COMPUTER SYSTEMS AND NETWORKS

Vol. 4, No. 1, 2022

DOI: https://doi.org/10.23939/csn2022.01.147

УДК 004.45

ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ МЕРЕЖНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ПРОТОКОЛУ SCTP

З.-А. С. Фещенко, І. Ю. Юрчак

Національний університет "Львівська політехніка",
кафедра електронних обчислювальних машин
E-mail: zakharii-andrii.feshchenko.mkisp.2021@lpnu.ua, Iryna.Y.Yurchak@lpnu.ua

У роботі наведено програмну систему для демонстрації роботи транспортного протоколу SCTP у порівнянні з його більш відомими аналогами – TCP та UDP. Для відтворення роботи вибраного протоколу створено API-сокет, який описує роботу SCTP, спираючись на стандарт RFC 6458.

Проведено аналіз транспортних протоколів TCP та UDP, що широко використовувалися до протоколу SCTP, їхні сильні та слабкі сторони. Розглядається, що вдалося покращити в протоколі SCTP.

Описано програмну модель, яка розподілена на клієнтську частину та серверну, яка була створена для демонстрації роботи транспортного протоколу SCTP. Клієнтська частина полягає в надсиланні повідомлення на сервер, який буде зчитувати дане повідомлення і скидати його. Розроблено можливість надсилання використовуючи безпосередньо протокол SCTP, або інкапсуляцію UDP, таким чином інкапсулюючи пакет SCTP у датаграму UDP. Наведено ефективність та доцільність використання програмної моделі та розглянуто альтернативні програмні моделі, які реалізовуються за допомогою транспортного протоколу SCTP. Обґрунтовано засоби, які використано для реалізації даного рішення, а також платформи та операційні системи, на яких це рішення можна відтворити, окрім Windows.

Ключові слова: транспортний протокол, SCTP, TCP, API-сокет, клієнт-серверна система.

Вступ

Загалом, під транспортним рівень розуміється 4-й рівень моделі OSI, призначений для доставки даних без помилок, втрат і дублювання в тій послідовності, в якій вони були передані. При цьому неважливо, які дані передаються, звідки й куди, тобто він надає сам механізм передачі. Блоки даних транспортний рівень розділяє на фрагменти, розмір яких залежить від протоколу, короткі поєднує в один, довгі розбиває. Протокол транспортного рівня може не використовувати усіх вищевказаних можливостей. Наприклад, протокол UDP не гарантує послідовність, відсутність втрат та дублювання, але завдяки відсутності цих можливостей не має так званого« «службового» трафіку [1, 2, 3, 4].

Після одержання повідомлення з прикладного рівня транспортний рівень розбиває його для успішної передачі на достатньо дрібні частини, призначає їм послідовні номери й пересилає їх. Взаємодія на рівні заголовка транспортного рівня зводиться до обговорення того, який пакет був надісланий, який прийнятий, скільки місця є в адресата для прийому подальших повідомлень, що варто послати повторно тощо [5, 6].

Існує невелика кількість готових програмних засобів, які відтворюють роботу транспортного протоколу SCTP чи хоча б пояснюють його зв'язок з аналогічними протоколами, окрім напевно використання даного транспортного протоколу в стеку протоколів, які використовуються при написанні та роботі операційних систем [9, 10]. Наукова-технічна новизна даного проєкту полягає

в детальнішому вивченню протоколу SCTP, дослідженню більшості його областей його застосування і будови, а також пошуку його слабких місць, які можуть бути покращені, у порівнянні зі стандартом RFC 6458 [5, 7, 8].

1. Аналіз застосування протоколу SCTP в програмних моделях

Якщо брати загальне використання протоколу SCTP, то в більшості випадків його яскраву імплементацію можна побачити в операційних системах, зокрема в: FreeBSD, Mac OS X, Microsoft Windows та Linux. Але можна побачити його використання у наступних застосунках.

NetFlow – це функція, яка була представлена на маршрутизаторах Cisco приблизно в 1996 році, і забезпечує можливість збору IP-мережевого трафіку під час входу або виходу з інтерфейсу. Аналізуючи дані, надані NetFlow, адміністратор мережі може визначити такі речі, як джерело та призначення трафіку, клас обслуговування та причини перевантаження.

З міркувань ефективності маршрутизатор традиційно не відстежує вже експортовані записи потоку, тому, якщо пакет NetFlow скидається через перевантаження мережі або пошкодження пакета, усі записи, що містяться, втрачаються назавжди. Протокол UDP не інформує маршрутизатор про втрату, тому він може знову надіслати пакети. Це може бути справжньою проблемою, особливо з NetFlow v8 або v9, які можуть об'єднувати багато пакетів або потоків в один запис. Одна втрата пакета UDP може сильно вплинути на статистику деяких потоків.

Ось чому деякі сучасні реалізації NetFlow використовують протокол передачі керування потоком (SCTP) для експорту пакетів, щоб забезпечити певний захист від втрати пакетів і переконатися, що шаблони NetFlow v9 отримані перед експортом будь-якого пов'язаного запису. Можна зауважити, що TCP не підходить для NetFlow, оскільки суворе впорядкування пакетів призведе до надмірної буферизації та затримок.

WebRTC (Web Real-Time Communication) – це безкоштовний проєкт із відкритим вихідним кодом, який надає веббраузерам і мобільним програмам зв'язок у реальному часі (RTC) через інтерфейс програмування додатків (API). Це дозволяє аудіо- та відеозв'язку працювати всередині вебсторінок, дозволяючи пряме однорангове спілкування, усуваючи потребу встановлювати плагіни або завантажувати рідні програми.

В цьому застосунку роль протоколу SCTP полягає у транспортуванні даних використовуючи DTLS (Datagram Transport Layer Security – протокол зв'язку, який забезпечує безпеку додатків на основі датаграм, які виключають можливість бути підслуханим, втрати чи підробці переданого повідомлення).

2. Постановка задачі

Метою статті є опис побудови програмної моделі, що надає можливість відтворити роботу транспортного протоколу SCTP у якості API-сокета з прив'язкою до стандарту RFC 6458. Описана програмна модель розподілена на клієнтську та серверну частини. Дане рішення повинно працювати не лише для операційної системи Windows, але і для операційних систем типу Linux, а також повинно бути досягнуто покращення швидкості надсилання повідомлення з клієнта і, відповідно, отримання надісланого повідомлення сервером.

3. Алгоритм роботи програмної моделі

Основною вимогою до програмної моделі є покращення швидкодії роботи транспортного протоколу SCTP, а саме зменшення можливого часу, який потрібний для відправки повідомлення клієнтом і його прийом сервером, а також демонстрація даних покращень шляхом показу часу відправки повідомлення з клієнту на сервер спершу до покращень, а пізніше – після. Ще однією особливістю, до якої треба прагнути, щоб розроблювальний застосунок працював швидко і оптимально та не навантажував ресурси комп'ютера.

Основна ідея створення API сокету, на базі стандарту RFC 6458, який окрім уже існуючих (спільних) функцій та викликів, буде також містити і нові виклики, відмінні від RFC 6458. RFC

(Request for Comments) – документ із серії пронумерованих інформаційних документів Інтернету, що містить технічні специфікації та стандарти, має широке застосування у всесвітній мережі.

Створюючи програмну систему, необхідно було детально дослідити технології, які можна використовувати при їх розробці. Ознайомитись із найпопулярнішими мовами програмування, бібліотеками та інструментами для підвищення ефективності мережного програмного забезпечення протоколу SCTP. На основі їх аналізу обрати найоптимальніші засоби розроблення.

На основі проведеного аналізу і досліджень, для створення API сокету, який буде покращувати роботу протоколу SCTP обрано наступний інструментарій:

- Інтегроване середовище розробки Microsoft Visual Studio 2022.
- Мову програмування C, яка найкраще підходить для написання подібного роду речей.
- Мову розмітки даних TeX.
- Крос-платформний відкритий генератор складання сценаріїв CMake, для подальшої генерації сценаріїв.
- Крос-платформну консольну утиліту, що автоматизує процес складання програмного забезпечення з вихідного коду Meson.

4. Реалізація методів надсилання і отримування повідомлення через SCTP

Для вирішення задачі підвищення ефективності мережного програмного забезпечення протоколу SCTP використано мову програмування C. C – мова програмування загального призначення, надзвичайно популярна, проста та гнучка у використанні. Це структурована мова програмування, яка не залежить від машини та широко використовується для написання різних програм, операційних систем тощо.

Використовуючи мову програмування C, було написано дві функції, які є головними, якщо хочемо протестувати транспортний протокол SCTP: функція usrsctp_sendv() і usrsctp_recvv().

Нижче подано функцію usrsctp_sendv(), а також роз'яснення щодо наданих їй аргументів.

```
ssize_t
usrsctp_sendv(struct socket *so,
              const void *data,
              size_t len,
              struct sockaddr *addrs,
              int addrcnt,
              void *info,
              socklen_t infolen,
              unsigned int infotype,
              int flags)
```

Загалом, ця функція приймає 9 аргументів на вхід:

– so: сокет для надсилання даних;

– data: оскільки зручніше надсилати дані в буфері, а не в структурі даних struct iovec, вирішено передавати дані як вказівник void;

– len: довжина даних;

– addrs: підтримується щонайбільше одна адреса призначення. У випадку підключеного сокета параметр addrs може бути встановлений на NULL;

– addrcnt: кількість адрес. Оскільки підтримується щонайбільше одна адреса, addrcnt дорівнює 0, якщо addrs має значення NULL, і 1 в іншому випадку;

– info: додаткова інформація для повідомлення зберігається в void *info. Типи даних struct sctp_sndinfo, struct sctp_prinfo та struct sctp_sendv_spa підтримуються, як визначено в RFC 6458;

– infolen: довжина інформації в байтах;

– infotype: визначає тип інформації, наданої в info.

Можливі значення:

- SCTP_SENDV_NOINFO
- SCTP_SENDV_SNDINFO
- SCTP_SENDV_PRINFO
- SCTP_SENDV_SPA;

– flags: прапори, як описано в RFC 6458.

– usrsctp_sendv() повертає кількість надісланих байтів або -1, якщо сталася помилка. Потім змінна errno встановлюється належним чином. -1 може відбутися лише у випадку, якщо використовуватиметься struct sctp_authinfo. Тому, з метою запобігти цьому, значення параметру erno було встановлено на EINVAL і повернеться -1, якщо буде використане.

Далі показано функцію usrsctp_recvv(), яка буде приймати і обробляти дані, а також зроблено акцент на її аргументах.

```
ssize_t
usrsctp_recvv(struct socket *so,
              void *dbuf,
              size_t len,
              struct sockaddr *from,
              socklen_t * fromlen,
              void *info,
              socklen_t *infolen,
              unsigned int *infotype,
              int *msg_flags)
```

Аргументи функції:

– so: сокет для отримання даних;

– dbuf: аналогічно usrsctp_sendv() дані повертаються в буфер;

– len: довжина буферу в байтах;

– from: вказівник на адресу, яка буде заповнена адресою відправника отриманого повідомлення;

– fromlen: вхідний/вихідний параметр, що описує довжину fromlen;

– info: покажчик на буфер для зберігання атрибутів отриманого повідомлення. Тип структури інформації визначається параметром infotype. Атрибути, що повертаються в info, повинні оброблятися так само, як зазначено в RFC 6458;

– infolen: вхідний/вихідний параметр, що описує розмір інформаційного буфера;

– infotype: після повернення *infotype встановлюється на тип інформаційного буфера. Поточними визначеними значеннями є:

- SCTP_RECVV_NOINFO
- SCTP_RECVV_RCVINFO
- SCTP_RECVV_NXTINFO
- SCTP_RECVV_RN (Детальний опис наведено в RFC 6458)

– flags: вказівник на ціле число, яке буде заповнено будь-якими прапорцями повідомлення (наприклад, MSG_NOTIFICATION). Зауважте, що це поле є вхідним/вихідним параметром. Параметри для отримання також можуть бути передані в значення (наприклад, MSG_EOR). Після повернення з виклику значення прапорів буде відрізнятися від початкового значення.

– usrsctp_recvv() повертає кількість отриманих байтів або -1, якщо сталася помилка. Далі змінна errno встановлюється належним чином.

5. Тестування роботи застосунку

Результати тестування застосунку

Вид тестування	Тип операційної системи	
	Windows	Linux
До покращення протоколу	0,025 с	0,037 с
Після покращення протоколу	0,02 с	0,03 с

В таблиці наведено результати тестування покращення часу обміну повідомленнями між відправником і отримувачем, у нашому випадку – між клієнтом і сервером, для транспортного протоколу SCTP, реалізованого для операційної системи Windows, а також операційної системи типу Linux. Якщо порівнювати отримані результати, то видно, що у порівнянні до покращення, час надсилання повідомлення зменшився після проведеного покращення, яке було досягнуто застосунком.

6. Генерація проєкту за допомогою CMake-сценаріїв

Для досягнення цілі використано два CMake-сценарії – для створення нової директорії (з метою перевірки роботи застосунку) і перенесення в неї файлів проєкту, а також побудова проєкту з нової директорії, а також один nmake сценарій для компіляції всього рішення.

Нижче подано результат виконання першого протоколу, який відповідає за компіляцію всього рішення і, як видно з коду, за створення програм, які будуть використовуватися для перевірки роботи транспортного протоколу SCTP, а також його майбутнього покращення. Нижче також подано команду, яка прописується у командному рядку розробника Microsoft Visual Studio за допомогою службової програми для обслуговування програм Microsoft – nmake, який створює проєкти на основі команд, що знаходяться в описовому файлі, який часто називають makefile:

```
                        nmake -f Makefile.nmake
ekr_loop.c
      link -out:ekr_loop.exe ekr_loop.obj programs_helper.obj
/LIBPATH:..\usrsctplib usrsctp.lib
Microsoft (R) Incremental Linker Version 14.33.31630.0
Copyright (C) Microsoft Corporation.  All rights reserved.

      cl /W3 /WX /I..\usrsctplib -DINET -DINET6 -c ekr_loop_upcall.c
ekr_loop_upcall.c
      link -out:ekr_loop_upcall.exe ekr_loop_upcall.obj programs_helper.obj
/LIBPATH:..\usrsctplib usrsctp.lib
Microsoft (R) Incremental Linker Version 14.33.31630.0
Copyright (C) Microsoft Corporation.  All rights reserved.

      cl /W3 /WX /I..\usrsctplib -DINET -DINET6 -c test_libmgmt.c
test_libmgmt.c
      link -out:test_libmgmt.exe test_libmgmt.obj programs_helper.obj
/LIBPATH:..\usrsctplib usrsctp.lib
Microsoft (R) Incremental Linker Version 14.33.31630.0
Copyright (C) Microsoft Corporation.  All rights reserved.

      cl /W3 /WX /I..\usrsctplib -DINET -DINET6 -c http_client.c
http_client.c
      link -out:http_client.exe http_client.obj programs_helper.obj
/LIBPATH:..\usrsctplib usrsctp.lib
Microsoft (R) Incremental Linker Version 14.33.31630.0
Copyright (C) Microsoft Corporation.  All rights reserved.

      cl /W3 /WX /I..\usrsctplib -DINET -DINET6 -c http_client_upcall.c
http_client_upcall.c
      link -out:http_client_upcall.exe http_client_upcall.obj
programs_helper.obj /LIBPATH:..\usrsctplib usrsctp.lib
Microsoft (R) Incremental Linker Version 14.33.31630.0
Copyright (C) Microsoft Corporation.  All rights reserved.
```

```
      cl /W3 /WX /I..\usrsctplib -DINET -DINET6 -c st_client.c
st_client.c
      link -out:st_client.exe st_client.obj programs_helper.obj
/LIBPATH:..\usrsctplib usrsctp.lib
Microsoft (R) Incremental Linker Version 14.33.31630.0
Copyright (C) Microsoft Corporation.  All rights reserved.

      cd ..
```

Другий сценарій, як було раніше сказано, відповідає за створення нової директорії, поза батьківською директорією, і за перенесення файлів проєкту в неї.

Для другого і третього сценаріїв було використано кросплатформову утиліту CMake, яка володіє вмінням автоматизації збірки програмного забезпечення з вихідного коду. Сам CMake не займається безпосередньою збіркою, а лише генерує файли збірки із попередньо написаного скрипт-файлу «CMakeLists.txt» і представляє собою простий єдиний інтерфейс управління. Також CMake здатен автоматизовувати процес встановлення і пакетування.

Команда, яка створює нову директорію і виконання наступного сценарію, який відповідає за перенесення файлів проєкту у новостворену директорію, з метою перевірки працездатності проєкту:

```
mkdir Cmake
cd Cmake
cmake C:\Users\zfs21\Downloads\usrsctp-master\usrsctp-master
```

Нижче показано результат виконання даних команд у форматі коду з командного рядка для розробників Microsoft Visual Studio 2022:

```
      -- Building for: Visual Studio 17 2022
      -- Selecting Windows SDK version 10.0.19041.0 to target Windows
10.0.19044.
      -- The C compiler identification is MSVC 19.33.31630.0
      -- Detecting C compiler ABI info
      -- Detecting C compiler ABI info - done
      -- Check for working C compiler: C:/Program Files/Microsoft Visual
Studio/2022/Community/VC/Tools/MSVC/14.33.31629/bin/Hostx64/x64/cl.exe -
skipped
      -- Detecting C compile features
      -- Detecting C compile features - done
      -- No build type selected, using DEBUG
      -- Looking for include file sys/queue.h
      -- Looking for include file sys/queue.h - not found
      -- Looking for include files sys/socket.h, linux/if_addr.h
      -- Looking for include files sys/socket.h, linux/if_addr.h - not
found
      -- Looking for include files sys/socket.h, linux/rtnetlink.h
      -- Looking for include files sys/socket.h, linux/rtnetlink.h - not
found
      -- Looking for 4 include files sys/types.h, ..., netinet/ip_icmp.h
      -- Looking for 4 include files sys/types.h, ..., netinet/ip_icmp.h
- not found
      -- Looking for 3 include files sys/types.h, ..., net/route.h
      -- Looking for 3 include files sys/types.h, ..., net/route.h - not
found
      -- Looking for include file stdatomic.h
      -- Looking for include file stdatomic.h - not found
      -- Looking for usrsctp.h
      -- Looking for usrsctp.h - found
      -- Performing Test have_sa_len
```

```
      -- Performing Test have_sa_len - Failed
      -- Performing Test have_sin_len
      -- Performing Test have_sin_len - Failed
      -- Performing Test have_sin6_len
      -- Performing Test have_sin6_len - Failed
      -- Performing Test have_sconn_len
      -- Performing Test have_sconn_len - Failed
      -- Compiler flags (CMAKE_C_FLAGS): /DWIN32 /D_WINDOWS /W4 /wd4100
/wd4127 /wd4200 /wd4214 /wd4706 /wd4245 /wd4389 /wd4702 /wd4701 /wd4244
/WX
      -- Performing Test has_wno_address_of_packed_member
      -- Performing Test has_wno_address_of_packed_member - Failed
      -- Performing Test has_wno_deprecated_declarations
      -- Performing Test has_wno_deprecated_declarations - Failed
      -- Looking for pthread.h
      -- Looking for pthread.h - not found
      -- Found Threads: TRUE
      -- link library: ws2_32
      -- Configuring done
      -- Generating done
-- Build files have been written to: C:/Users/zfs21/Downloads/CMake
```

І останній – третій сценарій відповідає за побудову файлів у новоствореній папці, що й продемонстровано наступним фрагментом коду:

```
      MSBuild version 17.3.1+2badb37d1 for .NET Framework
        Checking Build System
        Building    Custom    Rule    C:/Users/zfs21/Downloads/usrsctp-
master/usrsctp-master/usrsctplib/CMakeLists.txt
        sctp_asconf.c
        sctp_auth.c
        sctp_bsd_addr.c
        sctp_callout.c
        sctp_cc_functions.c
        sctp_crc32.c
        sctp_indata.c
        sctp_input.c
        sctp_output.c
        sctp_pcb.c
        sctp_peeloff.c
        sctp_sha1.c
        sctp_ss_functions.c
        sctp_sysctl.c
        sctp_timer.c
        sctp_userspace.c
        sctp_usrreq.c
        sctputil.c
        sctp6_usrreq.c
        user_environment.c
        Generating Code...
        Compiling...
        user_mbuf.c
        user_recv_thread.c
        user_socket.c
  Generating Code...
```

Висновки

Розглянуто основні методи, алгоритми та програмні моделі, які працюють за допомогою транспортного протоколу SCTP. Обґрунтовано доцільність використання цього протоколу на прикладі програмних засобів, які використовують у своїй роботі досліджуваний протокол. Описано модель, яка відповідає за реалізацію транспортного протоколу SCTP для різних платформ. Протестовано застосунок для надсилання і отримання повідомлень на операційних системах Windows, а також Linux. Порівняно з результатами, які показував протокол SCTP до покращення часу, необхідного для обміну повідомленнями між відправником і отримувачем, час надсилання повідомлень покращився на 0,005 секунд для операційної системи Windows. Для операційної системи Linux цей показник покращився на 0,007 секунд порівняно з результатом до покращення протоколу.

Список літератури

1. Transport protocol SCTP. Wikipedia. The Free encyclopedia. [Electronic resource]. – URL: https://uk.wikipedia.org/wiki/Stream_Control_Transmission_Protocol (accessed: 28 October 2022).

2. Development and implementation of network protocols : Study guide. B. Yu. Zhurakivskyi, I. O. Zeniv. Kyiv: KPI named after Igor Sikorskyi, 2020. – 462 p.

3. Computer networks : [study guide] / A. G. Mykytyshyn, M. M. Mytnyk, P. D. Stuhlyak, V. V. Pasichnyk. Lviv:«Magnolia 2006», 2013. – 256 p.

4. Burov E. V. Computer networks: a textbook / Evgeny Viktorovych Burov. Lviv: «Magnolia 2006», 2010. – 262 p.

5. Brian W. Kernighan and Dennis M. Ritchie: The C Programming Language, 2nd ed., Prentice Hall, 1988, p. 3.

6. RFC-6458. Sockets API Extensions for Stream Control Transmission Protocol. [Electronic resource]. – URL: https://datatracker.ietf.org/doc/html/draft-ietf-tsvwg-sctpsocket-32 (accessed: 28 October 2022).

7. CMake. [Electronic resource]. – URL: https://cmake.org/ (accessed: 28 October 2022).

8. C programming language. Wikipedia. The Free Encyclopedia. URL: https://en.wikipedia.org/wiki/C_(programming_language) (accessed: 28 October 2022).

9. NetFlow. [Electronic resource]. – URL: https://uk.wikipedia.org/wiki/Netflow (Accessed: 28 October 2022).

10. WebRTC. [Electronic resource]. – URL: https://en.wikipedia.org/wiki/WebRTC (Accessed: 28 October 2022).

IMPROVING THE EFFICIENCY OF NETWORK SOFTWARE OF THE SCTP PROTOCOL

Z.-A. S. Feshchenko, I. Y. Yurchak

Lviv Polytechnic National University,
Computer Engineering Department

The work presents a software system for demonstrating the operation of the SCTP transport protocol in comparison with its better-known analogues – TCP and UDP. To reproduce the operation of the chosen protocol, a socket API was created that describes the operation of SCTP, based on RFC 6458.

It also describes the transport protocols that are similar to the protocol that was chosen to improve performance, namely TCP and UDP, their strengths and weaknesses, and what has been improved in SCTP.

A test software model is described, which is divided into a client part and a server part, which was created to demonstrate the operation of the SCTP transport protocol. The client part consists in sending a message to the server, which will read this message and reset it. The ability to send using the SCTP protocol directly, or UDP encapsulation, thus encapsulating the SCTP packet into a UDP datagram, has been developed. The efficiency and expediency of using the software model are shown and alternative software models designed to implement the SCTP protocol are considered. The means used to implement this decision are justified as well as the platforms and operating systems on which this solution can be reproduced, other than Windows.

Keywords: transport protocol, SCTP, TCP, API socket, client-server system.