УДК 004.738

АНАЛІЗ МЕРЕЖЕВИХ ПРОТОКОЛІВ ТА МЕРЕЖЕВИХ ТЕХНОЛОГІЙ, ЯКІ ВИКОРИСТОВУЮТЬСЯ В ІГРАХ

Чорненький О.В.
Науковий керівник – к.т.н., доц. Шаповалов С.В.
Харківський національний університет радіоелектроніки, каф. МІРЕС,
м. Харків, Україна
тел. +38(098) 2388430, e-mail: oleksandr.chornenkyi1@nure.ua

Using the main transport protocols: TCP and UDP. TCP has many useful features: reliability, preservation of packet order, error detection. UDP doesn't have all of that, but TCP by its very nature has high latencies that are unacceptable for some games.

Існує два основні транспортні протоколи: TCP та UDP. TCP має безліч корисних особливостей: надійність, збереження порядку пакетів, виявлення помилок. У UDP всього цього немає, зате TCP за своєю природою має підвищені затримки, неприпустимі для деяких ігор. Тобто для забезпечення низьких затримок можна створити власний протокол на основі UDP або використовувати бібліотеку, що реалізує транспортний протокол на UDP і адаптовану для відеоігор. Вибір між TCP, UDP та бібліотекою залежить від кількох факторів. По-перше, від потреби гри: чи потрібні їй низькі затримки? По-друге, від вимог протоколу додатка: чи потрібний йому надійний протокол?

Проведемо аналіз які дані передавати і в якому форматі.

Сервер відправляє не повний, а відфільтрований стан із сутностями, які знаходяться поруч із гравцем. Він робить це з трьох причин. По-перше, повний стан може бути надто великим для передачі з високою частотою. По-друге, клієнтів в основному цікавлять візуальні та аудіодані, тому що більшість ігрової логіки симулюється на сервері гри. По-третє, у деяких іграх гравець не повинен знати певних даних, наприклад, позицію противника на іншому кінці карти, адже в іншому випадку він може зняти пакети і точно знати, куди рухатися, щоб його вбити.

Першим кроком до серіалізації буде перетворення даних, які ми хочемо відправити (введення або ігровий стан) у відповідний для передачі формат.

У багатьох джерелах рекомендується використовувати двійковий формат, який набагато компактніший. Тобто пакети матимуть лише кілька байтів. Тут потрібно враховувати проблему порядку байтів, що на різних комп'ютерах може відрізнятися.

Для серіалізації даних можна використовувати такі бібліотеки:

- FlatBuffers компанії Google
- Cap'n Proto компанії Sandstorm
- cereal Шейна Гранта та Рендольфа Вурхіса

Альтернативним рішенням може бути самостійна реалізація, вона не є особливо складною, особливо якщо в коді ви використовуєте орієнтований на підхід. Крім того, вона дозволить вам виконувати оптимізацію, яка не завжди можлива при використанні бібліотеки.

Кількість даних, що передаються між клієнтами та сервером, обмежена пропускною здатністю каналу. Стиснення даних дозволить передавати в кожному снепшоті більше даних, збільшити частоту оновлення або просто зменшити вимоги до каналу.

Перша техніка – це бітова упаковка. Вона полягає у використанні рівно тієї кількості бітів, яка потрібна для опису потрібної величини. Наприклад, якщо у вас є перелік, який може мати 16 різних значень, то замість цілого байта (8 біт) можна використовувати лише 4 біти.

Дискретизація - це техніка стиснення з втратами, яка полягає у використанні для кодування величини лише підмножини можливих значень. Найпростіше реалізувати дискретизацію округленням чисел із плаваючою комою.

Розглянемо алгоритми стиснення без втрат:

- Кодування Хаффмана з заздалегідь обчисленим кодом, яке є надзвичайно швидким і може давати хороші результати. Воно використовувалося для стиснення пакетів у мережевому двигуні Quake3.
- zlib – алгоритм стиснення загального призначення, який ніколи не збільшує обсяг даних. Він може і стати в нагоді, якщо вам потрібно відправляти клієнтам з сервера асети, довгі тексти або рельєф.
- Копіювання довжин серій — дуже ефективне для певних типів даних, і може використовуватися як етап попередньої обробки перед zlib. Він особливо підходить для стиснення рельєфу, що складається з тайлів або вокселів, у яких повторюється безліч сусідніх елементів.
- Крім того, вам може знадобитися шифрувати передачу інформації між клієнтами та сервером. На це є кілька причин:
- приватність/конфіденційність: повідомлення можуть бути прочитані тільки одержувачем, і жодній іншій особі, яка виконує сніфінг мережі, не вдасться їх прочитати.
- автентифікація: людина, яка бажає виконувати роль гравця, повинна знати її ключ.
- запобігання читерству: зловмисним гравцям буде набагато складніше створювати власні пакети для читерства, їм доведеться відтворювати схему шифрування та знаходити ключ (який змінюється при кожному з'єднанні).

Таким чином можна зробити висновки, що стиск зовсім необов'язковий і рішення про його використання залежить тільки від гри та необхідної пропускної спроможності каналу. Шифрування обов'язково, але в першому прототипі можна обійтися без нього.