client server communication. Other than that, the scalability in terms of the number of nodes connecting to the system, the number of players joining the game lobby is not limited as it is connected to the internet which means multiple players can join the game lobby page. However, if is run locally, the number of ports in that local IP will be the limitation for the number of users joining the server. Furthermore, in terms of efficiency of the number of users in the server and performance, it is not tested in this program, but it should be able to handle multiple users without affecting the performance of the system in terms of lag or delay.

Reliability:

The idea of reliability in the system is the detection of faults followed by fault tolerance and error handling.

The first and foremost is the communication between client and server which can be an issue if the client could not connect to the server. The program has implemented the unreachability command which means that every time a client tries to connect to the server, there will be a message indicating that it is connected if it is reachable. In the event of a client could not reach the server and in order to handle this situation, the client will keep trying to reconnect to the server. However, the server will continue running as usual for other clients and display error messages of the one client that could not connect. This is done so that other clients would not be affected when using the system. Other fault detection during communication such as dead letters was used for debugging purposes, sometimes an actor's message does not arrive consistently, or it is sent somewhere else. This could be an issue regrading seed-nodes or ports or even respective IP addresses of both client and server side. With the help of dead letters, the system can still run, and it helps the system to be more reliable in terms of message delivery after debugging.

After a connection is established, error handling in our Tetris game itself is important to show reliability.

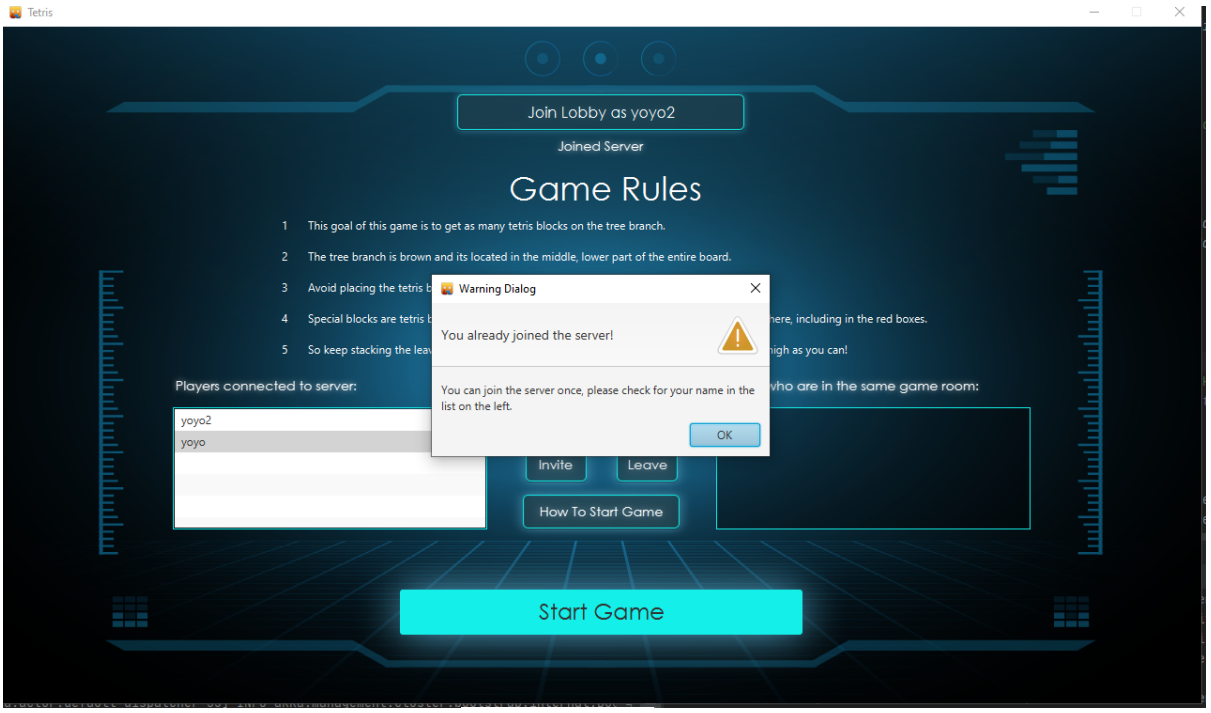


Figure 2.0

The first error handling in the game is that in a scenario where a client already joined the server as you can see from the image above. When the client press join lobby again, an error message will pop up indicating that the client has already joined the server. This is done to avoid duplicates of two same clients joining the server.

Figure 2.1

The image above is another handling where a client tries to invite himself which is not possible, the error message will pop up indicating player could not be invited, choose another player.

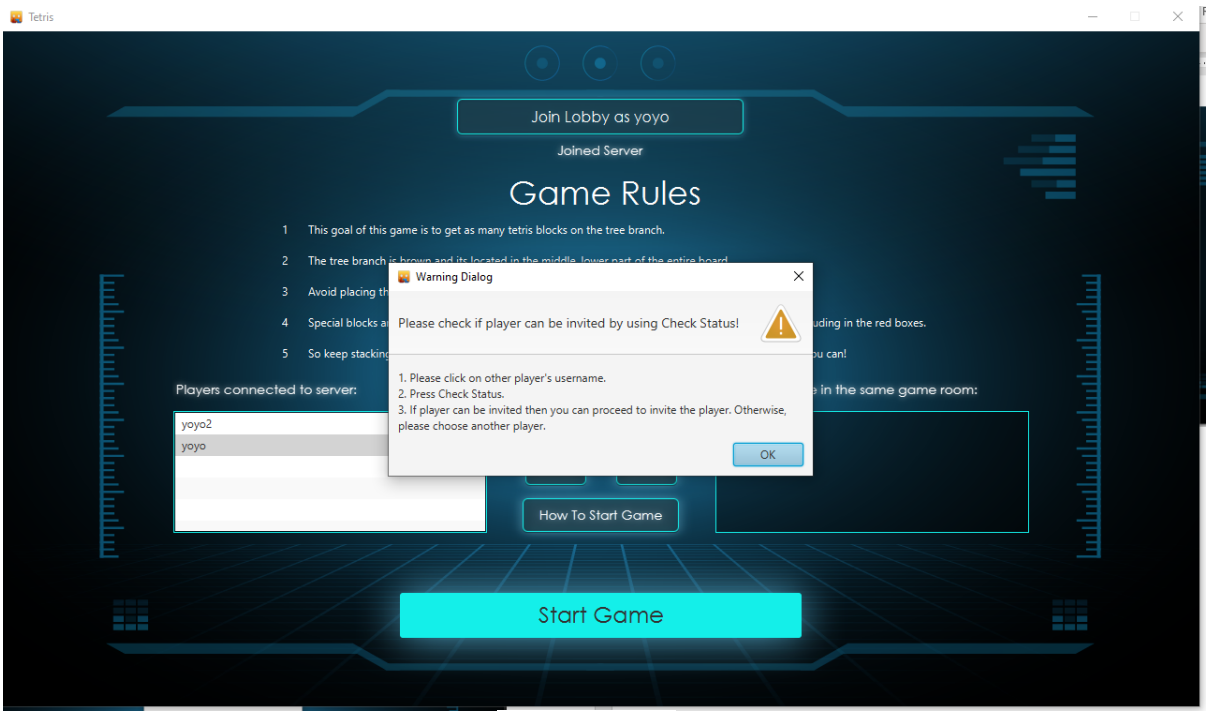


Figure 2.2

This is another error handling which is important in this program in a scenario where a player tries to invite another player into a game without checking the status of the invited player first. A pop-up message will warn the player that before inviting another player, the status of that player has to be checked first to indicate that if he is available for invites or he is currently in game and cannot be invited. This is done to avoid invites when the other player is in game.

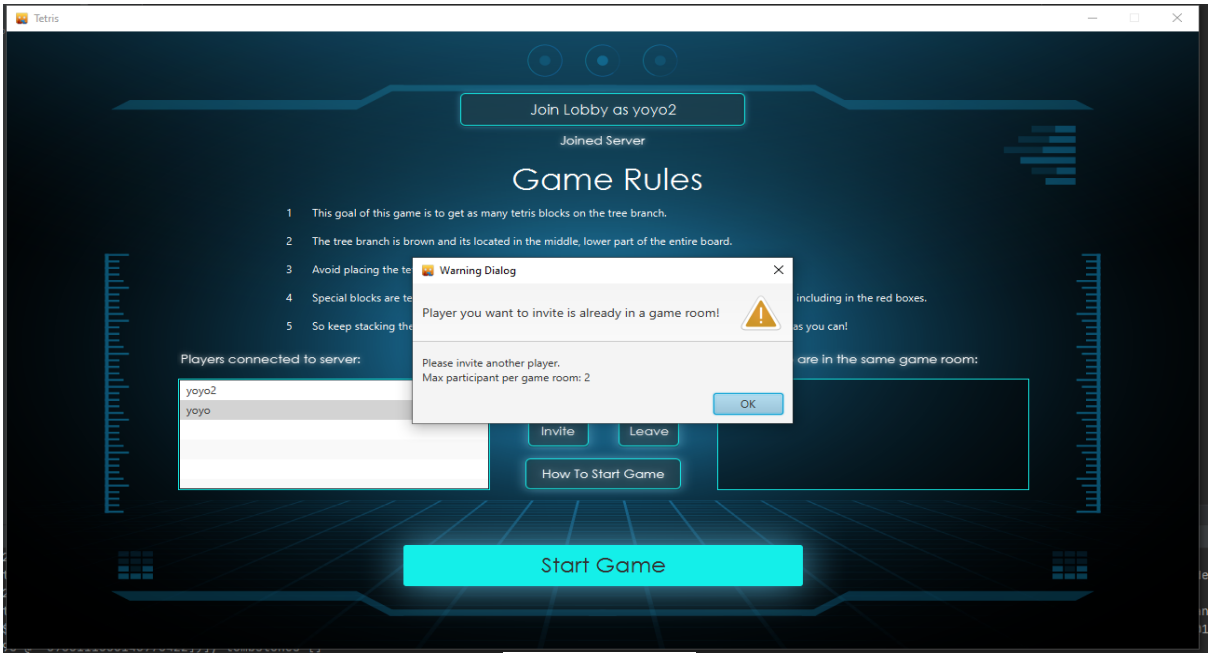


Figure 2.3

The error handling in this image above indicates a scenario where the player's status is already checked but the player still wants to invite the player that is currently in game. The warning message of player to be invited is in game, please invite another player.

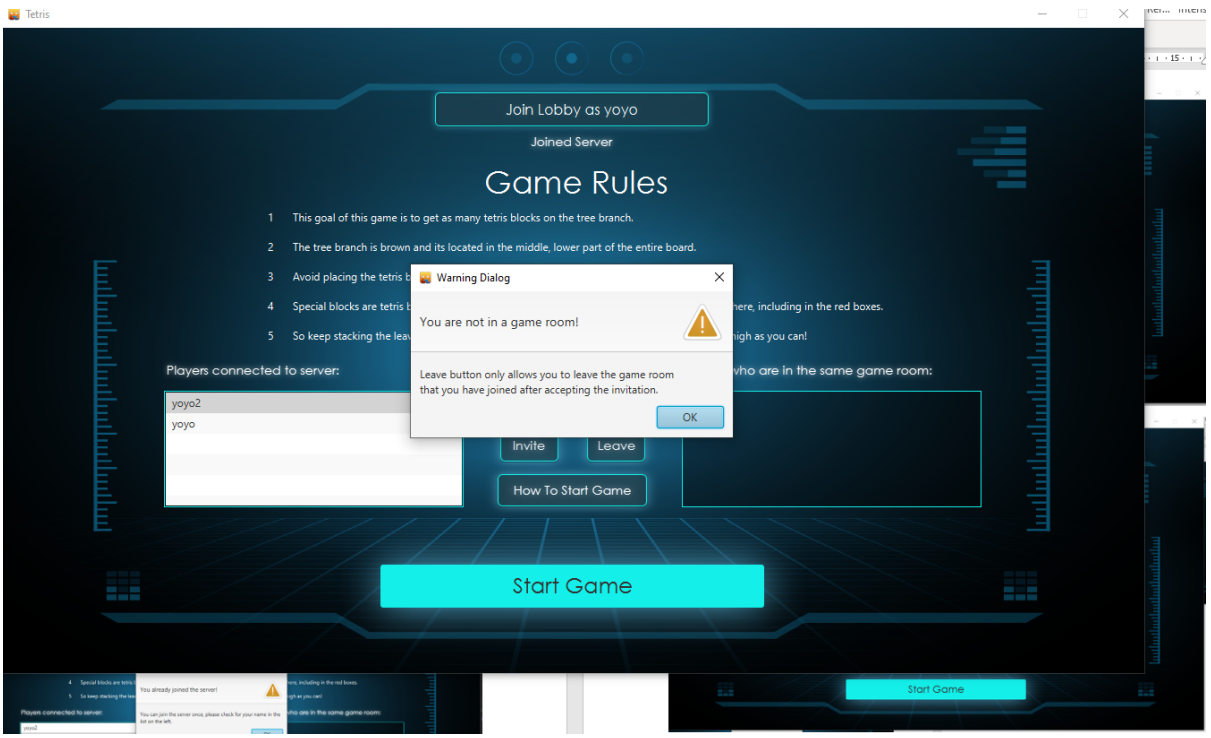


Figure 2.4

The warning message in the image above is when a player tries to click the leave button without even being in the game room. A warning message will tell the player that he is not even in the game room, he cannot leave the game.

Test cases:

Test cases video:

Video file name: TestCases.mp4

Please obtain TestCases.mp4 from Final Project's assignment page at MS Teams. Since the mp4 file is too big to be attached in OneNote.

Test cases specifications:

No.	Test Case Description	Test Steps	Expected Results	Actual Results
1.	Connect 2 clients to the server and play the game until there is a winner.	1. Start server by binding to a port. 2. Start both clients on different ports and connect to server. 3. Client 1 invites Client 2 and waits for them to accept. 4. Once accepted, Client 1 starts the game. 5. Both players play the game until game ends for both players. 6. Both clients disconnect from the server.	After the game ends, a pop-up will notify both clients of the winner and both clients will be redirected back to the main lobby.	As expected
2.	One client goes back to the main lobby while in the game.	1. Repeat steps 1 to 4 from Test Case 1. 2. While playing the game, one player clicks on the back button to return to the main lobby.	Other player is notified that their opponent has left the game. The game ends and the player is redirected back to the main lobby.	As expected
3.	One client pauses the game while in the game.	1. Repeat steps 1 to 4 from Test Case 1.	The game is paused for both players. The opponent will be notified that the	As expected

		2. While playing the game, one player clicks the pause button.	player has paused the game.	
4.	One client closes the game while in the game.	1. Repeat steps 1 to 4 from Test Case 1. 2. While playing the game, one client closes the application window.	Other player is notified that their opponent has left the game. The game ends and the player is redirected back to the main lobby.	As expected
5.	One client invites another client which is already in a game.	1. Repeat steps 1 and 2 from Test Case 1 but with 3 clients. 2. Repeat steps 3 and 4 from Test Case 1 with 2 of the clients. 3. Client 3 tries to invite either Client 1 or 2 to join a game.	Client 3 is notified that the other player is already in another game.	As expected
6.	One client closes the application by ending the task instead of closing the application window.	1. Repeat steps 1 and 2 from Test Case 1. 2. When both players are in the lobby, Client 2 closes the application by stopping the terminal.	The window for Client 2 closes and is removed from the list of players connected to the server.	As expected

Table 1.0

Personal Reflections:

Vincent Seaw (17028754)

An explanation of your understanding any distributed system concepts that you have apply in your assignment.

The distributed system that was designed and developed in this assignment fulfils a space coupling and time-coupled distributed system concept. This is because, it has properties in which communication is directed towards a specified receiver or receivers, and the receiver(s) must exist at that moment in time. Hence, the system performs communication within the cluster via unicasting, where the message is passed directly from the sender to the receiver.

A description of how you applied the distributed concepts in your assignment.

Store the actor reference that was clicked in the player's list into a variable and communicate with that actor only. The player's list will only show clients that are connected to the server and is within the same cluster. When a client leaves, instead of the server multicasting, the server will communicate to each actor's reference that are still connected in the cluster, and inform them of the new player's list in a one-to-one manner, then the clients will update their own player's list accordingly.

The problems encountered during this assignment and how you solved these problems.

1. Omission handling: notify the other player when a client leaves in the middle of the game.
Solution: When a client starts the game, all the client's reference in the same game room will be passed to the server, then the server will keep track if any of these client's reference invoked the leave case match. If yes, then notify the other client who is sharing the same game room based on the index.

2. Animation passing: To allow animation of the other client to be visible and shown correctly without altering own client's animation.
Solution: Create different functions to capture the other client's animation and pass the animation information to the other client whenever own client's animation is changing.

3. Storing information: Variables within a controller will be changed when the same fxml is required to be loaded again via a function, so clients are unable to keep track of another client's status correctly, and controllers are unable to pass information to each other.
Solution: Create an object that stores all the information that is required, then use the object's variables to pass the information around whenever necessary.

An evaluation of the strengths and weaknesses of your submitted work.

Strengths:

1. Communication is not handled by server, server is mainly in charge of being the seed node for clients to recognize and join the cluster, server is also in charge of notifying clients accordingly when a client leaves the cluster. Hence, single point of failure is reduced, making the system more reliable and scalable at the same time.
2. Workable on internet, the system was tested on both LAN and internet, it appears that under the correct configuration, and a proper server running that is visible on the internet, the system can work on the internet.
3. System can handle omissions such as when a player closes the application in the middle of the game, the other player will be notified, or when a player force closes the application via task manager, all the clients and server can detect that a client within the cluster is unreachable, and they will update the list of players connected to the server accordingly.
4. Program is not linearly designed, it will not force players to restart the application when game ends. Since, players can press back after the game ends to return to the main lobby page and initiate a new game with the correct referencing to other players within the same cluster.

Weakness:

1. If server is disconnected, players will not be notified.
2. Packet delivery is "at-most once", hence it is not guaranteed to be delivered.
3. Configuration required for system to be workable on the internet needs to be done manually, it is not automated.
4. Screen size is fixed at 720p.

Include each group's member contribution percentage.

Name:	**Student ID:**	**Contribution Percentage:**
Darryl Tan Zhe Liang	17023326	100%
Cherlynn Chew Hui Fen	16103988	100%
Yohannes Luke Koh	17018342	100%
Len Wei Xin	17039322	100%
Vincent Seaw	17028754	100%

Yohannes Luke Koh (17018342)

An explanation of your understanding any distributed system concepts that you have apply in your assignment.

One of the concepts of distributed system that was utilized in this assignment is the reliability. Reliability of a system is important as communication across the system will result in different type of errors and a good system that is reliable must have fault tolerance and even error handling. This is done so that systems do not experience any disruptions if a single machine fails.

A description of how you applied the distributed concepts in your assignment.

One key take away from the our system is the connection between the client and server. In the scenario of two clients are connected to the server and suddenly one client has a connection issue which will result in a failure from the client side. The server will then receive an unreachable message and the client will either keep on reconnecting or decided to be terminated. The server will still be able to run even if the client starts to fail as the server will still look for potential connections from clients and able to run the game with the other clients. This can be seen as a reliable system as the server will still be able to run even if a client suddenly failed in the middle of a game or a sudden disconnect before the game starts.

The problems encountered during this assignment and how you solved these problems.

The coding of the interaction of the game is not done by me as I have only worked on the connection to the internet. The problem that we encountered during this is to understand how seed nodes works and the port forwarding on the router. In order for a cluster to join another seed-node, you have to inform the cluster that is configured in the application config file, that server seed-node exist along with the port number so that they are able to connect to the seed-node.

For the port forwarding part, it seems that it only happened to some ISP. Initially, the ports were closed even if the server is running, it means that the port connected to the public IP is not visible to other people. This results in the failure of connecting through the internet to the server side itself. So configuring the port forwarding on our routers solve the problem as

the ports can finally be visible on the internet. This allow us to connect multiple clients to the server that helps us to play the game with each other through the internet.

Strength

1. Players can return to the lobby page without needing to restart the whole client code
2. Multiple clients can connect to the game lobby which allows many players to play the game but in a 1v1 fashion
3. Player 1 can reference correctly of the move of Player 2, and vice versa
4. Can be connected locally and through the internet

Weakness

1. If the server dies, the players cannot play the game anymore
2. Sometimes the termination of the game will result in the server could not update the list of players that are still available in the lobby

Include each group's member contribution percentage.

Name:	**Student ID:**	**Contribution Percentage:**
Darryl Tan Zhe Liang	17023326	100%
Cherlynn Chew Hui Fen	16103988	100%
Yohannes Luke Koh	17018342	100%
Len Wei Xin	17039322	100%
Vincent Seaw	17028754	100%

Darryl Tan Zhe Liang (17023326)

An explanation of your understanding any distributed system concepts that you have apply in your assignment.

We designed and developed a distributed system, 1v1 multiplayer Tetris game, which utilized both space and time coupling distributed system concepts. An initiator node can only communicate with a specific receiver that is distinguished based on its public address and port while requiring that the receiver must also exist as well during communication. The communication mentioned refers to unicasting whereby a sender or initiator must only communicate with only a single receiver at a time.

A description of how you applied the distributed concepts in your assignment.

In our distributed system, the player clicks on any other players that were also connected within the same server and cluster as well before sending an invitation to the chosen player for a game. The process describe beforehand was implemented with the system storing the actor reference of the chosen player and the initiator's client can only communicate with the chosen player via the reference. The outcome of the process results in changes within both the server room list and the game lobby list whereby the server room list changes will be updated for every players connected to the same server while the game lobby list changes can only be visible for the involved players that created the game room after the invitation. The server implementation required that it must communicate the changes occurred to all the players connected to it and the public cluster via a one-to-one communication using the corresponding actor references. The players' client that received the updates from the server would then update its own server room list to match the rest of the clients and server.

The problems encountered during this assignment and how you solved these problems.

One of the major problems encountered during the distributed system's development was omission handling in the case of a client leaving in the middle of match which requires notifying the other client in the same game room. The solution implemented was to store both clients' actor references into an element of game rooms set which would be sent and kept in the server. In the scenario of any client leaving during the middle of the game, the server would keep track and detect it before notifying the other client within the same game room if it happened. The other problem was the animation synchronization issue between

two clients' game board in a game that was not caused by latency but flawed logic in the implementation of the game controller. The implemented solution for this problem was to create functions devised on capturing the other client's board changes or animation and calling it whenever the local client's game board has changes in terms of animation.

An evaluation of the strengths and weaknesses of your submitted work.

The strengths of this distributed system are listed as follow:

Single point of failure is reduced significantly due to lack of dependence on the server for the distributed system to function as intended. Since clients communicate with one another in most cases while the server only responsible for detecting and notifying other clients on the leaving or joining of another client within the same cluster.

States transition is possible whereby players could go back and forth between the main game lobby window and the game itself without the need for restarting the game. Changes occurred during the mentioned process is also kept track among the other clients' and server within the same cluster.

The weaknesses of this distributed system are listed as follow:

Only elegant leaving such as pressing the "Back" button, internet disconnected, "x" button can be handled in terms of omission by notifying other clients' involved on the leaving of that particular client.

Packet delivery via TCP is "at-most-one" which meant that the delivery of message is not guaranteed and may cause latency and synchronization issue during a game.

Include each group's member contribution percentage.

Name:	**Student ID:**	**Contribution Percentage:**
Vincent Seaw	17028754	100%
Cherlynn Chew Hui Fen	16103988	100%
Len Wei Xin	17039322	100%
Yohannes Luke Koh	17018342	100%
Darryl Tan Zhe Liang	17023326	100%

Cherlynn Chew Hui Fen (16103988)

An explanation of your understanding any distributed system concepts that you have apply in your assignment.

One of the distributed system concepts that we have applied in our application is scalability. A distributed system should be scalable to be able to handle increasing load to the system such as resources or users without compromising the quality and effectiveness of services offered. In doing so, the system should prevent bottlenecks and reduce performance and resources loss while maintaining a reasonable cost.

A description of how you applied the distributed concepts in your assignment.

This concept is applied in our distributed system by using a peer-to-peer architecture among clients instead of a client-server architecture, thus there is no single point of failure among clients. This is implemented using Akka clustered systems where each server and client has a seed node and forms a cluster after connection. All nodes in a cluster can communicate with each other, so there are no fixed connections. Scalability is also implemented in the sense that the server has no limits to the number of clients that want to connect to it but is only limited to the number of ports available for binding.

The problems encountered during this assignment and how you solved these problems

The problems encountered during this assignment include passing data from one application window to another. This is because the main window page requires the user input data from the landing page, however Scala cannot reference methods from other classes. As a solution, a new object is created to store the data needed by the main window as well as the landing page so that the both controllers can reference the data needed via variable referencing.

An evaluation of the strengths and weaknesses of your submitted work.

- Among the strengths of this distributed system are:
- Clients can connect to the server either locally or through the internet.
- Omission handling so clients can close the application at any time and the server and other clients will be updated and notified.
- Low latency so clients can see their opponent's moves in real-time.

Some of the weaknesses of this distributed system are:

- Port configuration has to be done manually in the configuration files.
- Size of the application window is not resizable.
- No omission handling if server terminates before clients terminate. Clients are not notified when the server is terminated.

Include each group's member contribution percentage.

Name:	**Student ID:**	**Contribution Percentage:**
Darryl Tan Zhe Liang	17023326	100%
Cherlynn Chew Hui Fen	16103988	100%
Yohannes Luke Koh	17018342	100%
Len Wei Xin	17039322	100%
Vincent Seaw	17028754	100%

Len Wei Xin (17039322)

An explanation of your understanding any distributed system concepts that you have apply in your assignment.

A cluster is made by a set of nodes joined together. Akka Cluster provides a fault-tolerant decentralized peer-to-peer based Cluster Membership Service with no single point of failure or single point of bottleneck. It does this using gossip protocols and an automatic failure detector. It basically means that the system is able to remove or adding more nodes without stopping the system. This is important as it enables the system to have higher scalability.

A description of how you applied the distributed concepts in your assignment.

The cluster is created in the server where when a node is connected to the server, it automatically added them into the cluster. The cluster stores the unique reference of each node. In tetris, it is act as the game lobby of the server, whenever a user press the join lobby button, a connection will be establish between the user(node) and the reference will be added into the cluster. Which this allows all the nodes in the cluster will be able to see every each other in the server and invite them into a game which it will get the reference ID from the cluster and send direct message to other node.

The problems encountered during this assignment and how you solved these problems.

1. Changing scene in fxml: Due to the nature of actor model, a lot of dependencies of actor reference is needed throughout the whole session which complicates when a scene is needed to change. All of the reference and controller needed will need to update to the primary stage in order to maintain connection to the server. This issue is fixed by initialize empty root scene and update the stage's controller and references into the empty root scene whenever scene changes.

2. Synchronizing between nodes: During the gameplay, there are some features needed to synchronize with players, for example the pause button that will pause both of the player at the same time, animation will need to be transfer real time to the other player and winning and losing announcement. This is solved by creating functions in the game to detect the changes made and notify other node about the changes.

An evaluation of the strengths and weaknesses of your submitted work.

Strengths:

1. This system not only support LAN connection, but it also support internet connection.
2. It is horizontally scalable; it can connect and disconnect a node without interfere other connections.
3. Detailed omission handling is made, so that user will have smooth experience throughout the session.
4. This program allows user to go back to the previous scene which it will not require user to restart the client after every game.

Weakness:

1. Player will not know if the server is on or off.
2. Packet delivery is “at-most once”, hence packet loss will occur.
3. Even though the system can connect through LAN connection and Internet connection, there are some configurations and changes needed to be made manually in the code.
4. Screen size is fixed at 1280 pixels x 720 pixels.

Include each group’s member contribution percentage.

Name:	**Student ID:**	**Contribution Percentage:**
Darryl Tan Zhe Liang	17023326	100%
Cherlynn Chew Hui Fen	16103988	100%
Yohannes Luke Koh	17018342	100%
Len Wei Xin	17039322	100%
Vincent Seaw	17028754	100%