UDC 004.4

IMPLEMENTATION OF THE SERVER PART OF THE MOBILE MMO-GAME IN SPACE STYLISTICS

EVGENIY BOBROVICH, KANSTANTSIN RAKHANAU
Polotsk State University, Belarus

The paper formulates the requirements for the server part of the mobile MMO-game in space stylistics. It offers architectural solutions for implementing a network model, storing data, segmenting the system, which meet the requirements for the server part of the project.

Introduction. At present, computer games, especially multiplayer games, are becoming an important part of people's lives. They are played at home, in public transport, and even at work. This popularity is due to the fact that games can be addictive, can help to spend time with pleasure, to communicate with other people in the virtual world and to find new friends. Moreover, some types of computer games can also help in learning, enhancing memory, can improve attention to detail and analytical skills.

In multiplayer games people struggle to defeat real human opponents or cooperate to complete difficult game missions in the virtual universe. Massively multiplayer online (MMO) games have to manage large numbers of players at the same time and synchronize them efficiently in real time.

Multiplayer online games and games in space stylistics games showed their relevance to the example of popular games for personal computers (World of Warcraft, EVE Online, Star Wars: The Old Republic, Stellaris, Galactic Cilivization, Space Rangers) and are in demand at the moment. Therefore, the space MMORPG with the elements of the quest and arcade can gain popularity among users of mobile devices.

The game takes place in a certain galaxy, consisting of many star systems, and those in turn represent a certain number of inhabited/uninhabited planets. The character controlled by the player is the captain of the spaceship, which can be repaired, improved, equipped with equipment and team members for in-game currency. Each player when creating a character must choose one of the three existing races, each has the corresponding characteristics. Inhabited planets give the player the opportunity to take quests, buy/sell goods, upgrade/repair equipment. For performing tasks and participating in battles, players gain experience that is used to improve the characteristics of their characters. The battle between players takes place in turn-based mode, the essence of which is to alternately carry out any specific action, such as the use of weapons, repair droid, etc. The characters of the players interact with each other within the same system, and if necessary move into other systems of the galaxy.

To the main gaming mechanisms that provide the server part, it is necessary to include:

– client connection on the mobile internet or Wi-Fi network, which requires saving of consumed network traffic;
– multi-user work in real time affecting the state of the game world;
– storage of data about players and state of game world;
– registration and authorization of players;
– processing of messages and updating of true state of game world;
– joint or single-player game tasks;
– communication and battles between players;
– chatting;
– upgrading the characteristics of character and ship;
– progress in quests;
– turn-based battles;
– in-game currency and the acquisition of property.

Network interaction model. The protocol UDP (User Datagram Protocol) was chosen to organize the network interaction. UDP allows for continuous synchronization of the game world without delay. This protocol is unreliable and can cause problems when a game needs to send data that is important to all the players in the game. Therefore, there is a need for own implementation of the reliability system with sorting the data by importance [1].

Another important design decision was to utilize a client-server model. In a client-server model clients only communicate with the game server and not between each other, like in a peer-to-peer application [2].

Approach to implementation of network model was based on ideas that are described in articles «The DOOM III Network Architecture», «Quake 3 Source Code Review», «The TRIBES Engine Networking Model». The network model should allow to reduce data traffic, minimize packet loss and maintain the reliability of the UDP protocol.

Layers of networking implementation:

– a Connection Layer that deals with notification and delivery of packets between client and server;

– a Stream Layer which provides packet stream management. This layer employs different stream managers to deal with events, object replication, input move management and others;

– a Simulation Layer which manages all objects in the simulation of game world.

The main components of this network model are illustrated in figure.

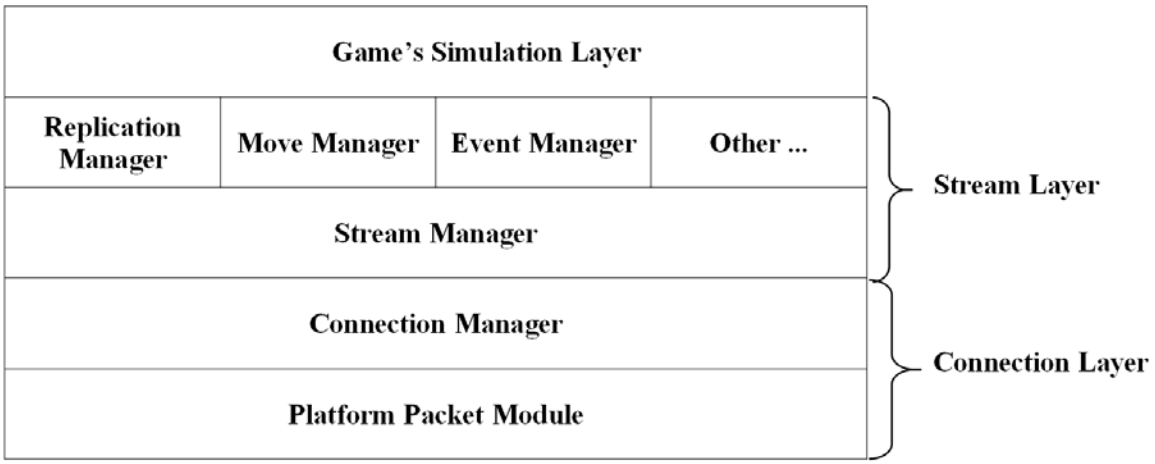

Fig. The main components of the network model

Platform Packet Module – is the lowest layer of the system. This layer is a wrapper for the standard socket, it constructs and sends various packet formats.

Connection Manager – it is to abstract the connection between two computers over the network. It receives data from the layer above it and transmits data to the layer below it, the platform packet module. Connection Manager does guarantee a delivery status notification. In this way, it is possible for the level above the connection manager (Stream Manager) to know whether or not particular data was successfully delivered.

Stream Manager – it is to allocate and transmit packets to the Connection Manager. To control bandwidth, each Stream manager has a packet update rate and size. Since other upper level systems will ask the stream manager to send data, it is also the duty of the stream manager to prioritize these requests. The move, event, and replication managers are given the highest priority when in a bandwidth-bound scenario. Once the stream manager decides on what data to send, the packets are dispatched to the connection manager. In turn, the higher-level managers will be informed by the stream manager regarding the status of delivery.

Replication Manager – it is to storage or replicate of duplicates (ghost) of dynamic objects that are deemed relevant to a particular client. In other words, the server sends information about dynamic objects to the clients, but only the objects that the server thinks the client needs to know about. Replication Manager must give the client the maximum number of relevant objects. It's very important that the ghost manager guarantees that the most recent data is always successfully transmitted to all of the clients. When an object becomes relevant, the ghost manager will assign some information to the object, which is appropriately called a ghost record. For transmission of the ghost records, the objects are prioritized first by status change and then by the priority level, after that Replication Manager determines the objects that should be sent, their data can be added to the outgoing packet.

Move Manager – it is to transmit player movement data as quickly as possible. Quick movement updates can be an important way to reduce the perception of latency on the part of player. The thread manager with the appearance of the movement data must add them to the outgoing packets first place, since this data has the highest priority. Each client is responsible for transmitting their move information to the server, where the simulation of the game world is performed. The client will be sent a confirmation of receipt of information about the movement, and the remaining clients will receive a new state of the world.

Event Manager – it is to maintain a queue of events that are generated by the game's simulation. These events produce functions on connected clients. After performing an action by one player, the server receives information about this event, checks its correctness and performs the appropriate simulation. Event Manager also prioritizes events and tries to keep a record of the maximum number of high priority events in the package.

Other Systems – support systems, which are not so important for understanding the overall architecture, but are involved in the implementation of game concepts.

Game's Simulation Layer – it is to process incoming commands and events, simulate the only true state of the game world. This layer has little to do with the network model. The server simulates the game in discrete time steps called ticks. During each tick, the server processes incoming user commands, runs a physical simulation step, checks the game rules, and updates all object states. After simulating a tick, the server decides if any client needs a world update and takes a snapshot of the current world state if necessary [2].

Data storage. Efficiently store and process data allows an approach that consists in the simultaneous use of two types of DBMS: In-Memory Database for operational work with dynamic data of the game world; Document-Oriented Database to store important, the loss of which would be critical for a player (improvement, money, quests), static and rarely changed data. Dynamic data should also be recorded for important with certain periodicity, and when a client logs out of the game.

Server partitioning. It allows to reduce the load, partially distribute it to servers, and reduce the amount of data that is transferred to each game client. This campaign is implemented by servicing each system of the galaxy by a separate server. When a player moves between systems, his data is replicated between the corresponding servers [1]. The processes of transition between systems, authorization and registration of the game are performed by the master server, which coordinates the operation of the entire system.

Conclusion. It provided a description of the game, on basis of which the main tasks for the implementation of the server part of the multiplayer online game were formulated. To solve the set tasks, architectural solutions were proposed, in particular, the approach to implementing the network interaction of the server and clients was considered. The network model describes ways to maintain the reliability of UDP protocol, reduce amount of data traffic and minimize packet loss.

REFERENCES

1. Глейзер, Дж. Многопользовательские игры. Разработка сетевых приложений/ Глейзер Дж., Мадхав С. — СПб. : Питер, 2017. — 368 с.
2. Source Multiplayer Networking [Electronic resource] / Valve Developer Community. – Mode of access: https://developer.valvesoftware.com/wiki/Source_Multiplayer_Networking. – Date of access: 14.01.2018.
3. LiGDX [Electronic resource] / LibGDX © 2017. – Mode of access: http://www.libgdx.ru/2013/08/goals-features.html. – Date of access: 14.01.2018.
4. Quake 3 Source Code Review [Electronic resource] / FABIEN SANGLARD'S WEBSITE. – Режим доступа: http://fabiensanglard.net/quake3/index.php. – Date of access: 14.01.2018.
5. Frohnmayer, M. The TRIBES Engine Networking Model [Electronic resource] / Mark Frohnmayer, Tim Gift. – Mode of access: http://gamedevs.org/uploads/tribes-networking-model.pdf/. – Date of access: 14.01.2018.