УДК 004.4

АРХИТЕКТУРНЫЕ РЕШЕНИЯ РЕАЛИЗАЦИИ СЕРВЕРНОЙ ЧАСТИ МОБИЛЬНОЙ ММО-ИГРЫ В КОСМИЧЕСКОЙ СТИЛИСТИКЕ

Е.П. БОБРОВИЧ
(***Представлено: канд. техн. наук, доц. А.Ф. ОСЬКИН***)

В статье формулируются требования к серверной части мобильной ММО-игры в космической стилистике. Предлагаются архитектурные решения реализации сетевой модели, организации хранения данных, сегментирования системы, которые удовлетворяют требованиям, предъявляемым к серверной части рассматриваемого проекта.

Введение. В настоящее время компьютерные игры, в особенности многопользовательские, являются важной частью жизни людей. Они играют дома, в общественном транспорте и даже на работе. Такая популярность объясняется тем, что люди могут провести время с удовольствием, общаться в виртуальном мире и находить друзей. Кроме того, некоторые игры способствуют обучению, развитию памяти, улучшению аналитических способностей и повышению внимания к деталям.

В многопользовательских онлайн-играх люди сражаются, чтобы победить своих противников, сотрудничают для достижения общих целей и выполнения сложных игровых миссий. ММО-игры должны справляться с большим количеством игроков и эффективно синхронизировать их в реальном времени.

Многопользовательские онлайн-игры и игры в космической стилистике показали свою востребованность на примере популярных игр под персональные компьютеры (World of Warcraft, EVE Online, Star Wars: The Old Republic, Stellaris, Galactic Cilivization, Космические рейнджеры) и пользуются спросом в настоящее время. Следовательно, космическая MMORPG с элементами квеста и аркады способна получить популярность у пользователей мобильных устройств.

Действие игры происходит в некоторой галактике, состоящей из множества звездных систем, а те в свою очередь представляют собой некоторое количество обитаемых/необитаемых планет. Управляемый игроком персонаж является капитаном космического корабля, который можно за внутриигровую валюту чинить, улучшать, комплектовать оборудованием и членами команды. Каждому игроку при создании персонажа необходимо выбрать одну из трех существующих рас, каждая имеет соответствующие ей характеристики. Обитаемые планеты предоставляют игроку возможность брать квесты, покупать/продавать товары, улучшать/чинить оборудование. За выполнение заданий и участие в сражениях игроки получают опыт, который используется для улучшения характеристик их персонажей. Сражение между игроками проходит в пошаговом режиме, суть которого сводится к поочередному выполнению каких-либо определенных действий, например, использование оружия, ремонтного дроида и т.д. Персонажи игроков взаимодействуют друг с другом в пределах одной системы, а при необходимости перемещаться в другие системы галактики.

К основным игровым механизмам, обеспечивающимися серверной частью, необходимо отнести:

- подключение клиента по мобильному интернету или сети WiFi, что требует экономии потребляемого сетевого трафика;
- многопользовательская работа в реальном времени влияющая на состояние игрового мира;
- хранение данных об игроках и о состоянии игрового мира;
- регистрация и авторизация игроков;
- обработка сообщений и обновление истинного состояния игрового мира;
- совместное либо одиночное прохождение игровых заданий;
- общение и сражения между игроками.
- общение в чате;
- улучшение характеристик персонажа и корабля;
- прогресс выполнения квестов;
- пошаговые сражения;
- внутриигровая валюта и приобретение, обмен имущества.

Сетевая модель взаимодействия. Для организации сетевого взаимодействия в качестве основного протокола был выбран UDP (User Datagram Protocol), который необходим для постоянной синхронизации игрового мира без задержек. Протокол UDP является ненадежным и может вызывать проблемы, когда требуется разослать данные, важные для всех игроков. Поэтому есть необходимость в собственной реализации системы поддержания надежности протокола с сортировкой данных по степени важности [1].

Другим важным архитектурным решением является реализация модели «клиент-сервер». В модели «клиент-сервер» все игроки подключаются к центральному серверу, тогда как в модели «точка-точка» каждый игрок связывается с каждым другим игроком [2].

Подход к реализации сетевой модели был основан на идеях, что описаны в статьях «The DOOM III Network Architecture», «Quake 3 Source Code Review», «The TRIBES Engine Networking Model». Сетевая модель должна позволять сократить траффик данных, минимизировать потерю пакетов и поддерживать надежность протокола UDP.

Сетевые операции можно разбить на несколько слоев:

– слой подключения, который имеет дело с уведомлением и доставкой пакетов между клиентом и сервером;

– слой потока, который обеспечивает управление потоком пакетов. В этом слое задействованы различные диспетчеры для обработки событий, входящих команд и другого;

– слой симуляции игры, который управляет всеми объектами в симуляции игрового мира.

Основные компоненты данной сетевой модели представлены на рисунке.

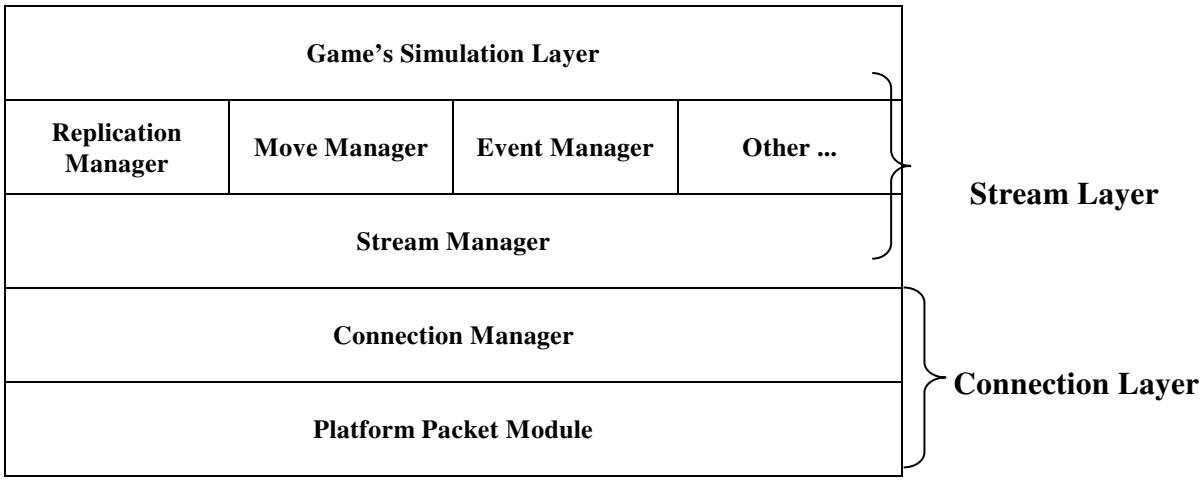

Рисунок. – Основные компоненты сетевой модели

Platform Packet Module — самый нижний уровень системы, который работает с сокетами, осуществляет конструирование и передачу пакетов различных форматов.

Connection Manager — предоставляет абстракцию сетевого соединения двух устройств. Он принимает данные от уровня, лежащего выше и передает их на уровень ниже. Connection Manager гарантирует возврат уведомления о доставке. Благодаря этому Stream Manager знает, были ли доставлены отправленные данные. Пакет, который помечен как потерянный, повторно не отправляется и удаляется.

Stream Manager — выполняет распределение и передачу пакетов в Connection Manager. Для управления пропускной способностью Stream Manager имеет скорость обновления и количество передаваемых пакетов. Все другие системы верхних уровней посылают свои данные через Stream Manager, который ранжирует пакеты по приоритетам. Информация от диспетчеров перемещений, событий и фантомов получает высший приоритет. После того как поток заполнен или все диспетчеры завершили свои действия, пакеты передаются на уровень ниже. В свою очередь диспетчеры более высокого уровня извещаются диспетчером потоков о состоянии доставки.

Replication Manager — хранит дубликаты динамических объектов, которые являются релевантными для конкретного клиента. Следовательно, сервер отсылает клиенту информацию об объектах, о существовании которых клиенту необходимо знать. Replication Manager должен передать клиенту максимальное число релевантных объектов. Очень важно, чтобы диспетчер фантомов гарантировал успешную передачу самых актуальных данных всем клиентам. Когда игровой объект становится релевантным, диспетчер записывает некоторую информацию о его состоянии в объект, который соответственно называется фантомной записью. Перед передачей фантомных записей объекты сначала упорядочиваются по изменениям состояния, а затем по уровню приоритета, после чего Replication Manager определяет, какие объекты должны передаваться, и их данные добавляются в исходящий пакет.

Move Manager — передает информацию о передвижении игроков. Быстрое обновление информации о перемещениях может служить важным способом снижения восприятия задержек игроками. С появлением

данных о перемещениях диспетчер потоков должен добавлять их в исходящие пакеты в первую очередь, так как эти данные имеют наивысший приоритет. Каждый клиент обязан передавать информацию о своих перемещениях на сервер, где выполниться симуляция игрового мира, клиенту будет выслано подтверждение приема сведений о передвижении, а остальные клиенты получат новое состояние мира.

Event Manager — управляет очередью событий, генерируемых уровнем симуляции игры Эти события вызывают функции на подключенных клиентах. После выполнения какого-либо действия одним игроком сервер получает информацию об этом событии, проверяет его корректность и выполняет соответствующую симуляцию. Event Manager так же ранжирует события по приоритетам и пытается вести запись максимального числа высокоприоритетных событий в пакет.

Other Systems — вспомогательные системы, которые не так важны для понимания общей архитектуры, но участвуют в реализации концепций игры.

Game's Simulation Layer — выполняет обработку поступающих команд и событий, осуществляет вычисление единственного верного состояния игрового мира. Данный слой в малой степени имеет отношение к сетевой модели. Сервер симулирует игру в дискретных временных шагах, называемых тиками. Во время каждого тика сервер обрабатывает входящие пользовательские команды, запускает этап физического моделирования, проверяет правила игры и обновляет все состояния объектов. После моделирования тика сервер решает, в обновлении каких объектов игрового мира нуждается каждый клиент, и делает снимок текущего состояния мира [2].

Хранение данных. Эффективно хранить и обрабатывать данные позволяет подход, который заключается в одновременном использовании двух видов СУБД: In-Memory Database для оперативной работы с динамическими данными игрового мира; Dcument-Oriented Database для хранения важных, потеря которых будет для игрока критичной (улучшения, деньги, квесты), статических и редко изменяемых данных. Динамические данные так же должны записываться к важным с определенной переодичностью и при выходе клиента из игры.

Сегментирование серверной части. Снизить нагрузку, частично распределить ее по серверам, а также сократить объем данных, передаваемых каждому клиенту игры позволяет сегментирование серверной среды выполнения. Данный поход реализуется путем обслуживания каждой системы галактики отдельным сервером. При переходе игрока между системами его данные реплицируется между соответствующими серверами [1]. Процесс перехода между системами, авторизация и регистрация игре выполняются мастер-сервером, который координирует работу всей системы.

Заключение. Приведено описание игры, на основе которого сформулированы основные задачи для реализации серверной части многопользовательской онлайн-игры. Для решения поставленных задач предложены архитектурные решения, в частности рассмотрен подход к реализации сетевого взаимодействия сервера и клиентов. В рамках сетевой модели описаны способы поддержания надежности протокола UDP, снижения объема траффика данных и сведения к минимуму потери пакетов.

ЛИТЕРАТУРА

1. Глейзер Дж., Мадхав С. Многопользовательские игры. Разработка сетевых приложений/ Глейзер Дж., Мадхав С. — СПб.: Питер, 2017. — 368 с.
2. Source Multiplayer Networking [Электронный ресурс] / Valve Developer Community – 2018 – Режим доступа: https://developer.valvesoftware.com/wiki/Source_Multiplayer_Networking – Дата доступа: 22.09.2019.
3. The DOOM III Network Architecture [Электронный ресурс] / FABIEN SANGLARD'S WEBSITE – 2012 – Режим доступа: http://fabiensanglard.net/doom3_documentation/The-DOOM-III-Network-Architecture.pdf – Дата доступа: 22.09.2019.
4. Quake 3 Source Code Review [Электронный ресурс] / FABIEN SANGLARD'S WEBSITE – 2012 – Режим доступа: http://fabiensanglard.net/quake3/index.php – Дата доступа: 22.09.2019.
5. The TRIBES Engine Networking Model [Электронный ресурс] / Mark Frohnmayer, Tim Gift – 2009 – Режим доступа: http://gamedevs.org/uploads/tribes-networking-model.pdf – Дата доступа: 22.09.2019.