УДК 004.75

Бублик В. В., Дученчук В. Б.

МЕТОДИ НЕЙТРАЛІЗАЦІЇ ВПЛИВУ МЕРЕЖЕВОЇ ЛАТЕНТНОСТІ В МУЛЬТИПЛЕЄРНИХ ІГРАХ

У роботі йдеться про дослідження методів нейтралізації впливу мережевої латентності в мультиплеєрних мультимедійних іграх з метою їх використання для вирішення основних проблем затримки синхронізації ігрових станів в умовах мережевих комунікацій. Розглянуто застосування наведених методів для створення ігор Dragon Sim Online та Dog Sim Online.

Ключові слова: мультимедійні ігри, мультиплеєрні ігри, Dragon Sim Online, Dog Sim Online, мережева латентність, мобільні операційні системи, комп'ютерні мережі, методи нейтралізації.

Вступ

Створення мультиплеєрних ігор стикається з цілою низкою проблем, пов'язаних як із самим характером індивідуальних можливостей її учасників, так і з особливостями виконання в умовах розгалужених мереж. У попередній роботі авторів було розглянуто деякі можливості підвищення ефективності застосування мультиплеєрних ігор шляхом організації ігрового досвіду [1]. Тепер зупинимося на основних негативних чинниках у мультиплеєрних іграх, які повинен брати до уваги їхній розробник, а саме: мережевої латентності разом із джитером при передачі даних.

Добре відомо, що латентність, тобто затримка, порівняно з очікуваним, реального часу відгуку на запит, – це одна з суттєвих мережевих проблем. Певний рівень латентності є завжди, мова йде про його зменшення, що для розробників ігор стане джерелом покращення ігрового досвіду її учасників. Мережева латентність є найвагомішою причиною затримок у багатокористувацьких іграх [5].

Наявність латентності призводить до появи джитеру – коливань часу (round-trip time, RTT) приймання-передачі від очікуваного значення. Джитер може викликати ситуацію, за якої пакети приходять у порядку, відмінному від очікуваного [4].

У роботі розв'язується задача зменшення впливу мережевої латентності в мультиплеєрних іграх шляхом застосування методів її нейтралізації та наводяться приклади їх практичного застосування в мультиплеєрних мультимедійних 3D-іграх Dragon Sim Online та Dog Sim Online. До розглянутих у роботі методів належать: стиснення та агрегація пакетів; інтерполяція та прогнозування на стороні клієнта; метод числення координат і прогнозування переміщень клієнта.

Стиснення та агрегація пакетів

Мета стиснення полягає в зменшенні числа бітів, необхідних для подання конкретної інформації. Таким чином, стиснення мережевих пакетів є інтуїтивним підходом до мінімізації мережевого трафіку.

Методи стиснення можуть бути класифіковані відповідно до їхньої здатності зберігати зміст інформації.

Алгоритми стиснення без втрат зберігають усю інформацію, отже, відновлені дані точно такі самі, як і дані перед стисненням. Зазвичай методи стиснення без втрат можуть зменшити розмір даних приблизно до половини початкового розміру.

Для досягнення більш високого ступеня стиснення можуть бути використані методи стиснення з втратами. Ідея полягає в тому, щоб випустити менш потрібну інформацію таким чином, щоб спотворення у відновлених даних стало непомітним. Це широко використовуваний метод, наприклад, у стисненні звуків та зображень [5].

Оскільки ми стискаємо мережеві пакети, також варто зазначити, як різні методи стиснення пов'язані з даними в пакетному форматі.

Внутрішнє стиснення концентрується на інформаційному змісті одного пакету без посилань на інші, раніше передані пакети. Таким чином, цей тип стиснення підходить для випадків, коли застосовується ненадійний протокол передачі даних, як, наприклад, User Datagram Protocol (UDP).

З іншого боку, зовнішнє стиснення може використовувати інформацію, яка вже була передана, отже, можна припустити, що ці дані доступні для отримувачів. Наприклад, ми можемо передавати лише дельти (зміни) або перехідну інформацію, яка потребуватиме менше бітів, ніж абсолютна інформація. Зовнішнє стиснення надає можливість перегляду великої кількості даних

у конкретний момент часу, усуваючи надмірності в інформаційному потоці. Отже, зовнішнє стиснення дозволяє отримати кращі результати компресії, ніж внутрішнє. Проте, оскільки зовнішнє стиснення базується на посиланнях на попередні пакети, воно вимагає використання надійного транспортного протоколу.

Одним із прикладів використання стиснення пакетів у грі Dragon Sim Online є стиснення з втратами при передачі даних про положення та поворот гравців; дані передаються з певною точністю з відкиданням значень після певного розряду. Варто зауважити, що в цьому випадку втрати даних не мають значного впливу на гру, адже гравцям майже неможливо помітити невелику різницю в позиції або повороті.

Агрегація пакетів знижує вимоги до пропускної здатності шляхом об'єднання кількох пакетів і передачі їхнього вмісту в одному великому пакеті. Таким чином, зменшуються накладні витрати на заголовки пакетів [5]. Економія в пропускній здатності залежить від розміру даних вихідних пакетів, розміру заголовків та розміру об'єднаних пакетів.

Є два основні підходи для визначення кількості об'єднаних пакетів: підхід, що базується на тайм-ауті, і підхід на основі кворуму.

У підході з тайм-аутом усі пакети, які були ініційовані до фіксованого періоду часу, об'єднуються. Такий підхід гарантує верхню межу затримки, викликаної агрегацією. Економія пропускної здатності в цьому випадку залежить від швидкості ініціювання пакетів, і в гіршому випадку ніякої економії не буде отримано, якщо жоден або лише один пакет буде ініційований протягом обраного періоду.

У підході, базованому на кворумі, завжди зливається фіксована кількість пакетів. Оскільки передача об'єднаного пакету затримується доти, доки достатня кількість пакетів не буде ініційована, час затримки передачі наперед невизначений. Хоча економія трафіку в цьому випадку передбачувана, тривалі затримки передачі можуть погіршити досвід гри.

Обмеження обох підходів можуть бути компенсовані шляхом їх поєднання. У цьому гібридному підході пакети об'єднуються щоразу, коли одна з умов виконується – або закінчується заданий період часу, або є достатня кількість пакетів для об'єднання [5].

Одним з прикладів агрегації пакетів у Dog Sim Online є передача позицій гравця іншим гравцям. Для більш точної репрезентації переміщення гравця нам необхідно мати більше позицій гравця, між якими ми будемо робити інтерполяцію, але передавати дуже багато пакетів за секунду ми не можемо через здебільшого слабкий мобільний Інтернет, тому вирішенням цієї проблеми може бути пересилка в одному пакеті даних про декілька позицій гравця.

Інтерполяція на стороні клієнта

Розглянемо ще один спосіб зменшення впливу латентності на досвід учасників гри. Рідкі оновлення стану з сервера можуть призвести до «нервозності», гравці відчуватимуть, що гра працює повільніше, ніж насправді. Одним зі способів покращення є інтерполяція на стороні клієнта. При використанні інтерполяції на стороні клієнта клієнт гри не буде одразу автоматично переміщений у нову позицію, отриману від сервера. Замість цього щоразу, коли клієнт отримує нову позицію свого об'єкта, плавна інтерполяція його переміщень до цього стану відбувається протягом певного періоду часу [7], що створює у гравця враження зменшення затримки перед наступним оновленням.

Нехай ПІ – період інтерполяції в мілісекундах, що виражатиме час інтерполювання клієнтом зі старого стану в новий. Нехай ПП – період пакету в мілісекундах, що виражає час очікування сервером між відправкою пакетів. Клієнт завершує інтерполяцію до стану пакету ПІ мілісекунд після прибуття відповідного пакету. Таким чином, якщо ПІ менше, ніж ПП, то клієнт припинить інтерполяцію до того, як прийде новий пакет, тобто гравець може, як і раніше, відчувати нервозність у рухах [3, с. 237].

Щоб переконатися в тому, що стан клієнта змінюється в кожному кадрі гри, а інтерполяція ніколи не зупиняється, час інтерполяції має бути не меншим за час отримання пакету. Таким чином, щоразу, коли клієнт закінчує інтерполяцію цього стану, він матиме наступний стан для інтерполяції і зможе почати процес знову.

Віддалений клієнт завжди відстає на половину RTT від сервера. Важливо розуміти, що якщо стан приходить, а клієнт не відображає його одразу, то віддалений клієнт відстає від сервера ще більше. Ігри, що використовують інтерполяцію стану на стороні клієнта, відстають приблизно на ½ RTT + ПІ від справжнього стану на сервері. Таким чином, щоб зменшити латентність, ПІ повинен бути якомога меншим. Ця вимога в поєднанні з фактом, що ПІ повинен бути більшим або дорівнювати ПП, щоб запобігти нервозності, означає, що ПІ повинен дорівнювати ПП [3, с. 237].

Сервер може або повідомити клієнта, як часто він має намір посилати пакети, або клієнт

може обчислити значення періодичності отримання пакетів емпірично, визначивши, наскільки швидко надходять пакети. Варто зауважити, що сервер повинен встановлювати період відправлення пакету на основі пропускної здатності, а не латентності. Сервер може посилати пакети так часто, як він вважає, що мережа між клієнтом і сервером зможе передати їх.

Якщо сервер відправляє 15 пакетів за секунду, період пакету 66.7 мс. Це означає, що потрібно додати 66.7 мс затримки до ½ RTT, щоб визначити, наскільки відстає віддалений клієнт від стану сервера. Проте все-таки гра виглядатиме набагато гладкішою з інтерполяцією, ніж без неї, і це зможе покращити досвід гри, оскільки гравці менше відчуватимуть на собі цю затримку [3, с. 237].

Інтерполяція на стороні клієнта все ще вважається консервативним алгоритмом. При такому підході може виникнути ситуація, коли стан клієнта в якийсь момент часу за рахунок перебігу інтерполяції не відтворює точно стан сервера на даний момент, залишаючись лише проміжним між двома станами сервера. У такий спосіб клієнт згладжує перехід від одного стану до іншого, але ніколи не робить припущень стосовно актуального стану сервера, а отже, ніколи не опиниться в зовсім некоректному стані.

Інтерполяція на стороні клієнта використовується в іграх Dragon Sim Online та Dog Sim Online для репрезентації плавного руху гравців у грі. Як уже було зазначено, ми не можемо пересилати іншим гравцям дані про позицію гравця кожний фрейм у грі, тому пересилки з даними про позиції гравця робляться декілька разів за секунду (здебільшого 5–10 разів для мобільної гри). Через це виникає необхідність плавного переходу між даними позиціями, для чого використовується інтерполяція на стороні клієнта.

Прогнозування на стороні клієнта

Інтерполяція на стороні клієнта може згладити досвід гри гравця, але такий підхід не наблизить їх до того, що насправді відбувається на сервері. Навіть із невеликим періодом інтерполяції стан віддаленого клієнта, як і раніше, відставатиме хоча б на ½ RTT перед тим, як він його побачить.

Для того щоб показувати актуальний стан гри, вона повинна перейти від інтерполяції до екстраполяції. За допомогою екстраполяції клієнт може взяти трохи відсталий стан, що був ним отриманий, наближено довести його до поточного, а потім застосувати його. Методи, що використовують цей вид екстраполяції, часто називають прогнозуванням на стороні клієнта [2].

Для екстраполяції поточного стану клієнт повинен мати можливість запускати один і той самий код, що працює на сервері. Коли клієнт отримує оновлення стану, він знає, що це оновлення відстає від того, що на сервері, на ½ RTT. Для того щоб зробити цей стан актуальнішим, клієнт просто запускає код симуляції для додаткового ½ RTT.

Для того щоб надати гравцю ближче наближення до справжнього ігрового стану сервера, клієнт застосовуватиме симуляцію кожного кадру. Отримавши наступний стан у пакеті з сервера, клієнт внутрішньо моделює його для ½ RTT, що ідеально відповідатиме стану, розрахованому на основі попередньо прийнятого [2].

Щоб виконати екстраполяцію на ½ RTT, клієнт повинен спочатку приблизно порахувати RTT. Оскільки годинник на сервері і на клієнті не обов'язково в стані синхронізації, наївний підхід, який ґрунтується на відправці з сервера пакету з даними про час, не працюватиме. Замість цього клієнт повинен розрахувати весь RTT і розділити його навпіл.

Клієнт посилає пакет на сервер, що містить позначку часу, ґрунтуючись на власному локальному часі клієнта. Отримавши цей пакет, сервер копіює отриману мітку часу в новий пакет і відправляє його назад клієнту. Коли клієнт отримує цей новий пакет, він віднімає стару мітку часу, що була зроблена, ґрунтуючись на локальному часі цього клієнта, від поточного часу на годиннику клієнта. Це дає точну кількість часу між моментом, коли клієнт послав пакет і коли він отримав відповідь – визначення RTT. За допомогою цієї інформації клієнт знає приблизно, наскільки стан, отриманий у пакетах із сервера, відстає від реального стану на сервері. Ці дані можуть бути використані для екстраполяції стану.

Метод числення координат

Більшість аспектів ігрового моделювання є детермінованими, отже, клієнт може відтворювати їх, просто виконуючи копію коду, який використовується на сервері, та імітуючи об'єкти гри, що керуються цим кодом, тримаючи їх у синхронізації з сервером. Проте є один клас об'єктів, які не повністю детерміновані і їх неможливо ідеально моделювати, – це люди-гравці [1].

Клієнт ніяк не може знати, про що думають віддалені гравці, що вони збираються зробити або куди вони рухатимуться. У цьому випадку кращим рішенням для клієнта буде зробити обґрунтоване припущення, а потім виправляти його за потреби, коли приходитиме оновлення з сервера.

У мережевих іграх метод числення координат (dead reckoning) – це процес прогнозування поведінки об'єкта, ґрунтуючись на припущенні, що він і далі робитиме те, що робить у даний момент часу. Якщо це гравець, це означає, що ми припускаємо, що гравець продовжуватиме рухатися в тому самому напрямку [6].

Коли об'єкт моделювання контролюється гравцем, числення координат вимагає запуску тієї ж імітації, що працює і на сервері. Це означає, що, крім реплікації позиції об'єктів, що контролюються гравцем, сервер також має робити реплікацію всіх змінних, що впливають на моделювання для підрахунку майбутніх позицій. Це включає в себе швидкість, прискорення, стан стрибка тощо.

Поки віддалені гравці продовжують робити саме те, що вони роблять, числення координат дозволяє клієнтам точно передбачати реальний стан світу на сервері. Проте, коли віддалені гравці виконують несподівані дії, моделювання на стороні клієнта відходить від істинного стану і повинно бути виправленим.

З огляду на те, що числення координат робить припущення щодо поведінки на сервері, перш ніж матиме всі факти, числення координат не вважається консервативним алгоритмом. Він відомий як оптимістичний алгоритм.

Коли клієнт виявляє, що його локальна симуляція не точна, виникає три способи виправити ситуацію: миттєве оновлення стану, інтерполяція або зміна параметрів моделювання. Зазвичай ігри використовують комбінацію цих методів [3, с. 241].

Оскільки вороги в іграх Dragon Sim Online та Dog Sim Online повністю детерміновані, то було вирішено відправляти лише відповідні команди до реплікованих об'єктів. Код відповідних команд був спроектований таким чином, що відповідні команди відправляються раз у декілька секунд для того, щоб уникати стану десинхронізації, а також мати можливість вирішити проблему синхронізації на рівні клієнта.

Прогнозування переміщення клієнта

Числення координат не може приховати час очікування для локального гравця. Розглянемо випадок, коли гравець починає бігти вперед. Числення координат використовує стан, отриманий сервером для симулювання, так що з моменту часу, коли гравець натискає відповідну кнопку бігу, необхідно ½ RTT для того, щоб вхідні дані прийшли до сервера, далі потрібно ще ½ RTT для того, щоб повернути необхідні дані клієнту, і тільки тоді клієнт зможе використовувати числення координат [3, с. 243]. Бачимо, що ми все ще маємо затримку перед тим, як гравець натискає кнопку і бачить результат.

Існує краща альтернатива. Гравець передає всі дані на вхід безпосередньо клієнту, таким чином гра на клієнті може просто використати ці дані для моделювання руху. Щойно гравець натискає на кнопку для запуску руху, клієнт одразу може починати моделювати біг. Коли вхідний пакет досягне сервера, він зможе почати моделювання також, оновлюючи стан гравця відповідно [2].

Проблема може виникнути, коли сервер посилає пакет назад клієнту, що містить реплікацію стану гравця. При моделюванні віддалених гравців клієнт може просто використати числення координат і оновлювати стан, припускаючи, що ніяких змін у вхідних даних не було. Зазвичай оновлений вхідний стан відповідатиме стану клієнта, що був передбачений. Якщо ж ні, клієнт може плавно інтерполювати гравця на нове місце. При цьому відбувається повне ігнорування стану сервера локальним гравцем. Але в такому разі може виникнути сильна десинхронізація з сервером, що може призвести до погіршення ігрового досвіду.

Існує краще вирішення цієї проблеми. Коли клієнт отримує стан із сервера, він може використати вхідні дані гравця для ресимуляції будь-яких змін, що були зроблені після того, як сервер порахував вхідний стан.

Висновки

У цій роботі зроблено аналіз методів покращення ігрового досвіду в мультиплеєрних іграх за рахунок зниження ефекту.

Алгоритми стиснення пакетів без втрат зберігають усю інформацію, отже, відновлені дані точно такі самі, як і дані перед стисненням. Зазвичай методи стиснення без втрат можуть зменшити розмір даних приблизно до половини початкового розміру. Для досягнення більш високого ступеня стиснення можуть бути використані методи стиснення з втратами.

Агрегація пакетів знижує вимоги до пропускної здатності шляхом об'єднання кількох пакетів і передачі їхнього вмісту в одному великому пакеті.

Використання інтерполяції на стороні клієнта забезпечує плавність перебігу переміщень протягом певного періоду часу.

Прогнозування на стороні клієнта може згладити досвід гри гравця, але може призвести до

розбіжностей зі станом сервера. Навіть із невеликим періодом інтерполяції стан віддаленого клієнта відставатиме від стану сервера.

Для того щоб показувати актуальніший стан гри, гра повинна перейти від інтерполяції до екстраполяції. Завдяки екстраполяції клієнт може взяти раніше отриманий трохи відсталий стан і довести до приблизно поточного перед тим, як застосувати його.

Проведені первинні випробування мультиплеєрних мультимедійних 3D-ігор Dragon Sim Online та Dog Sim Online продемонстрували дієвість розглянутих у роботі методів нейтралізації впливу мережевої латентності в мультиплеєрних іграх.

Список літератури

1. Бублик В. В. Організація ігрового досвіду при створенні мультиплеєрних 3D-ігор / В. В. Бублик, В. Б. Дученчук // Матеріали міжнародної наукової конференції, м. Київ, 5–9 грудня 2016 р. / редкол.: М. С. Нікітченко [та ін.]. – Кіровоград : Центр оперативної поліграфії «Авангард», 2016. – С. 39–43.
2. Bernier Y. Latency Compensating Methods in Client/Server In-game Protocol Design and Optimization [Electronic resource] / Yahn Bernier. – Mode of access: http://www.vis.uni-stuttgart.de/plain/seminare/computerspiele/latency.pdf. – Title from the screen.
3. Glazer J. Multiplayer Game Programming: Architecting Networked Games / J. Glazer, S. Madhav. – Boston, MA : Addison-Wesley Publ., 2015. – 384 p.
4. Influence of Network Delay and Jitter on Cooperation in Multiplayer Games [Electronic resource] / A. Beznosyk, P. Quax, K. Coninx, W. Lamotte. – Mode of access: https://uhdspace.uhasselt.be/dspace/bitstream/COMPENSATION42/13047/1/delay%20jitter%20(short%20paper).pdf. – Title from the screen.
5. Smed J. A Review on Networking and Multiplayer Computer Games [Electronic resource] / J. Smed, T. Kaukoranta, H. Hakonen. – Turku Centre for Computer Science, 2002. – Mode of access: http://staff.cs.utu.fi/~jounsmed/papers/TR454.pdf. – Title from the screen.
6. Smed J. Aspects of Networking in Multiplayer Computer Games [Electronic resource] / J. Smed, T. Kaukoranta, H. Hakonen. – Mode of access: http://web.cs.wpi.edu/~cs4513/d08/Papers/Smed%20et%20al,%20Networking%20in%20Multiplayer%20Games.pdf. – Title from the screen.
7. Source Multiplayer Networking [Electronic resource] // Valve Developer Community. – Mode of access: https://developer.valvesoftware.com/wiki/Source_Multiplayer_Networking. – Title from the screen.

V. Bublyk, V. Duchenchuk

METHODS OF NEUTRALIZING THE INFLUENCE OF NETWORK LATENCY IN MULTIPLAYER GAMES

Development of multiplayer games faces a number of problems related both to the nature of the individual capabilities of its participants and to their implementation in the conditions of branched networks. This paper focuses on the main negative factors in multiplayer games which should be taken into account by their developers, namely: network latency together with jitter in data transfer. The presence of latency leads to the appearance of jitter-RTT oscillations of the transmit-receive time from the expected value.

It is well known that one of the network problems is latency; that is, the delay, compared with the expected, of theb real-time response to the request. Network latency is the most significant reason for delays in multiplayer games. A certain level of latency is always presented; thus its reduction is still possible. Game developers can use this reduction as a source for improvement of experience for game participants.

The aim of the approaches suggested in the paper is to solve the main problems of delays in the synchronization of game states depending on conditions of network communications. Some neutralization methods of the influence of network latency in multiplayer multimedia games have been presented.

The methods discussed in the work include: compression and aggregation of packets, interpolation and client side prediction, which includes dead reckoning and client move prediction. Compression can minimise the size of packets to be transferred, while aggregation can minimise the network traffic by collecting packets together. The use of interpolation on the client side ensures smoothness of the state updates during a certain period of time by interpolating to them instead of using them immediately. In order to neutralize the influence of the network latency, the client side prediction uses extrapolation instead of interpolation to keep up with the server state. Dead reckoning enables a client to extrapolate a future state from the latest known state. Using client move prediction, a client can instantly simulate the input of a local player.

The application and approbation of the above methods by creating Dragon Sim Online and Dog Sim Online games has been considered.

Keywords: multimedia games, multiplayer games, Dragon Sim Online, Dog Sim Online, network latency, mobile operating systems, computer networks, methods of neutralization.

Матеріал надійшов 06.11.2017