УДК 004.7
DOI: 10.18523/2617-3808.2021.4.44-47

Алєксєєв А. В., Сініцина Р. Б.

КОМПЕНСАЦІЯ ЗАТРИМОК ТА ВТРАТИ ПАКЕТІВ У ДИНАМІЧНИХ ОНЛАЙН-ІГРАХ

У цій статті розглянуто основні причини застосування алгоритмів компенсації затримок в онлайн-іграх та основні аспекти їх реалізації. У роботі подано інформацію щодо впровадження таких алгоритмів і кроків, які необхідно зробити на різних рівнях TCP/IP. Також для наведених алгоритмів показано вузькі місця, певні недоліки та переваги.

Ключові слова: алгоритми, онлайн-ігри, клієнт-сервер, TCP/IP.

Вступ

Ще пару десятків років тому швидкість передавання даних у мережі вимірювали в кілобайтах за секунду, і вже тоді розробники онлайн-ігор мали певні проблеми зі втратою пакетів та затримками у процесі передавання. Нині швидкість передавання в сотні разів більша, а проблема компенсації затримок є ще актуальнішою.

Для багатьох динамічних онлайн-ігор затримка передавання вже у 20 мс може бути доволі помітною та негативно впливати на геймплей і емоції від гри, що може відштовхувати гравців. Проблема посилюється через те, що поряд із необхідністю компенсувати сам час доставлення пакетів, на стороні клієнта є інші немережеві фактори, які непідвладні розробникам і роблять загальну затримку на 5–10 мс більшою. Тому бажання якомога більше та якісніше позбутись мережевих затримок переростає у необхідність, і розробники вимушені шукати оптимальні шляхи розв'язання цієї проблеми.

Ця стаття показує вплив мережевих затримок на онлайн-гру та способи їх компенсації, зокрема наведено теорію щодо протоколів передавання даних у мережі та шляхи вирішення проблем, які постають у процесі розробки алгоритмів.

Принцип роботи онлайн-ігор і причини виникнення затримок

Мережевий код в іграх цілком зав'язаний на «діалозі» із сервером. Є кілька основних моделей такої взаємодії: виділений сервер, listen-сервер та peer-to-peer. У першій моделі є віддалений сервер, до якого приєднуються гравці, у другій – один із гравців бере на себе роль сервера, а в третій – немає центрального сервера, гравці з'єднуються безпосередньо один з одним. І кожна модель передбачає виконання певних розрахунків, на які вже витрачається деякий час, що є першою причиною виникнення затримок. Пакет має бути оброблений перед тим, як його надішлють іншим гравцям.

На цьому проблеми не закінчуються. У сучасному світі широкого застосування набула модель TCP/IP, що абстрагує певні задачі й ділить їх на рівні, кожен з яких підтримується безліччю протоколів, які можна взаємозамінювати, і кожен з яких, своєю чергою, вирішує відповідні завдання та передає дані. На рис. 1 зображено структуру багаторівневої моделі TCP/IP.

Рис. 1. Багаторівнева модель TCP/IP

Причин втрати пакетів може бути багато, і найбільш імовірні – ненадійність канального рівня, мережевого рівня та самого фізичного носія. Проблему з мережевим рівнем можна вирішити шляхом упровадження правильної архітектури та зменшення кількості пакетів зі збільшенням їхніх розмірів. Розглянемо проблеми на кожному рівні детальніше.

Проблема виникнення затримки на фізичному та канальному рівнях

У самому низу моделі TCP/IP розташований найбільш простий підтримувальний рівень – фізичний. Його завдання – забезпечити фізичний зв'язок між вузлами в мережі. І тут уже виникає

обмеження на швидкість передавання даних: незалежно від типу носія, інформація не може поширюватись швидше, ніж швидкість світла. Через це виникає перша причина затримок – *затримка поширення.* Вона становить приблизно 3.301 нс за кожен метр, який проходить пакет у мережі. Це означає, що для того, щоб пакет подолав відстань від Києва до Вашингтона, потрібно 25 мс. Вирішенням проблеми на цьому рівні може бути оптимальне розташування ваших серверів, для listen-сервера – визначити гравця, до якого середня відстань інших гравців найменша, а для peer-to-peer-архітектури – пошук гравців з одного регіону.

Канальний рівень – це те місце, звідки бере початок логічна частина моделі. Його завдання – реалізувати спосіб взаємодії між фізично зв'язаними вузлами. Тобто канальний рівень має надавати метод, за допомогою якого один вузол може упакувати інформацію і передати її на фізичному рівні так, щоб інший вузол міг, прийнявши пакет, відновити з нього передану інформацію. Також доставка кадру до вузла призначення лише можлива, але на цьому рівні вона не гарантована. Існує безліч факторів, що впливають на цілісність цього електричного сигналу. Пошкодження фізичного носія, електромагнітні перешкоди та відмова обладнання легко можуть призвести до втрати кадру. Канальний рівень *не передбачає* застосування будь-яких додаткових зусиль, щоб визначити факт доставлення кадру адресату або повторне його передання, якщо доставлення не було успішним. Однак проблема цього рівня не тільки в цьому. Сімейство протоколів Ethernet, яке є найпопулярнішим на цьому рівні, визначає власний стандарт розміру пакета (MTU), і тут він становить 1500 байт. Через це передавання пакетів якомога більшого розміру має верхній ліміт, і вплив заголовків пакета на затримку на передавання залишається.

На цьому рівні варто знати, скільки пакетів у секунду ви будете передавати, і, відштовхуючись від цього, визначити оптимальний розмір пакета. Але здебільшого принцип один – чим більший розмір пакета, тим краще.

Проблема виникнення затримки на мережевому рівні

Завдання мережевого рівня – забезпечити інфраструктуру для логічної адресації, щоб можна було легко змінювати апаратні компоненти вузла, групувати вузли в окремі підмережі і дозволяти вузлам у віддалених підмережах передавати повідомлення один одному. У наш час для реалізації необхідних функцій мережевого рівня найбільш широко використовують інтернет-протокол версії 4 (IPv4) та інтернет-протокол версії 6 (IPv6).

На цьому рівні можна виокремити такі дві причини виникнення затримок: *затримка на оброблення* та *затримка в черзі.* Перша спричинена тим, що маршрутизатор читає пакети з мережевого інтерфейсу, перевіряє IP-адресу отримувача, визначає наступний хост, куди передати пакет, і виводить пакет у відповідний інтерфейс. І час, потрібний на перевірку та дослідження адреси отримувача та визначення подальшого маршруту, і є причиною *затримки на оброблення.*

Друга затримка спричинена тим, що маршрутизатор має обмежену пропускну здатність. Якщо пакети занадто швидко надходять до мережевого інтерфейсу, вони розміщуються у приймальну чергу, звідки будуть оброблені. І навпаки, якщо занадто багато пакетів необхідно відправити, маршрутизатор розміщує їх у вихідну чергу, адже мережевий інтерфейс може відправляти пакети лише по одному.

Можна вважати, що *затримка на оброблення* є не в компетенції розробника, але насправді в сучасних маршрутизаторах все дуже оптимізовано, тому ця затримка в більшості випадків не перевищує 1 мс.

Затримку в черзі можна зменшити, мінімізувавши затрати на оброблення та передавання. Варто пам'ятати, що ця затримка однакова як для пакета розміром 1500 байт, так і для пакета розміром 100 байт. Через це замість 15 пакетів розміром 100 байт краще відправляти один пакет розміром 1500 байт, тим самим уникаючи накопичення пакетів у черзі.

Основні методи компенсації затримок

Пригальмовування, зумовлене низькою частотою оновлення стану сервером, може змусити гравців думати, що гра відбувається повільніше, ніж насправді. Один із способів виправити ситуацію – використовувати інтерполяцію на стороні клієнта. У такому разі ігрові персонажі не переміщуються так званими ривками з отриманням нових даних із сервера. Замість цього кожного разу, коли клієнт отримує новий стан об'єкта, він плавно інтерполює цей стан протягом деякого інтервалу часу. Це називають локальним фільтром сприйняття (local perception filter).

Період інтерполяції, тобто інтервал часу, протягом якого клієнт буде інтерполювати старий стан у новий, має бути якомога меншим, адже це

впливає на час, на який відстає клієнт від сервера. Також клієнт має знати, з якою частотою сервер надсилає пакети.

Інтерполяцію на стороні клієнта вважають консервативним алгоритмом: незважаючи на те, що іноді цей прийом може допомогти представити стан, який сервер ніколи не передавав явно, він відображає лише проміжні стани між двома точками, які сервер справді моделював. Клієнт згладжує перехід з одного стану в інший, але він ніколи не намагається вгадати, що робить сервер, і тому ніколи не виявиться у неправильному стані.

Іншим рішенням є *прогнозування на стороні клієнта.* Щоб показати стан, максимально наближений до актуального, гра має перемкнутися з інтерполяції на екстраполяцію. За допомогою екстраполяції клієнт може на основі прийнятого старого стану передбачити більш свіжий стан і відобразити його на екрані. Прийоми, які здійснюють таку екстраполяцію, часто називають *прогнозуванням на стороні клієнта*.

Щоб виконати екстраполяцію, потрібно спочатку визначити величину ½ RTT (Round Trip Time). Оскільки час на сервері і клієнті може бути розсинхронізованим, найпростішим буде прийом, коли сервер додасть у пакет позначку часу, а клієнт перевірить її. Однак такий підхід не спрацює. Замість цього клієнт повинен визначити повний період RTT і розділити його навпіл.

Коли клієнт отримає пакет зі станом, він викличе метод для оброблення кроків із позначками часу, які повернув сервер. Виявивши такий крок, він відніме позначку часу зі значення поточного часу, щоб визначити величину RTT, яка стане у пригоді для розрахунку траєкторії руху. Потім викличе RemovedProcessedMoves, щоб видалити кроки з позначками часу, меншими, ніж ця позначка, або які збігаються з нею. Це означає, що після завершення методу для оброблення кроків локальний список кроків на клієнті міститиме тільки ті з них, які ще не досягли сервера і тому повинні бути застосовані до будь-якого стану, отриманого від сервера. Однак використання екстраполяції без інтерполяції може призвести до ситуації, у якій один гравець вистрілить, але через прогнозування інший гравець вже буде дуже далеко від місця, де в нього влучила куля (рис. 2).

Рис. 2. Використання екстраполяції без інтерполяції може призвести до неочікуваних результатів

Використовуючи разом екстраполяцію і інтерполяцію, можна отримати найбільш очікуваний результат – майже повну синхронізацію між гравцями. Але є ще хитрощі, які полегшують життя розробникам.

Зокрема, є можливість використати приховування затримки із застосуванням оптимістичного алгоритму. Майже всі дії у відеоіграх мають деякі візуальні ознаки, які повідомляють, що щось сталося. Спалах повідомляє про постріл, а чарівники махають руками і вимовляють слова, перш ніж виплеснути свою магію. Зазвичай тривалість відтворення цих графічних ефектів ніяк не менше, ніж час передання пакета на сервер і отримання відповіді. Це означає, що гра на клієнті може дати локальному гравцеві миттєвий зворотний зв'язок у відповідь на будь-яке введення та відтворити відповідний анімаційний ефект, чекаючи, поки моделювання буде виконано на сервері.

Висновки

Наведені прийоми для розв'язання проблеми компенсації затримок можна застосовувати при проєктуванні та створенні онлайн-шутерів, стратегій тощо. Завдяки наведеним прийомам можна мінімізувати загальну затримку на передавання пакетів у мережі, завдяки чому гра на клієнті виглядає так, нібито гравець грає в Single Player моді. Однак не варто забувати і про безпеку передання даних, адже, нехтуючи нею, можна отримати ситуацією, у якій гравець зможе маніпулювати даними задля нечесної гри.

Список літератури

1. Гамбетта Г. Мультиплеєр в швидких іграх. Частина III: Поява ворога [Електронний ресурс] / Г. Гамбетта. – Режим доступу: https://habr.com/ru/post/302834/.
2. Глейзер Джошуа. Многопользовательские игры. Разработка сетевых приложений / Джошуа Глейзер, Санджай Мадхав. – Санкт-Петербург : Питер, 2017. – 368 с.
3. Скачков А. О. Методи зменшення мережевих затримок в динамічних онлайн-іграх [Електронний ресурс] / А. О. Скачков. – Режим доступу: https://moluch.ru/archive/216/52143/.

References

Gambetta, Gabriel. *Quickplay Multiplayer* (Part III: The Enemy Appears). Retrieved from https://habr.com/ru/post/302834/.

Glejzer, Dzhoshua, & Madhav, Sandzhaj. (2017). *Mnogopol'zovatel'skie igry. Razrabotka setevyh prilozhenij*. Sankt-Peterburg: Piter [in Russian].

Skachkov, A. O. *Metody zmenshennia merezhevykh zatrymok v dynamichnykh onlain-ihrakh.* Retrieved from https://moluch.ru/archive/216/52143/ [in Ukrainian].

A. Alexeev, R. Sinitsyna

COMPENSATION FOR DELAYS AND LOSSES OF PACKAGES IN DYNAMIC ONLINE GAMES

A couple of decades ago, data rates on the network were measured in kilobytes per second, and even then, online game developers had some problems with the packet loss and transmission delays. Now the transfer rate is hundreds of times higher, and the problem of delay compensation is even more relevant.

For many dynamic online games, a transmission delay of as little as 20 ms can be quite noticeable, negatively affecting the gameplay and emotions of the game, which can repel players.

The problem is exacerbated by the fact that along with the need to compensate for the time of delivery of packets, on the client side there are other non-network factors that are beyond the control of developers, which make the total delay 5-10 ms longer. Because of this, the desire to get rid of network delays as much and as well as possible becomes a necessity, and developers are forced to look for optimal ways to solve this problem.

The problem statement is as follows: to review the causes of delays in online games and possible solutions, as well as the advantages and disadvantages of certain approaches. The problem is considered at the 4 levels of the TCP / IP network model, as well as at the application level. The approaches are given for the most commonly used protocols for each layer, but basic ideas can be easily transferred to other implementations.

The main causes of delays under consideration: propagation delay, router queue delay, transmission delay, and processing delays.

This article shows the impact of network delays on the online games and the ways to compensate for them, along with the theory of data transmission protocols in the network and the ways to solve the problems that arise in the development of algorithms.

Recommendations for solving the compensation problem can be taken into account when designing and launching online shooters, strategies, etc. Thanks to the given receptions it is possible to minimize the general delay on the transfer of packets in a network, thanks to which the game on the client looks as if the player plays in the Single Player mode.

Keywords: algorithms, online games, client-server, TCP/IP.

Матеріал надійшов 15.06.2021