УДК 004.057.4

Причиненко М.А., ст. гр. СП-10-1, Міхайлуца О.М., доцент, науковий керівник

РОЗРОБКА ПРИКЛАДНОГО ПРОТОКОЛУ З ВИКОРИСТАННЯМ ФРЕЙМФОРКУ NETTY 4 ДЛЯ ПЕРЕДАЧІ ДАНИХ В МЕРЕЖАХ TCP/IP В РЕЖИМІ РЕАЛЬНОГО ЧАСУ

Запорізька державна інженерна академія, кафедра ПЗАС

Під час розробки програмних систем часто постає питання про мережну взаємодію між окремими застосунками, що входять до її складу. Це дає можливість обмінюватися командами та повідомленнями, передавати дані, синхронізувати свою роботу.

Для програмної реалізації мережного зв'язку можна використовувати протоколи прикладного рівня стеку TCP/IP. Кожний такий протокол має свої особливості, визначає певні правила та дії необхідні для передачі даних між двома або більше пристроями у мережі [1].

Передача даних у режимі реального часу передбачає створення між застосунками каналу. Після цього застосунок може вільно відправляти дані мережею. Дуже важливим фактором під час передачі інформації є збереження її цілісності. Тому, в якості транспортного протоколу слід обрати TCP, адже він надає можливості підтримки з'єднання, контролю за доставкою даних та гарантує збереження порядку пакетів. Крім того, під час розробки програми в об'єктно-орієнтованому стилі, що є досить популярним на даний момент і використовується, зокрема, у мові програмування Java, дані представляються у вигляді класів. Виходячи з цього, мережею необхідно передавати об'єкти цих класів. Тому, однією з вимог до протоколу передачі даних може стати можливість передавати та отримувати об'єкти різних класів.

Перед прикладним протоколом постають дві важливі задачі: представлення даних у вигляді послідовності байтів для передачі і розділення отриманої послідовності байтів на окремі повідомлення.

Для вирішення проблеми перетворення об'єктів класів у послідовність байтів та у зворотному напрямку призначений механізм серіалізації об'єктів. Серіалізований об'єкт можна представити у текстовому або бінарному вигляді. Для серіалізації, в залежності від вимог, можна використати як стандартні методи мови Java, так і сторонні рішення. Ключовими характеристиками, що можуть впливати на вибір того чи іншого варіанту є, час серіалізації об'єкту, час десеріалізації об'єкту, об'єм серіалізованих даних та незалежність від мови програмування. Зважаючи на те, що для вирішення цієї проблеми існує велика кількість підходів, доцільним є створення абстракції для виконання операцій з серіалізації та десеріалізації об'єктів. Таким чином, протокол не залежить від обраного методу серіалізації даних, і, при необхідності, його можна легко замінити на кращий.

Для коректного розбору вхідного потоку байтів слід описати формат повідомлення. Він визначає поля, що містяться у повідомленні, їхню довжину та призначення, ознаку кінця повідомлення. Для протоколу прикладного рівня, що призначений для передачі між застосунками даних у вигляді об'єктів, достатньою буде структура повідомлення, що зображена на рисунку 1. Об'єкти класів в такому випадку представляються у серіалізованому вигляді, а визначити клас серіалізованого об'єкта можна за допомогою вказаного типу.

Рисунок 1. Структура повідомлення протоколу

Поле ”Тип” має довжину 1 байт. Воно вказує на тип даних, що передається в даному повідомленні. Це дозволяє протоколу підтримувати до 256 різних типів даних. Поле “Довжи-

на повідомлення” має довжину 4 байти. Воно визначає розмір серіалізованого об'єкта у байтах. Це надає можливість точно визначити межі окремих повідомлень у вхідному потоці байтів. Поле “Тіло повідомлення” має довжину вказану в полі “Довжина повідомлення” та містить серіалізований об'єкт. Це останнє поле в повідомленні, всі байти після нього відносяться до наступних повідомлень.

Для реалізації протоколів за допомогою клієнт серверного Java фреймворку Netty 4 треба враховувати його особливості. Для обробки даних цей фреймворк пропонує використовувати архітектуру каналів та фільтрів. Користувач може створювати фільтри, що реалізують інтерфейси ChannelInboundHandler і ChannelOutboundHandler, та додавати їх у чергу фільтрів-обробників каналу SocketChannel[2]. Така архітектура ідеально підходить для реалізації протоколів, адже дозволяє чітко виділити етапи обробки даних.

В даному випадку можна виділити 2 етапи обробки даних: серіалізація об'єктів та формування пакетів. Для кожного з цих етапів необхідно створити енкодер і декодер, та розташувати їх у черзі так, як показано на рисунку 2.

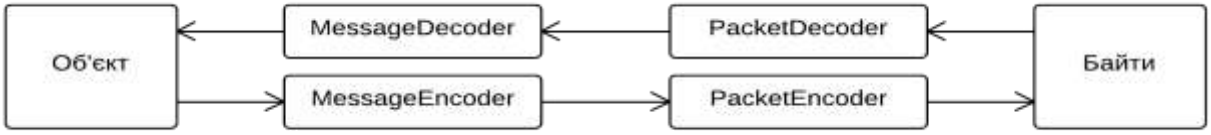

Рисунок 2. Принцип побудови черги фільтрів для обробки інформації.

Фільтр-енкодер повідомлень отримує об'єкт-контейнер повідомлення. Задачею цього фільтра є серіалізація об'єкта у послідовність байтів за допомогою заздалегідь визначеного серіалізатора. В результаті формується об'єкт класу-контейнера пакета, в якому серіалізовані дані передаються на подальшу обробку. Фільтр-енкодер пакетів приймає на вхід сформований на попередньому кроці об'єкт-контейнер пакета та створює з нього повідомлення. Він формує послідовність байтів згідно з визначеним вище форматом, записуючи відповідні поля у наданий фреймворком буфер. Надалі цими даними оперує вже сам Netty. Фільтр-декодер пакетів займається виділенням окремих повідомлень з вхідного потоку байтів. Завдяки чітко визначеному формату повідомлень це дуже просто зробити. Виділене повідомлення пакується в об'єкт класу-контейнеру пакета, що зрештою передається на подальшу обробку. Фільтр-декодер повідомлень отримує від попередника об'єкт-контейнер пакета, що містить серіалізовані дані та інформацію про їх тип. Задачею цього фільтра є десеріалізація даних з послідовності байтів в об'єкт. Для цього застосовується заздалегідь визначений десеріалізатор. В результаті формується об'єкт класу-контейнера повідомлення, що передається на подальшу обробку.

Залишилося лише зареєструвати ці фільтри у відповідному порядку в процесі ініціалізації каналу та створити для зручності основний обробник каналу, що прозоро створює об'єкти-контейнери повідомлення при передачі даних та розкриває їх при отриманні.

Слід також наголосити на важливості тестування коректності роботи проколу, адже помилки в його реалізації можуть призвести до неочікуваної поведінки програм, що його використовують.

Таким чином, за допомогою фреймворку Netty 4 реалізовано протокол прикладного рівня стеку TCP/IP для передачі мережею структурованих даних у вигляді об'єктів мови Java. Для спілкування у режимі реального часу між за стосунками створюється канал на базі транспортного протоколу TCP. Енкодери та декодери за допомогою серіалізатора конвертують об'єкти у послідовність байтів та навпаки, що дозволить легко змінювати алгоритм серіалізації даних у випадку необхідності.

Література

1. Craig Hunt. TCP/IP Network Administration. 3rd Edition. / Craig Hunt. - O'Reilly Media, Inc., 2002. – 746с.
2. User guide for 4.x [Електронний ресурс] / Netty project. - Електрон. текстові дан. і граф. Дан. - Режим доступу: http://netty.io/wiki/user-guide-for-4.x.html, вільний.