Networking for multiplayer games in RAGE

Server Instantiation (UDP example):

```java
import java.io.IOException;
import java.net.InetAddress;
import java.util.UUID;

import ray.networking.server.GameConnectionServer;
import ray.networking.server.IClientInfo;

public class GameServerUDP extends GameConnectionServer<UUID>
{
 public GameServerUDP(int localPort) throws IOException
 { super(localPort, ProtocolType.UDP); }
```
- -
```java
 @Override
 public void processPacket(Object o, InetAddress senderIP, int sndPort)
 {
  String message = (String) o;
  String[] msgTokens = message.split(",");

  if(msgTokens.length > 0)
  {
    // case where server receives a JOIN message
    // format:  join,localid
    if(msgTokens[0].compareTo("join") == 0)
    { try
      {  IClientInfo ci;
         ci = getServerSocket().createClientInfo(senderIP, senderPort);
         UUID clientID = UUID.fromString(messageTokens[1]);
         addClient(ci, clientID);
         sendJoinedMessage(clientID, true);
      }
      catch (IOException e)
      { e.printStackTrace();
    } }

    // case where server receives a CREATE message
    // format: create,localid,x,y,z
    if(msgTokens[0].compareTo("create") == 0)
    { UUID clientID = UUID.fromString(messageTokens[1]);
      String[] pos = {msgTokens[2], msgTokens[3], msgTokens[4]};
      sendCreateMessages(clientID, pos);
      sendWantsDetailsMessages(clientID);
    }

    // case where server receives a BYE message
    // format: bye,localid
    if(msgTokens[0].compareTo("bye") == 0)
    { UUID clientID = UUID.fromString(msgTokens[1]);
      sendByeMessages(clientID);
      removeClient(clientID);
    }

    // case where server receives a DETAILS-FOR message
    if(msgTokens[0].compareTo("dsfr") == 0)
    { // etc..... }

    // case where server receives a MOVE message
    if(msgTokens[0].compareTo("move") == 0)
    { // etc..... }
} }
```

- -
```java
public void sendJoinedMessage(UUID clientID, boolean success)
{ // format: join, success  or  join, failure
 try
 { String message = new String("join,");
   if (success)  message += "success";
   else  message += "failure";
   sendPacket(message, clientID);
 }
 catch (IOException e)  { e.printStackTrace(); }
}
```
- -
```java
public void sendCreateMessages(UUID clientID, String[] position)
{ // format:  create, remoteId, x, y, z
 try
 { String message = new String("create," + clientID.toString());
   message += "," + position[0];
   message += "," + position[1];
   message += "," + position[2];
   forwardPacketToAll(message, clientID);
 }
 catch (IOException e)  { e.printStackTrace();
}}
```
- -
```java
public void sndDetailsMsg(UUID clientID, UUID remoteId, String[] position)
{ // etc..... }
```
- -
```java
public void sendWantsDetailsMessages(UUID clientID)
{ // etc..... }
```
- -
```java
public void sendMoveMessages(UUID clientID, String[] position)
{ // etc..... }
```
- -
```java
public void sendByeMessages(UUID clientID)
{ // etc..... }
}
```

Networking driver class:

```java
import java.io.IOException;
import ray.networking.IGameConnection.ProtocolType;

public class NetworkingServer
{
    private GameServerUDP thisUDPServer;
    private GameServerTCP thisTCPServer;

    public NetworkingServer(int serverPort, String protocol)
    { try
      { if(protocol.toUpperCase().compareTo("TCP") == 0)
        { thisTCPServer = new GameServerTCP(serverPort);
        }
        else
        { thisUDPServer = new GameServerUDP(serverPort);
        }
      }
      catch (IOException e)
      { e.printStackTrace();
    } }

    public static void main(String[] args)
    { if(args.length > 1)
      { NetworkingServer app =
            new NetworkingServer(Integer.parseInt(args[0]), args[1]);
} } }
```