УДК 004.254:004.052(045)
Мельник В.М. к.т.н., Мельник К.В. к.т.н., Жигаревич О.К. асистент, Шклярський Б.М.
Луцький національний технічний університет

ПІДТРИМКА ОГОЛОШЕНОЇ/ВСТАНОВЛЕНОЇ КОМУНІКАЦІЇ В МЕРЕЖІ ЧЕРЕЗ СТАНДАРТНІ СОКЕТИ API

Мельник В.М., Мельник К.В., Жигаревич О.К., Шклярський Б.М. Підтримка оголошеної/встановленої комунікації в мережі через стандартні сокети API. В той час як адресно-орієнтовані датаграми і надійний потік послуг, що підтримуються UDP- і TCP-протоколами є основою розподілених обчислень, інші форми комунікації все більше використовуються для побудови сучасних систем і додатків. Популярною альтернативою датаграм і потоків є зв'язок на основі парадигми оголошення/встановлення (О/В) або публікації/підписки, де пересилання повідомлень і їх прийом замість адресних звернень здійснюється на основі заголовку або змістового опису. Кілька проміжних систем були побудовані для підтримки подібної форми комунікації, на вершині API сокета. Беручи інший підхід, здійснюється обговорення, як О/В мережі можуть підтримуватися за допомогою сокетів API, так що це може служити в якості універсального інтерфейсу для підтримки різних комунікаційних форм. На кінецьв роботі вводиться нове сімейство адресації та розширюється семантика обраних сокетних примітивів для здійснення підтримки О/В функцій. Також описується підтвердження концепції реалізації пропонованих сокетних розширень, в якому подаються два протоколи для цього випадку і О/В-зв'язку на основі інфраструктури, відповідно.

Ключові слова: сокети, мереже вапідтримка програм, оголошений/встановлений зв'язок, інформаційно-централізовані мережі, узагальнене програмування інтерфейсів.

Мельник В.М., Мельник Е.В., Жигаревич О.К., Шклярский Б.М. Поддержка объявленной/установленной коммуникации в сети через стандартне сокеты API. В то время как адресно-ориентированные датаграммы и надежный поток услуг, поддерживаемых UDP- и TCP-протоколами является основой распределенных вычислений, другие формы коммуникации все больше используются для построения современных систем и приложений. Популярной альтернативой датаграмм и потоков есть связь на основании парадигмы объявления/установки (О/У) или публикации/подписки, где пересылки сообщений и их прием вместо адресных обращений осуществляются на основе заголовка или содержательного описания. Несколько промежуточных систем были построены для поддержки подобной формы коммуникации, на вершине API сокета. Принимая другой подход, осуществляется обсуждение, как О/У сети могут поддерживаться с помощью сокетов API, так что это может служить в качестве универсального интерфейса для поддержки различных коммуникационных форм. На конец в работе вводится новое семейство адресации и расширяется семантика избранных сокетных примитивов для осуществления поддержки О/У функций. Также описывается подтверждение концепции реализации предлагаемых сокетных расширений, в которых подаются два протокола для этого случая и О/У-связи на основе инфраструктуры, соответственно.

Ключевые слова: сокеты, сетевая поддержка программ, объявленная/установленная связь, информационно-централизованные сети, обобщенное программирование интерфейсов.

Melnyk V.M., , Melnik K.V., Zhyharevych O.K., Shklyarskyy B.M. Support of the declared/set communication is in a network through the standard sockets of API. While the address-oriented datagram and reliable stream services supported by the UDP and TCP protocols are the foundation of distributed computing, other forms of communication are increasingly being used to build contemporary systems and applications. A popular alternative to datagram- and stream-based communication is the Publish/Subscribe (P/S) paradigm, where message forwarding and reception is done based on a topic or content descriptions instead of an address. Several middleware systems have been built to support this form of communication, on top of the socket API. Taking a different approach, the discussion is how P/S networking can be supported through the socket API, so that this can serve as a universal interface for supporting different communication abstractions. To this end, there are introduction of a new address family and extend the semantics of selected socket primitives to support P/S functions. In addition, a proof-of-concept implementation of the proposed socket extension is described, which features two protocols for ad-hoc and infrastructure-based P/S communication, respectively

Keywords: sockets, network programming support, publish/subscribe communication, information-centric networking, middleware, generic programming interfaces.

Постановка наукової проблеми. Сокети API були розроблені кілька десятиліть тому назад для того, щоб служити в якості універсального мережевого інтерфейсу для передавачів даних різних технологій і комунікаційних протоколів, що викликається через один і той же набір примітивів.На сьогоднішній день сокети використовуються для створення різних додатків, які зазвичай використовують UDP і TCP протоколи для взаємодії через Інтернет. Фактично, сокет API є, ймовірно, одним з найбільш широко використовуваних програмних інтерфейсів для різних операційних систем і мов програмування.

Значна кількість систем і додатків в наш час зосереджена навколо інформації, яка виробляється та споживається на різних місцезнаходженнях мережевої інфраструктури. В

результаті, змістове спілкування поступово стає таким же важливим, як і спілкування на основі адрес. Цей факт був визнаний співтовариством програмістів давним-давно, направляючи на розвиток оголошених/встановлених(О/В) систем[1], де потік повідомлень між вузлами (оверлеями) мережі визначається їх заголовком, а не адресою кінцевих точок, які генерують і споживають ці повідомлення. Заодно, мережева спільнота нещодавно ініціювала кілька науково-дослідницьких напрямків в колі інформаційно-централізованих мереж (ІЦМ) [2], де повідомлення або навіть маршрутизація пакетів і кешування відбувається на основі не адреси комп'ютера, а заголовку [3]. Цікаво відзначити, що О/В також став популярним в області робототехніки [4], в першу чергу через досягнення між різними компонентами складних систем, описаних в даній роботі. Загалом О/В парадигма стає все більш актуальною в ландшафті сучасної обчислювальної техніки.

З точки зору програмної інженерії О/В-функціональність, як правило, забезпечується проміжною системою, яка реалізується на верхньому рівні сокетного інтерфейсу і робить її послуги доступними програмісту-розробнику через власний API. Відомо, кожен такий проміжок на рівні апаратного забезпечення надає (злегка) різні API, з якими прикладний програміст повинен бути ознайомленим. Беручи інший підхід, виникає зацікавленість у вивченні того, як і якою мірою базова О/В-функціональність може бути введена через добре відомий сокетний інтерфейс, повністю на парах з встановленими базовими адресними датаграмами і потоковими побудовами.

Виклад основного матеріалу й обґрунтування отриманих результатів дослідження. Концепція. Ідеї, які розкриваються в роботі показані на рис. 1. Замість О/В функцій, реалізованих поверх сокети, як проміжний компонентний рівень з власним API (рис. 1а), ставиться за мету повторне використання інтерфейсу сокетів в такій міри, поки це буде можливим, і підтримка О/В зв'язку може здійснюватися через нього (рис. 1б). Це узгоджується з призначенням сокета, щоб виконувати роль універсального інтерфейсу для різних мережевих механізмів і протоколів, які відкрито реалізовані в комунікації. Наша мета – розширити "діапазон" сокетної побудови за рахунок включення О/В-функціональності не змінюючи сокетний API.

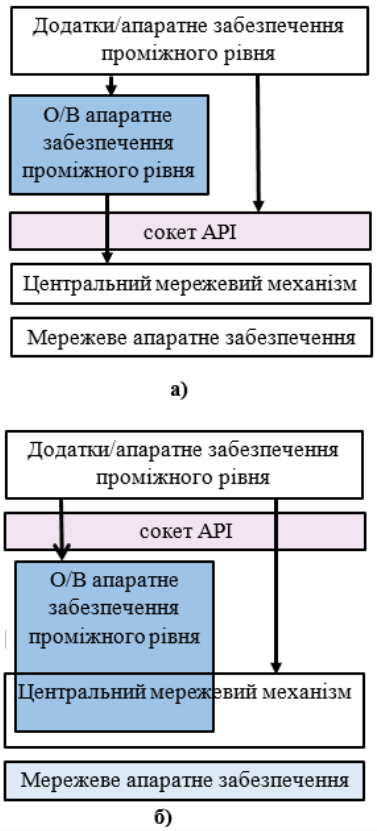

Рис. 1. Надання підтримки О/В мережі (а) на вершині сокета API (б) через/біля сокета API.

Є кілька причин, чому цей підхід може мати значення. По-перше, розробники програмного забезпечення можуть скористатися перевагами О/В через відомий сокетний API замість того, щоб вивчати інші можливості апаратного забезпечення, по можливості навіть і специфіку мови чи програмного інтерфейсу. Враховуючи, що більшість, якщо не всі програмісти знайомі з API сокетами, то накладні затрати на навчання написання програм з використанням О/В функцій найімовірніше будуть незначними. По-друге, О/В стає першокласним сервісом, який може бути використаний буквально в тісному поєднанні з послугами UDP і TCP на тому ж рівні побудови і через той же інтерфейс. По-третє, програмісту не потрібно знати, або керувати програмним забезпеченням, реалізованим з О/В функціями.

Звичайно, наявність універсального API для різних базових реалізацій є сьогодні ключовим для багатьох наукових досліджень. Та і програміст також повинен бути в курсі різних операційних семантик для того, щоб знати, як ті чи інші окремі примітиви повинні використовуватися у контексті конкретного базового механізму. Тим не менш, вважається, що сокетна побудова і інтерфейс досить багаті, щоб дозволити досить інтуїтивне відображення О/В функцій існуючих примітивів. Досвід роботи з UDP і TCP сокетами вказує на те, що такі поняття як поліморфізм побудови комунікацій і перевантаження відповідних примітивів є цілком керованими.

Слід відзначити, що рис. 1б являється концептуальним, який виявляє особливий підхід у реалізації О/В функцій через сокетну побудову. У плані реалізації, О/В компонент може бути невід'ємною частиною ядра ОС або модулем, який може бути динамічно завантажений в ОС. Він також може підходити як компонент на рівні користувача встановлений для додатків через IPC, а цей напрям міг би найбільш підходити для мікро-ядерної архітектури.

Сокетні розширення дляО/В

Для підтримки О/В через сонет API необхідно ввести нову схему адресації, яка може використовуватися у поєднанні з вже існуючими примітивами. Також слід розширити семантику цих примітивів відповідно таким чином, щоб охопити ключові О/В-функції, що базуються на основі обміну повідомленнями. У таблиці 1 наведено огляд запропонованого підходу. Надалі докладно будуть розглянуті відповідні сокетні примітиви.

socket()

Сокетний примітив традиційно використовується для створення кінцевої точки для конкретного комунікаційного механізму. Механізм задається через кортеж, що об'єднує бажане сімейство адрес, тип сокета і протокол. Програміст відповідає за комплектування допустимого поєднання, яке підтримується базовою платформою і мережевим стеком.

Щоб створити сокет для О/В-зв'язку, слід вказати адресу сім'ї– AF_PSand,а тип сокетів – SOCK_DGRAM. Також використовувалося спеціальне сімейство адрес, так як О/В працює на основі заголовка, а не машинної адреси. Враховуючи, що О/В зв'язок, як правило, на основі повідомлень і ненадійний, то слід повторно використовувати датаграму типу сокета.

Підтвердження концепції реалізації включає також в себе два протоколи, PS_ADHOC і PS_IFRSTR, виключно для цієї тільки реалізації і інфраструктури на основі О/В відповідно,про що буде описано в наступному розділі.

sockaddr_ps

Як вже згадувалося, О/В-зв'язок заснований на заголовках, які є замість кінцевої адреси вільно обраними програмними додатками. Важливо підкреслити, що заголовки не мають ніякого відношення до мережевої адреси локальної машини. Для підтримки цих різних форм "рішення" вводиться спеціальна структура sockaddr_ps для визначення цього заголовка, яку представлено нижче:

```
structsockaddr_ps
{
sa_family_tfamily; //AF_PS
inttopic_major;
inttopic_minor;
};
```

Дескриптор теми відображає два рівні ієрархічної структури, яка включає основний і другорядний ідентифікатор, що кодуються як цілі числа. Ті ж самі структури даних використовуються при публікації повідомлення за заданим заголовком, також використовуються

для повідомлень встановлення (підпису) та повідомлень отримання по цьому заголовку. Публікації повинні мати повний зміст зазначеної теми, тобто головна і підлегла частини повинні приймати позитивні значення. Для підписки (встановлення), є три варіанти: (1) повна тема (заголовок) уточняється з метою відповідності відповідним публікаціям в точності; (2) вказується лише у головна тема, в той час як підтеми залишаються рівними 0 при невідповідності всіх публікацій головної теми до цих підтем; (3) як головна так і підлеглі теми залишаються рівні 0, щоб порівняти всі видання, які будуть випущені.

Таблиця 1. Примітиви сокетів, що використовуються вО/В мережах

Примітив сокета	**Використання для О/В мереж**
intsocket(intdomain, inttype, intprotocol);	Створює сокет для О/В. Домен повинен бути AF_PS, тип SOCK_DGRAM, і протокол PS_ADHOC або PS_IFRSTR.
intbind(intsockfd, conststructsockaddr *addr, socklen t addrlen));	Додає/видаляє підписку або оголошення по заголовку (темі). Тема, зазначена в параметрі *add*з використанням структури *sockaddr_ps* та *addrlen*, встановлених відповідно.
intconnect(intsockfd, conststructsockaddr *addr, socklen t addrlen));	Асоціює з мережею для передачі повідомлень між видавцями та передплатниками. Мережа, зазначена в параметрі ADDR, використовуючи структуру sockaddr_intaaddrlen, встановлених відповідно.
intsendto(intsockfd, constvoid* buf, size_tlen, intflags, conststructsockaddr *addr, socklen_taddrlen));	Посилає/публікує повідомлення на тему (заголовок). Тема, зазначена в параметрі ADDR за допомогою структури sockaddr_pstaaddrlen, встановлених відповідно.
intrecvfrom(intsockfd, void* buf, size_tlen, intflags, structsockaddr *addr, socklen_taddrlen));	Отримує повідомлення за темою (заголовком). Тема копіюється в параметр ADDR за допомогою структури sockaddr_pstaaddrlen, встановлених відповідно.
intclose (intsockfd);	Знищує сокет, видаляючи всі його підписки/ повідомлення.

Зверніть увагу, що якщо ідентифікатори головної та другорядних тем розглядаються як ідентифікатор типу single (long), то стає можливим підтримувати дуже велику кількість тем (що приблизно дорівнює 64-бітовому цілому). Також, іменовані схеми більш високого рівня можуть бути представлені значеннями типу integer, використовуючи відповідні функції конвертування заголовків.

bind()

Примітив bind() традиційно використовується, щоб зв'язати кінцеву точку з адресою або ідентифікатором на локальній машині. Ця адреса може бути використана віддаленим об'єктом при зворотному зверненні до цієї кінцевої точки.

В О/В зв'язку bind() використовується з метою зв'язати сокет з підпискою на задану тему (чи додати підписку), так щоб можна отримувати повідомлення, опубліковані по цій темі. Тема передається як аргумент, використовуючи sockaddr_ps структуру. Як описано вище, основні/підлеглі деталі можуть бути залишені відкритими в залежності від конкретної підписки. Сокетом може бути пов'язано кілька підписок через мультивиклики bind().

Відокремити сокет від підписки (видалити підписки), можна в принципі знищивши сокет, створити новий сокет і згодом зв'язати його з рештою підписок. Це досить грубо і надзвичайно неефективно. Краще можна використовувати традиційний підхід, використовуючи відв'язки для сокета: для видалення підписки, таким же чином використовується примітив bind() з проходженням через тему в структурі sockaddr_ps, але зі значенням сімейства AF_UNSPEC.

Деякі О/В системи використовують концепцію оголошень [5]. У цьому випадку видавець попередньо оголошує теми для яких він має намір видавати дані,а представник підписки

попередньо оголошує теми для яких він бажає отримати дані. Додавання і видалення повідомлення може також підтримуватися через bind(), точно так само, як для підписок. Теми (заголовки) повідомлень повинні бути повністю вказані (як це має місце для публікацій).

connect()

Підключення примітивів традиційно використовується, щоб зв'язати сокет з віддаленою кінцевою віддаленою точкою. Семантика цього об'єднання залежить від типу сокета і базового протоколу, наприклад, у TCP сокетів, це встановлює з'єднання, а в UDP сокетів, установка за замовчуванням пункту призначення для подальшої передачі повідомлення.

У O/B зв'язку підключення використовується для того, щоб зв'язати сокет з мережевою інфраструктурою, яка буде використовуватися для передачі повідомлень між видавцями і передплатниками. Для обох протоколів нашої реалізації передачі повідомлень мережа задається через стандартну структуру sockaddr_in, тобто IP-адресу та номер порта. Тлумачення цієї адреси обговорюється в наступному розділі.

Зверніть увагу, що підключення слід викликати перш ніж надсилати або отримувати повідомлення через сокет. Це не обов'язково відноситься до наявності передплати та повідомлень, деякі реалізації можуть дозволити це зробити без підключення сокета до мережі. У будь-якому випадку наявність передплати та оголошень можна вільно додавати і видаляти після виклику connect().

sendto(), recvfrom(), close()

Примітив sendto() зазвичай використовується для відправки повідомлення іншій кінцевій точці. УО/В-зв'язкувін використовується для публікування повідомлення на певну тему. Тема передається (місце адреси кінцевої точки) за допомогою структури sockaddr_ps. В залежності від базового протоколу, можливо, доведеться об'являти тему перед відправкою повідомлення.

Примітив recvfrom(), зазвичай, використовується для отримання повідомлення від іншої кінцевої точки. У O/B-зв'язку він використовується для прийому повідомлення, опублікованого на одній з підписаних тем. Якщо програміст не зацікавлений в отриманні даної теми, відповідний аргумент може мати значення NULL.

Нарешті, close() умовно закриває сокет і звільняє відповідні ресурси, що використовувались базовим механізмом. УО/В-зв'язку, він використовується таким же чином. Однак найголовніше – закриття сокета буде також автоматично видалити всі підписки та оголошення.

Обговорення

Після цього коротко деякого описаного рішення, було прийнято деякі варіанти вибору щодо використання примітивів сокета, які варто обговорити більш докладно, вказуючи також на деякі альтернативи. Найважливіше рішення стосується вибору теми та підписки. Ми вибрали просту дворівневу класифікацію теми, яка базується на числових ідентифікаторах. Можна було б описувати теми, використовуючи зручні для читання імена і складнішу структуру. Також було б можливо підтримувати більш вигідні підписки у формі логічних виразів, що відносяться до окремих частин/полів тематичних дескрипторів. Ці альтернативи не змінять концепцію істотним чином. Насправді, відповідна підтримка може бути реалізована програмістом через один і той же набір примітивів в поєднанні з відповідними структурами "адрес" і мовної підтримки для заголовків підписки, вказавши вирази підписки. Тим не менш вважається, що запропонована схема видає хороший баланс між виразністю і простотою, і цілком достатня для багатьох додатків.

Ще однією проблемою є диференціація між передплатою та оголошенням. Для початку що використовується bind() для додавання/видалення підписки та оголошення – це хороший вибір у відповідності з роллю bind() в адресній комунікації. Також, з огляду на нашу тематичну схему опису/підписки, природно визначити підписки та оголошення з використанням тієї ж структури. Те, що явно менше інтуїтивно – це неявна типізація оголошень через значення підпорядкованої частини заголовка. Замість цього, можна було б ввести окреме поле в sockaddr_psдля задавання підписки і оголошення в явному вигляді. Цей варіант не прийнято, тому що код програми спростити не реально, програміст все ж повинен позначити оголошення (і підписки), а зверху необхідно внести відповідні символічні константи.

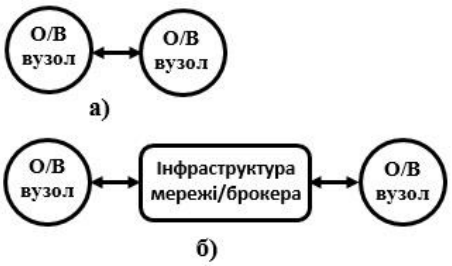

Рис.2. (а)О/В для власного випадку;(б)О/В на основі інфраструктури

Що інколи потребує деяку мотивацію – це асоціація О/В сокета з мережею передачі повідомлення через з'єднання. Можна стверджувати, що такаасоціація, при необхідності, повинна визначатися підпискою та інформацією оголошення відкритим способом. З теоретичної точки зору, це дійсно так. за таких умов можна припустити глобальну систему іменування, яка дозволяє уникнути колізій та/або випадкового перехресних перешкод між програмами, які не мають ніяких співвідношень між собою. На практиці, дозвіл контролювати додаткам цю асоціацію вводить додатковий ступінь поділу, що полегшує уникнення перешкод між програмами, які намагаються використовувати одні й ті ж заголовкові коди. Це також дає можливість ввести підтримку більш високого рівня іменування над механізмами протоколювання нижчого рівня. Примітив connect() підходить для цієї мети, так як його використання також стосується форми асоціації. Правда, немає прямої аналогії між кінцевою точкою UDP/TCP таО/В-повідомленнями мережі передавання, і те, що bind() може бути викликаний після connect()є проти звичайного використання цих примітивів.

Нарешті,замість використання bind(), підписки і оголошення можуть бути оброблені за допомогою примітиву setsockopt в міру того, як сокети UDP приєднують адреси розгалужень. Розглянемо запропоноване використання bind() з більшим узгодженням з тим фактом, що О/В обертається навколо тем, а не адрес кінцевих точок. Крім того, в нашому підході recvfrom() природним чином повертає тему повідомлення, а не адресу відправника. Якщо хтось стежить за звичайними мультиадресними UDP конвенціями, то тема прийнятого повідомлення, що в О/В є набагато важливішою інформацією для додатка порівняно з адресою видавця, що буде маловажним при отриманні.

Реалізація

Досліджуючи сокети API з запропонованим розширенням було реалізовано два різних протоколи, по одному для власної прямої комунікації (PS_ADHOC), і один на основі інфраструктури та комунікації через посередника (PS_IFRSTR). Наша реалізація відбувається як на рівні бібліотеки користувача, яка надає той же інтерфейс і на рівні стандартів сокетів API. Вона включає в себе обробку відповідного протоколу, діючи в якості оболонки для звичайних сокетів.

Протокол для власного випадку

Протокол для власного випадку (PS_ADHOC) підтримує прямі зв'язки між видавцями і передплатниками у припущенні плоскої мережевої моделі, як показано на рис. 2а. Для зручності поточна реалізація ґрунтується на розгалуженнях UDP, що дозволяє обмін одиничними власними повідомленнями в локальній мережі з можливістю трансляції. У цьому випадку connect() очікує розгалужену IP-адресу та номер порта.

Кожен вузол управляє підписками локальних кінцевих точок. Коли приходить повідомлення з мережі, ця інформація використовується для перевірки, чи має відповідна тема інших передплатників. Якщо це так, то повідомлення поміщається в буфер до тих пір, поки воно не буде отримане через всі підписані сокети, інакше воно буде видалено. Повідомлення також оновлюються по мірі необхідності, коли додаються або видаляються підписки. Протокол не підтримує оголошень.

Повідомлення передаються по мережі тільки один раз. Немає сигналізації для виявлення та/або повторної передачі втрачених повідомлень. На приймаючій стороні, якщо місце в буфері закінчується, старі повідомлення (які мають бути отриманими через один або кілька сокетів)

видаляються, щоб звільнити місце для нових. Як результат, деякі повідомлення можуть бути втрачені або не прийматися деякими абонентами.

Інфраструктура на базі протоколу

Інфраструктура на базі протоколу (PS_IFRSTR) підтримує непряму комунікаційну модель, в якій видавці і передплатники спілкуються через інфраструктуру, як показано на рис. 2b. Ця інфраструктура обслуговує як мультирозгалужену мережу з функцією маршрутизації і брокерським обслуговуванням, так що тільки відповідні повідомлення подорожують по мережі і прибувають в місця, де є очікуючі абоненти. В нашій реалізації інфраструктура складається з одного сервера. Зв'язок між вузлами видавця і передплатника (далі також-клієнти) і сервера здійснюється через TCP. Таким чином, підключення, повинно бути викликане за допомогою IP-адреси сервера і номера порта.

Клієнт відправляє підписку на місцеві сокети сервера: сервер буде переслати повідомлення до клієнта, тільки якщо він знає, що існують місцеві клієнти на цю тему. Клієнт також зберігає підписки локально щоб узгоджено діяти зі швидкозмінними умовами, які можуть виникнути в зв'язку з повільними або асинхронними оновленнями підписок. Оголошення локальних сокетів також відправляються на сервер, який перевіряє, чи є які-небудь відповідні підписки від інших клієнтів. Якщо ні, то сервер інформує клієнта для придушення передачі повідомлень на ці відіслані теми. Аналогічно, як тільки сервер отримує відповідну підписку від іншого клієнта, то він повідомляється щоб дозволити транзакцію передачі повідомлень за темою.

При запиті підключення клієнт відкриває TCP-з'єднання з сервером і реєструє необхідний унікальний ідентифікатор. З'єднання залишається відкритим для обміну повідомленнями та оновлення інформації підписки та оголошень. Якщо він залишається не задіяним протягом деякого часу,то він закривається і звільняються системні ресурси. Клієнт відкриває нове підключення, коли він хоче відправити повідомлення або оновити підпис кучи оголошення, або після мовчання, щоб опитувати сервер на наявність нових повідомлень і оновлень, або відновити раніше невдалі спроби з'єднання. Внутрішнє управління з'єднанням є прозорим для програми (за замовчуванням опитування/повторювання періодично можуть бути перевірені і змінені через примітиви getsockopt і setsockopt).

Коли сокет закривається, клієнт скасовує реєстрацію, отриману від сервера. Слід звернути увагу, що клієнт може не правильно скасувати реєстрацію. З цієї причини сервер в односторонньому порядку скасовує реєстрацію клієнта, якщо він не виявив жодних ознак активності (тобто не взаємодіє з сервером) довгий час (знову ж таки, це мовчання може бути також встановлене через setsockopt).

Буфери підписок/оголошень клієнта і сервера оновлюються і повідомлення додатків, які не можуть бути відправлені в інші місця по причині проблем зі зв'язком чи його наявністю. Це робить можливим прозоро діяти з (коротким) мережевими/серверними перебоями. Більше того, оскільки завжди клієнт ініціює взаємодію з сервером, реалізація може підтримувати мобільні обчислювальні сценарії через публічні точки доступу Wi-Fi та мережевий стільниковий зв'язок.

Сумісність

Наша реалізація функціонально еквівалентна стандартній (POSIX) реалізації для "застарілих" типів сокетів і мережевих операцій. Це досягається через захоплення O/B сокетних дескрипторів в переліку, який переглядається кожен раз, коли викликаються операції сокета. Якщо дескриптор сокета знайдений в списку, то відповідна O/B-функція виконується (помилка повертається у випадку, якщо операція незастосовна до O/B-сокетів). Інакше сокет являється спадковим сокетом і подається запит на відповідний POSIX-примітив.

Примітиви нашої бібліотеки іменуються префіксом (ps_), щоб відрізнити їх від примітивів API стандарту POSIX. Звичайно, наша реалізація не відкрита програмісту-розробнику, який знає, що він прямо не використовує підтримку сокетів на рівні системи. Повна прозорість може бути досягнута, якщо O/B функціональність була вкладена в операційну систему, але це і не дає пріоритету і не є практичним для наших прототипних цілей.

Приклади коду

Написано кілька простих додатків для налагодження нашої реалізації, а також оцінено використання O/Bсокета. Нижче також наведено фрагменти коду, що ілюструють використання O/Bпримітивів .Для стислості викладу включення, перевірки на наявність помилок та ін. –

опускаються. Створення О/В сокета для PS_IFRSTR протоколу і його асоціації з певною мережевою службою проводиться наступним чином:

```
s=ps_socket(AF_PS,SOCK_DGRAM,PS_IFRSTR);
bzero(&saddr,sizeof(saddr));
saddr.sin_family = AF_INET;
saddr_aton(srvIPaddr, &saddr.sin_addr);
saddr.sin_port=htons(srvPort);
ps_connect(s,(structsockaddr *)&saddr,sizeof(saddr));
```

Оголошення і публікація повідомлень (в даному екземплярі, прості ASCII-кодові рядки) на тему робляться наступним чином:

```
topic.sa_family=AF_PS;
topic.topic_major=-majorTopicID;
topic.topic_minor=minorTopicID;
ps_bind(s,(structsockaddr *)&topic,sizeof(topic));
topic.topic_major=majorTopicID;
ps_sendto(s,strbuf,strlen(strbuf)+1,0,(structsockaddr *)&topic,sizeof(topic));
```

Нарешті, підписки і отримання повідомлень за темою робиться наступним чином:

```
topic.sa_family=AF_PS;
topic.topic_major=majorTopicID;
topic.topic_minor=minorTopicID;
ps_bind(s,(structsockaddr *)&topic,sizeof(topic));
topiclen=sizeof(topic);
ps_recvfrom(s,strbuf,sizeof(strbuf),0,(structsockaddr *)&topic,&topiclen);
```

Як видно з вищенаведених прикладів коду, код, який повинен бути написаний програмістом для використання О/В мережі,являється дійсно простим. Також вважається, що програмування з використанням О/В-сокета повинно бути інтуїтивно зрозумілим і простим для тих, хто має базовий досвід роботи з сокетом API.

Роботи подібного характеру

Сучасні операційні системи та мережі розроблені з рекомендацією використання універсальних інтерфейсів з прозорим доступом до їх основних механізмів і послуг. Найбільш широко використовуються файли і сокети разом з їх примітивами. Файли використовуються для доступу до пристроїв і послуг, як локально на персональному комп'ютері, так і віддалено по мережі [6]. Сокети використовуються для виклику різних механізмів мережевої взаємодії та реалізації протоколу на локальній платформі [7]. Дана робота стосується сокетів API і пропонує повторне використання/перезавантаження його відомих універсальних примітивів в цілях підтримки О/В-зв'язку.

Через універсальну сокетну побудову сокет API є сильним кандидатом для інтерфейсу майбутнього Інтернету. Однак, протокольна незалежність, яка, здається, ключовою вимогою мережі наступного покоління не легко досягається за допомогою традиційних API, в основному, через їх сильну схожість з домінуючими UDP- і TCP-протоколами. Щоб розрізнити сокет API від специфіки основних вживаних протоколів науково-дослідні роботи [8] та [9] пропонують API-розширення для більш універсальних адресних структур і нових функцій, при намаганні зберегти якомога більше оригінальної семантики API і примітивів. Хоча деякі з цих розширень знадобилися б О/В мережі, однак краще звести до мінімуму API зміни і встановити/розширити семантику примітивів, які здаються найбільш підходящими для експонування функціональних можливостей О/В для програміста.

Цікаве дослідження в цьому напрямку, яке розширює сокет API для підтримки мульти комунікації стеків зв'язку представлене в роботі [10]. Ідея полягає в тому, щоб дозволити

програмістам здійснювати вибір між різними TCP- і UDP-реалізаціями з тим же підписаним стеком в умовах мережевого домена, типу сокета і протоколу. Це досягається шляхом зміни поведінки стандартних сокетних примітивів, схожих на ті, що демонструються в цій роботі. Зокрема, того ж підходу слід дотримуватися, щоб дозволити програмістам вибирати між різними доступними мережевими стеками для організації О/В комунікації.

Багато систем були розроблені для підтримки різних варіацій О/В [11], пропонуючи багатогранність API на рівні проміжного апаратного забезпечення. Однак, досі немає широкого консенсусу на інтерфейс, який може служити в якості інтернет-стандарту. Спроба визначити API такого типу наведена в [12], де пропонується три так званих рівні відповідності. Рівень L1 відображає основну функціональність О/В і складається з: (1) основних операцій для подій публікування, підписки та відмови від підписки на/від події отримування повідомлень, коли відбуваються події, що стосуються інтересів, а також (2) необов'язковість операцій декларування/відгуку наміру видавця для випуску специфічних подій (додавання/видалення оголошення), продовження часу підписки та оголошень, вибираючи кращі варіанти доставки. Деякі популярні системи О/В побудовані у відповідності до рівня L1, наприклад Siena [13] izeroMQ [14].Сокет API, запропонований у даній роботі, також відповідає рівню L1, як це безпосередньо відповідає всім основним і навіть деяким додатковим операціям. Слід звернути увагу, що цілком можливо вказати варіанти доставки повідомлень за допомогою таких стандартних сокетів API через примітив setsockopt з урахуванням можливостей протоколу використання (наприклад, базовий протокол нашої інфраструктури може бути розширений для підтримки надійного обміну повідомленнями).

Мережеве співтовариство приділяє велику увагу на інформаційно-центричну природу сучасних додатків. О/В парадигма відіграє ключову роль у розробці майбутнього Інтернету і ведуться спроби для того, щоб наполягти на тому, щоб хоч якусь частину функціональності О/В впровадити до частини мережевої структури із залученням маршрутизаторів для виконання брокерської роботи та завдання переадресації повідомлень на більш високому рівні [2] [3] [15]. Наша робота узгоджена з цією тенденцією, так як вона піднімає О/В для побудови першого класу мережі та обслуговування нарівні з датаграмами і надійним потоком комунікації. Вона також дозволяє застосування основних протоколів, вертикально інтегрованих з низьким рівнем підтримки ICN, яка, ймовірно, стане частиною майбутнього стека Інтернету. Використання простого кодування для підписок тем є ще більш важливим у цьому відношенні, тому що спрощує пошук між підписками та публікаціями/ оголошеннями, які повинні бути виконані основними елементами мережі (замість його ребер). Дійсно, подібний до нашого підхід був також прийнятий в прототипі Блекедера [16], який був нещодавно розроблений з метою великомасштабних розширення, де фільтрація пакетів і маршрутизація здійснюється на основі чисельних меж масштабу і ідентифікаторів співмірних послуг.

Нарешті, підхід майже "подвійний" до нашої пропозиції, представлений в [17], де UDP- і TCP-сокети є земульовані на вершині технології О/В. У цьому випадку передбачається, що О/В є основною мережевою технологією, на вершині якої розроблено підтримку загальновживаних датаграми і надійних послуг потокової взаємодії. В даній роботі не дається порада будь-якого ієрархічного/домінованого співвідношення між О/В і однаковими датаграми та надійними потоками. Вважається, що це по суті різні форми спілкування, і, отже, повинні бути підтримані пліч-о-пліч на тому ж рівні побудови як послуги системного рівня. Відомий API сокетів, здається, цілком доречним для відтворення всіх трьох форм спілкування з програмістом.

Висновки та перспективи подальшого дослідження. О/В та ІЦМ стають все більш важливими в сучасних розподілених обчислювальних системах. В роботі пропонується представлення О/В функціональності через стандартний API сокет, так що він може бути використаний пліч-о-пліч з традиційними датаграмами та надійним потоковим обслуговуванням через споріднену сокетну побудову та інтерфейс. Дана робота містить реалізацію та підтвердження здійснення концепції і може бути використана для отримання перших перспектив в застосуванні цього підходу, що дозволяє написання програм, які використовують О/В сокети, зберігаючи при цьому сумісність з іншими типами сокетів.

На майбутнє можна здійснити включення функціональності О/В на рівні системної служби, які можуть також забезпечити можливість провести деякі корисні дослідження продуктивності.

Можна б також реалізувати версію поданої інфраструктури на основі механізмів ІЦМ мережевого рівня, таких як наведені в роботі [16].

1. Jacobson V., Smetters D.K., Thornton J.D., Plass M.F., Briggs N., Braynard R. Networking named content. *Proc. ACMCoNEXT*, pages 1-12, December 2009.
2. Ahlgren B., Dannewitz C., Imbrenda C., Kutscher D., Ohlman B. A survey of information-centric networking. *IEEE Communications Magazine*, 50(7), pages 26-36, July 2012.
3. Carzaniga A., Papalini M., Wolf A.L. Content-based publish/subscribe networking and information-centric networking. *Proc. ACM SIGCOMM Workshop in Information- Centric Networking*, pages 56-61, August 2011.
4. Kramer J.,Scheutz M. Development environments for autonomous mobile robots: a survey. *Autonomous Robots*, 22(2), pages 101-132, February 2007.
5. Avvenuti M., Vecchio A., Turi G. A cross-layer approach for publish/subscribe in mobile ad hoc networks. *Proc. 2nd International Workshop on Mobility Aware Technologies and Applications*, pages 203-214, LNCS 3744, October 2005.
6. Pike R., Presotto D., Thompson K., Trickey H. Plan 9 from Bell Labs. *Proc. UKUUG Summer Conference*, pages 1-9, July 1990.
7. Stevens W. R., Fenner B., Rudoff A.M. *UNIX Network Programming: The sockets networking API*. Addison-Wesley, 2004.
8. Metz C. Protocol independence using the sockets API. *Proc. USENIXAnnual Technical Conference*, pages 37-47, June 2000.
9. Siddiqui A.A., Mueller P. A requirement-based socket API for a transition to Future Internet Architectures. *Proc. 6th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, pages340-345, July 2012.
10. DavoliR., M. Goldweber. msocket: multiple stack support for the Berkeley socket API. *Proc. 27th Annual ACM Symposium on Applied Computing*, pages 588-593, March 2012.
11. EugsterP.Th., Felber P.A., Guerraoui R., Kermarrec A.-M. The many faces of publish/subscribe. *ACM Computing Surveys*, 35(2), page 114-131, June 2003.
12. Pietzuch P., Eyers D., KounevS.,Shand B. Towards a common API for publish/subscribe. *Proc. Distributed Event-Based Systems*, pages 152- 157, Canada, June 2007.
13. Carzaniga A., Rosenblum D.S., Wolf A.L. Design and evaluation of a wide-area event notification service. *ACM Transactions on Computer Systems*, 19(3), pages 332-383, August 2001.
14. Sustrik M. *Design of PUB/SUB subsystem in 0MQ*. White paper, June 2011, http://250bpm.com/pubsub
15. Fotiou N., Polyzos G.C., Trossen D. Illustrating a pub- lish-subscribe Internet architecture. *Telecommunication Systems*, 54(4), pages 233-245, Springer, December 2012.
16. PURSUIT FP7 project, Blackadder prototype,http ://www. fp7-pursuit.eu/PursuitWeb/?page id=338
17. XylomenosG.,Cici B. Design and Evaluation of a Socket Emulator for Publish/Subscribe Networks. *Proc. 3rd Future Internet Symposium*, pages 11-19, LNCS 6369, Springer, September 2010.