ОСОБЕННОСТИ АРХИТЕКТУРЫ МНОГОПОЛЬЗОВАТЕЛЬСКИХ ИГР

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Жешко А.А., Близнюк Н.М.
Искра Н.А. - старший преподаватель

Обзор многопользовательских игр, топологий и типичных проблем.

При проектировании многопользовательской игры наибольший акцент ставится на пропускные способности сети, а также на ее топологию.

Для обеспечения высокой пропускной способности предлагается ряд мер как аппаратных, так и программных [1]: минимальное расстояние от сервера до клиента, размер пакета должен стремится к MTU, сжатие информации в пакетах и т.д.

На сегодняшний день, существует несколько топологий сетей в многопользовательских играх:

1. “клиент-сервер”, самая популярная. Обусловлено тем, что только сервер имеет представление о полной картине игры. Это делает невозможным читерство в игре. Код игры делится на клиентский и серверный.
2. “точка-точка”. Каждый клиент одновременно является и сервером с точки зрения геймплея. Существует мастер-клиент для избежания коллизий внутренних id для игроков. Требует повышенной пропускной способности, однако растет она линейно.
3. “точка-точка через ретранслятор”. Проблема топологии “точка-точка” заключается в том, что каждому клиенту сообщается ip-адрес других клиентов. При подключении этим клиентам может возникнуть проблема с технологией NAT.

Для многопользовательских игр очень большую роль играет выбранный сетевой протокол. На данный момент выбор осуществляется из двух: TCP и UDP. TCP обладает огромными возможностями по повторной отправке пакетов, сохранению очередности и т.д. Однако в большинстве случаев для игр с высоким и средним требованием к пропускной способности не все данные нужно передавать с такой точностью (данные о местоположении). UDP же не обладает этими возможностями, но это “чистый лист”, на котором можно программно создать свои алгоритмы повтора, сохранения очередности пакетов и т.д.

Пример архитектуры карточной игры.

В игре используется протокол TCP, т.к. игра пошаговая и нетребовательна к пропускной способности сети.

По аналогии с сетевыми протоколами, удобно разделить сетевой код на слои и дать вышестоящему слою удобное API для работы [3].

Каждый слой отвечает за свое взаимодействие с сетью и игрой:

1. Application - сама игра, с сетью взаимодействует исключительно через методы NetworkManager.
2. Network Manager - предоставляет API для взаимодействия с сервером/клиентом (для каждой игры это API различается). Уведомляет игру о новых пакетах. Формирует высокоуровневые пакеты-классы и передает их HLAPI.
3. HLAPI - принимает высокоуровневые пакеты от Network Manager, формирует очередь высокоуровневых пакетов, трансформирует низкоуровневые пакеты из LLAPI в высокоуровневые пакеты при приеме и высокоуровневые в низкоуровневые при передаче LLAPI. Постоянно опрашивает LLAPI на наличие новых пакетов.
4. LLAPI - работает через сокеты Беркли, формирует поток байтов (низкоуровневые пакеты) и передает их HLAPI. Передает низкоуровневые пакеты по сети.

Для передачи пакетов используется собственная реализация протокола прикладного уровня. Структура пакета состоит следующих компонент [4]:

- стартовые байты (фиксируют начало пакета);
- название пакета (уникальный идентификатор для пакета и его обработки);
- тег сообщения (для формирования программных очередей пакетов);
- данные пакета (информация, которая состоит преимущественно из набора информации простейших типов данных).

Список использованных источников:

1. Глейзер Дж., Мадхав С., Многопользовательские игры. Разработка сетевых приложений. - СПб.: Питер, 2017. - 368 с.
2. Искра Н.А., Близнюк Н.М., Жешко А.А. Подход к обучению объектно-ориентированному проектированию и программированию через реализацию игрового приложения. / Объектные системы. 2016. № 13. С. 25-31.
3. Game programming patterns [Электронный ресурс]. - Режим доступа: https://gameprogrammingpatterns.com/
4. Networking. Gaffer on Games [Электронный ресурс]. - Режим доступа: https://gafferongames.com/tags/networking/