Good Team : P2P Bombermine Game

Faisal Arshad (Matrikel Nr: 2611358)
faisal.arshad@stud.tu-darmstadt.de

Kamran Yaqub (Matrikel Nr: 2408121)
kamran.yaqub@stud.tu-darmstadt.de

Ehsan ul Haq Zaffar (Matrikel Nr: 2731306)
ehsan.zaffar@stud.tu-darmstadt.de

Noorulla Sharief (Matrikel Nr: 2641865)
noorulla.sharief@stud.tu-darmstadt.de

August 31, 2014

1 Introduction

Massive Multiplayer Online Games (MMOGs) are growing due to advances in the generation of attractive content and the availability of high speed and capacity networks. One of the main characteristics of the MMOGs is that they enable users to become members of active communities with common interests, shared adventures and common objectives. Enabling thousands of users to communicate with each other creates large network demands, in terms of required bandwidth and low latency. We can experience all these aspects on Planet Lab where a virtual setup can be accessed on the nodes scattered around the globe with the real network experience.

2 Game

2.1 Motivation

We have developed a Peer to Peer Multiplayer game with self-organizing peers. Our ultimate goal was to scale this game for hundreds of players. We reckon that concurrent self-organization of such huge number of players may not scale.

2.2 Game Logic

We have built a Bombermine game with large GameSpace, but each player will be viewing only a part of the whole space forming his/her Locality of Interest. On the Locality of Interest the player will be able to move while this locality of interest will change as player tries to moves out of current locality of interest. Some rules of the games are as followed.

- Players movement will be restricted by Blocks and Walls.
- Players can place time bombs at any location, these time bombs will destroy any player and walls on explosion within in its range while any block in the bomb's range is indestructible.
- Players cannot move over the Blocks, Walls or any other player

Our Idea is to manage the GameSpace in form of a grid. This grid would be managed by all peers, there are some immutable objects like Blocks, mutable objects like players, walls and bombs. We implemented the following types of events i.e. position change, player to player to interaction and player to object interaction. All these events would change the games state which must be conveyed to all players consistently at any given instance.

3 Game Design

3.1 Architecture

Our P2P game is developed with MVC architecture. The GUI is built with Java Swing and Overlay is built with KaZaA Architecture. For network messaging We have selected Netty Library.

The game GUI is initialized at the start from a configuration file. Configuration file contains the initial game space information which is updated upon joining an existing game. Among other parameters some important parameters worth explaining are isSuperNeighbor, portNo, activeRegion. The very first Peer starting the game should set isSuperNeighbor parameter to true, while other Peers joining this game can connect as Ordinary Peer with isSuperNeighbor parameter set to false.The portNo parameter will set the incoming port of the current Peer. The activeRegion will set the region the Peer will be for the first time but after successful connection with Super Peer, the Peer is redirected to the region of Super Peer and gets the latest game state.

For automated testing the game GUI has two modes GUI mode and Console mode which can be selected by setting parameter consolMode in configuration file. GUI mode gives control to user while Console mode automatically moves the peer according to the type of motion set in the configuration file with wayPoint parameters by setting to one of right, down, point, randomPoint, verticalRegion, horizontalRegion. if it is set to point the parameter srcDesPoints in terms of start and end coordinates (rows and columns) needs to be set also. for tesing purpose Consol mode one instance can run multiple Peers by setting the peersToConnect parameter to ≥ 1.

KaZaA overlay is based on the concept of Super Peers and Ordinary Peers. Ordinary Peers connect to Super Peers with TCP protocol and Super Peers

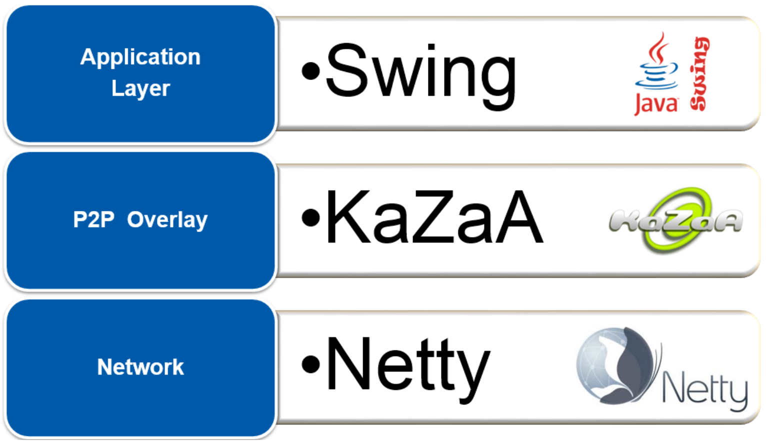


Figure 1: P2P Bombermine game architecture

connect to other Super Peers using TCP protocol. Super Peers in KaZaA overlay track the content of Ordinary Peers.

KaZaA Overlay is built using Netty.io (NIO) framework. NIO framework facilitates a quick and easy development of protocol servers and clients and simplifies network programming (TCP, UDP).

3.2 Overlay

Our peer to peer Bombermine game is based on KaZaA over-lay, where every node represents a player. Similar to KaZaA, the Super Peer acts like a peer in addition to coordinating the region under its responsibility in the gamespace.

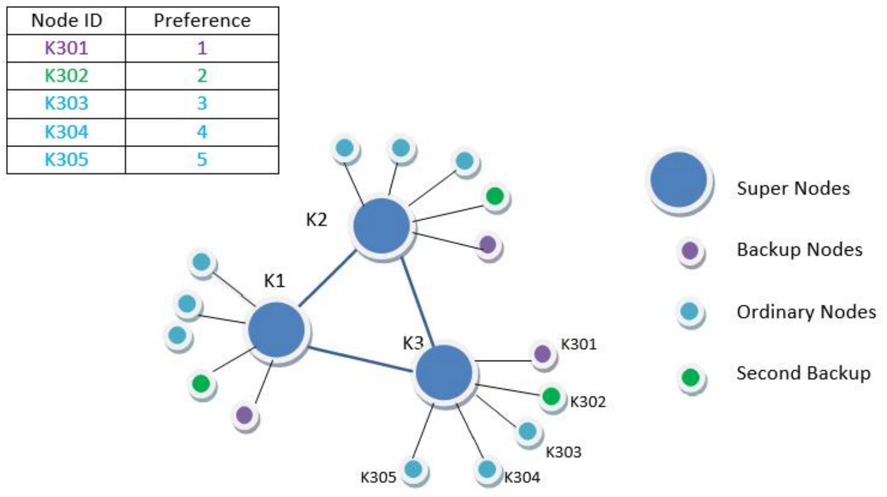
The connection between Super Peers is maintained in form of a mesh and connection of Ordinary Nodes with Super Peer is maintained in star topology. One Ordinary node can only connect to only one Super Peer in the overlay.

We have selected this approach to cater for abrupt disconnection of peers from the gamespace. But on the other hand it provides single point of failure. To handle this situation we are using a Backup Node which acts like a mirror of the Super Peer.

Super Peer is responsible for maintaining routing table for a game region. Super Peer is selected for a region on the basis of first-peer-entering in the region. It waits for more peers to enter in the region, as soon as a new peer enters, it is given a status of Backup Node. All following peers are considered as normal peers. To maintain the structure of the Super Peer and Backup Node the entry time of each node is maintained at the Super Peer.

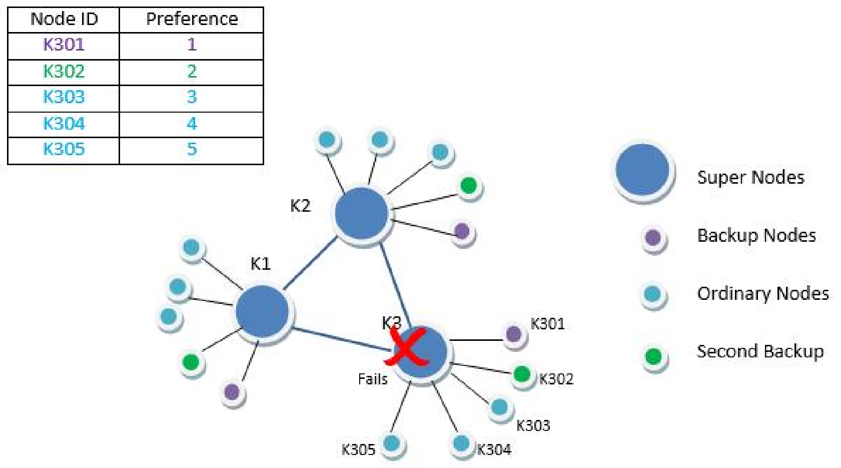
3.3 Protocol

We have selected TCP as our communication protocol. We reckon that TCP has its own overhead of message ACKs but as we have designed our overlay to



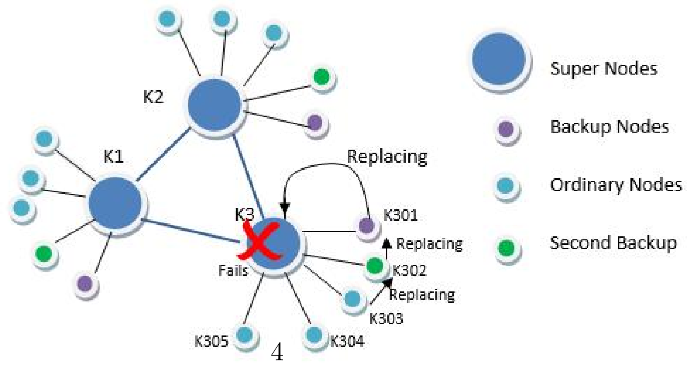
Node ID	Preference
K301	1
K302	2
K303	3
K304	4
K305	5

(a) Overlay maintained in KazaA style, Every node maintaines a queue of Backup Nodes



Node ID	Preference
K301	1
K302	2
K303	3
K304	4
K305	5

(b) Super node may fail, leaving region orphan



(c) Top node in Back up nodes queue takes over as Super Peer

Figure 2: Bombermine Overlay Structure

be dependent on Super Peer, all the messages are being routed via Super Peer. Our assumption with this node is that it will be able to handle the messages passing within region under its control.

3.4 Network Messages

During the development we encountered a high message frequency of player movement messages in ratio to other messages. Player movement messages to move across 1 tile generated about 8 messages. To avoid this we adopted the game to send summary messages but this approach resulted in complicated message handling at GUI and lag in game play. Then we adopted the game to send only one message for a movement over a gamespace tile. This reduces the message frequency by a factor of 8 to the original approach. e.g. If we send a message for every 4 pixels movement of a player, one tile distance covered will send about 8 messages (8x4=32 pixels, the size of a tile).

As we have divided our game space into regions in order to better manage the messaging in the overlay. There are two types messages: Intra-Region and Inter-Region messages.

3.4.1 Intra-Region Messages

- **Player Movement:** This message is used to update player position in gamespace using navigation keys. This message has messageId PU and it updates a player's position (xPlayerPos, yPlayerPos) and region Id.

- **Player Bomb Placement:** This message is used to send the place bomb message to other players with space bar. This message has messageId PB, and it sends a placed bomb position (xBombPos, yBombPos).

- **Player Join:** This message is sent to Super Peer, Super Peer then includes this player in the list of existing players which is then forwarded to other players on next state update. Other Ordinary Peers will add this new node in the queue of Backup Super Peers. This message has messageId JN and playerId, superIp, superPort.

- **Player State Update:** In reply of messageId PJ the Super Peer replies with the current state of the game including existing players already in the game. In addition to this the object state of the new region is also included in this message. With this message the joining peer extracts the list of backup Super Peer form existing Peers. This message has messageId SU.

- **Player Leave:** This message is used for graceful leave of players. We have two different scenarios, first, when an ordinary player leaves, and second, when Super Peer leaves. When ordinary player leaves, it sends player leave message with messageId LV to its respective Super Peer. Super Peers and subsequently Ordinary Peers in gamespace remove the leaving player from their screens and routing tables. When Super Peer leaves, all ordinary nodes are notified with leaving of Super Peer. Now the top entry (Ordinary Peer) from the backup queue connects to other ordinary peers and top entry in back up queue will take over as Super Peer. In case of ungraceful leave of super peer, all the ordinary nodes are notified on network level and top node from the backup queue takes over as Super Peer and all other nodes connect to it.

- **Region Leave:** message Id of this type of message is RL. This message is sent to Ordinary Peers by Super Peer indicating that Super Peer is leaving. This will trigger the top back up Peer to take over as Super Peer of that region and connects to other Super Peers, and other Ordinary Peers will connect to new Super Peer.

3.4.2 Inter-Region Messages

- **Neighbor Update:** This message has messageId NU. It is sent from Super Peers to a new Super Peer of neighboring region. This message contains the information of all the Super Peers in the gamespace.

3.5 Bootstrap

We have made an assumption that any peer joining will already know an existing peer (Ordinary or Super Peer), new peer will contact any Super Peer to join the game.

3.6 Join/Leave

We have handled joining and leaving of the peers by signaling to Super Peers, while joining the region or game, the peers will send Player Join Message to Super Peer of that particular region and it will subsequently enter the region. Super node would also handle the leaving of the player.

3.7 Message Sequence

3.7.1 Ordinary Peer Join

When an Ordinary Peer joins the game it sends PEER_JOIN message to Super Peer (depicted in Figure: 3) it is supposed to know already. Super Peer in return send the STATE_UPDATE to the joining Peer and informs other Ordinary Peers under its region about joining of a new peer by sending PEER_JOIN message.

3.7.2 Player Movement

Whenever any peer changes its position in gamespace, (Figure: 4) it sends "PLAYER _POSITION_UPDATE" to its Super Peer which forwards the message to other Ordinary Peers under its region.

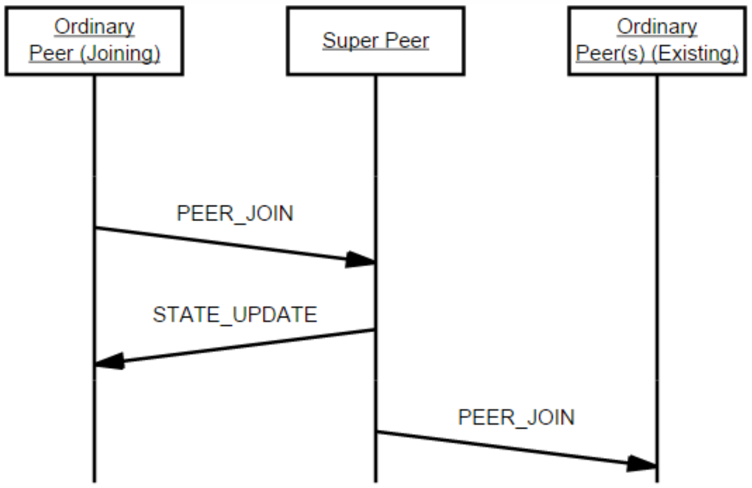


Figure 3: Message Sequence: Ordinary Peer Join

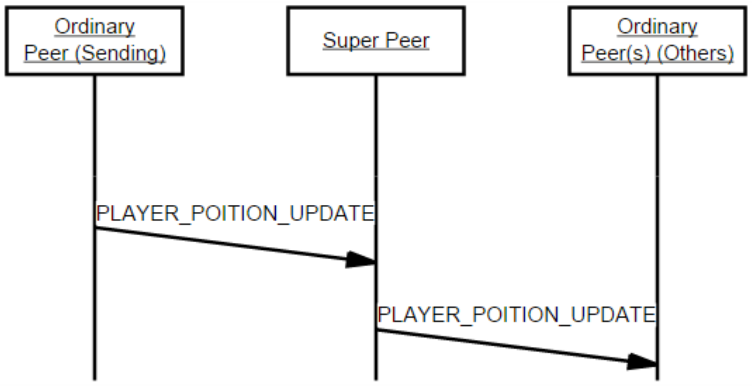


Figure 4: Message Sequence: Player Movement

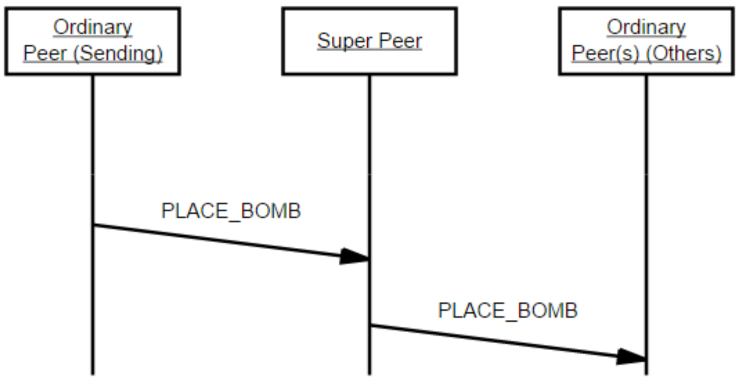


Figure 5: Message Sequence: Bomb Placement

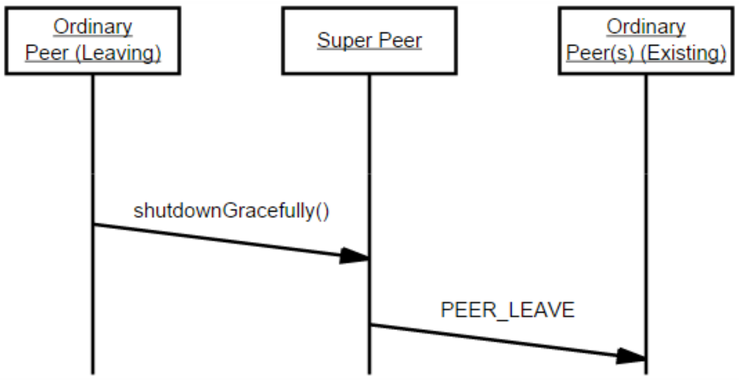


Figure 6: Message Sequence: Graceful Leave - Ordinary Peer

3.7.3 Bomb Placement

Whenever any peer places a bomb in gamespace, it sends ”PLACE_BOMB” to its Super Peer which forwards the message to other Ordinary Peers under its region. This Scenario is shown in Figure: 5.

3.7.4 Graceful Leave - Ordinary Peer

When any Ordinary Peer is leaving the game, it sends shuthdwonGracefully() method call (Figure: 6) at network level to its Super Peer which sends PEER_LEAVE message to other Ordinary Peers under its region.

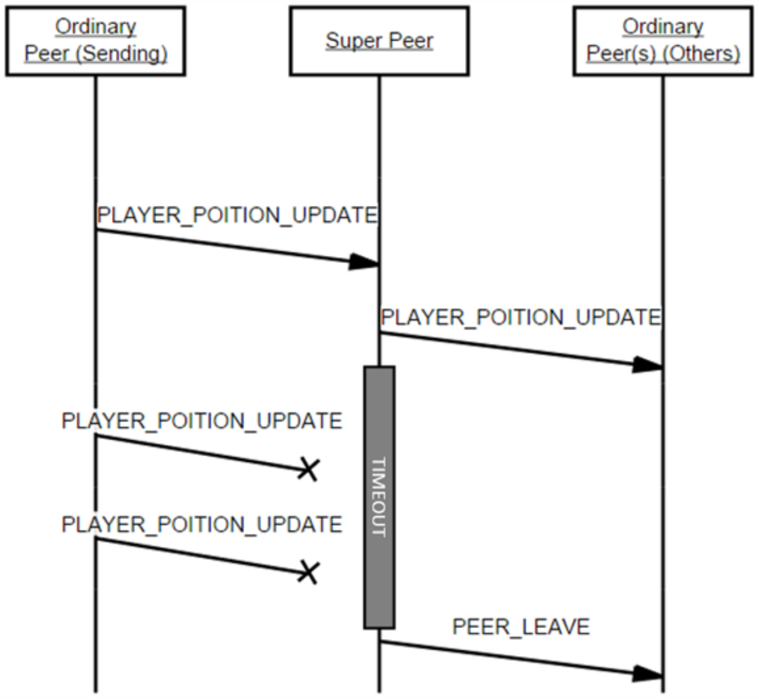


Figure 7: Message Sequence: Ungraceful Leave - Ordinary Peer

3.7.5 Ungraceful Leave - Ordinary Peer

When any Ordinary Peer has left ungracefully, the Super Peer of the region detects it by timeout of its connection and sends "PEER_LEAVE" message to Ordinary Peers under its region indication that a Peer has left the game (Figure 7)

3.7.6 Change Region to an Empty Region - Ordinary Peer

When any Ordinary Peer moves to an empty region, it sends a shuthdwon-Gracefully () method call to its Super Peer and leaves the region and Super Peer sends PEER_LEAVE message to notify other Ordinary Peers in its region that the peer has left. The leaving Ordinary Peer checks its queue of neighboring Super Peers and detects that there is no peer in the region its moving into, it assumes itself as Super Peer of this region and connects to its previous Region's Super Peer as a new Super Peer on its Current Region. The Super Peer of that region informs this to its Ordinary Peers with NEIGHBOR_UPDATE message. (Figure: 8)

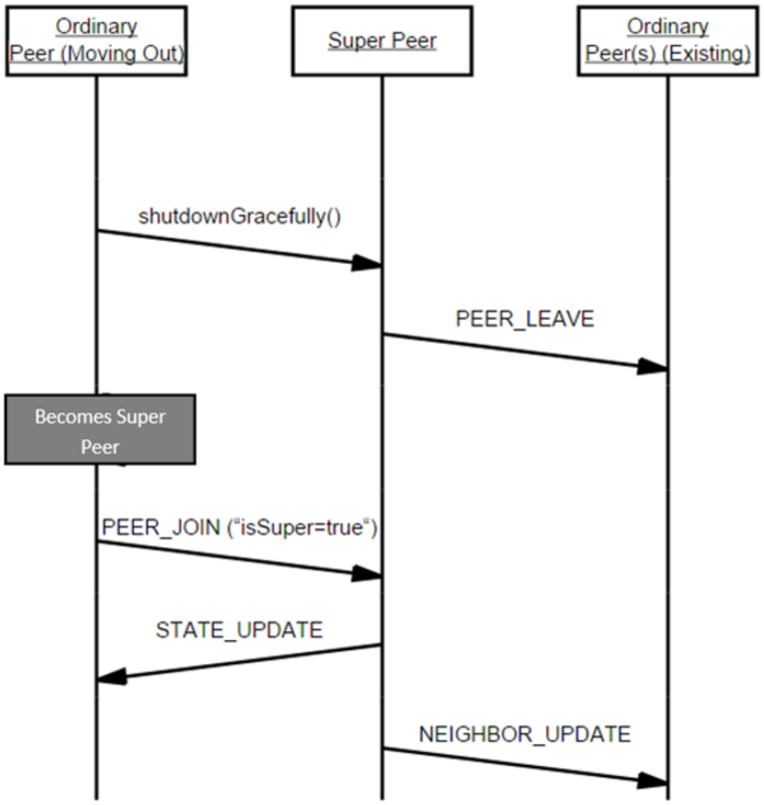


Figure 8: Message Sequence: Change Region to an Empty Region - Ordinary Peer

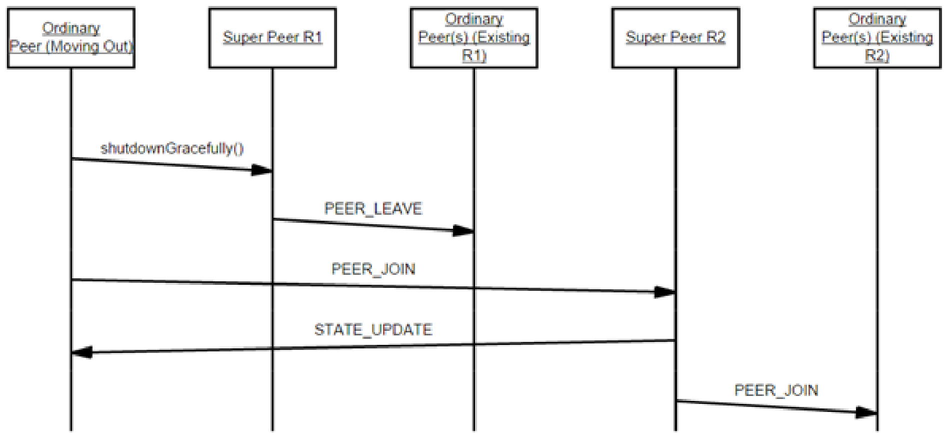

Ordinary Peer (Moving Out)
Super Peer R1
Ordinary Peer(s) (Existing R1)
Super Peer R2
Ordinary Peer(s) (Existing R2)
shutdownGracefully()
PEER_LEAVE
PEER_JOIN
STATE_UPDATE
PEER_JOIN

Figure 9: Message Sequence: Change Region to an Empty Region - Ordinary Peer

3.7.7 Change Region to a Populated Region - Ordinary Peer

When any Ordinary Peer moves an already populated region, it is treated as a graceful leave from previous region followed by a join in new region. As shown in Figure: 9

3.7.8 Change Region to a Populated Region - Super Peer

In a scenario where Super Peer moves out to an already populated region, it sends a REGION_CHANGE message to all connected Ordinary Peers in its current region and all connected Super Peers. The Super Peers of the neighboring regions notify this change to their Ordinary Peers so that they can update their neighboring Super Peers queue. As the Super Peer moves out and Ordinary Peers are notified, they connect to the new Super Peer which is at the top of their backup Super Peer queue. The Super Peer moving out joins the new region as an Ordinary Peer as shown in Figure: 10

3.7.9 Ungraceful Leave - Super Peer

If Super Peer leaves ungracefully, it is detected by other peers at network level and peers join the Backup Super Peer in their queue. The new Super Peer sends PEER_JOIN as Super Peer and gets STATE_UPDATE in return. The other Super Peer(s) notify their Ordinary Peers about change in the neighboring region with a NEIGHBOR_UPDATE message (Figure 11).

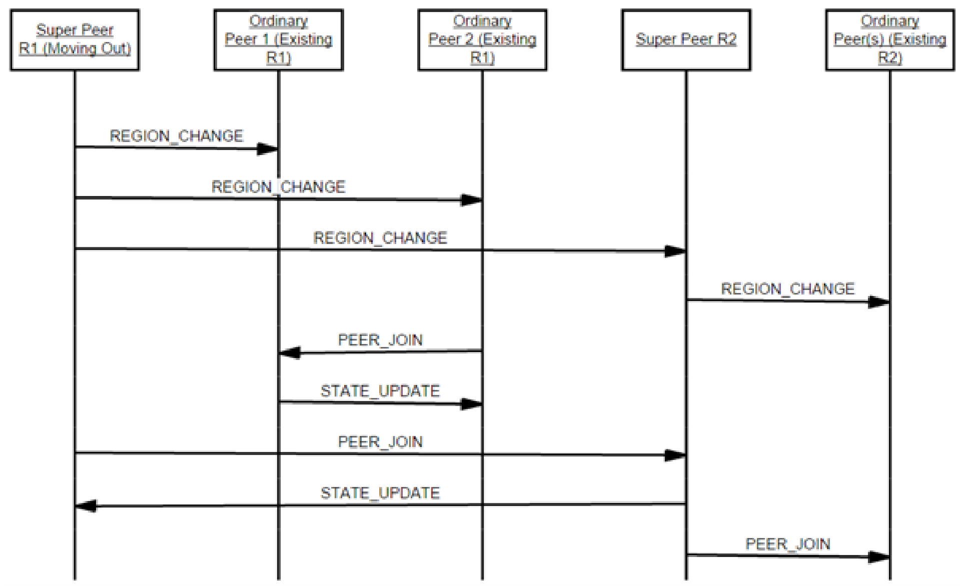


Figure 10: Message Sequence: Change Region to an Empty Region - Ordinary Peer

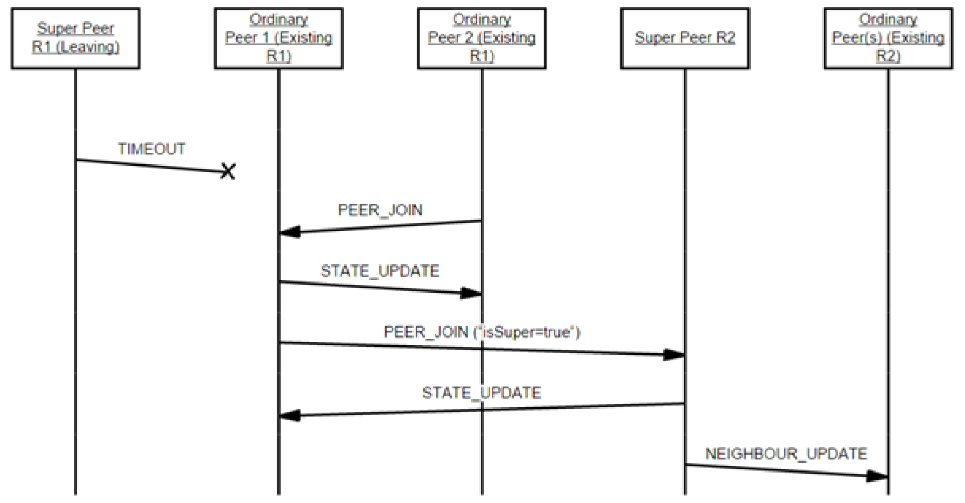

Super Peer R1 (Leaving)
Ordinary Peer 1 (Existing R1)
Ordinary Peer 2 (Existing R1)
Super Peer R2
Ordinary Peer(s) (Existing R2)
TIMEOUT
PEER_JOIN
STATE_UPDATE
PEER_JOIN ("isSuper=true")
STATE_UPDATE
NEIGHBOUR_UPDATE

Figure 11: Message Sequence: Ungraceful Leave - Super Peer

4 Testing

4.1 LAN Testing

We tested our game over LAN with about 100 peers and performance observed was good. No noticeable lag in the game was observed.

4.2 Planet Lab Testing

We tested our game on $\tilde{4}0$ PlanetLab nodes across Europe, Asia and North America. We observed arbitrary behavior of our game from these nodes at different testing instances. We observed that response from these nodes were different at different times of the day. Furthermore we observed a higher round trip time from PlanetLab nodes in North America as compared to nodes in Europe.

5 Limitations and Future Enhancements

- **Region Limitation**
  The stand-alone version of the game was tested on multiple combination of the regions. But we tested our network game on 4 regions (2 rows x 2 columns). This can be extended to multiple regions, preferably to grow exponentially in 2x2, 3x3, 4x4 regions.

- **Regeneration of Gamespace**
  As more focus of the development was on the overlay development we did not implement the regeneration of the game space. If during the game all immutable objects like Bricks are destroyed, they should ideally redrawn based on some criteria.

- **Time Synchronization**
  We plan to synchronize the time between the Super Peer and Ordinary Peers using some well know synchronization algorithm like Berkley algorithm or NTP algorithm.

- **UDP Testing**
  We may later develop and test UDP based messaging after analyzing impact of TCP overhead on actual game. More frequent messages like PLAYER_POSITION_UPDATE and PLACE_BOMB can be sent using UDP.