# СЕКЦІЯ КОМП'ЮТЕРНИХ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

## Підсекція комп'ютерної та програмної інженерії

УДК 004.7

### МЕТОД КОДУВАННЯ ІНФОРМАЦІЙНИХ СТРУКТУРОВАНИХ ПОВІДОМЛЕНЬ ЗА ДОПОМОГОЮ АДАПТИВНИХ ВЕКТОРІВ СТАНУ

**Андрущенко Р.Б.,** аспірант
Науковий керівник: **Зайцев С.В.,** д.т.н., доцент
*Чернігівський національний технологічний університет*

Структуровані повідомлення широко використовуються в інформаційних системах для передачі даних по мережевим каналам. Інформація, яку необхідно передати по мережевому каналу проходить через декілька етапів трансформації, серед них можуть бути, наприклад: серіалізація, компресія, шифрування, завадостійке кодування. Ці процеси зазвичай розглядаються окремо один від одного, слідуючи принципу єдиної відповідальності. Плюс такого підходу в універсальності та спрощеній процедурі розуміння внутрішніх алгоритмів і, відповідно, пошуку помилок. Платою за такий підхід є додаткові шари абстракції, які зменшують продуктивність програмного забезпечення. В даній роботі розглядається можливість покращити характеристики процесу передачі даних за рахунок прийняття до уваги внутрішньої структури повідомлень.

Алгоритми серіалізації, компресії та завадостійкого кодування розглядають дані, які необхідно передати, лише як набір бітів, байтів, блоків або потоку байтів/блоків, без аналізу їх структури та семантики. З одного боку цей підхід спрощує процес кодування/декодування, з іншого – ігнорується частина інформації, яка б могла бути використана для підвищення характеристик процесу передачі даних: швидкості передачі, пропускної здатності і т.д. [1]

Розглянемо поняття протоколів без станів (stateless-протоколи) та протоколів з підтримкою стану (statefull-протоколи).

Stateless-протокол передачі даних – це такий протокол, у якому жоден учасник протоколу не має ніякої інформації про стан інших учасників. Відправник передає пакет даних та не очікує підтвердження від отримувача. Для кодування/декодування фрагменту даних достатньо інформації, закладеної в алгоритмі кодування/декодування та в самому пакеті передачі даних. Прикладом stateless-протоколів є протоколи HTTP, UDP [2, 3].

Statefull-протокол – це такий протокол, у якому учасники передачі даних можуть бути в курсі стану інших учасників. Інформація про стан учасників може бути отримана з алгоритму роботи протоколу або/та передаватись поряд з іншими даними по мережевому каналу. Інформація про стан може використовуватись для кодування/декодування даних. Прикладом statefull-протоколів є протоколи TCP, Telnet, FTP.

Варто зазначити, що statefull- та stateless-протоколи можна комбінувати в довільному порядку. На базі statefull-протоколу можна побудувати stateless-протокол (реальним прикладом такого стеку є stateless-протокол HTTP, який в якості транспорту використовує statefull-протокол TCP) [3,4].

Statefull-протокол можна перетворити в stateless-протокол за допомогою включення вектору стану $S = [s_1,s_2,\ldots,s_N]$ до вхідних аргументів $X = [x_1,x_2,\ldots,x_N]$ функції кодування/декодування даних:

$$f_{statefull}(t(X), S_1, S_2) = f_{stateless}(t(X + S_1 + S_2)) \quad (1)$$

Тут: $X$ – дані, які необхідно передати, $S_1$ – стан передатчика, $S_2$ – стан приймача, $t(\ldots)$ – функція, яка здійснює відправку/прийом даних, $f_{statefull}(\ldots)$ – функція декодування повідомлення для протоколу зі станом, $= f_{stateless}$ – функція декодування повідомлення для протоколу без стану.

Об'єм інформації, що передається по мережевому каналу, очевидно, в даному випадку збільшується. Справедливе й зворотне твердження: якщо з даних, які передаються по мережевому каналу, можна виділити вектор стану, то об'єм передаваємої інформації теоретично можна зменшити. В наш час розробляються та покращуються як stateless-протоколи, так і statefull-протоколи [4].

Приймемо, що час, необхідний на кодування/декодування повідомлення ($t_к$) значно більший за час передачі повідомлення між хостами ($t_п$). Тобто:

$$t_к >> t_п \quad (2)$$

В більшості випадків так і є (за винятком, якщо генерація повідомлення підпорядковується складному алгоритму), оскільки обчислювальні можливості сучасних процесорів значно перевищують можливості каналів передачі даних навіть всередині обчислювальних пристроїв.

Структуровані повідомлення в своїй структурі мають неділимі частини (наприклад, поля, пари ключ-значення) деякого фіксованого або динамічного типу, які часто називають токенами. Токени мають деякий алфавіт значень, які вони можуть приймати. Тому, як мінімум, в кожному повідомлені можна проаналізувати та виділити:

1) Фактично використовувану множину значень $V = \{v_1, v_2, ..., v_N\}$, яка є підмножиною всіх можливих значень $U$ даного токену: $V \subset U$

2) Ідентифікатор токену $i \in I$

Тоді, повідомлення можна представити у вигляді $m = \{ i_1, \{v_{11}…v_{n1}\}, …, i_k, \{v_{k1} … v_{kn}\} \}$, і його можна розбити на схему $s$ та дані $d$.

$$s = \{ i1, b(V1), i2, b(V2), …, in, b(Vn) \} \quad (3)$$

$$d = \{ f(\{v11…vn1\}), f(v12…vn2), …, f(v1k…vnk) \} \quad (4)$$

Тут: $b(…)$ – функція знаходження границь (bounds) токену, $f(…)$ – функція декодування токену.

Розбивши повідомлення на 2 частини способом, вказаним вище, можна надіслати при першому з'єднанні другій стороні схему s, а при всіх наступних – відправляти лише дані $d$. Графічно можна зобразити так:


*Рис. 1. Розбивка повідомлення m на вектор стану (схему s) та дані (d), де різним кольором показано різні токени*

Як бачимо, при цьому з'являється буферна зона, яку можна або виключити з повідомлення, або використати для інших цілей, наприклад, для збереження надлишкової інформації, яка додається при завадостійкому кодуванні.

Тим не менш, у даного методу є деякі недоліки:

1) Підвищується обчислювальна складність алгоритму кодування/декодування. Даний недолік можна виправити за допомогою, наприклад, динамічного завантаження класів даних (для таких мов як Java) або за допомогою перекомпіляції та динамічного лінкування об'єктних файлів, які містять опис структур даних.

2) З'являється необхідність зберігання векторів стану в пам'яті та необхідність синхронізації векторів.

Оскільки при передачі даних з'являється необхідність коригування схеми та синхронізація вектору стану, то ефективність методу не є сталою в часі та залежить від того, наскільки постійною є структура даних. На графіку нижче показано, як змінюється у часі розмір закодованого повідомлення.

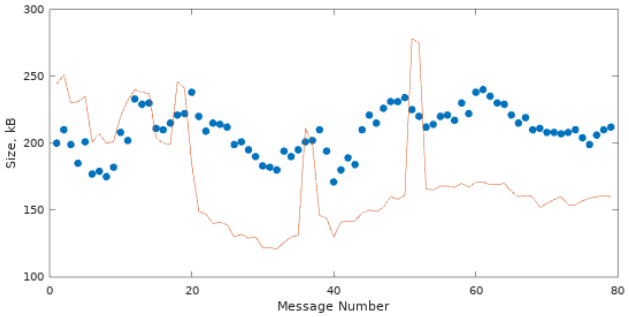


*Рис. 2. Моделювання на базі повідомлень у форматі JSON. Лінією показано фактичний розмір даних після розбивки повідомлення на вектор стану s та дані d*

З рисунку видно, що поки набирається статистика для структури повідомлення, загальний розмір даних після розбивки більший, аніж вихідне повідомлення. Однак після того, як схема даних була успішно виділена, отримуємо економію трафіку. Піки відповідають моментам корекції вектору стану у разі знаходження невідповідності між вектором та токенами у повідомленні, що призводить до коригування вектору стану та його передачі поряд з повідомленням. Загалом, при постійному навантаженні піки з часом будуть зустрічатись все рідше, а це означає, що чим більше часу виконується програмне забезпечення та чим рідше вносяться зміни до структури даних повідомлення, тим краще працює розбивка повідомлень.

**Список використаних джерел**

1. Фред Иди. Сетевой и межсетевой обмен данными с микроконтроллерами / Додэка XXI (перевод О. Тихонова). – с. 378. – 2007.

2. Philip Koopman. 32-Bit Cyclic Redundancy Codes for Internet Applications / The International Conference on Dependable Systems and Networks (DSN). – P. 459–468. – July 2002.

3. Olivier Bonaventure. Computer networking: Principles, Protocols and Practice / Saylor Foundation. – P. 268. – 2011

4. Lydia Parziale. TCP/IP Tutorial and Technical Overview / David T. Britt, Chuck Davis, Jason Forrester, Wei Liu, Carolyn Matthews, Nicolas Rosselot / IBM Redbooks. – P. 1004. – 2006.