Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
Факультет інформаційних технологій
Кафедра безпеки інформації та телекомунікацій

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеню бакалавра

студента *Данильченко Романа Володимировича*

академічної групи *172-16ск-1*

спеціальності *172 Телекомунікації та радіотехніка*

спеціалізації[1]

за освітньо-професійною програмою *Телекомунікації та радіотехніка*

на тему *Вдосконалення способу адаптивного управління пакетним трафіком протоколу транспортного рівня*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	к.ф.-м.н., проф. Гусєв О.Ю.			
розділів:				
спеціальний	к.ф.-м.н., проф. Гусєв О.Ю.			
економічний	к.е.н., доц. Романюк Н.М.			
Рецензент				
Нормоконтролер	к.ф.-м.н., проф. Гусєв О.Ю.			

Дніпро
2019

ЗАТВЕРДЖЕНО:
завідувач кафедри
безпеки інформації та телекомунікацій
_____ д.т.н., проф. Корнієнко В.І.

«_____»_____20___року

ЗАВДАННЯ
на кваліфікаційну роботу
ступеня бакалавра

студенту ***Данильченку Роману Володимировичу*** **академічної групи** ***172-16зск-1***
(прізвище ім'я по-батькові) (шифр)

спеціальності *172 Телекомунікації та радіотехніка*
(код і назва спеціальності)

на тему ***Вдосконалення способу адаптивного управління пакетним трафіком протоколу транспортного рівня***

затверджену наказом ректора НТУ «Дніпровська політехніка» від _____ № ____

Розділ	Зміст	Термін виконання
Розділ 1	Розробити програмну імітаційну модель, відтворюючу основні характеристики мережі, які і визначають функціонування в ній транспортного протоколу	20.03.2019
Розділ 2	Визначити характеристики ARTCP, а також вказати наявність властивості самоподібності у трафіку ARTCP. Порівняти результати експерименту між протоколом ARTCP та TCP по основним критеріям	30.05.2019
Розділ 3	У розділі необхідно визначити розрахунок витрат на впровадження нового транспортного протоколу ARTCP	15.06.2019

Завдання видано _____ _____
(підпис керівника) (прізвище, ініціали)

Дата видачі: 08.01.2019р.

Дата подання до екзаменаційної комісії: 17.06.2019р.

Прийнято до виконання _____ _____
(підпис студента) (прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: ___ с., __рис., __ табл., __ додатків, __ джерел.

Об’єкт розробки: транспортний протокол ARTCP.

Мета дипломної роботи: розробити програмну імітаційну модель, відтворюючу основні характеристики мережі, які і визначають функціонування в ній транспортного протоколу.

У спеціальній частині визначено найважливіші характеристики ARTCP, а також показано наявність властивості самоподібності у трафіку ARTCP. Результати експерименту дозволяють стверджувати, що ARTCP перевершує TCP по основним критеріям.

У роботі наведені:

- керування швидкістю потоку в різних режимах;
- топологічна схема з одним потоком і парою кінцевих систем;
- ілюстрація функціонування протоколу ARTCP в даних умовах.

В економічному розділі виконаний розрахунок витрат на створення розроблювального транспортного протоколу та проведено порівняльну характеристику з аналогам.

Практичне значення роботи полягає в підвищенні ефективності транспортного протоколу, за рахунок впровадження нового протоколу.

ПРОТОКОЛ, МАРШРУТИЗАТОР, КАНАЛ, ІНТЕРФЕЙС, МЕРЕЖА, АЛГОРИТМ, ТАЙМЕР.

РЕФЕРАТ

Пояснительная записка: ___ с, ___ рис., ___ табл., ___ прил., ___ источников;

Объект разработки: транспортный протокол ARTCP.

Цель дипломной работы: разработать программную имитационную модель, воспроизводящую основные характеристики сети, которые и определяют функционирование в ней транспортного протокола.

В специальной части определены важнейшие характеристики ARTCP, а также показано наличие свойства самоподобия у трафика ARTCP. Результаты эксперимента позволяют утверждать, что ARTCP превосходит TCP по основным критериям.

В работе приведены:

– управления скоростью потока в различных режимах;

– топологическая схема с одним потоком и парой конечных систем;

– иллюстрация функционирования протокола ARTCP в данных условиях.

В экономическом разделе выполнен расчет затрат на создание разрабатываемого транспортного протокола и проведена сравнительная характеристика с аналогам.

Практическое значение работы состоит в повышении эффективности транспортного протокола, за счет внедрения нового протокола.

ПРОТОКОЛ, МАРШРУТИЗАТОРЫ, КАНАЛ, ИНТЕРФЕЙС, СЕТЬ, АЛГОРИТМ, ТАЙМЕР.

ABSTRACT

Explanatory note: ___ p., ___ pic., ___table, ___appendices, ___ sources;

Subject: transport protocol ARTCP.

The aim of the thesis: to develop a software simulation model which reproduces the main features of networks that determine the functioning and it transport protocol.

In a special set of essential features ARTCP, and shows the presence of self similarity properties in traffic ARTCP. Experimental results suggest that TCP ARTCP superior to the basic criteria.

In the present study:

— control flow rate in different modes;

— topological diagram of a flow and a pair of end systems;

— illustration of the operation protocol ARTCP in these conditions.

In the economic section Calculating the cost of creating a developed transport protocol and the comparative characterization of analogues.

The practical importance of the work is to improve the efficiency of the transport protocol by introducing a new protocol.

PROTOCOL, ROUTER, CHANNEL, INTERFACE, NETWORK, ALGORITHM, TIMER.

СПИСОК УМОВНИХ СКОРОЧЕНЬ

БМ — бездротові мережі;

ДКО — диспетчер кротового обслуговування;

ЕМВВС — еталонна модель взаємодії відкритих систем;

ІС — інформаційні системи;

ІТ — інформаційні технології;

КЗ — контрольована зона;

ЛОМ — локальна обчислювальна мережа;

ПЗ — пропускна здатність;

СПЗ — сервер пропускної здатності.

ТДС — точка доступу до сервісу;

ТДТР — точка доступу до сервісу транспортного рівня;

ТПП — таймер повторної передачі;

ЗМІСТ

с.

ВСТУП

Одним з найважливіших напрямів науково-технічного прогресу в наварте час є комунікаційні системи, що представляють собою мережі передачі інформації. Координацію процесів передачі інформації в розподіленій системі, якою є мережа, здійснюють комунікаційні протоколи.

Прийнято розділяти комунікаційні протоколи за ступенем спільності завдань, що вирішуються ними, на кілька рівнів, упорядкований набір яких утворює мережеву архітектуру. Найпоширенішою і універсальною мережевої архітектурою є архітектура TCP/IP [43, 1]. В рамках TCP/IP всі системи в мережі діляться на кінцеві системи, між якими відбувається інформаційний обмін, і проміжні системи, які не є кінцевими або вихідними точками обміну. Кінцеві системи називаються вузлами мережі, а проміжні - маршрутизаторами.

Двосторонній потік інформації між парою суміжних систем у мережі забезпечується каналом, що зв'язує дві системи. Канали характеризуються швидкістю інформаційного потоку (пропускною спроможністю), затримкою передачі та ймовірністю бітових помилок. У кожній точці підключення маршрутизатора до каналу є буфер, в якому організовується чергу даних очікують відправки по цьому каналу. Буферне простір і пропускна здатність (ПС) є колективні ресурси мережі. Якщо зростає прибуття інформації в маршрутизатор перевищує максимально можливу швидкість її відправки, то відбувається перевантаження мережі, висловлює в переповненні буферів і втрати інформації.

Протокол транспортного рівня займає найважливіше положення в будь-який мережевий архітектурі, в тому числі і в TCP/IP, оскільки він забезпечує надійну і ефективну передачу інформації безпосередньо між кінцевими системами мережі. Для цього транспортний протокол задає погодити ванний набір правил поведінки для учасників інформаційного обміну. Ці

правила регулюють спільний доступ вузлів до ресурсів мережі, тому ефективність транспортного протоколу визначає ефективність роботи всієї мережі в цілому. Програма, що реалізує алгоритм протоколу, називається об'єктом протоколу.

Транспортним протоколом в архітектурі TCP/IP є TCP (Transmission Control Protocol) [4, 5, 6], який забезпечує надійний двосторонній зв'язок з контролем швидкості передачі. Джерело TCP потоку отримує інформацію від користувача у вигляді послідовності бітів, формує з неї блоки кінцевої довжини, звані сегментами і відправляє їх до TCP одержувачу. Одержувач, приймаючи сегменти, формує з них результатна послідовність і передає її своєму користувачеві.

Для здійснення обміну TCP встановлює логічне з'єднання між парою вузлів мережі, на кожному з яких виконується алгоритм TCP. Потік сегментів по TCP з'єднання може проходити через впорядковану послідовність маршрутизаторів і каналів. Пропускна здатність з'єднання в цілому обмежена мінімальної з ПС каналів, через які проходить з'єднання. Алгоритм керування потоком, що є частиною TCP, прагне відправляти дані зі швидкістю, що не перевищує менше з ПЗ з'єднання і швидкості споживання інформації одержувачем.

Набір з'єднань транспортного протоколу, які поділяють загальний канал, являє собою складну систему, що самоорганізується в сенсі Г. Хакена [24]. Поведінка кожного з об'єктів протоколу в цій системі визначається алгоритмом протоколу, однак, поведінка всієї системи, як цілого, взагалі кажучи, не описується сукупністю дій її компонентів. Кожен об'єкт протоколу прагне максимально ефективно адаптуватися до доступних ресурсів мережі в умовах кооперації з іншими об'єктами цього протоколу.

На сьогоднішній момент відомий ряд істотних недоліків алгоритму управління потоком протоколу TCP:

Для оцінки доступною ПС алгоритм управління потоком TCP постійно збільшує швидкість відправки сегментів, штучно викликаючи перевантаження мережі. Це призводить до частих втрат пакетів і, при стійкому переповненні буферів, до збільшення затримок сегментів в мережі.

TCP інтерпретує втрату сегмента як ознака перевантаження мережі і реагує на будь-яку втрату даних зниженням швидкості передачі, що веде до суттєвих обмежень ефективності TCP в мережах, де ймовірність втрати сегментів через виникнення помилок відмінна від нуля. Це стосується, зокрема, до всіх бездротових мереж.

Локальні нерівномірності у відправці сегментів TCP призводять до з підвищення ймовірності втрати сегментів при максимальному заповненні буферів.

Усунення наведених вище недоліків TCP є темою великого числа досліджень. У роботах на цю тему пропонуються різні варіанти удосконалення транспортного протоколу. Більшість протоколів, пропонованих для використання в мережах з ненульовою ймовірністю бітових помилок, не є сумісними з TCP і вимагають введення додаткових елементів в структуру мережі, ускладнюючи її та порушуючи основний принцип транспортного протоколу, що складається в тому, що на транспортному рівні з'єднання встановлюється між безпосереднім джерелом і одержувачем інформації.

Таким чином, найважливішим завданням є розробка нового транспортного протоколу в рамках архітектури TCP/IP, який більш ефективний, ніж TCP. Новий протокол повинен бути універсальним в сенсі можливості використання його як в дротяних, так і бездротових мережах, що особливо важливо в світлі подальшого розвитку мережевих технологій і розширення областей їх застосування.

У дипломі розроблено новий транспортний протокол ARTCP. Розроблено-на універсальна об'єктно-орієнтована імітаційна модель для

конструювання мереж з топологією будь-якої складності. Проведені експерименти роботи протоколу ARTCP для ряду сценаріїв показали, що він майже завжди перевершує стандартний протокол TCP.

РОЗДІЛ 1. СТАН ПИТАННЯ. ПОСТАНОВКА ЗАДАЧІ

1.1 Комунікаційні транспортні протоколи

1.1.1 Сучасні комунікаційні мережі

Важливість мереж передачі даних і мультимедіа інформації неможливо переоцінити сьогодні, на початку інформаційної епохи розвитку людства. Системи збору, обробки і розподілу інформації є найважливішою точкою докладання наукового знання. Розвиток технології усунуло розходження між обробкою і передачею інформації, а найбільш актуальним завданням в комунікації стала специфікація та верифікація комунікаційних протоколів, а також підвищення їх ефективності.

Мережі передачі даних і мультимедіа інформації в сучасному об-суспільстві застосовуються повсюдно у виробництві та у сфері управління, у дослідницькій середовищі і в побуті. Економічна вигода від підвищення ефективності комунікаційних протоколів може виявитися дуже істотною, особливо з урахуванням подальшого росту і розвитку комунікацій-ційних систем.

Будь-яка комунікаційна система складається з двох основних компонентів: апаратної та логічної частин.

1.1.2 Апаратна інфраструктура мереж

Апаратна частина це фізична інфраструктура, за допомогою якої здійснюється розповсюдження фізичних сигналів кодують інформаційні повідомлення. Фізична інфраструктура може або здійснювати інформаційний обмін між усіма учасниками мережі одночасно (широкомовна мережа), або тільки між двома вузлами (мережа типу точка-точка). В залежності від масштабу і топології мережевий інфраструктури виробляють її подальшу класифікацію: локальні обчислювальні мережі (ЛОМ), територіальні мережі (WAN), сотові мережі і т.д.

Багато властивостей сімейства протоколів для комунікаційної системи перебувають у прямій залежності від її фізичного середовища і топології, тому, параметри фізичної інфраструктури мережі ретельно враховуються при розробці її протоколів.

1.1.3 Системи протоколів

Найважливішим компонентом мережі є комунікаційні протоколи, що становлять її логічний компонент. В якості неформального визначення протоколу можна привести наступне: протокол являє собою угоду за процедурою обміну інформацією в розподіленої системи.

Завданням будь-якого протоколу є управління спільним використанням ресурсів. Протоколи подібні з мовами. Визначення протоколу дається наступним чином через визначення його функціональних компонентів [1]:

– протокол визначає точний формат допустимих повідомлень (синтаксис);

– протокол визначає процедурні правила для обміну інформації цией (граматика);

– протокол визначає словник правильних повідомлень та їх значення (семантика).

1.1.3.1 П'ять елементів протоколу

На додаток до визначення, даному вище, розробка протоколу передбачає детальну специфікацію сервісу надається протоколом користувачеві, яким може бути прикладна програма або протокол більш високого рівня, а також знання характеристик середовища, в якому протоколі буде виконуватися.

Таким чином, виділяють п'ять компонентів протоколу, кожен з яких повинен бути заданий тим чи іншим методом формального опису:

1 Сервіс, що надається користувачеві;

2 Передбачувана характеристика середовища виконання протоколу;

3 Словник допустимих повідомлень;

4 Кодування повідомлень - формат кожного повідомлення зі словника;

5 Процедурні правила, контролюючі обмін повідомленнями.

Визначення протоколу через "пов'язані кінцеві автомати"

На низькому рівні абстракції протокол можна представити у вигляді ко-кінцевих автоматів. У своїй монографії [1] Хольцман пропонує так звану модель пов'язаних кінцевих автоматів для формалізованого опису протоколу. В цьому випадку завдання розробки, формальної верифікації та перевірки на сумісність зводяться до знаходження бажаних/небажаних станів і переходів. Кожен з пов'язаних конечних автоматів може отримувати символи на вхід, здійснювати перехід в новий стан і генерувати символьну інформацію на виході. Окремі кінцеві автомати зв'язуються між собою за допомогою обмежених FIFO черг, через які вихідний сигнал одного автомата надходить на вхід іншого. Так моделюється асинхронний зв'язок, характерний для більшості комунікаційних систем.

Визначення черги: черга повідомлень (символів) представляється трійкою (S, N, C), де S - обмежене безліч визначальне словник черги, N - ціле число позицій у черзі, C - упорядкований безліч елементів з *S*. *S* і *C*: безлічі повідомлень. Якщо модель системи вимагає наявності кількох черг, то їх словники повинні утворювати непересекається множини. Якщо *M* безліч всіх системних черг, індекс $1 \le m \le |\,M\,|$ нумерує черги, а $1 \le n \le N$ позиції всередині черги, C_n^m - n-ний символ в m-тої черги. Системний словник тоді виражається через словники кожної з черг і нульовий символ $\varepsilon : V = Y_{m-1}^{|M|} S^m Y_\varepsilon$. В [1] дається визначення пов'язаних кінцевих автоматів як набору (*Q, q0, M, T*), де *Q* - кінцеве непорожнє безліч станів, *q0* - елемент *Q* – початкове стан, *M* - безліч символьних черг, визначених вище, T - відношення переходу. T має два аргументи *T (q, a)*, де q поточний стан і a - дія (одне з: введення, висновок, пусте дія). Реалізація переходу з діями типу введення/виведення залежить від стану символьних черг. При їх реалізації змінюється стан однієї з черг.

Ставлення переходу T задає безліч (яке може бути порожнім) всіх можливих результуючих станів. Достатньою умовою реалізації *T (q, e)* є те, що поточний стан є q.

Розширення моделі пов'язаних кінцевих автоматів безліччю внутрішніх змінних, а набору дій булевими операціями та операціями присвоєння значення дає еквівалентну модель, але істотно зменшується число станів. Для такої моделі визначені алгоритми мінімізації та виконання. Проте уявлення реальних складних протоколів, наприклад TCP, у вигляді кінцевих автоматів вкрай складно і неефективно, тому для їх розробки і специфікації застосовують менше формальні методи.

Ієрархії протоколів

Принципи розробки і дослідження будь-якої складної системи переважають розчленування її на частини меншого обсягу і складності, які можна досліджувати окремо. Маючи на увазі принципи взаємодії частин, можна узагальнити знання на всю систему. Системи протоколів в комунікації мають складну ієрархічну структуру. Кожен з протоколів в такій системі є певний рівень абстракції, при цьому нижчі рівні приховують низько рівневі завдання від верхніх рівнів. Ієрархічний принцип дозволяє висловити логічну структуру протоколів, розділяючи завдання нижніх і верхніх рівнів.

Таким чином, для зниження складності розробки комунікаційні системи організуються у вигляді декількох рівнів (шарів). Кількість, зміст і функції шарів різні для різних типів мереж. Метою існування кожного з рівнів є надання певних типів сервісів рівнями розташованим вище їх і захист верхніх рівнів від деталей реалізації цих сервісів.

Рівень *N* на одному пристрої логічно перебуває в стані промена з рівнем N на іншому мережевому пристрої. Набір законів і угод, прийнятих при такому обміні називається протоколом рівня N.

Рисунок 1.1 – Приклад інкапсуляції даних одного протоколу в форматі повідомлень іншого. Протокол 2 інкапсулює повідомлення протоколу 1.

В реальності ніякого обміну даними не відбувається між шарами на одному рівні. Замість цього кожен рівень передає дані і контрольну інформацію на рівень знаходиться прямо під ним, поки не буде досягнутий нижчий рівень, що помістить інформацію безпосередньо на фізичний носій для передачі її по мережі.

Між кожною парою суміжних шарів знаходиться інтерфейс, керуючий якими операціями і сервісами нижнього рівня може скористатися вищерозміщений рівень. В процесі розробки мережі дуже важливим моментом є чітке і чисте визначення інтерфейсів між шарами. Для цього кожен рівень повинен реалізовувати певний набір добре продуманих функцій. Ретельне опрацювання інтерфейсів не тільки робить можливою заміну одного типу внутрішньої реалізації рівня на інший за умови збереження семантики інтерфейсу, але і сприяє зведенню до мінімуму кількості контрольної інформації, що пересилається між рівнями. У цьому випадку заміна може бути здійснена прозоро для суміжних рівнів.

Набір шарів і відповідних їм протоколів називається архітектура мережі. Упорядкований набір протоколів визначених у рамках конкретної мережної архітектури називається стеком протоколів. Кожен з *N* рівнів однієї системи здійснює обмін з рівнем *N* кінцевої системи, використовуючи для цього сервіси рівня *N-1*, при цьому кожен з проміжних рівнів інкапсулює дані верхнього рівня всередині свого формату повідомлень рисунок 1.1. Таким чином, на

кожному рівні *N* процедура, що відправляє або отримує дані від рівня *N* інший системи, насправді відправляє інформацію для обробки на рівень *N-1* своєї системи, забезпечивши дані своєї контрольної інформацією.

Загальні властивості рівнів ієрархічної системи

У функції кожного з рівнів входять декілька типів операцій, які, реалізуючись порізному на кожному рівні абстракції, мають схожі ознаки. Так кожному рівню необхідний механізм для ідентифікації відправника і одержувачів певна система адресації і мультиплексування потоків.

Природно, що і відправник і одержувач повідомлень повинен здійснювати проміжну буферизацію даних. У даній роботі пропонується нова схема організації контролю швидкості передачі потоку даних протоколу транспортного рівня.

Сервіси та інтерфейси

Надання сервісів верхнім рівням є головним завданням кожного з рівнів. Активні елементи всередині кожного з рівнів називаються об'єктами протоколу. Об'єкт може являти собою програмний процес або частину функціональності апаратури, наприклад інтелектуальний контролер вводу/ виводу. Об'єкти рівня *N* реалізують сервіси, які використовуються рівнем *N +1*. У цьому випадку рівень *N* є постачальник послуг, а рівень *N+1* користувачем. Рівень *N* для виконання свого завдання з надання набору сервісів рівню *N+1* може сам виступати користувачем послуг рівня *N-1*. Можливе надання декількох типів сервісу, наприклад швидкої і ненадійною зв'язку поряд з повільною і надійною. Сервісами рівня можна скористатися через інтерфейс, званий точкою доступу до сервісу (ТДС). Кожен з можливих ТДС рівня *N* це інтерфейс, на якому об'єкт рівня *N +1* може мати доступ до сервісів рівня *N*. Кожна ТДС ідентифікується унікальною адресою. Щоб два суміжних рівня могли обмінюватися інформацією необхідна наявність набору правил регламентують розподіл доступа і інтерфейса між ними рисунок 1.2.

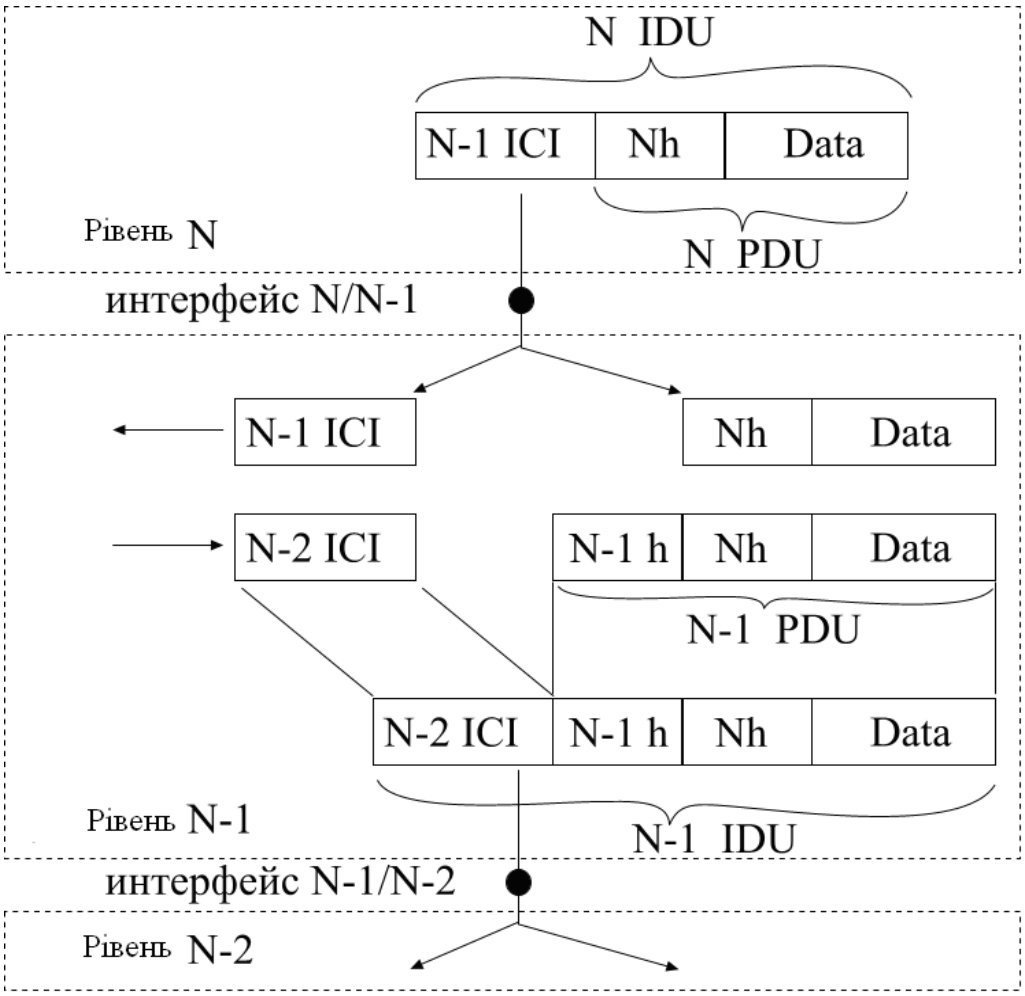

Рисунок 1.2 – Перетворення потоку інформації на інтерфейсі NSAP між суміжними рівнями мережевої архітектури

На рисунку 1.2 обозначено:

IDU - формат блоку даних на відповідному інтерфейсі;

ICI - контрольна інформація для нижчого рівня;

PDU - повідомлення, визначене протоколом відповідного рівня;

IDU - повідомлення в форматі інтерфейсу між рівнями;

h - заголовок, що несе контрольну інформацію для протоколу відповідного рівня;

Data - передані дані.

Механізм обміну наступний: сутність рівня *N* постачає дані заголовком несе інформацію для протоколу рівня *N*. На приймальній стороні з'єднання заголовок *Nh* буде використаний рівнем *N* для відновлення даних в

початковому вигляді, після чого самі дані будуть передані користувачеві рівня *N*. Дані спільно з заголовком утворюють спільнота протоколу рівня *N (PDU)*. Крім того, рівень *N* передавальної сторони постачає *N PDU* додаткової контрольної інформацією для рівня *N-1 (N-1 ICI)*. Структура, що містить *ICI* і *PDU*, утворює так зване повіщення в форматі інтерфейсу між рівнями *(IDU)* яке передається нижчого рівня.

На рівні *N-1* від повідомлення відділяється контрольна інформація *N-1 ICI*, яка визначає спосіб обробки даного повідомлення. Після цього рівень *N*-1 генерує власну контрольну інформацію для рівня *N-2 (N-2 ICI)* і додає її до повідомлення разом із заголовком протоколу рівня *N-1 (N-1 h)*. Після цього вийшла структура даних передається на рівень *N-2* для подальшої обробки.

Розглянемо як визначається сервіс, що надається рівнем свойому користувачеві.

Типи з'єднань

Рівні ієрархічної архітектури можуть надавати два кардинально розрізняються типу сервісів рівнями знаходяться над ними: сервіс з встановленням логічного з'єднання та сервіс, при якому логічне з'єднання не встановлюється.

У разі сервісу з встановленням логічного з'єднання обмін даними може початися лише після того, як від відправника до одержувача встановлено логічно виділений канал. Протокол транспортного рівня TCP функціонує саме за такою схемою.

Сервіс без установки логічного з'єднання заснований на моделі поштовой системи. Кожне повідомлення має повну адресу одержувача і в потоці повідомлень не дотримується черговість доставки. Таким характеристика відповідає протокол IP застосований в мережах з комутацією пакетів, який надає свій сервіс протоколу TCP.

Причина, по якій вивчення аспектів взаємодії протоколів суміжних рівнів надається багато уваги в тому, що ця взаємодія є одним з необхідних компонентів протоколу, а саме характеристикою середовища його виконання.

Примітиви сервісів

Сервіс формально визначається набором примітивів, що визначають операції, доступні користувачеві. Примітиви є командами об'єктивним ту сервісу вчинити певну дію або видати звіт про виконання дії. Одним із способів класифікації операцій сервісу є організації їх примітивів в чотири основні класи:

Таблиця 1.1 – Способі класифікації операцій сервісу

Запит	Об'єкт повинен виконати певну задачу
Індикація	Запитуючий повинен бути проінформований про подію
Відповідь	Запитуючий бажає прореагувати на подію
Підтвердження	Прибуття відповіді на попередній запит

Приклад - встановлення та припинення зв'язку для найпростішого протоколу можуть бути активовані за допомогою наступного набору: (примітиви можуть мати параметри)

CONNECT.request - Запити на з'єднання

CONNECT.indication - сигнал викликається стороні

CONNECT.response - застосовується викликається стороною для підтвердження / скасування встановлення з'єднання

CONNECT.confirm - повідомлення викликає стороні про прийом запиту на з'єднання

DATA.request - запит на передачу даних

DATA.indication - сигнал про прийом даних

DISCONNECT.request - запит на роз'єднання

DISCONNECT.indication - сигнал іншій стороні про роз'єднання

Зв’язок сервісів і протоколів

Сервіс це набір примітивів (елементарних дій) які рівень *N* надає користувачу - тобто рівню *N+1*. Сервіс відноситься до інтерфейсу між двома рівнями. Протокол, з іншого боку, є набір правил, що визначають формат і значення повідомлень, якими обмінюються об'єктами комунікаційних вузлів, що знаходяться на однаковому рівні і пов'язаних через мережу. Об'єкти користуються протоколами для реалізації своїх сервісів. За аналогією з об'єктними мовами програмування сервіс є класом, в якому визначені операції, які можуть бути здійснені, але не обговорюються деталі їх внутрішньої реалізації, яка й є тим чи іншим протоколом.

1.1.4 Еталонні моделі

Модель ISO OSI RM

Міжнародною організацією зі стандартизації (ISO) з метою уніфікації методів розробки протоколів була запропонована так звана еталонна модель взаємодії відкритих систем (OSI RM) рисунок 1. 3.

Еталонна модель OSI RM складається з семи рівнів:

Фізичний рівень

У завдання цього рівня входить коректна передача бітового потоку з фізичної лінії зв'язку. Коли відправник передає «1», біт приймаюча сторона повинна отримати його як «1» біт, а не «0». Специфікації цього рівня задають як логічні, так і фізичні і схемотехнічні параметри. Фізичний рівень розглядає інформацію як безперервний бітовий потік.

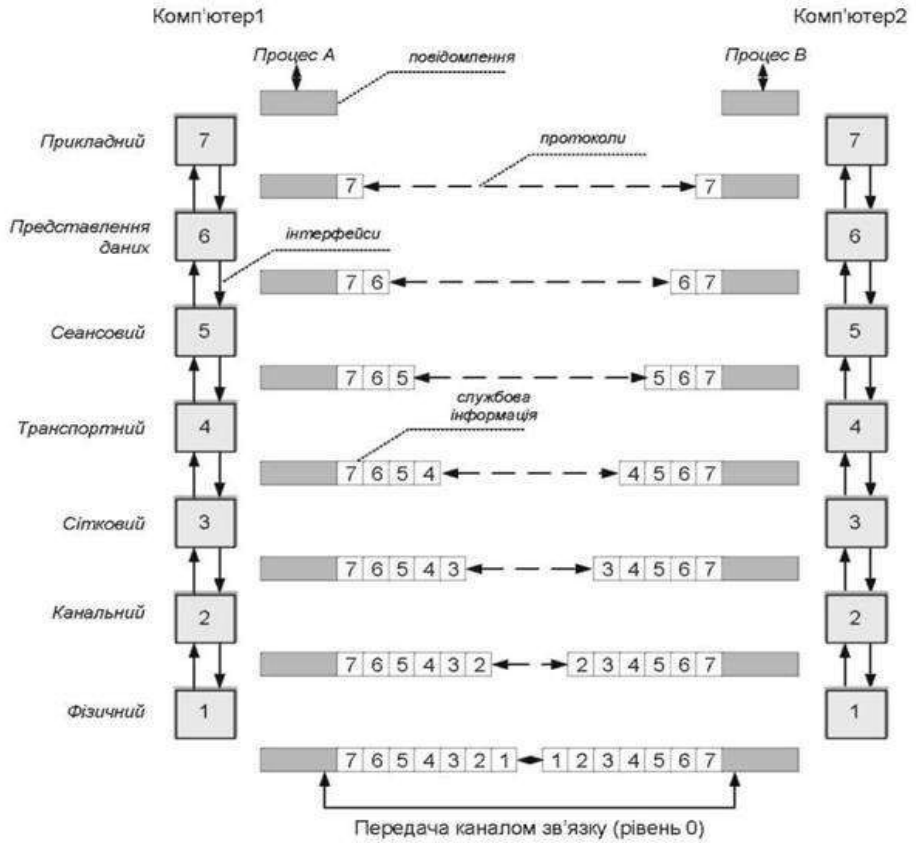

Рисунок 1.3 – Еталонна модель взаємодії відкритих систем (EMBOC). Рівні і напрям передачі даних.

Канальний рівень

Цей рівень створює логічний канал, який не має помилок і збоїв, для потреб мережевого рівня. Відправляється інформація розбивається на блоки, звані кадрами, які пересилаються по черзі. В деяких випадках рівень створює і обробляє кадри з підтвердженням коректного прийому даних. Оскільки фізичний рівень має справу з неструктурірованним потоком бітів, то канальний рівень повинен піклуватися про створення і відстеження кордонів кадрів. Також канальний рівень може застосовувати різні алгоритми для визначення та по можливості корекції спотворення інформації.

Канальний рівень бере на себе всі турботи про акуратному надавати пріоритет лении інформації мережевого рівня. Інша проблема, яку доводиться вирішувати на багатьох рівнях, включаючи канальний, це, як не допустити

перенання повільного одержувача даними від швидкого відправника. Необхідний певний механізм, що дозволяє відправникові інформації мати дані про наявність вільного буферного простору в одержувача і діяти відповідно до цього знанням. Мережі з широкомовної передачі вносять додаткові складності в пристрій канального рівня - управління множинним доступом до каналу.

Мережевий рівень

Завданням мережевого рівня є управління роботою базової підмережі. Найважливіша проблема - розрахувати шлях від точки відправлення то одержувача. Такий маршрут може бути заснований на статичних таблицях зберігаються в пам'яті пристроїв підмережі або ж маршрут може визначатися на початку кожній сесії, альтернативним варіантом є високо-динамічна маршрутизація, коли шлях заново визначається для кожного окремого пакета. Якщо в базову подсеть потрапляє більше пакетів, ніж мережа може обробляти, то виникає перевантаження, боротьба чи профілактика якої також є завданням мережевого рівня. Одиниця інформаційного обміну на мережевому рівні називається пакетом.

Транспортний рівень

Транспортний рівень надає послуги (і приховує від верхніх рівнів) по надійної транспортування даних по мережі. Даний рівень забезпечує відкриття, підтримку і нормальне відключення віртуальних каналів, передає повідомлення про помилки та збої, здійснює управління швидкістю потоків інформації, щоб уникнути переповнення буферів мережних пристроїв і приймачів і, відповідно, втрат даних. Відмінною особливістю транспортного рівня є те, що об'єктом ти протоколу транспортного рівня здійснюють обмін безпосередньо, незалежно від базової підмережі. Одиниця інформаційного обміну на транспортному рівні називається сегментом або датаграммой.

Сеансовий рівень

Цей рівень синхронізує, відкриває, закриває і маніпулює сеансами зв'язку, активними об'єктами рівня презентації (яких може бути декілька). Рівень сеансу присвоює ступінь терміновості повідомленням, керує прискореної передачею повідомлень про виникли помилки вищого рівня.

Рівень презентації

Рівень презентації здійснює кодування даних, забезпечуючи сумісність для різних форматів представлення інформації.

Рівень програми

Цей рівень відрізняється від інших тим, що його об'єктами є безпосередньо програми, виконувані на обчислювальному вузлі. Функція цього рівня полягає у визначенні клієнтів, яким необхідна комунікація, синхронізації комунікації, визначенні наявність ресурсів для проведення сеансу комунікації та детектування помилок.

1.1.4.1 Модель TCP / IP

В середині 1970-х американської військової дослідницької організацією DARPA було прийнято рішення про створення мережі з комутацією пакетів для забезпечення з'єднання дослідних установ на території Сполучених Штатів. У той час дослідники та виробники вперше зіткнулися з проблемою організації в мережу різноманітних і гетерогенних комп'ютерних систем. З метою вироблення протоколів зв'язку для різнорідних систем DARPA початку фінансування досліджень проведених в Стенфордському університеті по створенню сімейства мережевих протоколів. Другою метою було створення мережі має можливість продовжувати функціоніувати навіть при виході з ладу істотної частки її апаратної частини. В результаті цієї роботи з'явилися протоколи сімейства Internet protocols. Здатність нової розробки з'єднувати мережі, засновані на раз-них технологіях, абсолютно прозорим чином була основним завданням розробників з самого початку. Найбільш практичними і

широко використовуючими членами цієї родини є протоколи Transmission Control Protocol (TCP) - протокол контролю передачі і Internet protocol (IP) протокол інтернету, а сама архітектура стала відомою під назвою еталонної моделі TCP/ IP.

Сімейство протоколів TCP/IP застосовується для комунікації через будь-яку кількість проміжних ЛОМ. Протоколи TCP/IP ідеально підходять як для комунікації в ЛОМ, так і для глобальних обчислювальних мереж. Набір протоколів TCP/IP містить специфікації не тільки протоклів низького рівня, таких як TCP або IP, але також і таких широко поширених додатків як електронна пошта, емуляція віддаленого терміналу, протокол передачі файлів і багато чого іншого.

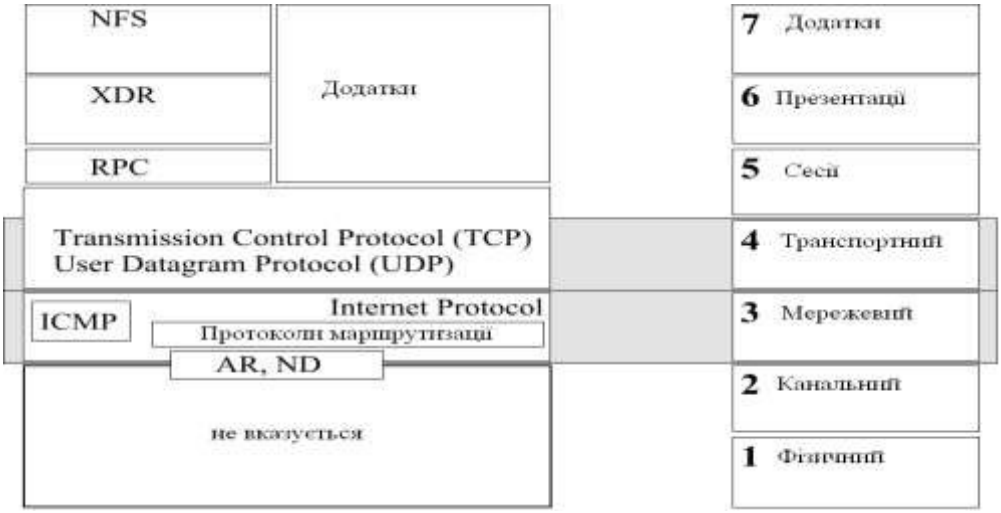

Рисунок 1.4 - Зіставлення еталонних моделей мережевих архітектур EMBOC і TCP / IP.

Як видно з рисунок. 1.4 рівні моделі TCP / IP не повністю збігаються з рівнями моделі OSI. Основні складові моделі TCP / IP відповідають мережевого і транспортного рівнів EMBBC. Ці протоколи - IP і TCP є ключовими для концепції сучасної комунікаційної архітектури рисунок 1.5.

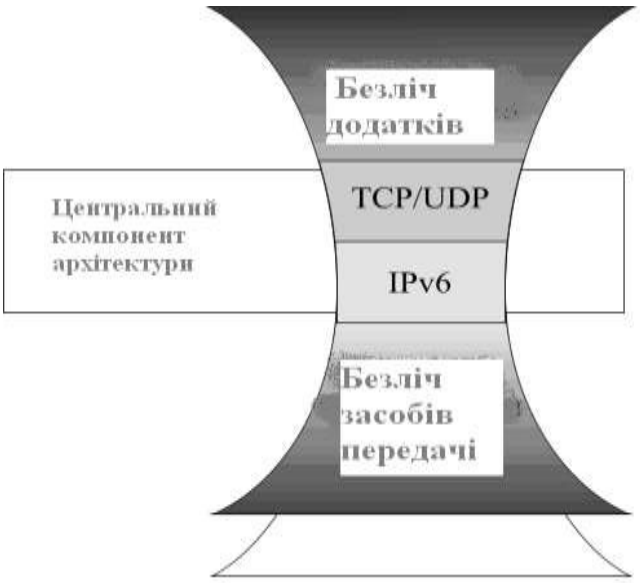

Рисунок 1.5 - Сучасна концепція мережевої архітектури

Мережевий протокол моделі TCP / IP: протокол IP

Протокол IP є єдиним протоколом мережевого рівня сімейства TCP/IP, тому всі транспортні протоколи стека TCP/IP використовують сервіси протоколу IP. Тобто сервіс, що представляється IP, є середовищем, в якому виповнюється протокол TCP.

Протокол IP надає користувачеві ненадійний сервіс по передачі пакетів. На додаток до функцій маршрутизації, тобто визначення оптимального маршруту пакетів від відправника до одержувача на підставі інформації про топологію мережі, IP також відповідальний за фрагментацію та зборку пакетів і повідомлення про збійних ситуаціях.

Транспортні протоколи моделі TCP / IP

Транспортний рівень набору протоколів інтернету складається з двох протоколів: TCP (Transmission Control Protocol - протокол контролю передачі) і UDP (User Datagram Protocol - протокол користувальницьких датаграм). TCP надає надійний транспорт з установкою логічного сполуки.

TCP є найбільш важливим транспортним протоколом моделі TCP/IP він забезпечує повністю дуплексний зв'язок з кумулятивним підтвердження

прийому. Він переміщує інформацію у вигляді безперервного неструктурованого потоку, де байти ідентифікуються порядковим номером. Об'єкт протоколу TCP може одночасно підтримувати безліство сеансів інформаційного обміну для протоколів вищих рівнів, здійснюючи мультиплексування потоків.

1.1.5 Еволюція комунікаційних протоколів

Розглянемо процес розвитку мережевих протоколів на прикладі стека TCP/IP. Нас цікавить питання наступності в розвитку протоколів, і їх зворотної сумісності. На розвитку реальних протоколів відображається безліч факторів: це ступінь складності протоколу, якість його специфікації, верифіковані даного протоколу і результати його тестування на відповідність стандартам. Крім того, ряд практичних характеристик протоколу може істотно вплинути на його реальне використання і одним з найважливіших факторів тут є зворотна сумісність нового протоколу з попередніми версіями або реалізаціями.

Зворотна сумісність нової версії протоколу означає, що його реалізація зможе взаємодіяти з старими версіями без втрати потужності, причому поліпшення характеристик роботи системи буде відбуватися при взаємодії нових версій протоколів або в деяких випадках вже при взаємодії старої версії з нової. Доцільність вимоги зворотної сумісності цілком виправдана, з іншого боку, діалектичний розвиток протоколів комунікаційних систем призводить до необхідності зміни одного протоколу на інший, несумісним з колишнім на певному етапі розвитку системи. При цьому зворотна сумісність або не зберігається зовсім, або забезпечується за рахунок временного застосування додаткових механізмів не є частиною системи протоколів. Така ситуація спостерігається в даний час, коли мережевий протокол в мережі Інтернет IP версії 4 замінюється на нову версію IPv6 [2], яка не передбачає зворотної сумісності зі старим протоколом Інтернет. Такий підхід цілком виправданий, оскільки завдання забезпечення зворотної сумісності вимагає ускладнення багатьох компонентів протоколу - його словника і процедурних правил, істотно

ускладнює його аналіз і верифікацію. У разі IPv6 було вирішено пожертвувати обратної сумісністю для забезпечення мінімальності і простоти безлічі процедурних правил протоколу.

Перша специфікація транспортного протоколу TCP була дана в роботі [3] в 1980 році. За минулі 30 років протокол TCP піддавався великій кількості оптимізацій і доповнень, які або вирішували очевидні проблеми виявляються по ходу застосування протоколу, або покращували його характеристики для систем вузької спеціалізації. Всі ці зміни залишали протокол сумісним зі старими версіями. В результаті складність протоколу TCP зросла настільки, що повний перебір досяжних станів кінцевого автомата моделює протокол і навіть контрольований вибірковий перебір не є можливими для автоматизованої верифікації. Внаслідок цього утруднена не лише автоматизована верифікація протоколу TCP, але і ручний аналіз навіть обраних наборів його станів.

Основне нововведення пропонованого в даній роботі протоколу транс-кравця рівня ARTCP полягає в заново створеному алгоритмі керування швидкістю потоку, який використовує зовсім відмінні від TCP принципи. Реалізація протоколу ARTCP може забезпечувати сумісності зі стандартним TCP.

1.1.6 Транспортний рівень: його роль та компоненти

Транспортний рівень є центральним для всієї ієрархії протоколів. Завдання цього рівня - забезпечувати надійну та ефективну транспортування інформації від вихідного до кінцевого пристрою поза залежно від фізичних особливостей мереж чи обмежень накладених протоколами БМ.

Сервіс транспортного рівня

Послуги, що надаються користувачеві

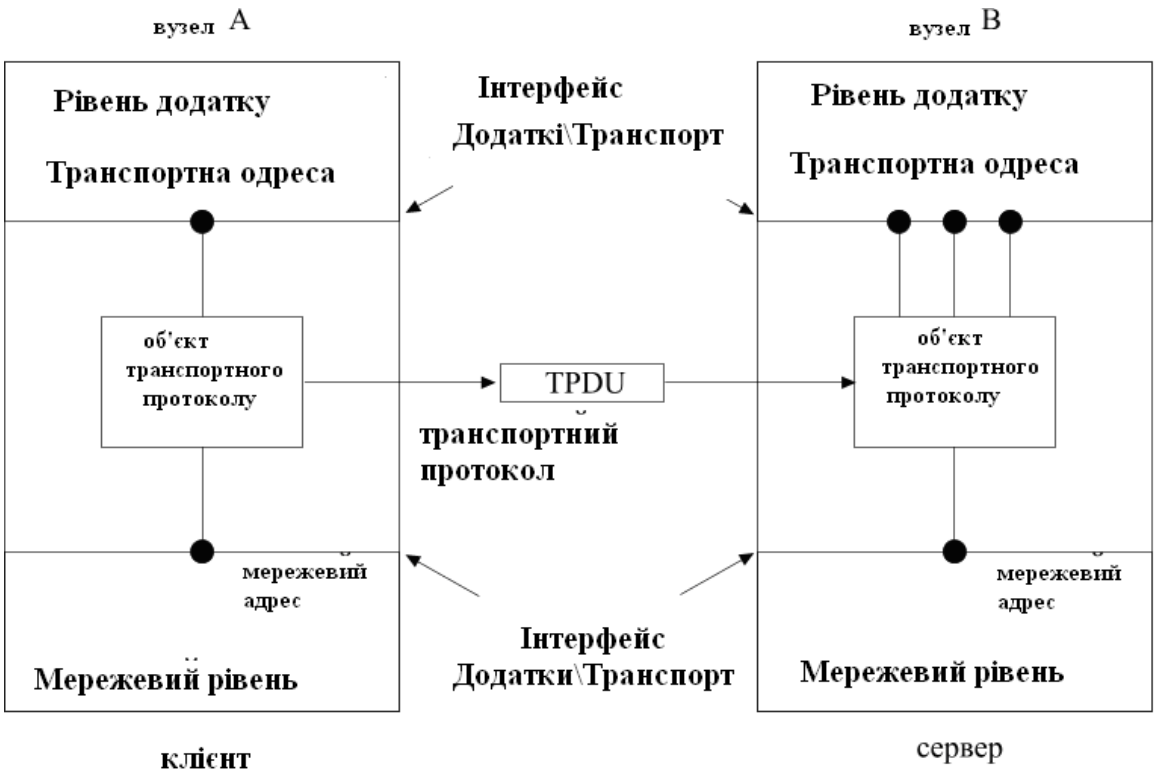

Рисунок 1.6 – Обмін даними на рівні транспортного протоколу. Формат даних транспортного протоколу TPDU

Головне завдання транспортного рівня полягає в наданні ефективного і надійного сервісу для відповідних користувачів - головним чином для користувача процесів рівня програми. Для виконання цих завдань транспортний рівень покладається на сервіси мережевого рівня. Апаратне і/або програмне забезпечення, яке виконує функції транспортного рівня називається транспортним об'єктом. Цей об'єкт може бути частиною ядра операційної системи, окремим процесом в користувацькому просторі, у вигляді бібліотечних об'єктів скомпонований з одними програмами або функцією окремої інтерфейсної плати. Блок даних, за допомогою якого відбувається обмін між об'єктами транспортного протоколу, називається TPDU (повідомлення транспортного протоколу) рисунок 1.6.

На відміну від всіх нижніх рівнів, транспортний рівень функціонує безпосередньо між учасниками обміну інформацією, безпосередньо і дозволяє цим системам обмінюватися інформацією незалежно від проміжних мереж і систем. Користувачі, як правило, не мають контроля над БМ, тому єдиний

спосіб, за допомогою якого вони можуть впливати на якість послуг БМ, це використання додаткового рівня над мережним.

Транспортні протоколи роблять можливим надавати користувачеві транспортний сервіс набагато більш надійний, ніж використовуваний мережевий сервіс. Наявність втрачених та пошкоджених пакетів контролюється і компенсується транспортним рівнем. Крім того, сервісні примітиви транспортного рівня можуть бути розроблені таким чином, щоб бути повністю незалежними від мережевих сервіс примітивів, котрі радикально різняться для різних типів мереж. Завдяки наявності транспортного рівня, користувальницькі додатки можуть розроблятися із застосуванням стандартного інтерфейсу (стандартного набору викликів сервіс примітивів) і використовуватися без змін на самих різних типах мережевих технологій, що і має місце на практиці. Транспортний рівень ізолює прикладні програми від особливостей технології, розробки, ненадійності і різнорідності мереж. Саме тому існує поділ рівнів моделі OSI RM на дві групи з 1-го по 4-й і вище 4-й. Перша називається постачальником транспортних послуг, друга - користувачем транспортних послуг. Проведення такого розходження, по-перше, впливає на специфіку реалізації рівнів та протоколів, а по-друге, ставить транспортний рівень на місце ключової ланки в ієрархічній моделі, оскільки він знаходиться на інтерфейсі між постачальником і користувачем надійного транспортного сервісу. Основне завдання транспортного рівня в тому, щоб максимально поліпшити якість послуг мережевого рівня.

Розглянемо, яким чином протокол TCP покращує параметри сервіса порівняно з сервісом мережевого рівня. Як було сказано раніше сукупний вплив фізичного, канального і мережевого рівнів на передані дані виражається в появі трьох типів помилок даних:

- втрата;
- дублювання;
- спотворення порядку відправки.

Протокол TCP повністю усуває ці помилки. З точки зору користувача транспортного протоколу, мережа виглядає як повністю дуплексний канал, який переміщує потік байтів користувача без втрат, причому байти з'являються з каналу в тому самому порядку, в якому вони надійшли в нього на передавальній стороні і лише одноразово. Виконання всіх цих функцій ніяк не залежить від використовуваних технологій мережевого, канального та фізичного рівнів, яких може бути багато на маршруті від відправника до одержувача.

Додатка, що використовує сервіс TCP, досить лише запросити зв'язок з іншим додатком, а потім почати передавати свою послідовність даних після установки логічного з'єднання, не піклуючись про втрати, помилки, управління швидкістю передачі і т.д.

Примітиви транспортного сервіс*у*

Примітиви транспортного сервісу вже є достатньою формалізацією для опису сервісного компонента протоколу. Через ці примітиви користувачі (додатки) отримують доступ до послуг транспортного рівня. Кожен тип транспортного сервісу має свій набір примітивів. На відміну від ненадійної зв'язку без установки ВК на мережевому рівні, транспортний рівень пропонує надійний зв'язок з побудовою логічного каналу.

Оскільки реальні мережі не можуть гарантувати відсутність помилок, то завдання транспортного рівня як раз в тому, щоб забезпечити надійний зв'язок, користуючись ненадійною. Наприклад, два процеси використовують іменний комунікаційний канал (pipe) [4] в середовищі ОС UNIX [5], у своїй роботі припускають, що з'єднання є ідеальним. Розробнику програми реалізує ці процеси немає необхідності піклуватися про відстеження підтверджень, втрат повідомлень, перевантажень. З'єднання для нього виглядає як абсолютно надійне. Аналогічно, транспортні протоколи являють ненадійний канал як бітовий потік вільний від помилок для користувача.

В якості додаткового сервісу транспортний рівень може пропонувати і ненадійну зв'язок без підтверджень прийому та управління потоком, в середовищі TCP/IP такий сервіс реалізується протоколом UDP.

Різні також і користувачі транспортного та мережевого рівня. Мережовий сервіс використовується тільки об'єктами транспортного рівня. Вкрай рідко програми розробляються так, щоб використовувати сервіси мережового рівня безпосередньо. Основна маса програм розробляється саме в розрахунку на використання сервісів, а, отже, і примітивів транспортного рівня. Тому специфікація сервісу транспортного рівня повинна бути універсальною і простий у використанні, як, наприклад, інтерфейс *Berkeley Sockets*.

Для того щоб скласти уявлення про примітиви транспортного сервісу розглянемо реальний інтерфейс *Berkeley Sockets* [6, 7]

Це інтерфейс для взаємодії додатків з об'єктами транспортних протоколів, включаючи протокол TCP. Спочатку був розроблений для роботи з TCP в операційній системі сімейства BSD UNIX [5] в 1980 році. Зараз інтерфейс Berkeley Sockets є промисловим стандартом і використовується в більшості операційних систем.

Користувач отримує доступ до даних примітивам на UNIX системі у вигляді системних викликів, оскільки об'єкт протоколу TCP в системі UNIX є частиною ядра операційної системи [4].

Виклик *SOCKET* в разі успішного завершення повертає дескриптор файлу і виділяє місце під таблиці в контрольних блоках на транспортному рівні. Параметри виклику вказують на тип і характеристики необхідного сервісу.

Таблиця 1.2 – Команд виклику

Примітив	Пояснення
SOCKET	Створити нову точку доступу до системи зв'язку
BIND	Присвоїти локальну адресу ТДС
LISTEN	Оголосити про готовність до прийому сполук, встановити розмір черги

Продовження таблиці 1.2

Примітив	Пояснення
ACCEPT	Блокувати ініціатора до надходження запиту на
CONNECT	Активна спроба встановлення з'єднання
SEND	Передача даних
RECEIVE	Отримання даних
CLOSE	Завершити сеанс зв'язку

Користувач отримує доступ до даних примітивам на UNIX системі у вигляді системних викликів, оскільки об'єкт протоколу TCP в системі UNIX є частиною ядра операційної системи [4].

Виклик *SOCKET* в разі успішного завершення повертає дескриптор файлу і виділяє місце під таблиці в контрольних блоках на транспортному рівні. Параметри виклику вказують на тип і характеристики необхідного сервісу.

Після отримання дескриптора, що характеризує одну сторону з'єднання, йому присвоюється локальну адресу за допомогою виклику *BIND*. Причина використання окремого виклику в тому, що деякі процеси вимагають використання стандартного адреси, для інших же цього не потрібно.

LISTEN оголошуючи про готовність приймати з'єднання, виділяє місце під чергу прийнятих сегментів. *LISTEN* не є блокуючим викликом. Виклик *ACCEPT* блокує процес викликав його в очікуванні з'єднання.

Розрив з'єднання відбувається при виконанні примітиву *CLOSE* і є симетричним.

1.1.6.1 Характеристика середовища використання

Для того, щоб визначити середу виконання протоколу TCP або його запропонованої модифікації ARTCP необхідно розглянути принципи функціонування рівнів ієрархії, сервісами яких користується транспортний протокол. Згідно еталонної моделі OSI RM транспортному рівню передують мережевий, канальний і фізичний рівні ієрархії. Оскільки завдання кожного з

рівнів в тому, щоб максимально ізолювати свого від проблем нижніх рівнів, досить розглянути лише сервіс мережевого рівня.

Сервіси мережевого рівня

Доступ до сервісів мережевого рівня можливий на інтерфейсі між мережним і транспортним рівнями. Важливість даного інтерфейсу визначається тим, що часто цей інтерфейс відокремлює споживача комунікаційних послуг від їх постачальника, тобто оператора базової мережі. Оператор базової мережі має повний контроль над протоколами попередніми транспортному рівню. Саме з цієї причини інтерфейс повинен опрацьовуватися особливо ретельно. Сервіси мережевого рівня будь комунікаційної системи повинні володіти наступними основними характеристиками:

– сервіси не залежать від канальної технології базової мережі;

– транспортний рівень повинен бути екранований від кількості, типів і топологій різних базових мереж;

– якщо мережеві адреси є доступними для транспортного рівня, то вони повинні укладатися в межі стандартної схеми адресації, в якій кожен адреса унікальний.

Розглянемо коротко функціонування протоколу IP, і проаналізуємо його з точки зору середовища виконання транспортного протоколу.

Отримуючи дані від вищого рівня у вигляді блоків кінцевого розміру, протокол IP інкапсулює їх в пакет, додаючи до даних користувача свій заголовок. На рисунку 1.7 наведені поля заголовка протоколу IPv6. Для нас становить інтерес те, що в заголовку відсутнє поле коду циркулярного контролю. Це пов'язано з тим, що завдання перевірки та корекції помилок передачі покладено на канальний рівень, який завжди виконує цю перевірку. Протокол IPv4 здійснював перевірку цілісності даних за допомогою коду контролю парності, але з функціональності IPv6 дана операція була вилучена, оскільки вона дублює аналогічну або більш потужну систему канального рівня.

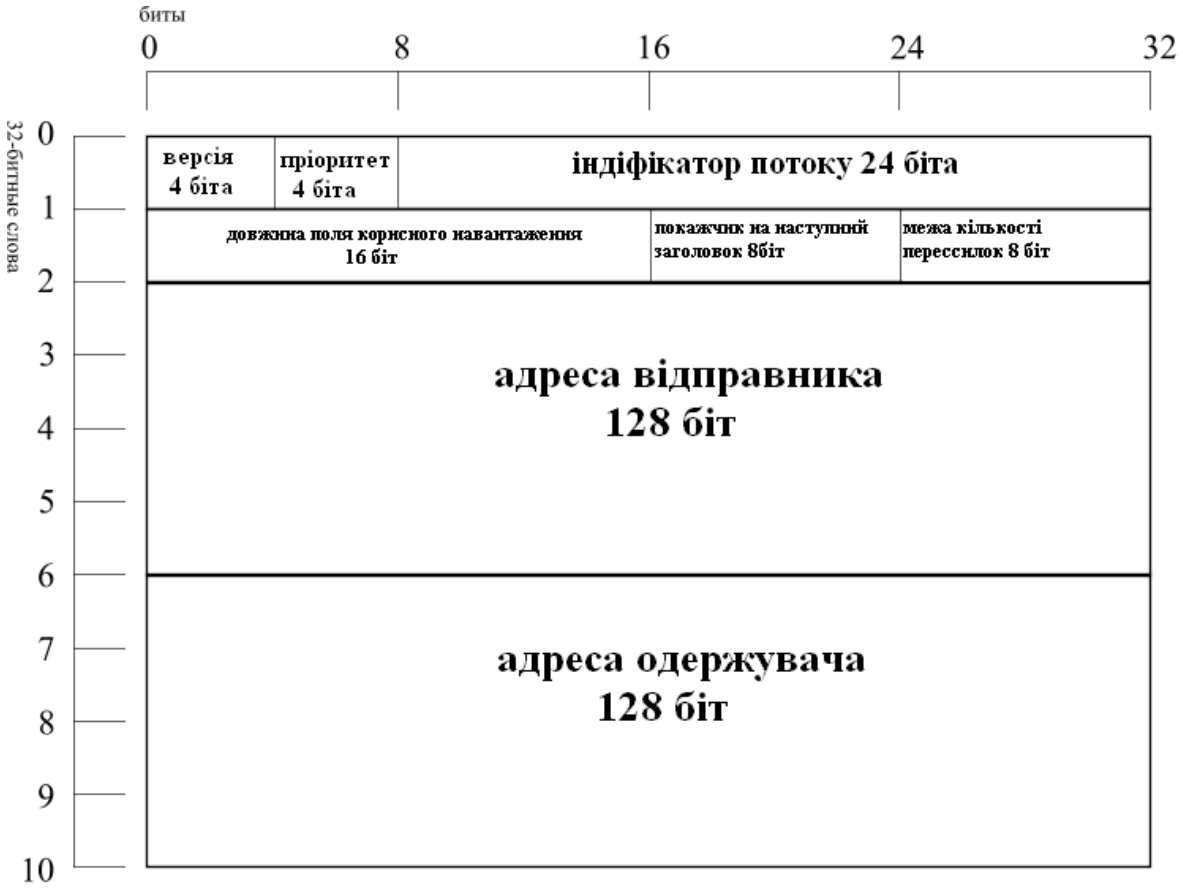

Рисунок 1.7 – Формат заголовка пакета протоколу IP (зображений для IP версії 6)

Закінчивши компоновку пакета, IP передає його в мережу на маршрутизатор топологічно більш близький до вузла якому адресовано повідомлення. Кожен IP пакет в заголовку містить повну адресу відправника і одержувача. Маршрутизатори в складі мережі обмінюються топологічної інформацією за допомогою протоколів маршрутизації і завдяки цьому мають знанням топології мережі. Отримуючи IP пакет, кожен маршрутизатор порівнює адресу вузла призначення з наявною у нього таблицею топологічної інформацією і направляє пакет на відповідний вихідний порт. Оскільки різні маршрутизатори можуть володіти різним уявленням про топологію мережі, то і маршрутів між двома вузлами в мережі може бути кілька. Якщо на вихідний порт маршрутизатора надходить більше пакетів в одиницю часу, ніж може його покинути, то пакети організуються в чергу. Оскільки ресурси буферного

простору маршрутизаторів кінцеві, то переповнення черги призводить до втрат пакетів.

Виникнення і корекція помилок

Низьке співвідношення енергії сигналу до енергії шуму на лінії призводить до спотворень сигналу і як наслідок до високої ймовірності бітових помилок на приймачі. Помилки каналу передачі даних проявляються як:

1 Вставлені дані: дані, отримані приймачем, але ніколи не передавалися відправником;

2 Втрачені дані: дані відправлені, але не дійшли до одержувача (зниклі на фізичному рівні);

3 Дубльовані дані: дані, передані один раз і отримані в декількох примірниках;

4 Спотворені дані: дані, пошкоджені в транзиті;

5 Розупорядкування дані: дані, послідовність отримання яких не збігається з послідовністю передачі.

Застосовувані на канальному рівні алгоритми розпізнавання і корекції помилок з достатньою ймовірністю дозволяють відобразити помилки типу вставлених і перекручених даних на решту три типи помилок, а саме втрата, дублювання і разупорядочіваніе даних.

Мережевий рівень стека TCP/IP надає користувачеві ненадійний сервіс без установки логічного з'єднання. Сказане означає, що IP пакети можуть почати надходити в мережу без затримки, як тільки інформація готова до відправки (пакет сформований). Надіслані пакети не підтверджуються одержувачем. Пакет може бути втрачено, і ніколи не дійти до одержувача, з кількох причин: він може бути відкинутий з переповненого буфера маршрутизатора внаслідок перевантаження останнього, або дані в складі пакета можуть бути зруйновані в процесі передачі по ненадійному каналу. Таким чином, мережевий рівень може привнести додаткові помилки типу втрат даних. Також внаслідок можливості наявності кількох маршрутів до одержувача

порядок прибуття пакетів може не збігатися з порядком їх відправки, якщо потік пакетів будуть доставлятися по маршрутах з різною затримкою. Більше того, існує можливість доставки одержувачу декількох копій одного пакета.

Протокол TCP, таким чином, повинен самостійно відслідковувати виникнення трьох типів помилок даних і здійснювати самостійне відновлення в ситуаціях, коли сегмент не приходить взагалі, коли сегменти доставляються мережею не в тому порядку, в якому були відправлені і коли мережа доставляє кілька копій одного сегмента.

Управління потоком

Кожен з нижчих рівнів здійснює управління швидкістю передачі даних. Фізичний рівень відповідальний за синхронізацію запису і сканування середовища передачі. Канальний рівень управляє швидкістю передачі пакетів між двома вузлами, застосовуючи схему управління потоком тієї чи іншої складності, в залежності від призначення - Xon / Xoff, Alternating bit [8], sliding window [9] і т.п. Мережевий рівень також має примітивну схему управління потоком. Маршрутизатор в стані перевантаження може відправити своїм топологічним сусідам повідомлення про наявність перевантаження за допомогою протоколу ICMP. Проте всі алгоритми канального і мережевого рівнів здійснюють управління швидкістю потоку лише локально. TCP, як протокол транспортного рівня, здійснює управління потоком по всій довжині логічного каналу між передавальної і приймаючої системами.

У протоколі TCP існує механізм, що обмежує швидкість відправки сегментів в мережу, для того щоб уникнути переповнення буферів проміжних маршрутизаторів і буфера приймача.

Процедурні правила

Сервіс транспортного рівня реалізується транспортним протоколом між двома об'єктами транспортного рівня. Процедурні правила, необхідні для реалізації цього сервісу дуже громіздкі, тому їх формальний опис наприклад у

вигляді кінцевих автоматів було б занадто громіздким. Далі ми дамо опис процедурних правил TCP у вигляді окремих алгоритмів. Окремими важливими аспектами транспортного протоколу, які повинні бути враховані в його процедурних правилах є:

- адресація і мультиплексування;
- ініціалізація та закриття зв'язку;
- буферизація і керування потоком.

Словник транспортного протоколу

Словник найпростішого транспортного протоколу, що забезпечує надійний канал зв'язку і контроль швидкості передачі, повинен дозволяти передавати такі типи повідомлень [10, 1]:

- дані (DATA);
- підтвердження (ACK);
- розмір приймального вікна (WND).

Таким чином, мінімальний словник транспортного протоколу забезпечує надійний зв'язок такий: V = {DATA, ACK, WND}. Якщо розглядати кожен напрямок полнодуплексного транспортного сполучення роздільно, то дані передаються в напрямку від відправника до одержувача, а підтвердження і поновлення вікна в протилежному напрямку.

Оскільки кожне індивідуальне повідомлення транспортного протоколу инкапсулируется мережевим рівнем в окремий пакет, то вигідно об'єднати якомога більше повідомлень в кожному TPDU. Реально в кожному повідомленні об'єднуються всі ці типи. Формат повідомлень транспортного протоколу такий:

{DATA, SEQ}

{ACK, SEQ, WND}, де SEQ і WND - цілі числа.

Оскільки транспортні сполучення розраховані на двосторонній обмін повідомленнями, то об'єднуються всі поля в єдиному форматі повідомлення:

{DATA, SEQ, ACK, SEQ, WND}

Число SEQ на другій позиції нумерує передані дані, а в третій позиції - підтверджуються дані.

1.1.6.2 Кодування повідомлень на транспортному рівні

Отже, словник транспортних протоколів складається з повідомлень - так званих TPDU, які інкапсулюють передані дані. Сам TPDU в свою чергу укладений в пакет мережевого рівня.

Формат сегмента, як правило, наступний: заголовок фіксованої довжини вирівняний на 32 бітної кордоні передує полю даних змінної довжини. Більш конкретна схема кодування сегмента і деталізація додаткових полів наведена у схемі заголовка TCP сегмента рисунок. 1.8.

Заголовок сегмента транспортного протоколу містить поле, що дозволяє перевірити цілісність даних і заголовка прийнятого сегмента. У протоколі TCP це 16-ті бітове поле несе перевірочну суму, отриману підсумовуванням по модулю 2 всіх 16-ті бітних слів частині заголовка і поля даних.

При отриманні сегмента перевірочна сума обчислюється заново і якщо результат не дорівнює нулю, то сегмент містить помилку відкидається.

Рисунок 1.8 - Формат заголовка сегмента TCP.

Адресація

Для установки зв'язку з віддаленим додатком потрібно вказати його адресу. Адресація необхідна для всіх видів мереж. Існує метод, що дозволяє встановлювати транспортний адресу, за якою додаток-сервер очікує прибуття запиту на з'єднання. Для архітектури TCP/IP це пара: IP адреси та номер локального порту. Об'єкти транспортного рівня допускають використання декількох транспортних адрес рисунок1.9, при цьому мультиплексування потоків в межах вузла здійснюється самим транспортним протоколом. Постійно існуючі серверні додатки адресуються безпосередньо, для роботи з додатками, що запускаються лише на час існування сесії, використовується так званий «протокол початкового з'єднання» або інтернет сервер (UNIX inetd) [11].

Установка з'єднання

Без здатності БМ до накопичення пакетів завдання встановлення з'єднань звелася б до двох дій - надіслати запит на з'єднання - дочекатися позитивної відповіді на нього. Насправді ж проблема значно складніша. Якщо БЬ перевантажена, і підтвердження не встигають прибути до відправника вчасно, то й запити на встановлення з'єднання можуть бути відправлені декілька разів. Оскільки маршрутизація кожного пакета відбувається незалежно, то існує ймовірність затримки деяких пакетів протягом більш тривалого часу, ніж інших і, відповідно, збоїв при встановленні з'єднання. Одинична транзакція може, таким чином, випадково відбутися два і більше рази.

Використовуваний в архітектурі TCP/IP метод поєднує з ідентифікацію кожного пакета з вимогою обмеження часу життя пакетів в мережі. Причому ідентифікація кожного пакета увазі наявність у нього порядкового номера.

В реальності необхідно гарантувати, що не тільки сам пакет зник з мережі, але і всі його підтвердження, тому використовується проміжок часу T, кратний максимальному часу життя пакета. Якщо пройшов час T з моменту

відправки пакета, то ми можемо бути впевнені, що ні сам пакет, ні його підтвердження не існують на мережі.

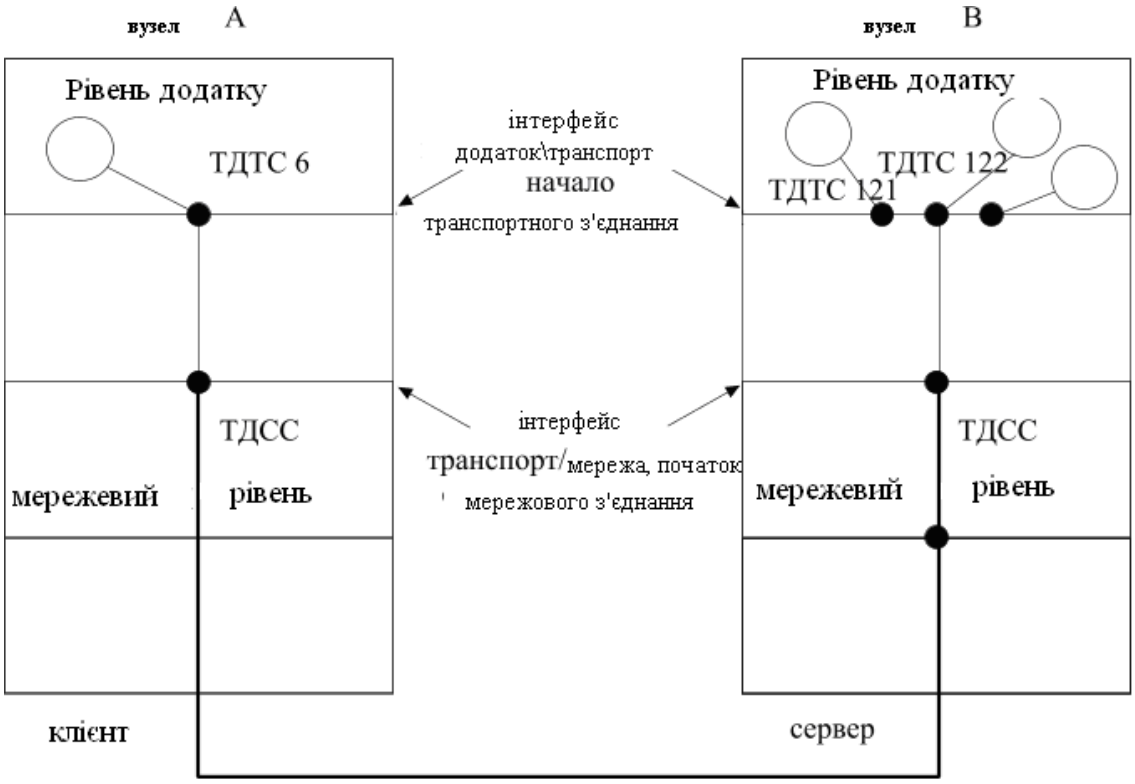

Рисунок 1.9 - Точки доступу до сервісу транспортного рівня (ТДТР). Зображені інтерфейси транспортного рівня з рівнем програми та мережевим рівнем

Якщо спиратися на таку передумову, то можна розробити надійний спосіб безпечної установки з'єднання. Метод був вперше запропонований [12] в 1975 і вдосконалений [13] в 1978. Метод називається «тристоронній обмін» (Three-way handshake). Цей метод не потребує використання обома сторонами одних і тих же номерів. Нормальна процедура установки з'єднання ілюстрована на рисунок 1.9 (частина А.) Пристрій А посилає запит на з'єднання (CR) пристрою В, вказуючи при цьому початковий порядковий номер який буде використовуватися транспортним протоколом пристрою А для передачі даних. Пристрій В, отримавши запит, реагує відправкою пакета, повідомляючого про згоду встановити з'єднання, вказує свій власний початковий номер *y* і підтверджує номер *x*. Після цього в першому пакеті з даними, що має номер *x*, машина А підтверджує прийом повідомлення з номером *y*. Варіанти В та С

ілюструють поведінку транспортних об'єктів при отриманні дубліката запиту на з'єднання та дубліката підтвердження.

Схема установки з'єднання наведена вище застосовується транспортним протоколом TCP. Альтернативна схема надійної установки з'єднання в умовах наявності затриманих копій повідомлень описана [14].

Розрив з'єднання

Для закриття транспортного з'єднання використовується так званий симетричний розрив, при якому кожен напрямок з'єднання вважається незалежним і закривається окремо.

Рисунок 1.11 ілюструє процес закриття з'єднання з використанням таймерів. Незважаючи на відносну надійність такого протоколу, він може не спрацювати у випадку втрати початкового DR і всіх *N* повторних передач. В цьому випадку ініціює сторона закриє з'єднання, протилежна ж не отримавши інформації про закриття буде активною. Такий стан називається напіввідкрите з'єднання. Для запобігання виникнення таких станів вводиться правило, за яким з'єднання буде автоматично закриватися, якщо протягом певного часу на ньому не зареєстровано жодної активності.

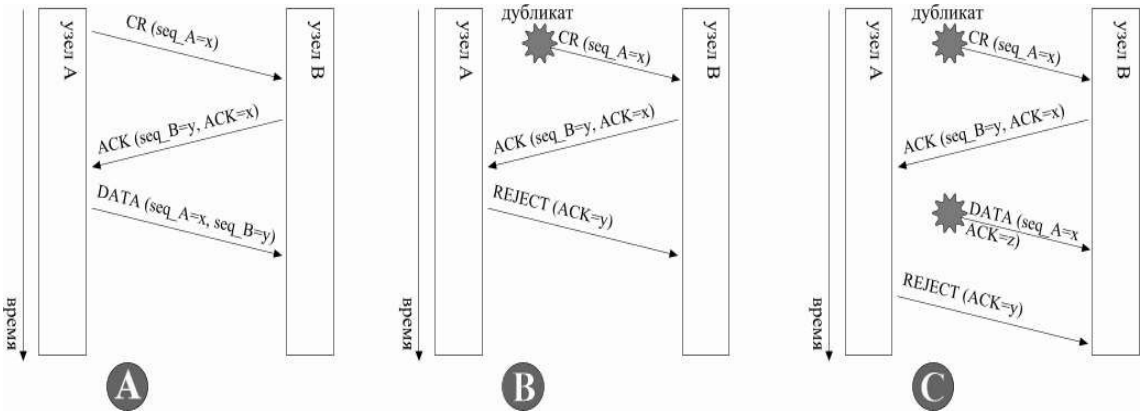

Рисунок 1.10 - Установка з'єднання по методу потрійного обміну. Частина A - нормальна установка з'єднання. B - реакція на дублікат запиту. C - реакція на дублікат запиту і даних

Рисунок 1.11 ілюструє процес закриття з'єднання з використанням таймерів. Незважаючи на відносну надійність такого протоколу, він може не спрацювати у випадку втрати початкового DR і всіх *N* повторних передач. В цьому випадку ініціює сторона закриє з'єднання, протилежна ж не отримавши інформації про закриття буде активною. Такий стан називається напіввідкрите з'єднання. Для запобігання виникнення таких станів вводиться правило, за яким з'єднання буде автоматично закриватися, якщо протягом певного часу на ньому не зареєстровано жодної активності.

Частина A - сценарій без помилкових ситуацій. B, C і D - різні сценарії виникнення помилок. Втрачений сегмент зображується зірочкою.

Якщо ж з'єднання повинно залишатися відкритим протягом тривалого часу, чекаючи появи даних для передачі, то сторонам доведеться періодично обмінюватися порожніми повідомленнями, які утримують протилежну сторону від закриття.

Управління потоком і буферизація даних

Управління потоком безпосередньо пов'язано з буферизацією. Завданням процесу управління потоком є не допустити переповнення одержувача і проміжних вузлів інформацією, яку ті не встигали б обробляти. У тому випадку, якщо БМ надає ненадійний Датаграммним сервіс, то відправник повинен буферізовивать надіслані повідомлення на випадок виникнення необхідності повторної відправки. Одержувач, знаючи, що повідомлення залишаються в пам'яті відправника поки не надійде підтвердження їх прийому, може виділяти чи не виділяти буферне простір для кожного з'єднання. У разі буферизації повідомлень у одержувача, що дозволяє істотно підвищити продуктивність системи, виникає питання про схему виділення пам'яті під буфери. Це може бути і система буферів рівних розмірів (якщо всі повідомлення приблизно одного розміру) або великий циркулярний буфер або набір буферів різних розмірів.

Оптимальне рішення щодо буферизації у відправника і одержувача залежить від характеристик потоку. Наприклад, для трафіку породженого інтерактивної роботою з віддаленим терміналом, що характеризується невеликою швидкістю, краще взагалі не виділяти буферів, а одержувати і передавати інформацію додаткам по мірі її надходження (природно, що передане повідомлення має залишатися в пам'яті відправника до отримання підтвердження). З іншого боку для потоку створеного не інтерактивної передачею даних, наприклад - завантаженням віддаленого файлу, ефективність може бути істотно підвищена, якщо одержувач виділить максимальну кількість буферного простору, щоб якомога більше підвищити швидкість передачі.

Таким чином, з плином часу відправник та одержувач повинні узгоджувати свої схеми виділення буферного простору. Відправник повинен мати можливість запитувати одержувача про виділення певної кількості буферного простору або ж одержувач (не має інформації про потреби відправника) повинен повідомляти, що для з'єднання зарезервовано *X* буферів.

Виділення буферів повинно відбуватися незалежно від приходу підтверджень (на відміну від більшості протоколів канального рівня). Ця схема передбачає наявність «Вікна» змінного розміру. Одержувач виділяє таку кількість буферного простору у відповідь на запит відправника, яке дозволяють його ресурси. Кожен раз, посилаючи повідомлення, відправник повинен зменшити вказану кількість і тимчасово припинити передачу, якщо розмір вільного буферного простору в одержувача зменшиться до нуля. Одержувач, в свою чергу повинен вказувати поточний розмір вікна і номер підтвердження в потоці йде в зворотному напрямку. При збільшенні кількості пам'яті доступної для буферизації мережевих з'єднань вже не недолік вільного буферного простору буде обмежувати максимальну швидкість передачі даних, а доступна пропускна здатність БС. Таким чином, необхідний механізм управління потоком, який враховує також і несучу здатність БС, а не тільки кількість буферного простору у кінцевих учасників обміну. Якщо використовується

механізм обмежує вікна, то його розмір повинен відображати поточну пропускну здатність БС.

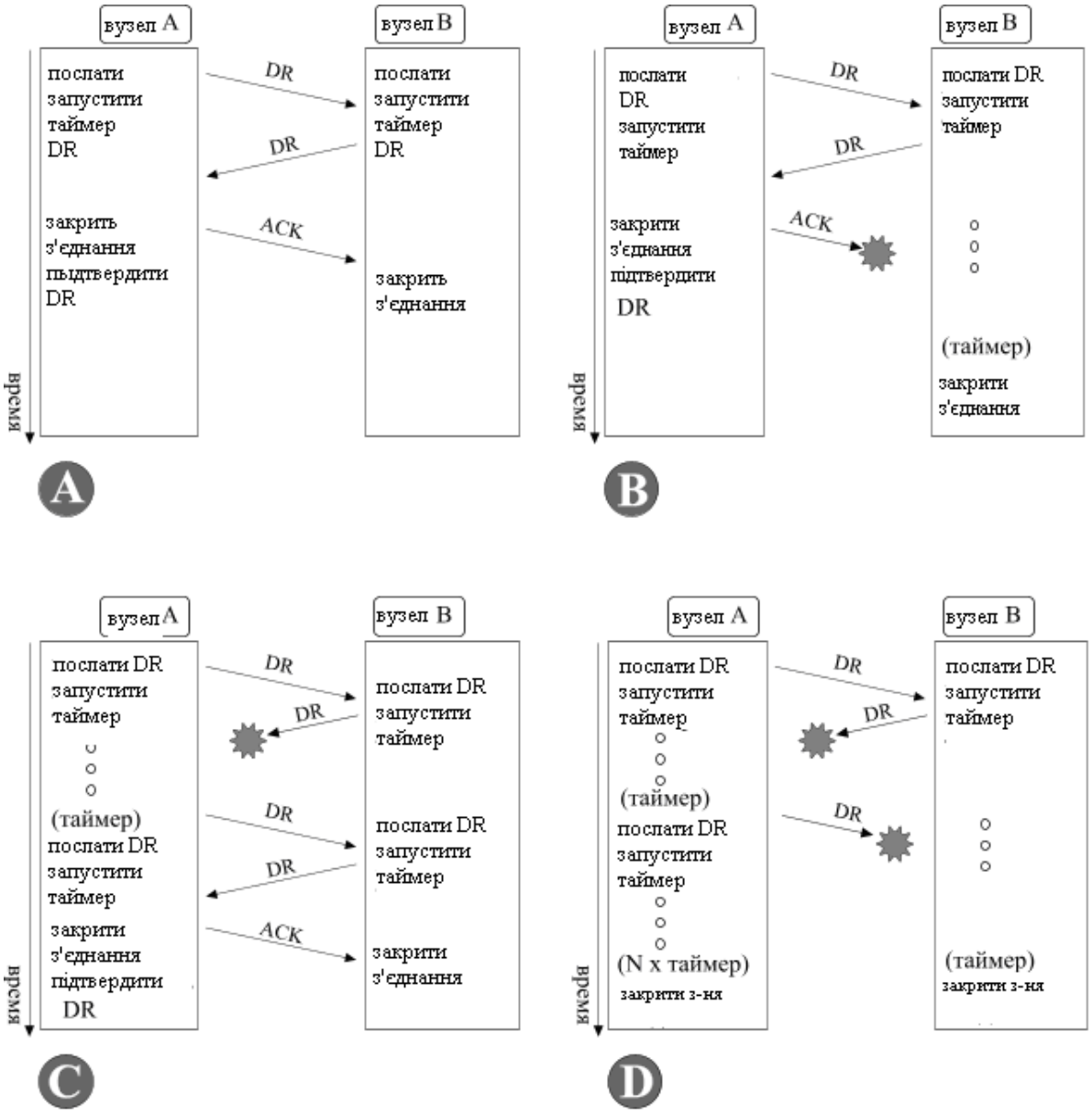

Рисунок 1.11 - Сценарії розриву з'єднання з використанням підтверджень і таймерів

1.1.7 Протокол TCP

Сімейство протоколів TCP/IP включає в себе два протоколи транспортного рівня це TCP - Transmission Control Protocol - і UDP User Datagram Protocol. UDP досить простий, по суті це невелике доповнення до IP, і

не використовує віртуальний канал. TCP це надійний протокол, який використовує попередню установку віртуального каналу і доставляє дані користувача без помилок. TCP був спеціально розроблений, щоб забезпечувати надійну передачу байтового потоку від відправника до одержувача через ненадійну БС. БС може складатися з різних мережевих компонентів, що володіють різними топологічними схемами, технологіями, пропускною здатністю, затримками.

Формально протокол TCP був визначений в RFC 793 [3]. З часом накопичилися помилки і недоліки були враховані і знайшли вираз у вимогах до пристроїв інтернет в RFC 1122 [15]. Надалі було проведено велику кількість досліджень з протоколом. Частина модифікацій стала стандартом і нові специфікації протоколу міститься в RFC 1323 [16].

Кожен пристрій, що підтримує TCP, містить в собі об'єкт протоколу, який управляє TCP потоками і взаємодіє з рівнем IP. Цей об'єкт приймає потоки даних від локальних процесів, розбиває їх на повідомлення певної довжини (сегменти) і передає сегмент на рівень IP для його передачі у вигляді окремого IP пакета. Після прибуття на адресуються машину інформація з цих пакетів передається TCP об'єкту, який реконструює оригінальний байтовий потік і передає її користувальницької програмі. Рівень IP не гарантує доставку пакетів, тому TCP повинен відстежувати втрачені пакети і здійснювати повторну відправку. Пакети можуть прибувати і в також є реконструкція порядку в бітовому потоці. Для управління потоком та запобігання перевантажень БМ TCP застосовує механізм ковзаючого вікна [9].

1.1.7.1 Модель обслуговування TCP

Доступ до сервісу TCP можна отримати шляхом створення на кінцевих машинах точок доступу (Sockets). Кожна така точка має адресу, що складається з IP адреси машини та 16-ти бітного номера, унікального в межах пристрої, що ідентифікує порт. TCP з'єднання встановлюється між двома точками доступу і пара транспортних адрес однозначно ідентифікує з'єднання. Кілька сполук

можуть закінчуватися на одній точці доступу. Порти з номерами нижче 256 однозначно відповідають набору конкретних додатків. Порти з іншими номерами виділяються додаткам динамічно. Всі TCP з'єднання повнодуплексні і мають тип точка-точка. З'єднання є неструктурований байтовий потік, тобто кордону між повідомленнями не зберігаються.

Коли додаток передає дані TCP, протокол може або відразу здійснити їх відправку, або накопичити достатню їх кількість в буфері. Якщо ж додатком необхідно надіслати інформацію миттєво, то воно може використовувати прапор PUSH. Крім того, існує можливість пересилати термінову інформацію з використанням прапора URGENT. Коли додаток встановлює цей прапор, то TCP передає всі наявні дані без зволікання, крім того, TCP одержувача генерує переривання для програми якій призначається термінова інформація і вказує на становище її в отриманому потоці.

Кожен байт TCP з'єднання має унікальний 32-х бітний порядковий номер. Ці номери використовуються не тільки для підтверджень, але і для механізму управління вікном. Об'єкти TCP обмінюються інформацією у вигляді сегментів. Сегмент складається з 20-ти байтного заголовка (+ необов'язкова частина) і нуля або більше байтів даних рисунок 1.8. Протокол сам приймає рішення щодо розміру сегмента. Існують два обмеження розміру сегмента. Перше: сегмент повинен уміщатися в межах максимального розміру поля корисного навантаження IP -216 байт. Друге: кожна мережна технологія має так звану максимальну одиницю передачі (Maximum Transfer Unit - MTU) в яку і повинен вміститися повний IP пакет, що включає в себе IP і TCP заголовки і дані.

Одночасно з передачею сегмента запускається таймер повторної передачі. Коли сегмент прибуває до одержувача, TCP об'єкт останнього відправляє сегмент, який містить підтвердження прийому цього сегмента. Якщо таймер повторної передачі для сегмента спрацьовує раніше часу приходу підтвердження, то даний сегмент ретранслюється.

1.1.7.2 Формат заголовка TCP.

Заголовок сегмента TCP містить наступні поля рисунок 1.8:

Порядковий номер ідентифікує перший байт даних в цьому пакеті, може також використовуватися для вказівки першого номера в серії який буде використовуватися в наступних передачах.

Номер підтвердження - містить порядковий номер наступного байта даних, який очікується на приймаючому пристрої.

Прапори - різноманітна контрольна інформація:

URG - термінові дані, покажчик на термінові дані в цьому випадку вказує на їх положення в сегменті

ACK - 1 означає, що поле підтвердження містить правильну інформацію PSH - одержувач повинен доставити дані додатком якомога швидше RST - з'єднання повинно бути закрито

SYN - використовується для установки з'єднання, запит на установку містить SYN = 1, ACK = 0, відповідь на цей сегмент містить SYN = 1, ACK = 1.

FIN - відправник сигналізує про бажання закрити з'єднання. Після відправки FIN машина може ще необмежено довго продовжувати отримувати інформацію. SYN і FIN сегменти мають порядкові номери, що гарантує їх обробку в правильному порядку.

Розмір вікна повідомляє про те, як багато байтів можна відправити, починаючи з останнього підтвердженого байта. Цей розмір може бути дорівнює нулю.

Управління з'єднанням

Установка з'єднання в TCP використовує процедуру тристороннього обміну. Для відкриття з'єднання пасивна сторона виконує примітиви LISTEN і ACCEPT, а інша відкриває з'єднання в активному режимі, виконуючи примітив CONNECT. Коли запит на відкриття з'єднання, прибуває до одержувача, той перевіряє, чи є процес зареєстрований для прийому з'єднань за вказаною порту. Якщо такий процес існує, то з'єднання встановлюється рисунок 1.12 частина А.

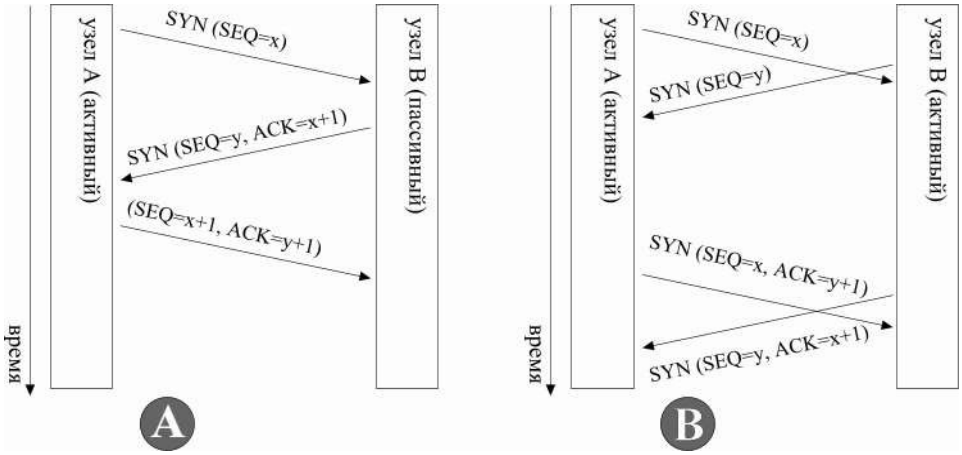

Рисунок 1.12 – Сценарії установки з'єднання TCP. Частина A - нормальна процедура, частина B - установка з'єднання при зіткненні запитів

Хоча TCP з'єднання повнодуплексні, закриваються з'єднання в кожному напрямку окремо. У нормальній ситуації потрібно чотири сегмента, щоб повністю закрити з'єднання (два FIN і два ACK). По кожному з симплексних з'єднань дані можуть передаватися необмежено довго. Для запобігання появи напіввідкритих з'єднань використовуються кілька таймерів. Якщо відповідь не сегмент FIN не приходить протягом двох максимальних періодів життя пакета, відправник сегмента FIN розриває з'єднання.

Управління передачею

Управління вікном в TCP не прив'язане безпосередньо до підтвердженням (як у багатьох протоколу канального рівня).

Підтвердження, що використовуються протоколом TCP, є кумулятивними, тобто кожне підтвердження відповідає останньому отриманому в правильному порядку байту потоку. Прихід підтвердження з номером N означає, що всі байти [0 ... N-1] були успішно доставлені отримувачу.

Управління вікном і тактика підтверджень

На ранніх стадіях експлуатації протоколу було відмічено, що часто в мережах виникали перевантаження при дуже низькому ефективному використанні ресурсів [17]. Були виявлені три незалежні причини цього: програма викликають протокол для відправки незначної кількості інформації, оновлення вікна TCP дуже маленькими порціями і викликають відповідні сегменти з малою корисним навантаженням, і сам протокол посилає занадто багато підтверджень. Було запропоновано використовувати так званий алгоритм SWS [15] для вирішення проблеми з оновленням вікна.

А Алгоритм SWS у одержувача, поєднаний із затримкою підтвердження наказує: оновлення вікна разом з підтвердженням відправляється тільки якщо звільнився обсяг буфера перевершує певну частку загального буферного простору (30-50%) або 2-3 максимальних розміру сегмента.

Б Алгоритм SWS у відправника наказує: відправник намагається приблизно оцінити розмір буфера одержувача, відстежуючи максимальний розмір вікна відкривається одержувачем на даному з'єднанні. max (SND.WND). Використовуване вікно, дані в межах якого можуть бути відправлені, обчислюється U = SND.UNA + SND.WND - SND.NXT. Якщо D кількість даних буферизують протоколом, але ще не відправлених, то TCP відправляє дані якщо:

– є можливість послати сегмент максимального розміру min (D, U)> = Eff.snd.MSS;

– встановлений прапор PUSH і [SND.NXT = SND.UNA] && PUSHED && D <= U [SND.NXT = SND.UNA] умова накладається алгоритмом Nagle [8];

– як мінімум частка Fs від максимального вікна може бути відправлена [SND.NXT = SND.UNA] && min (D, U)> = Fs * Max (SND.WND);

– встановлений прапор PUSH і спрацював таймер.

В Алгоритм Nagle: ефективний коли додаток передає малі обсяги інформації. Згідно з цим алгоритмом, якщо є непідтверджені дані (SND.NXT> SND.UNA), то TCP відправника буферизует дані без відправки, незважаючи на

наявність прапора PUSH, до моменту, коли або приходить підтвердження, або стає можливим відправити сегмент максимального розміру.

Г Алгоритм затримки підтверджень вимагає:

Підтвердження повинно затримуватися не менше ніж на 0.5 с. Підтвердження затримуються до тих пір, поки не виникне можливість вказати нове вікно або дані відправляються в протилежному напрямку. Таймер генерує безумовне підтвердження кожні 200 мс. Крім того, кожен другий сегмент повинен підтверджуватися. Також сегменти, буферізованние в одержувача (прибули в неправильній послідовності) генерують підтвердження миттєво для включення механізму швидкої ретрансляції.

Управління потоком і запобігання перевантажень

Якщо сумарна швидкість надходження даних в на мережу перевищує максимальну швидкість, з якою мережа передає ці дані, то виникає перевантаження мережі. Якщо стан перевантаження триває протягом часу, достатнього для переповнення буферів в мережі, то відбуваються втрати даних в мережі. Втрачені сегменти ретранслюються протоколом TCP і сприяють продовженню зростання перевантаження. Алгоритм керування потоком транспортного протоколу повинен запобігати виникненню перевантаження і сприяти виходу мережі з станів перевантаження, в разі його виникнення.

Перевантаження мережі може бути відвернена за допомогою використання принципу збереження пакетів [18] - якщо мережа завантажена повністю, то не передавати наступний пакет раніше, ніж попередній покине мережу (тобто буде доставлений за призначенням). TCP прагне слідувати цьому принципу, змінюючи розмір вікна.

Схема визначення настання стану перевантаження в TCP грунтується на припущенні, що причина всіх втрат пакетів полягає в переповненні черг маршрутизаторів, тобто перевантаження мережі. Таким чином, алгоритм управління потоком в TCP вважає втрати пов'язані з руйнуванням даних через шум на лінії несуттєвими і реагує на всі втрати як на ознаку перевантаження.

Такий механізм є неефективним, особливо в разі бездротових з'єднань, таких як радіо та супутникові канали, ІК приймач.

TCP оперує двома вікнами, одне визначає розмір буфера відправника, інше (CWND) є результатом спроб протоколу визначити пропускну здатність мережі на даному з'єднанні. Для відправки даних використовується мінімальна з двох вікон.

Механізми Slow start, congestion avoidance, fast retransmit, fast recovery [6] є стандартом для сучасних реалізацій TCP.

Slow start (уповільнений запуск) застосовується для поступового збільшення швидкості передачі інформації після відкриття з'єднання або втрати пакета. Спочатку розмір вікна CWND встановлюється рівним одному сегменту. Після отримання кожного підтвердження CWND збільшується на розмір одного сегмента. Алгоритм повільного запуску відключається коли відбувається втрата сегмента або досягається верхня граніцаокна одержувача. Час, що витрачається для повного відкриття вікна

$$t = RTT * \log 2W, \quad (1.1)$$

де W – розмір вікна в сегментах.

Congestion avoidance (запобігання перевантаження) використовується для визначення наявності додаткової пропускної здатності. Зазвичай цей алгоритм діє спільно з алгоритмом повільного старту. Для фіксації моменту переходу з повільного старту на запобігання перевантаження використовують змінну SSTHRESH. SSTHRESH ініціалізується розміром максимально можливого вікна при відкритті нового з'єднання. Повільний старт триває поки значення CWND не досягне SSTHRESH, після цього режим зростання CWND змінюється на лінійний, а саме, для кожного підтвердження CWND збільшується на значення (1 / CWND).

Fast Retransmit (швидка ретрансляція). Використовує послідовне отримання підтвердження одного і того ж сегмента, як ознака втрати, ретранслює цей сегмент до спрацьовування таймера ретрансляції.

Fast recovery (швидке відновлення). Після спрацьовування механізму швидкої ретрансляції TCP переходить в режим запобігання перевантаження, а не повільного старту. Після приходить третє дубліката підтвердження SSTHRESH встановлюється рівній $\frac{1}{2}$ CWND, і відсутній сегмент ретранслюється, CWND встановлюється рівним SSTHRESH +3 (кількість сегментів покинули мережу). Потім на кожне підтвердження CWND збільшується на розмір одного сегмента. Якщо розмір CWND дозволяє, передаються додаткові сегменти. Коли приходить підтвердження, що включає в себе ретранслюють сегмент, CWND встановлюється рівним SSTHRESH і TCP переходить в режим запобігання перевантаження. Описані вище механізми є стандартною частиною протоколу TCP, їх застосування регламентується RFC 1323 [16].

Управління таймерами

Об'єкти TCP концептуально використовують кілька таймерів для виконання своїх функцій. Найбільш важливим з них є таймер повторної передачі (ТПП). ТПП перезапускається кожен раз при відправленні сегмента. Якщо підтвердження прийому сегмента не приходить до спрацьовування таймера, то цей сегмент ретранслюється.

Визначення сумарного часу витрачається сегментом на досягнення одержувача і часу витрачається підтвердженням, щоб повернутися до відправника (RTT) само по собі досить складно в реалізації, ще складніше навіть знаючи цей час правильно встановити інтервал таймера. Якщо ТПП встановлений на занадто короткий інтервал, то повторна передача відбудеться ще до того як може прибути підтвердження і це ще більше збільшить ступінь завантаження БМ. Якщо ж таймер встановлений з дуже тривалу затримку, то

деяку частину часу канал буде використовуватися неефективно. Крім того, середнє значення вимірюваного RTT і його дисперсії вкрай динамічно змінюється з плином часу в залежності від ступеня завантаження БС.

У протоколі TCP використовується високо динамічний алгоритм, який безперервно коригує значення ТПП, відстежуючи час звернення сегментів в мережі. Цей алгоритм функціонує наступним чином [18]: для кожного з'єднання TCP зберігає значення найкращої поточної оцінки часу обігу сегментів у змінній RTT. При відправці сегмента встановлюється таймер для контролю повторного відправлення і для вимірювання RTT. Якщо підтвердження прибуває до спрацьовування таймера, то, вимірюючи час, який минув до приходу підтвердження (M), TCP оновлює середнє значення RTT за формулою:

$$RTT * \alpha * RTT * (1 - \alpha) * M, \quad (1.2)$$

де α ваговий коефіцієнт застосовується до старого значенням (з типовим значенням 7/8).

Для того, щоб значення RTT не спотворювалося через обліку ретранслюють сегментів, Карно [19] був запропонований алгоритм, який додав до TCP наступне правило: не оновлювати значення RTT за даними, які належать до ретранслюють сегментам, натомість значення RTT подвоюється з кожною ретрансляцією . Цей принцип отримав назву "Exponential Backoff".

1.2 Управління трафіком

1.2.1 Загальні принципи

Управління потоками в комунікаційних мережах позначає регулювання швидкості відправки даних в мережу з метою досягнення максимального використання ресурсу мережі та мінімізації втрат даних.

До складу протоколу TCP входить велика кількість різноманітних механізмів, які відповідають за різну функціональність і знаходяться в певному взаємодії між собою в силу концептуально-необхідного зв'язку або зв'язку, що виникла при реалізації. На рисунок 1.13 наведена схема основних механізмів,

що входять до складу найпоширенішою реалізації протоколу TCP - Reno в складі 4.4 BSD UNIX [54].

Задачу управління швидкістю передачі даних можна умовно розбити на два компоненти: не допущення переповнення приймаючої сторони, і не допущення переповнення мережі. Іншими словами, метою системи управління потоком є вирівнювання швидкості передачі даних зі швидкістю їх прийому. Механізм контролю перевантаження відправляє дані в мережу не швидше, ніж мережа може їх доставляти в місце призначення і не швидше, ніж одержувач може їх обробляти.

Рисунок 1.13 - Набір функцій і алгоритмів у складі стандартного протоколу TCP

Перевантаження в мережі з комутацією пакетів це такий стан, при якому продуктивність системи падає у зв'язку з переповненням ресурсів мережі - комунікаційних каналів, процесорного часу і буферного простору. Наслідки перевантаження проявляються в збільшенні часу доставки даних, падінні

ефективності використання мережевих ресурсів і так званому мережевому колапсі, коли процес передачі корисних даних в мережі повністю припиняється.

Вивчення систем управління потоками і запобігання перевантажень, дуже важливо для розвитку мереж. З безлічі запропонованих схем лише кілька стали стандартами та отримали широке поширення: IBM Systems Networking Architecture (SNA) [20], Digital's Networking Architecture (DNA) [21], і модель TCP/IP [22]. Найбільш повна класифікація різних методик управління потоком наводиться в [23].

Перевантаження і методи її контролю в мережах з комутацією пакетів

Перевантаження є проблемою неефективного спільного використання поділюваних ресурсів. У мережі ресурси розподілені між усіма вузлами, комутаторами і каналами передачі даних. Будь-який з цих трьох компонентів може стати вузьким місцем у мережі і викликати її перевантаження. З одного боку мережа повинна обслуговувати всі користувальницькі запити на передачу даних, які, як правило, не можуть бути передбачені детерминистически і виражаються у вигляді сплесків по відношенню до часу виникнення, швидкості і об'єму переданої інформації. З іншого боку всі фізичні ресурси мережі мають обмежену потужність і повинні бути керованими для досягнення оптимального загального використання багатьма сесіями обміну даними. Більш формальний опис стану перевантаження мережі можна отримати розглянувши продуктивність її роботи залежно від навантаження рисунок 1.14.

Після досягнення мережею насиченого стану (точка А) пропускна здатність перестає рости, а час відповіді (RTT) продовжує, оскільки відбувається заповнення буферів всередині мережевих пристроїв. Потужність мережі досягає піку в точці А - це оптимальний режим роботи мережі.

Механізми управління потоками повинні утримувати режим роботи мережі на оптимальному значенні навантаження і забезпечувати можливість виходу зі стану перевантаження, якщо перевантаження таки відбулася. Механізм управління потоками в мережах TCP/IP забезпечується як самими

кінцевими системами, в яких виконуються об'єкти протоколу TCP, так і за допомогою певних механізмів управління чергами та диспетчеризації потоків в маршрутизаторах.

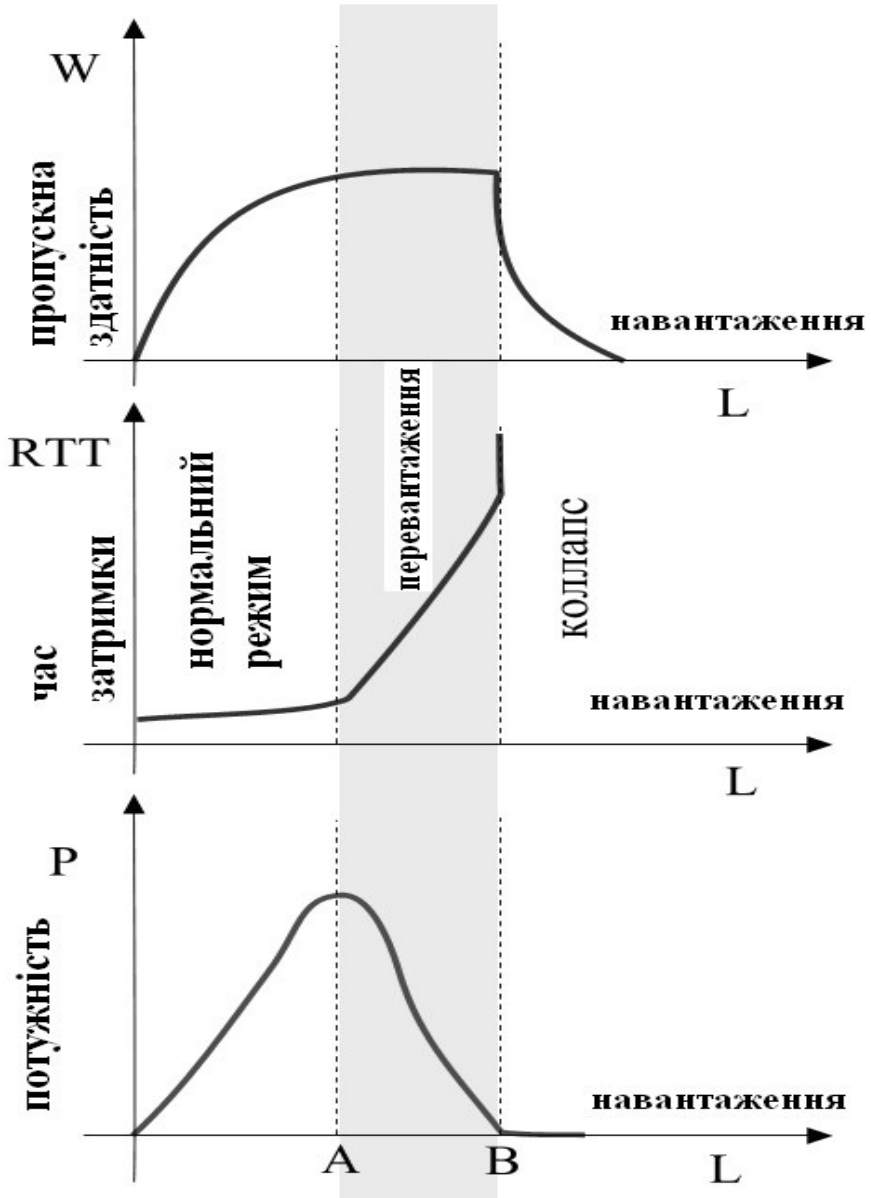

Рисунок 1.14 - Залежність характеристик мережі від навантаження. Пропускна спроможність (у горі), час затримки (в середині), потужність мережі (внизу)

1.2.2 Управління затримкою мережі

Затримка даних при їх передачі по мережі складається з двох компонентів. Для передачі мультимедіа інформації дуже важливо щоб значення затримки і її дисперсія для індивідуальних пакетів були мінімальними. Ця вимога виражається у вимозі мінімальності черг в маршрутизаторах.

Оскільки TCP в процесі передачі цілком заповнює чергу маршрутизатора, то за рахунок компонента Dq загальна затримка в мережі істотно зростає. Тому в мережах, де величина затримка актуальна, крім управління швидкістю засобами протоколу транспортного рівня необхідно використовувати механізми, розташовані в маршрутизаторах [23].

Алгоритм RED

Оскільки засобами одного протоколу TCP забезпечити мінімальність середньої довжини черг в мережевих пристроях неможливо, то в схемі управління трафіком повинні бути задіяні самі мережеві пристрої. Для реалізації такого управління використовується алгоритм випадкового раннього виявлення (Random Early Detection - RED) [24].

Алгоритм RED застосовується для запобігання перевантажень в мережах передачі даних працюють за принципом комутації пакетів. Маршрутизатор визначає близькість стану перевантаження, обчислюючи середню довжину черги. Маршрутизатор оповіщає з'єднання, що проходять через нього, про стан перевантаження шляхом відкидання пакетів. Ймовірність відкидання залежить від середньої зваженої довжини черги, поки її значення знаходиться в заданих межах і стає рівною одиниці при перетині верхньої межі.

Маршрутизатор застосовує алгоритм RED дозволяє утримувати низьку середню довжину черги, при цьому допускаючи прийняття в чергу сплески пакетів. В режимі перевантаження ймовірність оповіщення певного транспортного з'єднання, шляхом відкидання або відмітки пакета, з метою зменшення вікна контролю потоку для цього з'єднання, пропорційна частці цього з'єднання в загальній пропускної здатності каналу. Алгоритм RED

призначений для роботи з транспортним протоколом, який управляє своєю пропускною здатністю, реагуючи на втрати пакетів, які протокол вважає ознакою перевантаження, зниженням своєї швидкості передачі. RED не допускає дискримінації нерівномірних потоків і призводить до глобальної синхронізації багатьох сполук.

Однак застосування алгоритму RED в маршрутизаторах ускладнює їх і збільшує їх вартість.

Альтернативні схеми управління потоком

TCP VEGAS

Vegas представляє собою поліпшену реалізацію протоколу TCP в порівнянні з поширеними реалізаціями Tahoe і Reno. Поліпшення, запропоновані в системі Vegas

В роботах [25, 26], зачіпають в основному механізм швидкої ретрансляції. В алгоритмі Vegas використовується більш точний спосіб визначення необхідності швидкої ретрансляції втрачених сегментів. Цей алгоритм використовує значення тимчасової мітки у заголовку сегментів.

Таким чином, Vegas краще, ніж TCP, реагує на ситуації в яких відбувається втрата більш ніж одного сегмента в межах вікна, і з якими не може ефективно боротися стандартний механізм прискореної ретрансляції [27, 28, 29]. Також в рамках цього механізму Vegas більш точно управляє розміром вікна в процесі ретрансляції. Для реалізації контролю перевантаження алгоритм Vegas зберігає значення мінімального RTT у змінній BaseRTT = min (BaseRTT, RTT). Очікуване значення пропускної здатності визначається за формулою Expected = размер_окна/BaseRTT. Далі, реальна пропускна здатність періодично обчислюється як кількість байт переданих з моменту відправки певного сегмента і до приходу його підтвердження. Різниця реальної й очікуваної пропускної спроможності визначає вибір режиму лінійного збільшення або зменшення вікна.

Оскільки основний алгоритм Vegas повністю взятий від TCP, то всі основні недоліки, притаманні TCP, характерні і для TCP Vegas.

TRUMP

Даний механізм розроблений в докторській дисертації В.К. Тумей [30] і в нього входить не тільки специфікація транспортного протоколу, а й ціла система управління потоком на мережевому рівні. Ця система включає в себе механізм вимірювання завантаження вузлів з явною реверсивної зворотним зв'язком і власний формат даних. Цікавим аспектом системи TRUMP є те, що швидкість відправки даних в мережу регулюється не ковзним вікном, а значенням дозволеної швидкості, що задається мережевими пристроями. До недоліків системи потрібно віднести той факт, що системи з явною передачею контрольної інформації існували й раніше [31, 32, 33]. Спроби реально впровадити такі системи для мережі Інтернет не мали успіху, оскільки це вимагало б введення додаткових функцій маршрутизаторів, в той час як TRUMP передбачає не тільки суттєва зміна мережевих пристроїв, але і повна відсутність сумісності з архітектурою TCP/IP.

PP (Packet Pair Flow Control)

Управління потоком за методом Packet Pair Flow Control [34] являє собою оригінальну ідею вимірювання доступної пропускної здатності мережі. Схема призначена для мереж з віртуальними каналами, які ізолюють транспортні потоки за допомогою диспетчера квотовано обслуговування (ДКО) черг [35, 36, 37]. Мотивація для створення такої схеми управління потоками в тому, що надання інтегрованих сервісів стає домінуючим напрямком у сучасній мережевий парадигмі. Відповідно до неї мережі повинні одночасно обслуговувати стабільні потоки з постійною швидкістю передачі даних одночасно з потоками, робота яких характеризується сплесками і нерівномірністю. Перший тип обмінів повинен отримувати обслуговування від мережі з гарантованою якістю, а другий тип трафіку використовувати ресурси, що залишилися. Це основна концепція побудови мереж з інтеграцією сервісів

[30]. Для її реалізації, міжмережеві пристрої повинні застосовувати один з алгоритмів рівноправного обслуговування черг. Як правило це алгоритм зваженої рівноправній диспетчеризації WFQ [36].

Робота по PP на додаток до механізму визначення пропускної здатності, описує оціночну функцію для згладжування отриманих вимірювань, функцію управління, метод управління з'єднанням, методи управління потоком і передачею. Пристрій, де виповнюється алгоритм (ДКО), називають сервером пропускної здатності (СПЗ). В такому пристрої пакети кожного окремого транспортного потоку поміщаються в окрему чергу на вихідному інтерфейсі.

Алгоритм PP не підходить для використання в мережах TCP/IP, оскільки в таких мережах відсутній логічна ізоляція потоків на мережному рівні. Однак PP цілком може знайти застосування в мережах ATM, при наявності роздільної буферизації кожного з віртуальних каналів.

NETBLT

Протокол NETBLT [39, 40] розроблений з метою надання користувачеві сервісу аналогічного сервісу протоколу TCP, а саме надійної передачі великого об'єму інформації з підтвердженнями і контролем швидкості потоку. У протоколі NETBLT представлено декілька принципово нових ідей.

По-перше, швидкість потоку регулюється заданої одержувачем величиною швидкості. Передача здійснюється з попередньо сформованих банків інформації. Після передачі кожного банку (які можуть передаватися незалежно), одержувач підтверджує коректно отримані дані або запитує вибіркову ретрансляцію втрачених пакетів. Також в залежності від спостережуваного рівня втрат NETBLT встановлює нову швидкість передачі.

По-друге, сама передача регулюється кількома параметрами - частотою і тривалістю сплесків, кожен з яких може складатися з декількох пакетів. За рахунок цього підвищується ефективність використання системних ресурсів, зокрема зменшується число переривань використовуються для диспетчеризації передачі. Цей підхід можна застосувати і для протоколу ARTCP.

Таким чином, протокол NETBLT здатний забезпечити гарні результати при роботі по каналах, що характеризується великим значенням RTT * BW. Однак NETBLT не вільний і від недоліків властивих TCP, наприклад обидва протоколи інтерпретують втрату пакета як ознака перевантаження, що невірно в загальному випадку.

Tri-S

Алгоритм Slow Start and Search (Tri-S) [41] заснований на механізмі уповільненої старту протоколу TCP. Автори проаналізували недоліки уповільненої старту і спробували усунути неприйнятно високу амплітуду осциляцій розміру вікна, яка призводить до великої кількості втрат і значного збільшення компонента затримки.

За результатами моделювання схема Tri-S дозволяє поліпшити показник справедливості розподілу ПС в порівнянні з TCP. Однак вибір граничних значень надає сильний вплив на стабільність системи. Крім того, як і стандартний TCP, схема Tri-S інтерпретує втрату пакета як ознака перевантаження.

DUAL

Даний метод був запропонований як спосіб усунення проблем з осциляцією вікна, які спостерігаються з алгоритмом уповільненої старту [36]. DUAL використовує зміну вимірюваного RTT разом з відображенням втрат пакетів для визначення перевантаження.

Висновки

Кожен з цих методів має певними недоліками і не застосовується в стандартних протоколах. Головний недолік Vegas, Tri-S, DUAL в тому, що в них використовується протокол TCP, який реагує на втрату сегмента зниженням швидкості, а у відсутності втрат лінійно збільшує навантаження на мережу, приводячи до переповнення буферів. Схема TRUMP передбачає розширення функціональності маршрутизаторів механізмом, що повідомляють

джерела про перевантаження в явному вигляді, що означає відхід від парадигми мереж з пакетною комутацією. PP пропонує ефективний спосіб вимірювання завантаження мережі, однак непридатний для TCP/IP, оскільки передбачає роботу в мережі з віртуальними каналами на мережевому рівні.

Принципи AIMD і STA

Більшість алгоритмів керування потоками розрізняються способом визначення настання стану перевантаження і реагують на зміни стану мережі шляхом аддитивного (лінійного) збільшення навантаження і мультиплікативного скидання, у разі визначення настання перевантаження. В літературі такий підхід отримав назву Addititve Increase and Multiplicative Decrease (AIMD). Обгрунтування застосування принципу AIMD викладені в роботі [18].

У разі відсутності перевантаження, мережа ніяк не повідомляє з'єднанням про наявність додаткових ресурсів, у зв'язку з цим, за відсутності сигналів про перевантаження, з'єднання повинні збільшувати навантаження, до тих пір, поки втрата пакет не повідомить про початок перевантаження. Експоненціальне зростання завжди призводить до нової перевантаження, тому автор [18,55, 56] рекомендує лінійний ріст або адитивна збільшення.

1.3 Недоліки протоколу TCP

До найбільш істотних недоліків протоколу TCP в галузі управління потоками відноситься наступне:

1 До основному недоліку протоколу TCP слід віднести проблему не стільки реалізації протоколу, скільки самої логіки функціонування. Так механізми корекції помилок і управління потоком в TCP, що реалізують різні функції, виявилися пов'язаними. Причиною цього є інтерпретація протоколом факту втрати сегмента (і, відповідно, факту спрацьовування механізму корекції помилок) як ознаки перевантаження мережі. Внаслідок цього TCP реагує на будь-яку втрату даних зниженням швидкості передачі. Результатом є вкрай

низька ефективність застосування протоколу TCP для систем, де втрати даних відбуваються не тільки внаслідок перевантаження - всіх систем, де існує ненульова ймовірність втрати даних на фізичному-канальному рівнях. Це, зокрема, всі види бездротових мереж: супутникові, стільникові, оптичні.

2 Пріменяемий в TCP метод AIMD наказує постійне лінійне збільшення навантаження на мережу з метою визначення моменту початку перевантаження. Внаслідок цього мережа постійно знаходиться або в стані перевантаження, або в стані виходу з неї. Це негативно позначається на з'єднаннях у вигляді збільшеного середнього RTT, великий дисперсії вимірюваних значень RTT, постійному наявності втрат, за допомогою яких мережа сигналізує про початок перевантаження.

3 В більшості умов протокол TCP здійснює відправку даних дуже нерівномірно. Фактично всі дані в межах вікна відправляються за невеликий час у вигляді сплеску пакетів [43]. Нерівномірний режим відправки пакетів приводить до підвищення числа втрат. Оскільки середня довжина черг в мережевих пристроях близька до максимальної, то ймовірність втрати сегментів в межах сплеску підвищується.

4 Механізми управління передачею TCP залежить від потоку підтверджень в зворотному напрямку, прибуття яких змушує переміщатися вікно і дозволяє відправку нових сегментів. Такий режим називають синхронізованим по підтвердженням. Його ефект не помітний для симетричних каналів, однак для каналів, чиї властивості в різних напрямах різні синхронізація по підтвердженням веде до обмеження ефективності використання ресурсів. Це відноситься до таких асиметричним системам, як «супутниковий / наземний канали», «кабельний модем / комутоване з'єднання».

Оскільки недоліки TCP широко відомі, то безліч робіт було присвячено створенню окремого транспортного протоколу для асиметричних інфраструктур [48] або бездротових мереж [44, 45, 46, 47]. Більшість пропонованих протоколів не є сумісними з TCP і вимагають наявності агентів-посередників транспортного рівня. Тим самим порушується основний постулат

Інтернет полягає в тому, що мережа повинна забезпечувати безперешкодний зв'язок на транспортному рівні між безпосередніми учасниками обміну. Також зайво ускладнюється вся структура мережі.

Дипломна робота спрямована на створення нового протоколу, названого ARTCP, який міг би стати ефективною заміною існуючого TCP, усунувши найважливіші недоліки останнього.

1.4 Постановка завдання

Завдання дипломної роботи в створенні модифікації механізма управління потоком для транспортного протоколу в архітектурі мережі з комутацією пакетів (TCP/IP). Новий алгоритм управління потоком має виправити основні недоліки, притаманні протоколу TCP наведені вище. При цьому новий протокол (ARTCP) повинен зберегти всі зовнішні властивості та інтерфейси існуючого транспортного протоколу. Новий протокол повинен існувати в типовій середовищі виконання транспортного протоколу Інтернет і надавати користувачеві той же сервіс.

Для вивчення характеристик протоколу ARTCP і порівняння їх з TCP, в рамках даної роботи необхідно розробити програмну імітаційну модель, відтворюючу основні характеристики мережі, які і визначають функціонування в ній транспортного протоколу: затримку, мультиплексування, втрати і помилки передачі.

На створеній імітаційної програмної моделі необхідно провести модельній експеримент для визначення основних параметрів протоколу ARTCP в різних умовах і порівняння їх з TCP.

1.5 Висновок

Для вивчення характеристик протоколу ARTCP і порівняння їх з TCP, в рамках даної роботи необхідно розробити програмну імітаційну модель, відтворюючу основні характеристики мережі, які і визначають функціонування в ній транспортного протоколу: затримку, мультиплексування, втрати і помилки передачі.

На створеній імітаційної програмної моделі необхідно провести модельній експеримент для визначення основних параметрів протоколу ARTCP в різних умовах і порівняння їх з TCP.

РОЗДІЛ 2. СПЕЦІАЛЬНИЙ РОЗДІЛ

2.1 Розробка системи передачі і топології мережі

Формалізуємо модель на якій будемо вивчати протокол ARTCP. Має мережу, в топологічну схему якої входять декілька вузлів, два маршрутизатора і набір каналів з'єднують вузли рисунок 2.1. На кожному вузлі виконується об'єкт протоколу ARTCP. Вузли об'єднані у дві локальні обчислювальні мережі (ЛОМ), кожна з яких підключена до одного маршрутизатора. Маршрутизатори пов'язують локальні мережі передаючи трафіка по каналу з малою ПЗ і великим значенням затримки.

Кожен з вузлів в одній з ЛОМ відправляє сегменти із заданим меж-сегментним інтервалом, який і визначає швидкість передачі. Ця швидкість задається алгоритмом управління потоком ARTCP. Передбачається, що джерела потоків завжди мають інформацію для відправки. Джерела і приймачі трафіку знаходяться в різних ЛОМ. Приймачі трафіку ARTCP відправляють підтвердження в протилежному напрямку у вигляді сегментів, не містять даних.

Завдання маршрутизатора в тому, щоб здійснювати передачу сегмента за адресою отримувача в його заголовку. У буфері маршрутизатора R1 організовується FIFO чергу сегментів, які відправляються далі до маршрутизатора R2. Черга має кінцеву довжину. Сегмент, що надходить на вихідний інтерфейс, поміщається в чергу, якщо вона може вмістити цей сегмент, інакше сегмент втрачається. Черга маршрутизатора R1 обслуговується зі швидкістю каналу, що з'єднує маршрутизатори.

Ми розглядаємо такі характеристики каналів: ПЗ, затримку передачі і ймовірність бітової помилки. Пропускна здатність каналу означає швидкість надходження бітів в канал, затримка передачі характеризує тривалість інтервалу між надходженням певного біта в канал і появою його з каналу. Ймовірність бітової помилки BER визначає ймовірність втрати сегмента як 1 - (1 - BER) S в залежності від імовірності помилки при передачі.

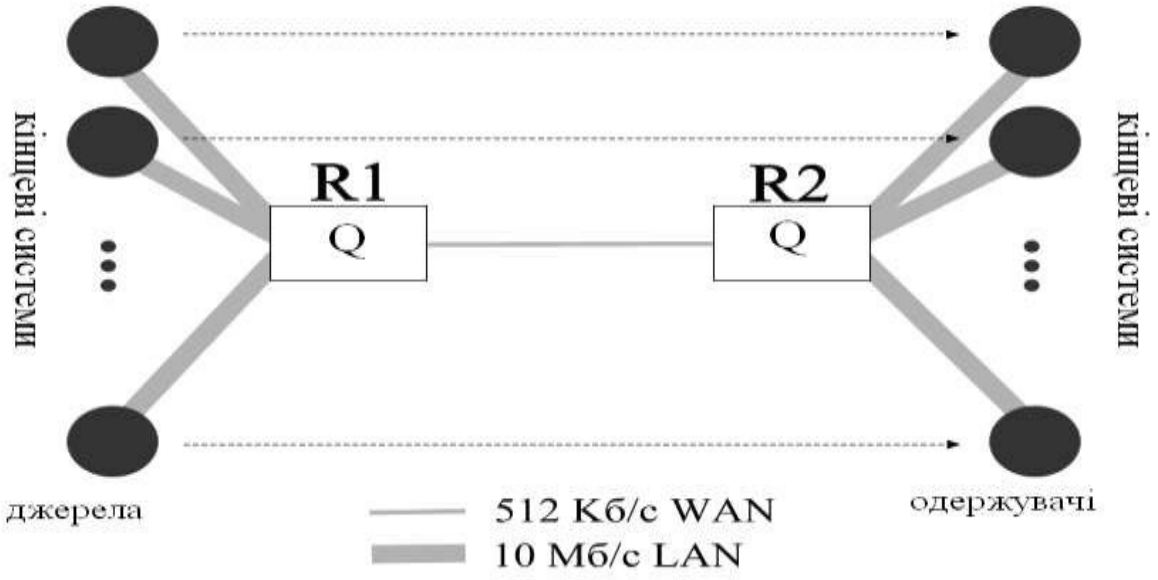

Рисунок 2.1 – Формальна модель досліджуваної мережі. Стрілками вказано напрям передачі даних

2.2 Основні характеристики протоколу

В роботі розглядаються наступні основні характеристики протоколу:

1 Відносне число втрат сегментів (відношення числа втрачених до загальної кількості відправлених сегментів)

2 Коефіцієнт використання пропускної здатності каналів (відношення числа успішно прийнятих бітів до максимально можливого їх числа).

3 Коефіцієнт рівноправності поділу ресурсів,

$$F = (\sum_{i=1}^{n} b_i)^2 \Big/ n \times (\sum_{i=1}^{n} b_i^2), \tag{2.1}$$

де b_i частка пропускної здатності, зайнята i-м з'єднанням.

4 Середня довжина черги Q в маршрутизаторі R1.

Порівняння ARTCP і TCP проводиться за цими основними характеристиками.

Наведені вище характеристики протоколу є стандартними і використовуються в багатьох дослідженнях протоколів [49].

ARTCP відрізняється від стандартного TCP тим, що сегменти відправляються в мережу не в вигляді сплеску в межах вікна, а розділені тимчасовими проміжками, тривалість яких визначається поточним значенням швидкості. Швидкість потоку регулюється не розміром змінного вікна, а значенням швидкості, зміною якої здійснюється адаптація алгоритму відповідно до умов. Механізм ковзаючого вікна в ARTCP застосовується тільки для запобігання переповнення буферів одержувача. На розмір вікна в ARTCP не накладено жодних обмежень ні в одному з режимів роботи. Вікно обмежена лише наявністю буферного простору в одержувача, тому для одержувача рекомендується встановлювати вікно на максимальний розмір.

У випадку, коли одержувач обмежений в буферному просторі, ARTCP буде вести себе як звичайний протокол TCP, обмежений вікном. Таким чином, протокол ARTCP містить в собі метод ковзаючого вікна для регулювання управління потоком з кінця в кінець і метод контролю швидкості для підстроювання під ПЗ проміжних вузлів з'єднання.

Другим найважливішим – ARTCP є те, що в якості сигналу про стан перевантаження або наявності додаткових ресурсів у мережі використовуються темпоральні характеристики потоку - вимірювання шпаруватості потоку в одержувача і зміни часу RTT. Втрата пакета ніяк не відбивається на роботу ARTCP крім здійснення ретрансляції втраченого пакета механізмом корекції помилок. Як і в TCP про втрати пакета повідомляють два можливі події: спрацьовування RTT або послідовне отримання двухподтвержденій одних і тих же даних.

Таким чином, у порівнянні зі своїм попередником, ARTCP володіє наступними перевагами:

– ARTCP не потрібно доводити мережу до стану перевантаження, щоб визначити доступну частку ПЗ, тому виключені втрати пакетів пов'язані з цим процесом;

– ARTCP істотно знижує вимоги до міжмережевих пристроїв. По-перше, для нормального функціонування даного протоколу потрібно менший

обсяг буферного простору, ніж для TCP, оскільки режим передачі є згладженим. По-друге, ARTCP не вимагає і не залежить від наявності будь-яких механізмів диспетчеризації або управління чергами, таких какREDiliWFQ;

– ARTCP не інтерпретує втрату пакета як ознака перевантаження мережі, використовуючи замість цього темпоральні характеристики потоку. Тому ARTCP повинен особливо ефективно працювати в системах бездротового зв'язку, там де використання TCP неефективно;

– На відміну от TCP новий протокол не покладається цілком на потік підтверджень в зворотному напрямку для синхронізації процесу передачі.

У зв'язку з цим можлива реалізація ARTCP з меншою частотою підтверджень, яка не обмежувала б швидкість в асиметричних системах.

2.3 Розробка модифікації транспортного протоколу

Аналіз робіт в області транспортних протоколів і зокрема механізму PP дозволив зробити висновок, що недоліки протоколу TCP вельми істотні і є наслідком самого алгоритма, що лежить в основі протоколу TCP. Тому модифікація TCP без заміни його основних алгоритмів не може привести до істотного поліпшення характеристик протоколу.

Тому в цій роботі було вирішено створити новий алгоритм транспортного протоколу, який залишається, однак, повністю сумісним з архітектури TCP/IP.

Для того щоб усунути недоліки, властиві TCP, необхідно було знайти спосіб отримання інформації про стан мережі, відмінний від застосування в цих цілях втрат сегментів. Найбільш добре на роль індикатора стану мережі підходять тимчасові характеристики потоку: час RTT і межсегментні інтервали. З використанням міжсегментних інтервалів можна також визначити частку ПЗ каналу. Для цього потрібно запам'ятовувати межсегментний інтервали потоку у відправника і вимірювати їх у одержувача. Порівняння значень інтервалів характеризує стан мережі, а мінімальне значення вимірюваних інтервалів в одержувача дозволяє визначити доступну частку ПЗ.

Таким чином, виходить наступна схема: установка швидкості потоку відправником за допомогою ретельної диспетчеризації сегментів, вимірювання швидкості прибуття потоку в одержувача і передача цієї інформації відправникові разом з рештою контрольної інформацією. Різниця старого і нового значень швидкості відправки потоку ARTCP на кожному кроці задається випадкової змінної, однак, при наявності сигналу про перевантаження мережі ймовірність зниження швидкості перевищує ймовірність її збільшення на кожному новому кроці.

2.3.1 Параметри і змінні

Нехай *r* часовий інтервал між послідовними трансляціями пакетів.

Завдання функції диспетчеризації сегментів в тому, щоб затримувати відправку чергового сегмента на час τs після початку передачі попереднього сегмента. Позначимо всі змінні, пов'язані з відправника індексом *s*, і *r* - пов'язані з одержувачу. Отже, τs часовий інтервал між моментами початку відправки в мережу сегмента *i*+1 і *i*-го, а τr інтервал між послідовно прибулими до одержувача сегментами.

Нехай швидкість каналу зв'язку, безпосередньо до якого підключений відправник R_s тоді час йде на відправку одного сегмента (з моменту початку передачі до моменту її закінчення)

$$t_{ls} = \frac{S}{R_{ls}}, \tag{2.2}$$

де S - розмір переданого сегмента. Очевидно, що максимально можлива швидкість потоку

$$R_s^{\max} = R_{ls} = \frac{S}{\tau_s^{\min}} \tag{2.3}$$

Мінімальне значення міжсегментного інтервалу в цьому випадку буде $\tau_s^{\min} = t_{ls}$, коли пакети відправляються в мережу без затримок з максимальною швидкістю канального рівня. Шляхом зміни τ_s в межах $[\tau_s^{\min}, \infty)$, ARTCP може

контролювати швидкість потоку в межах $[R_{ls,},0)$. Ілюстрація принципу управління швидкість приведена на рисунку 2.3.

Затримка відправки готових сегментів проводиться за допомогою системного таймера. Наприклад, для повного використання ПС каналу в 512 Кб / с при розмірі сегмента в 1000 байт кожен сегмент необхідно відправляти з затримкою $\tau = 0{,}015625$ с, щоб швидкість потоку склала 64 пакета / с.

Таблиця 2.1 - Змінні і параметри моделі ARTCP

S	Розмір кадру канального рівня містить сегмент
τ_S	Часовий проміжок між послідовними трансляціями сегментів (від першого біта до першого біта)
τ_R	Часовий проміжок, який вимірюється між послідовними моменту прибуття сегментів (від останнього біта до останнього біта)
$R_S(t)$	Швидкість потоку, що встановлюється відправником
$R_R(t)$	Швидкість потоку, що обчислюється одержувачем
$R_e(t)$	Оцінка доступної пропускної здатності в момент t-RTT
$R_s^{'}$	Перша похідна швидкості за часом, встановлюється відправником
R_{pe}	Значення оцінки доступної пропускної здатності в момент i-1
A_C	Площа області компенсації
SSGR	Параметр експоненціального зростання RS в режимі SS
RTT	Часовий проміжок між моментом відправки сегменту і моментом приходу його підтвердження
ERTT	Зважене ковзаюче середнє RTT
SmoothedRe	Зважене ковзаюче середнє Re
MaxERTT	Максимальне значення згладженого RTT

Продовження таблиці 2.1

MinERTT	Мінімальне значення згладженого RTT
MDFACTOR	Коефіцієнт, який використовується при мультиплікативному зниженні R_S (t)
R_S^{and}	Значення швидкості відправки даних безпосередньо на виході режиму MD1
R_S^{bmd}	Значення швидкості відправки даних безпосередньо перед входом в режим MD1
R_e^{and}	Значення оцінки доступної пропускної здатності в режимі MD1

2.3.2 Формат повідомлення

Формат повідомлень використовуваних ARTCP може в точності збігатися з форматом пакета TCP. Стандарт TCP [3] передбачає наявність додаткового-тільних полів у заголовку сегмента між стандартним заголовком і полем даних. ARTCP може передавати додаткову інформацію в цих полях, що буде гарантувати сумісність з TCP. Всього протокол ARTCP вимагає використання лише двох нових полів: значення попереднього порядкового номера "PS" в напрямку від відправника до одержувача і значення шпаруватості "TI" в напрямку від одержувача до відправника. Значення "TI" можна передавати у вигляді опції тимчасової мітки [16], а значення "TI" вимагає поля, позволяє помістити порядковий номер сегмента.

2.3.3 Структурна схема протоколу

У протоколі ARTCP повністю перероблені всі механізми управління потоком. Механізм корекції помилок передачі в ARTCP не впливає на ско-зростання передачі. Від TCP збережені віконний механізм для управління завантаженням одержувача, алгоритми визначення RTT і установки таймера

ТПП. Ознакою втрати сегмента служить спрацьовування ТПП або прихід двох після-послідовних підтверджень одного сегмента. Алгоритм керування швидкістю включає в себе: функції диспетчеризації сегментів, вимірювання швидкості та адаптації швидкості рисунок 2.2. Далі розглянемо ці функції докладно.

У завдання функції диспетчеризації сегментів входить відправка сегментів в мережу зі строго заданою швидкістю, яка виражається в значеннях межсегментних тимчасових інтервалів.

Функція вимірювання швидкості визначає шпаруватість потоку поступають до одержувача та інформує відправника про темпоральних параметрах прибуває потоку. Крім того, відправник сам виробляє вимір часу RTT (в рамках стандартної функціональності протоколу TCP).

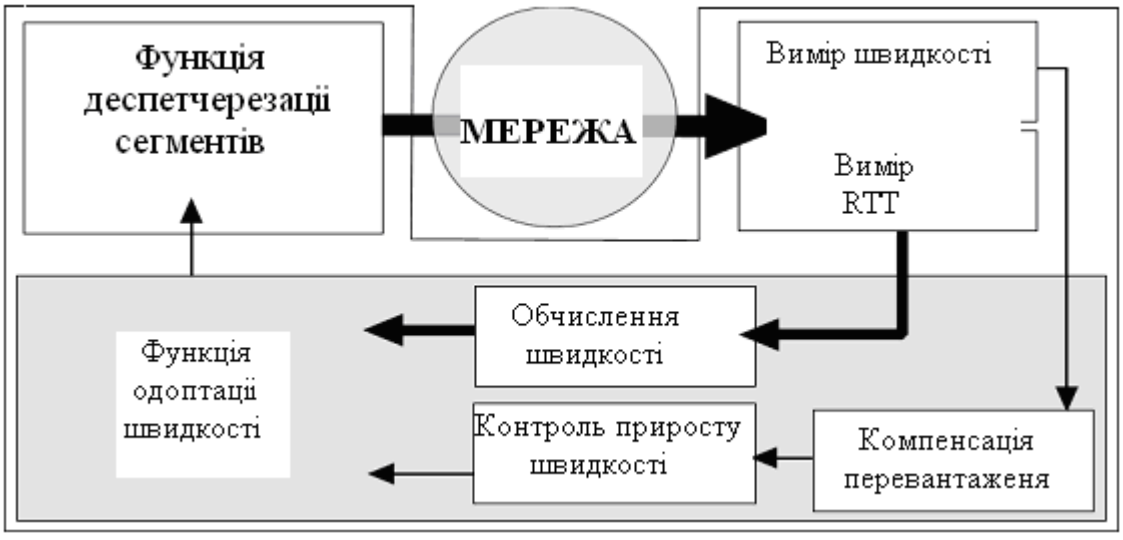

Рисунок 2.2 - Функціональна схема механізму управління потоком ARTCP. (Жирна стрілка вгорі позначає напрямок передачі даних. Більш тонкі стрілки: напрямок передачі керуючої інформації)

Значення шпаруватості потоку виміряної одержувачем і часу RTT виміряного відправником надходять на вхід функції адаптації, яка визначає

нове значення швидкості відправки потоку відповідно до отриманих на вхід значеннями і своїм станом в цей момент часу.

ARTCP використовує дві ознаки початку перевантаження мережі, коли середня швидкість прибуття запитів порівнюється з середньою швидкістю обслуговування і початком зростання черги: початок зростання RTT і стабілізацію *Rr (t)* при збільшення *Rs (t)*. Одержувач ARTCP в сегментах з підтвердженнями вказує значення швидкості прибуття потоку. Отримуючи підтвердження сегмента через час RTT після його відправлення, джерело ARTCP отримує інформацію про значенні швидкості, з якою потік, що містить цей сегмент, прибув до одержувача і використовує *Rr (t)* в якості оцінки *Re (t)* ПС мережі.

2.3.3.1 Диспетчеризація сегментів

Диспетчер сегментів відправляє сегменти на лінію через строго задані міжсегментні тимчасові інтервали. Значення інтервалів визначаються швидкістю відправки потоку, яка задається функцією адаптації.

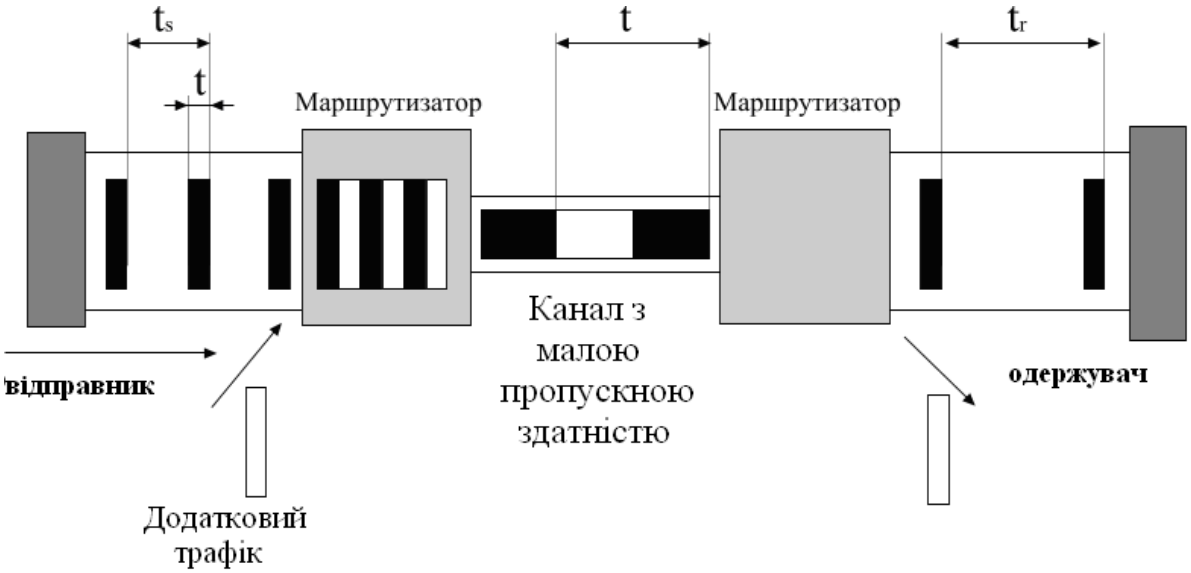

Рисунок 2.3 –Диспетчеризація сегментів і вимірювання швидкості потоку алгоритмом ARTCP

Чорні прямокутники позначають сегменти одного з'єднання. Раз-відмінність ширини прямокутників відображає відмінність швидкостей каналів. Меншій швидкості відповідає більш тривалий час передачі.

2.3.3.2 Вимірювання швидкості

Очевидно, що швидкість прийому потоку одержувачем не може бути вище за швидкість обслуговування потоку на ділянці з найменшою ПЗ, через яку проходить з'єднання. Таким чином, знаючи швидкість прибуття потоку до напівотримувача, можна визначити доступну пропускну здатність мережі. Для коректного виміру швидкості необхідно не враховувати випали з потока, тобто втрачені сегменти, а також сегменти, що доставляються мережею в зміненому порядку. Для виконання цієї умови в поле "PS" кожного відправляємого сегмента записується порядковий номер (або зсув) від попереднього сегмента.

Отримавши сегмент i, одержувач обчислює різницю поточного часу і часу прибуття попереднього (j) сегмента τ_r і у випадку, якщо поле "PS" i-го сегмента містить значення j, поміщає $R_r = \frac{s}{\tau_r}$ в полі "TI" підтвердження наступного в протилежному напрямку рисунок 2.3. Одержувач витягує значення поля "TI" з одержуваних підтверджень і використовує його для керування швидкістю передачі.

2.3.3.3 Функція адаптації

Опис алгоритму роботи функції адаптації, безпосередньо здійснюється управління потоком ARTCP, було дано в роботах [50, 51, 52, 53].

Спосіб управління потоком має досягти, по-перше, швидкої реакції потоку на мінливі умови з'єднання і, по-друге, стабілізувати швидкість передачі, коли вона дорівнює максимальній швидкості мережі. Алгоритм управління потоками ARTCP функціонує в кількох режимах.

Робота ARTCP починається з режиму швидкого збільшення швидкості, аналогічної механізму уповільненого старту стандартного TCP, для максимально швидкого досягнення з'єднанням верхньої межі доступної пропускної здатності. Після того, як верхня межа досягнуть, алгоритм ARTCP

переходить в режим точного налаштування, протягом якої утримує швидкість на рівні доступною ПЗ. У разі визначення зменшення доступною ПЗ, ARTCP здійснює мультиплікативне зниження швидкості, яке в разі тривалого стану перевантаження триває експоненціально. Отже, адаптація швидкості передачі потоку протоколом ARTCP відбувається в п'яти режимах рисунок 2.4.

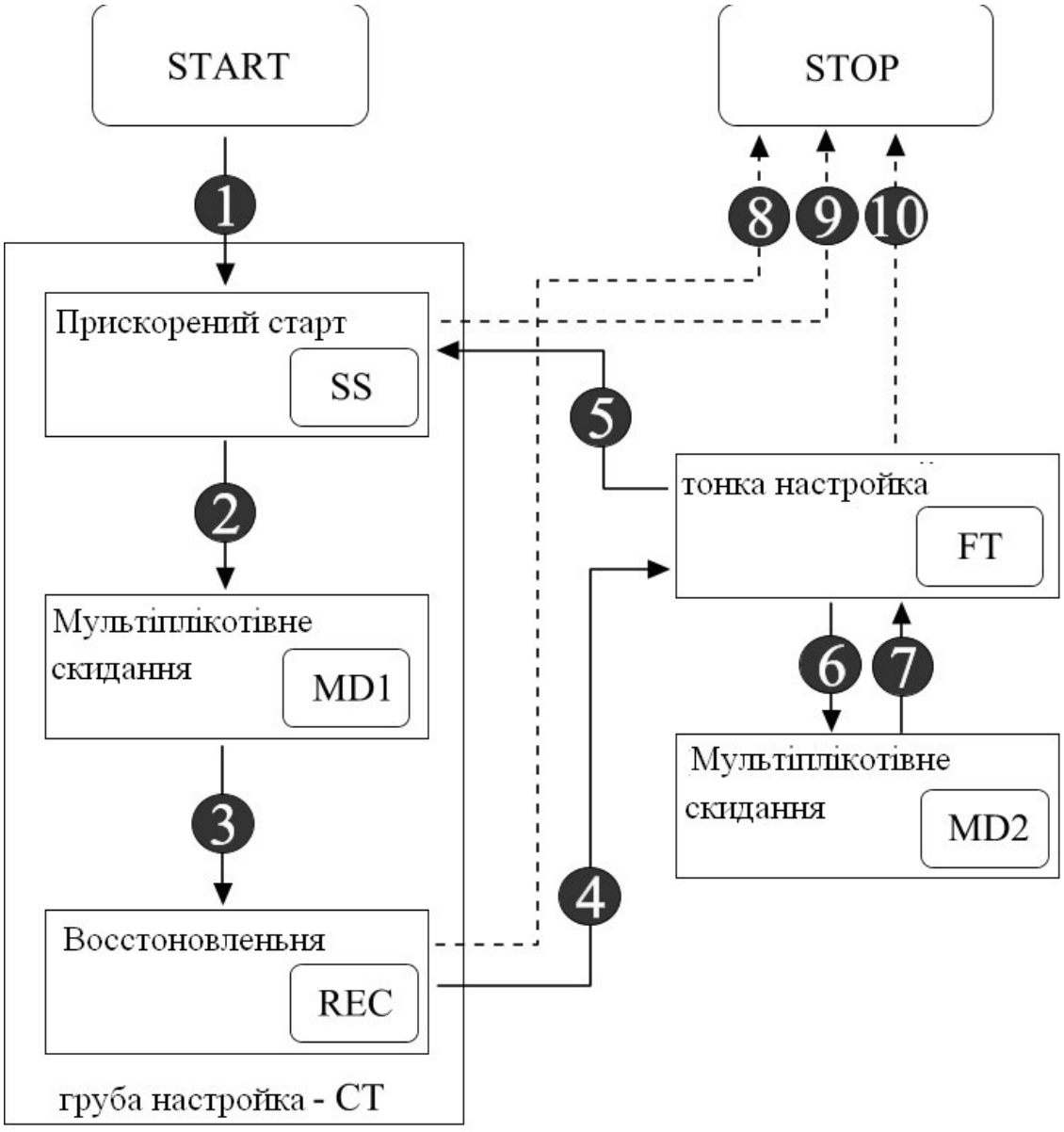

Рисунок 2.4 - Діаграма режимів алгоритму адаптації протоколу ARTCP. Штрихована лінії позначають можливі переходи в стан зупинки системи

Режим прискореного старту (SS) має мету максимально швидко збільшити швидкість потоку від мінімального значення до значення, рівного

або перевершує ПЗ каналу відразу після ініціалізації з'єднання. Для цього швидкість збільшується експоненціально:

$$R'_s(t + RTT) = R'_s(t) * SSGR. \quad (2.4)$$

$$R_s(t_i) = R_s(t_{i-1}) + R'_s(t_i) * (t_i - t_{i-1}). \quad (2.5)$$

Початкове значення швидкості встановлюється після синхронізації сполуки як:

$$R_s^{\min} = \frac{SEGSIZE}{\min RTT}. \quad (2.6)$$

Вихід з режиму SS відбувається, коли $R_e(t_i) < (1-\varepsilon) * R_s(t_i - RTT)$. Після реалізації переходу 2, алгоритм переходить в стан мультиплікативного скидання MD1.

Режим мультиплікативного скидання (MD1) слід за режимом SS. Після виходу з SS значення $R_s(t_i)$ будуть перевищувати $R_e(t)$ тому в режимі MD1 швидкість потоку стрибкоподібно встановлюється свідомо нижче $R_e(t)$:

$$R_s(t_i) = R_e(t_i) - MDFACTOR * (R_s(t_{i-1}) - R_e(t_i)). \quad (2.7)$$

Після зниження швидкості алгоритм переходить в режим відновлення.

Режим відновлення (REC) має на меті, лінійно збільшуючи швидкість, довести її до вже відомого значення ПС каналу: $R_e(t)$, компенсуючи виникла в режимі SS перевантаження. В режимі REC обчислюється значення площі області компенсації $A_c(t_i)$ як площі фігури, утвореної значеннями $R_s(t)$ над прямий $R_e(t_i)$ за час, поки $R_s(t_i) > R_e(t_i)$ в режимі SS: Значення площі області компенсації одно сумі площ набору трапецій освічених значеннями $R_s(t)$ над прямий $R_e(t_i)$. Площа кожної трапеції

$$S_T = \frac{\Delta t_i}{2(2R_S(t_i) - R'_S(t_i) * \Delta t_i - 2R_e(t_i))}, \quad (2.8)$$

де Δt_i час, протягом якого не відбувалося змін значенія $R'_S(t_i)$

$$A_c(t_i) = \frac{1}{2}\Delta t_0[2R_S(t_i) - \Delta t_0 R'_s(t_i) - 2R_e(t_i)] + \frac{1}{2}RTT[2R_S(t_i) - \frac{R'_S(t_i)}{SSGR} - RTT\frac{R'_S(t_i)}{SSGR} - R_e(t_i)] +$$

$$+\frac{1}{2}RTT[2R_S(t_i) - \frac{R'_S(t_i)}{SSGR^2} - RTT\frac{R'_S(t_i)}{SSGR^2} - R_e(t_i) + \frac{1}{2}\frac{[R_S(t_i) - \frac{R'_s(t_i)}{SSGR^N}}{\frac{R'_S(t_i)}{SSGR^N}}, \quad (2.9)$$

де проміжок часу між моментом t_i і моментом попереднього зміни $R'_S(t_i)$ и N - індекс доданка при якому:

$$R_S(t_i) - \frac{R'_S(t_i)}{SSGR^N} > R_e(t_i). \quad (2.10)$$

Отже, перший доданок наведеної формули є площа трапеції з висотою меншою Rrr, останнє доданок - площа трикутника, складові від 2 до N-1 площі трапецій з висотою рівній Rrr.

Наприклад, у випадку, наведеному на рисунок 2.5, $A_c(t_i)$ дорівнює сумі площ трапеції DCBE і трикутника ABE. Δt_0 дорівнює відрізку ED.

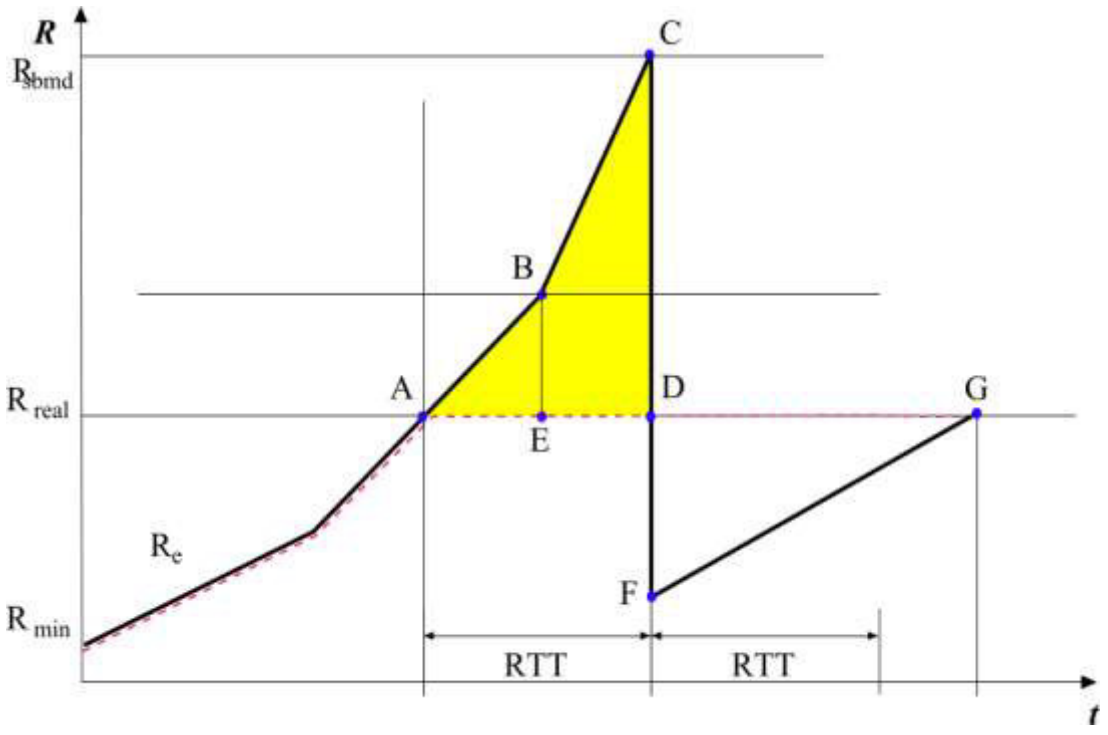

Рисунок 2.5 – Залежність швидкості від часу в фазі грубої настройки ARTCP. Зафарбована область, що складається з ABE і BCDE є площа області компенсації та висловлює обсяг даних, накопичених в буфері.Штрихове лінія позначає поточну оцінку ПС мережі протоколомARTCP

Значення $A_c(t_i)$ застосовується для визначення величини $R'_C(t_i)$ в стані відновлення REC. Ідея в тому, що через затримки інформації про стан мережі на час RTT, в стані швидкого збільшення швидкості відправки сегментів потік викличе наповнення буферів мережі. Пакети будуть накопичуватися в мережі протягом часу, коли швидкість відправки сегментів перевищує ПС мережі відрізок AD на рисунку 2.5. Перебування з'єднання в стані відновлення необхідно для того, щоб мережа впоралася з виниклою до зменшення швидкості відправки перевантаженням. Очевидно, що кількість даних, накопичених в буферах мережі, визначається площею області $A_c(t_i)$, тому швидкість відправки сегментів в стані REC повинна бути знижена таким чином, щоб площа фігури DFG дорівнювала $A_c(t_i)$. Швидкість в стані REC змінюється лінійно, обумовлена значенням:

$$R'_S(t_i) = \frac{[R_{pe}(t_i) - R_s(t_i)]^2}{2A_C(t_i)}. \qquad (2.11)$$

У стані REC швидкість відправки даних зростає лінійно від значення отриманого в попередній стадії MD1 за законом

$$R_s(t_i) = R_S^{and} + t * R'_S(t_i). \qquad (2.12)$$

Вихід зі стану REC (перехід 4) здійснюється в тому випадку, коли

$$R_S(t) \geq R_e^{and}. \qquad (2.13)$$

Режим тонкої настройки (FT) слід за режимом REC, в режимі FT швидкість відправки даних повільно підлаштовується під ПС каналу. Відношення коефіцієнтів speedup і *slowdown* в стані *FT* визначає ймовірність зниження або підвищення швидкості на кожному кроці. Коефіцієнт speedup, відповідальний за підвищення швидкості обернено пропорційний швидкості даного з'єднання. Коефіцієнт *slowdown,* що відповідає за зниження швидкості, пропорційний відношенню вимірюваного *RTT* до мінімального значення *RTT*. Значення speedup більше при менших значеннях $R_S(t)$, що дає повільним з'єднанням перевагу для отримання доступу до більшої відносної частці ПС. Значення slowdown однаково для всіх з'єднань і росте при зростанні *RTT*. Таким

чином, імовірність підвищення швидкості для повільних з'єднань більше, а ймовірність зниження швидкості однакова для всіх з'єднань. Вихід з режиму *FT* відбувається в разі стрибкоподібного зміни вимірюваного *RTT*.

У стані FT значення швидкості передачі визначаються за законом:

$$R_S(t_i) = midpoint + [2 + Rand - \frac{slowdown}{speedup}] * \frac{speedup}{slowdown} * INTEVAL * midpoint. \qquad (2.14)$$

де Rand - рівномірно розподілена випадкова величина, що генерується функцією drand48 () з областю значень [0; 1]. Після попадання в стан FT (реалізації переходу 4) значення midpoint встановлюється рівним R_e^{and}, надалі значення *midpoint* встановлюється рівним sR_S. Змінні *speedup* і *slowdown* визначають напрям зміни швидкості відправки даних в залежності від зміни часу RTT.

Коефіцієнти *slowdown* і *speedup* для використання у формулі (2.14) визначаються за наступними формулами:

$$speedup = [1 + \frac{S}{\min RTT * sR_S}]^2, \qquad (2.15)$$

де другий доданок представляє собою відношення мінімальної кількості даних в транзиті по з'єднанню (*S* байт за час *RTT*) до реального їх кількості (добуток мінімального часу *RTT* на середню швидкість відправки потоку). Відповідно зростання реальної швидкості потоку виражається в зменшенні значення коефіцієнта *speedup*, таким чином, при інших рівних умовах ймовірність зростання *RS* для з'єднання з меншою швидкістю буде більше. За рахунок цього усувається нерівномірність використання ресурсів різними сполуками.

Коефіцієнт slowdown обчислюється таким чином:

якщо

$$RTT > \min ERTT * (1 + PRECISION), \qquad (2.16)$$

тоді

$$slowdown = 2 * (\frac{RTT}{\min ERTT}) * speedup, \qquad (2.17)$$

інакше $slowdowm = 1$.

Ставлення *speedup/slowdown* визначає знак відхилення миттєвого значення швидкості від середнього. Якщо *speedup*>*slowdown* то відхилення від середнього значення для миттєвого значення швидкості буде позитивним, тобто швидкість потоку буде збільшуватися. У разі *speedup*<*slowdown* швидкість потоку буде знижуватися. Також, в стані *FT* максимальне відхилення миттєвого значення швидкості відправки пакетів від середнього за попередній період, згідно з формулою (2.18), пропорційно середньому значенню швидкості. У зв'язку з цим потік, здійснюючи перехід 4 в стані *FT* при більшому значенні оцінки доступною ПС, пристосовується до невеликих змін ПС більш інтенсивна.

Умовою переходу з *FT* в режим мультиплікативного скидання MD2 (перехід 6) є:

$$ERTT > K * (FT \max RTT - \min ERTT). \quad (2.18)$$

Режим мультиплікативного скидання (MD2) необхідний для швидкого зниження швидкості за умови різкого зростання RTT:

$$midpo\text{int}_i = midpo\text{int}_{i-1} * MDFACTOR \quad (2.19)$$

Після цього протокол переходить в стан FT, реалізуючи перехід 7. У тому випадку, якщо умова (2.19) продовжує залишатися істинним, то мультиплікативне зменшення триває, оскільки послідовність переходів 6 - 7 реалізується неодноразово, висловлюючись в експоненційному зменшенні швидкості передачі даних.

Завершення роботи протоколу може статися з будь-якого стану {SS, REC, FT} - переходи (8, 9, 10).

Очікуване поведінку алгоритму керування швидкістю потоку в різних режимах зображено на рисунку 2.6

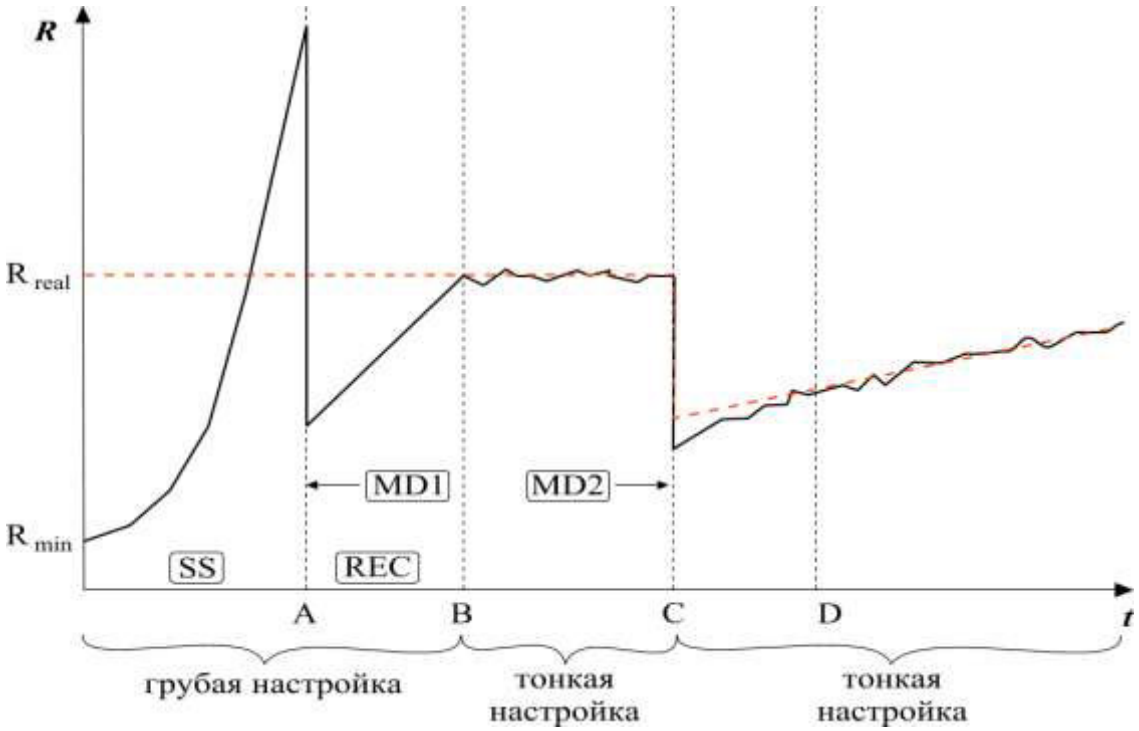

Рисунок 2.6 - Очікуване поведінку алгоритму керування швидкістю потоку (залежність швидкості від часу). Значення t в точках A, B, C позначають моменти переходу в новий режим

2.3.4 Працює з TCP

Система, що підтримує ARTCP, може бути також сумісна з TCP. Для цього, ініціатор з'єднання, що підтримує ARTCP, поміщає в заголовку синхронізуючого пакета опцію "PS".

Наявність поля "PS" в заголовку SYN пакета повинно включати механізм ARTCP одержувача. Якщо така можливість є, то одержувач включає в сегмент SYN-ACK поле "TI", якщо ні, то відсутність опції "TI" у відповіді одержувача відключає механізм ARTCP у ініціатора і подальший обмін відбувається по протоколу TCP. Якщо ж опція "TI" присутній у відповіді, то ініціатор впевнений, що і одержувач задіяв механізм ARTCP.

2.3.5 Порівняння ARTCP і TCP на основі аналізу алгоритму

Протокол TCP не може обходитися без втрат сегментів, які створюються самим протоколом шляхом переповнення черги, і які обмежують подальше зростання швидкості TCP.

Протокол ARTCP не створює втрат сегментів, якщо

Qmax / R> ε, де *Qmax* - максимальна довжина черги, ε - граничне значення, яке використовується алгоритмом ARTCP, а *R* - швидкість каналу.

Втрата сегмента може відбутися лише в разі надходження чергового сегмента в чергу обслуговуючого пристрою, коли розмір сегмента перевищує різницюQ_{max}- *Q*. Якщо *D* сума затримок передачі в каналах на шляху від відправника до одержувача, то у припущенні, що канали в системі симетричні і трафік передається лише в одному напрямку, мінімальний час *minRTT* визначається як 2*D*. У разі якщо середня швидкість відправки потоку *RS* перевищує швидкість R каналу, який обслуговує чергу, виникає чергу сегментів в маршрутизаторі. Поява черги довжини Q призведе до збільшення часу *RTT* на величину *Q / R*. Однак для ARTCP, якщо різниця *RTT-minRTT* перевищує деяке граничне значення ε, ймовірність зниження швидкості відправки потоку RS стане відмінною від нуля і швидкість потоку буде знижуватися, поки значення дозволеного перевищення *RTT* над *minRTT* не стане нижче граничного значення. Таким чином, виконується нерівність: *Q <ε * R*. Вибираючи досить мале значення *ε*, алгоритм ARTCP гарантує, що довжина черги не перевищить певного значення, меншого, ніж максимальна довжина черги *Q <Qmax*. Для виконання цього, необхідно вибрати ε, так, щоб *ε <Qmax / R*.Следовательно, при такому виборі граничного значення, R відсутність втрат сегментів при роботі ARTCP гарантовано.

При числі потоків більшому 1, протокол ARTCP на відміну від TCP, може бути налаштований так, щоб взагалі не створювати втрат сегментів.

Для протоколу TCP факт втрати сегмента служить індикатором виникнення перевантаження мережі. Значення ймовірності втрати сегмента визначає розвивається TCP з'єднанням швидкість передачі. Будь-яка втрата сегмента викликає стрибкоподібне зменшення розміру вікна і, отже, зниження швидкості передачі TCP. В умовах, коли причиною втрат є виключно переповнення черги, зниження швидкості TCP при виникненні втрат призводить до того, що виконується рівність середньої швидкості TCP і

швидкості обслуговування каналу. Очевидно, що зниження швидкості внаслідок додаткових втрат, викликаних помилками передачі, призведе до того, що середня швидкість TCP потоку стане менше, ніж швидкість каналу. Оскільки TCP не може визначити причину втрати сегмента і реагує зниженням швидкості на будь-яку втрату, то його ефективність в мережах, де втрати сегментів можуть бути наслідком помилок передачі, буде тим менше, чим вище ймовірність втрати сегмента.

Протокол ARTCP на відміну від TCP не знижує швидкість передачі потоку при виникненні втрати сегмента. Втрачені дані ретранслюються, не надаючи впливу на швидкість передачі. Внаслідок цього, втрати сегментів не впливають на швидкість потоку ARTCP.

Сказане вище можна сформулювати в наступному вигляді:

На відміну від TCP, протокол ARTCP не чутливий до втрат сегментів.

2.4 Топологія

Для дослідження поведінки протоколу ARTCP в умовах вільних про впливи інших транспортних потоків використовується топологія, зображена на рис.(2,7). Стрілкою вказано напрямок передачі даних. Вузол, відзначений знаком S, є відправником, вузол R - одержувачем. Набір каналів 0, 1 моделює дуплексну комутовану ЛВС або виділений канал у відправника, 2, 3 відповідно в одержувача. Набір каналів 4, 5 моделює канал типу точка-точка між двома вузлами територіальної мережі.

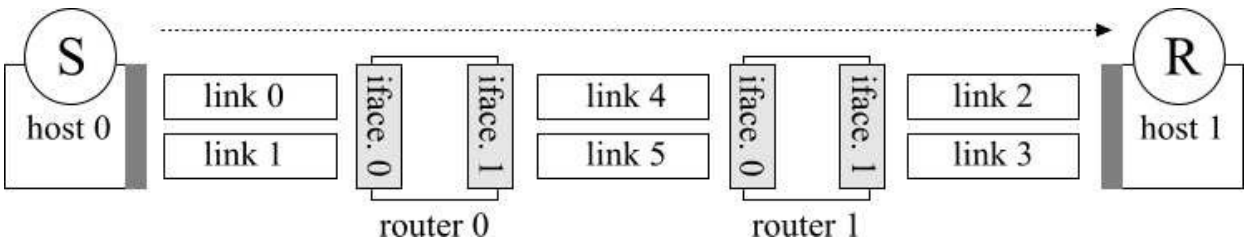

Рисунок 2.7 – Топологічна схема 1 з одним потоком і парою систем.

Експеримент 1: 32 Кб / с

Таблиця 2.2 – Ілюстрація функціонування протоколу ARTCP в даних умовах

Параметри	Значення
ПЗ каналів 0, 1, 2, 3	10 Мб\с
Затримка каналів 0, 1, 2, 3	0.01 с
ПЗ каналів 4, 5	32 Кб\с
Затримка каналів 4. 5	0.1 с
Час модулювання	300 с
Макс. розмір черги маршрутизатора	16 Кбайт
BER	0
Результати	
Характеристика	Значення
Коефіцієнт використання ПЗ в стані FT	97.81%
Середня швидкість потоку	31302.64 б\с
Число втрачених сегментів	0
Число переданих	1173
RTT (усереднення за час експерименту)	0.614±0.146 с
Мінімальне RTT	0.519
Q (усереднення за час експерименту)	459.833±716.51 байт

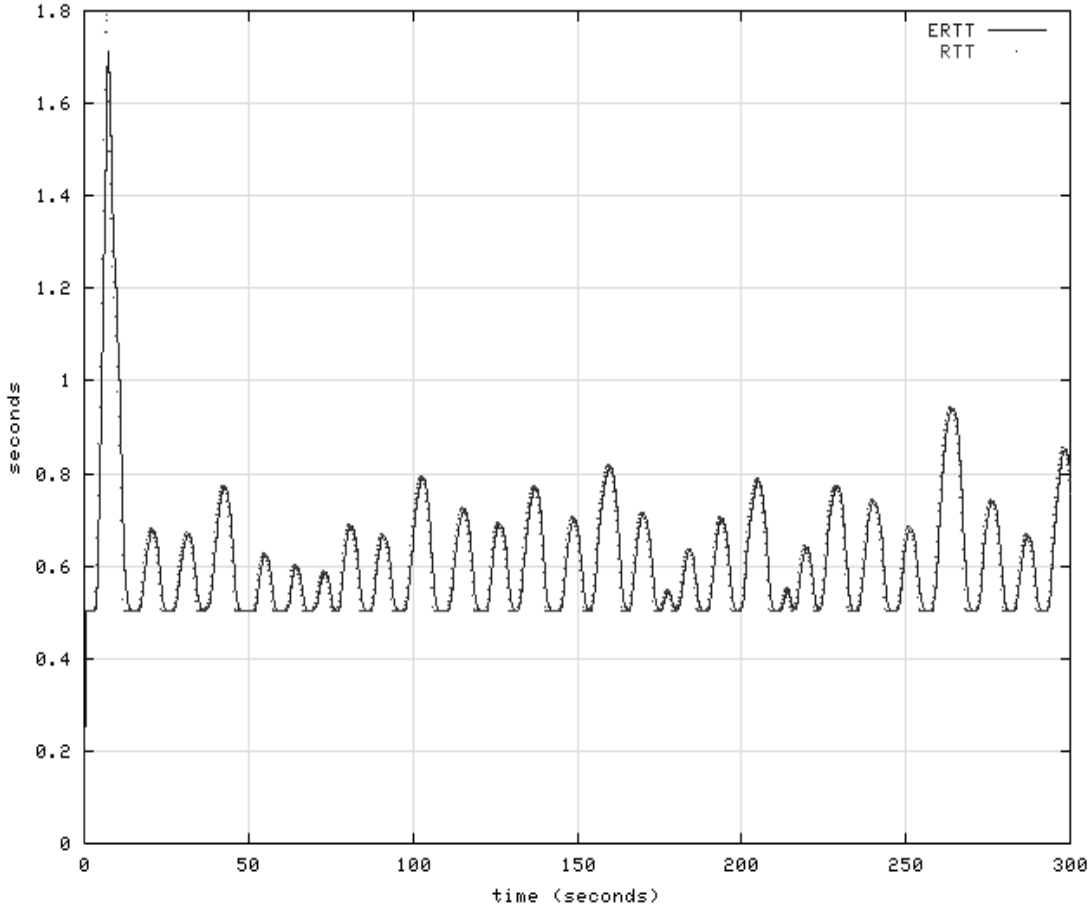

Рисунок 2.9 – Залежність швидкості потоку від часу при швидкості каналу 32Кб/с

Залежність середнього та миттєвого значення RTT від часу при швидкості каналу 32 Кб / с. Sendingrate - встановлюється швидкість відправки, measuredreceivedrate - вимірювана швидкість прийому потоку.

2.5 Сумісність с TCP

Система, що підтримує ARTCP, може бути також сумісна з TCP. Для цього, ініціатор з'єднання, що підтримує ARTCP, поміщає в заголовку синхронізуючого пакета опцію "PS". Наявність поля "PS" в заголовку SYN пакета повинно включати механізм ARTCP одержувача. Якщо така можливість є, то одержувач включає в сегмент SYN-ACK поле "TI", якщо немає, то відсутність опції "TI" у відповіді одержувача відключає механізм ARTCP у ініціатора і подальший обмін відбувається по протоколу TCP. Якщо ж опція

"TI" присутній у відповіді, то ініціатор впевнений, що і одержувач задіяв механізм ARTCP.

2.6 Висновок

Експерименти показали, як ізольований протокол ARTCP адаптується до доступної ПЗ каналу. При цьому протокол ARTCP здійснює переходив необхідні режими в повній відповідності з описаним раніше алгоритмом управління потоком. У експериментів трат пакетів не відбувається.

РОЗДІЛ 3. ЕКОНОМІЧНИЙ РОЗДІЛ

У дипломній роботі проводилось дослідження існуючих методів транспортного протоколу, були отримані нові результати, щодо покращення цих процесів. З їх урахуванням запропоновано новий транспортний протокол, який би покращив процес передачі сигналів, підвищило якість передачі і пропускну спроможність каналу.

При розробці нового транспортного протоколу важливо визначити, наскільки економічно ефективним він опиниться, для того, щоб конкурувати в сучасних ринкових умовах, у порівнянні з аналогами, що представлені на ринку. Конкурентоспроможність програмного продукту визначається багатьма чинниками, зокрема його якістю і ефективністю.

Найбільше вимог перед будь-яким протоколом встановлює користувач, але при введені в експлуатацію необхідно враховувати всі чинники, які можуть впливати на життєздатність протоколу. Тому що його якість, окрім позиції користувача, також визначається:

- з позиції використання ресурсів і їх оцінки;
- з позиції виконання умов на програмний продукт.

Оцінка якості програмного продукту з погляду користувача визначається необхідною на стадії функціонування кількістю оперативної пам'яті ЕОМ, витратами машинного часу, пропускною спроможністю каналів передачі даних. Оцінка використання ресурсів на стадії створення продукту включає визначення трудомісткості, часу налаштування і вартості створення програми.

3.1 Техніко-економічне обґрунтування створення і використання модифікованого транспортного протоколу

3.1.1 Визначення трудомісткості розробки транспортного протоколу

Нормування праці в процесі створення програмного забезпечення утруднене через творчий характер праці програмістів. Тому трудомісткість

розробки програмного забезпечення може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість створення програмного продукту можливо розрахувати по формулі (3.1):

$$t = t_О + t_И + t_А + t_П + t_{ОТЛ} + t_Д,\ людино-годин\ , \qquad (3.1)$$

де $t_О$ - витрати праці на підготовку і опис поставленого завдання;

$t_И$ - витрати праці на дослідження алгоритму рішення задачі;

$t_А$ - витрати праці на розробку блок-схеми транспортного протоколу;

$t_П$ - витрати праці на програмування за розробленою блок-схемою;

$t_{ОТЛ}$ - витрати праці на налагодження програми на ЕОМ;

$t_Д$ - витрати праці на підготовку документації по завданню.

Складові витрат праці визначаються на підставі умовної кількості операторів в програмному продукті. Причому до цього числа входять ті оператори, які програмістові необхідно написати в процесі роботи над завданням з урахуванням можливих уточнень в постановці завдання і удосконалення транспортного протоколу.

Умовна кількість операторів розраховується за формулою (3.2):

$$Q = q \cdot c \cdot (1 + p), \qquad (3.2)$$

де q - передбачувана кількість операторів;

c - коефіцієнт складності програми;

p - коефіцієнт корекції програми в ході її розробки.

Коефіцієнт складності програми з визначає відносну складність програм завдання по відношенню до типового завдання, складність якого прийнята рівній одиниці. Діапазон його зміни 1,25…2,0.

Коефіцієнт корекції програми p визначає збільшення об'єму робіт за рахунок внесення змін до алгоритму або програми в результаті уточнення постановки завдання. Величина p знаходиться в межах 0,05…0,1, що відповідає внесенню 3…5 корекцій, що спричиняють за собою переробку 5-10 % готової програми.

Проведемо розрахунок умовної кількості операторів в програмному продукті, що розробляється:

$$Q = q \cdot c \cdot (1 + p) = 70 \cdot 1{,}5 \cdot (1 + 0{,}1) = 116 \ опер. \tag{3.3}$$

Оцінка витрат праці на підготовку і опис завдання залежить від конкретних умов і визначається на основі експертних оцінок.

Зважаючи на той факт, що дослідження, пов'язані з обробкою складних сигналів в телекомунікаційних системах, охоплюють великий пласт інформації, приймаємо $t_O = 160$ *людино – годин*.

Витрати праці на вивчення опису завдання визначаються з урахуванням уточнення опису і кваліфікації програміста по формулі (3.4):

$$t_И = \frac{Q \cdot B}{(75...85) \cdot K}, \tag{3.4}$$

де B - коефіцієнт збільшення витрат праці унаслідок недостатнього опису завдання, B=1,2…1,5;

K - коефіцієнт кваліфікації програміста, визначуваний залежно від стажу роботи по даній спеціальності.

Він складає при стажі роботи, років:

— до 2 - 0,8;

— від 2 до 3 - 1,0;

— від 3 до 5 - 1,1…1,2;

— від 5 до 7 - 1,3…1,4;

$$t_{И} = \frac{116 \cdot 1{,}5}{80 \cdot 0{,}8} = 2{,}72, \textit{людино} - \textit{годин}$$

Витрати праці на розробку транспортного протоколу поставленої задачі розраховуються за формулою:

$$t_{А} = \frac{Q}{(20...25) \cdot K}, \textit{людино} - \textit{годин} \qquad (3.5)$$

Отже витрати праці на розробку транспортного протоколу поставленої задачі складуть:

$$t_{А} = \frac{116}{25 \cdot 0{,}8} = 5{,}8, \textit{людино} - \textit{годин}$$

Витрати праці на складання транспортного протоколу за розробленою блок-схемою:

$$t_{П} = \frac{Q}{(20...25) \cdot K} = \frac{116}{25 \cdot 0{,}8} = 5{,}8, \textit{людино} - \textit{годин} \qquad (3.6)$$

Витрати на налагодження програми на ЕОМ $t_{ОТЛ}$ розраховуються за наступними формулами:

– за умови автономної налагодження одного завдання:

$$t_{ОТЛ} = \frac{Q}{(4...5) \cdot K}, \textit{людино} - \textit{годин} \qquad (3.7)$$

– за умови комплексному відкладанню завдання:

$$t_{ОТЛ}^{K} = 1{,}5 \cdot t_{ОТЛ}, \textit{людино} - \textit{годин} \qquad (3.8)$$

Оскільки підбір параметрів програмного забезпечення, що розробляється, є комплексним завданням, розрахунок витрат на налаштування програми проведемо за формулами (3.8) та (3.7):

$$t_{ОТЛ} = \frac{Q}{(4...5)\cdot K} = \frac{116}{5\cdot 0{,}8} = \frac{116}{4} = 29, людино-годин$$

$$t_{ОТЛ}^{К} = 1{,}5\cdot t_{ОТЛ} = 1{,}5\cdot 29 = 43{,}5, людино-годин$$

Витрати праці на підготовку документації по завданню $t_{Д}$ визначаються за формулою (3.9):

$$t_{Д} = t_{ДР} + t_{ДО}, людино-годин, \qquad (3.9)$$

де $t_{ДР}$ - трудомісткість підготовки матеріалів рукопису, $t_{ДР} = \frac{Q}{(15...20)\cdot K}$;

$t_{ДО}$ - трудомісткість редагування, друку і оформлення документації, $t_{ДО} = 0{,}75\cdot t_{ДР}$.

Отже, трудомісткість створення транспортного протоколу, відповідно до (3.9) складає:

$$t_{Д} = t_{ДР} + t_{ДО} = \frac{Q}{(15...20)\cdot K} + 0.75\cdot(\frac{Q}{(15...20)\cdot K}) = \frac{116}{20\cdot 0{,}8} + 0{,}75\cdot\frac{116}{20\cdot 0{,}8} =$$

$$= 12{,}69, людино-годин$$

Таким чином, визначивши трудомісткість окремих показників, розрахуємо сумарну трудомісткість розробки транспортного протоколу за формулою (3.1):

$$t = 160 + 2{,}72 + 5{,}8 + 5{,}8 + 43{,}5 + 12{,}69 = 230{,}51, \ людино-годин$$

3.1.2 Розрахунок витрат на створення транспортного протоколу

Витрати на створення транспортного протоколу включають витрати на заробітну плату виконавців програми і вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$K_{ИЗ} = З_{ЗП} + З_{МИ}, \; грн. \tag{3.10}$$

Заробітна плата виконавців визначається по формулі:

$$З_{ЗП} = t \cdot C_{ПР}, \; грн, \tag{3.11}$$

де t - загальна трудомісткість розробки транспортного протоколу, визначувана за формулою (3.1), людино-годин;

$C_{ПР}$ - середня годинна заробітна плата програміста (основна і додаткова) з урахуванням єдиного соціального внеску, грн/год.

Для визначення мінімальної середньої годинної оплати програміста необхідно спочатку визначити його річний фонд заробітної плати з урахуванням єдиного соціального внеску. Це можна зробити, знаючи мінімальний щомісячний оклад програміста.

На 01.12.2018 р. він складає 25000 грн. Отже, заробітна плата програміста з урахуванням премій (20%) і можливих надбавок (10%) складає 32500 грн.

Таким чином, річний фонд заробітної плати – 390000 грн. Єдиний соціальний внесок складає 38%, тобто 148200 грн.

Разом, річний фонд заробітної плати з урахуванням відрахувань на соціальні потреби склав 538200 грн.

Визначимо номінальний річний фонд робочого часу, при цьому прийнявши середню тривалість робочого дня рівної 8 годинам:

$$F_{Н} = (T_{К} - T_{ПР} - T_{ВИХ} - T_{ОТП}) \cdot 8, \; годин, \tag{3.12}$$

де $T_К$ - кількість календарних днів в році, $T_К = 365 днів$;

$T_{ПР}$ - кількість святкових днів в році, $T_{ПР} = 10 днів$;

$T_{ВИХ}$ - кількість вихідних днів в році, $T_{ВИХ} = 104 дні$;

$T_{ОТП}$ - календарна тривалість відпустки $T_{ОТП} = 24 дні.$

Отже, річний фонд часу дорівнює:

$$F_Н = (365 - 10 - 104 - 24) \cdot 8 = 1816, годин$$

Середня заробітна плата програміста за годину визначається співвідношенням (3.13), яке має вигляд:

$$C_{ПР} = \frac{ФЗП_{СН}}{F_Н}, грн/год, \quad (3.13)$$

де $ФЗП_{СН}$ - річний фонд заробітної плати з урахуванням відрахувань на соціальні потреби;

$F_Н$ - річний фонд робочого часу.

$$C_{ПР} = \frac{538200}{1816} = 296{,}37, грн/год.$$

Таким чином, витрати на оплату праці розробника складають з урахуванням формули (3.11) отримаємо:

$$З_{ЗП} = 296{,}37 \cdot 230{,}51 = 68316{,}25, грн$$

Розрахунок вартість машинного часу, необхідного для налагодження програми на ЕОМ здійснюється по формулі:

$$З_{МИ} = (t_{ОТЛ} + t_Д) \cdot C_{МЧ}, \quad (3.14)$$

де $t_{ОТЛ}$ - трудомісткість налагодження програми на ЕОМ, яка визначається за формулами (3.7) і (3.8), людино-годин;

$t_Д$ - витрати праці на підготовку документація по завданню, що визначаються за формулою (3.9);

$C_{МЧ}$ - вартість машино-часу ЕОМ, грн/год.

Для розрахунку вартості машино-часу необхідно знати вартість ЕОМ та ПЗ на момент їх придбання і введення в експлуатацію, і вартість споживаної електроенергії. Відповідні дані представлені в таблиці 3.1.

Таблиця 3.1 – Вартість необхідного програмного та апаратного забезпечення

Найменування	Вартість за 1 шт., грн
Персональний комп'ютер	8645
MS Windows XP Professional Rus DSP SP3 (OEM)	1264
Мережевий кабель	30
Разом:	9939

Разом, загальна вартість ЕОМ і потрібного ПЗ на момент придбання складає 9939 грн.

Сума річних амортизаційних відрахувань буде визначатися за формулою (3.15):

$$САМ = СПЕР \cdot Н_А, \text{ грн} , \quad (3.15)$$

де *СПЕР* - первинна вартість ЕОМ і необхідного програмного забезпечення;

$Н_А$ - норма амортизації, $Н_А = 0{,}6$

Таким чином, амортизаційні відрахування в 2019 р. склали:

$$САМ_{2012} = 9939 \cdot 0{,}6 = 5963{,}4, грн \quad (3.16)$$

Отже, залишкова вартість ЕОМ і необхідного програмного забезпечення на кінець 2019 р. складе:

$$Ф_{ост} = 9939 - 5963{,}4 = 3975{,}6, грн$$

Розрахунок вартості машино-часу ЕОМ проведемо по формулі:

$$С_{м.ч} = \frac{Ф_{ост}}{Ф_{год.раб.ч.}} + W \cdot Ц_э \quad (3.17)$$

де $Ф_{год.раб.ч.}$ - річний фонд корисного часу роботи ЕОМ;

W - настановна потужність ЕОМ, $W = 0{,}4\,кВт$;

$Ц_э$ - вартість $1кВт*год$ електроенергії.

Річний фонд корисного часу роботи ЕОМ дорівнює річному фонду робочого часу програміста і складає 1816 годин. 2,198

Таким чином, вартість машино-години ЕОМ складе:

$$С_{м.ч} = \frac{3975{,}6}{1816} + 0{,}4 \cdot 1{,}23 = 2{,}69, грн/год$$

Враховуючи відому вартість машино-години проведемо розрахунок вартості машинного часу, яке є необхідним для налаштування програмного забезпечення на ЕОМ по формулі (3.14):

$$З_{МИ} = (43{,}50 + 12{,}69) \cdot 2{,}69 = 151{,}15, грн$$

Отже, витрати на створення транспортного протоколу розраховують за формулою (3.10):

$$К_{ИЗ} = 68316{,}25 + 151{,}15 = 68467{,}4, грн$$

Визначені таким чином витрати на створення транспортного протоколу є одноразовими капітальними витратами на створення транспортного протоколу і складають 68467,4 грн.

3.2 Висновок

В економічному розділі було розраховано, кількість часу, що необхідно для розроблення транспортного протоколу, заробітну плату робітників, вартість капітальних затрат, на основі чого було проведено порівняння з аналогами і було зроблено висновок, що затрати на транспортного протоколу, що розробляється складають 68467,4 грн.

При використанні розробленого транспортного протоколу соціальний ефект виявляється у вигляді якіснішої і швидшої роботи мережі, в порівнянні з аналогами, представленими на ринку України.

ВИСНОВКИ

У даному дипломній роботі було розглянуто експерименти, проведеного на імітаційної моделі, показують істотну перевагу адаптивного алгоритму керування швидкістю потоку протоколу ARTCP в порівнянні з TCP. Особливо добре ARTCP повинен функціонувати в бездротових мережах. Виявлене у трафіку моделюємої мережі, в якій функціонує протокол ARTCP властивість самоподібності, по-перше, свідчить про те, що модель добре відтворює властивості реальних мереж, а по-друге, служить підставою використання саме методу модельного експерименту для дослідження нового протоколу.

В економічному розділі було розраховано, кількість часу, що необхідно для розроблення транспортного протоколу, заробітну плату робітників, вартість капітальних затрат, на основі чого було проведено порівняння з аналогами і було зроблено висновок, що затрати на транспортного протоколу.

При використанні розробленого транспортного протоколу соціальний ефект виявляється у вигляді якіснішої і швидшої роботи мережі, в порівнянні з аналогами, представленими на ринку України.

СПИСОК ЛІТЕРАТУРИ

1 Holzmann G. Design and Validation of Computer Protocols. Prentice Hall, New Jersey, 1991.

2 Deering S., Hinden R. Internet Protocol Version 6 (IPv6) Specification. // RFC 2460. -1998.

3 Postel J. Transmission Control Protocol. // RFC793 (STD7). 1981.

4 Stevens R. Advanced Programming in the UNIX Environment. Addison-Wesley. 1992.

5 McKusick M., Bostic K., Karels M., Osterman J. The Design and Implementation of the 4.4 BSD Operating System. Addison-Wesley, 1996.139

6 Comer D. Internetworking with TCP/IP, Vol. II: Design Implementation and Internals. Prentice Hall, NJ. 1994.

7 Comer D. Internetworking with TCP/IP, Vol. III: Client - Server Programming and Applications. Prentice Hall, NJ. 1994.

8 Bartlett K., Scantlebury R., Wilkinson P. A note on reliable full-duplex transmission over half-duplex lines. // Comm. of the ACM. Vol. 12, No. 5. -1969.- p. 260-265.

9 Cerf V., Kahn R. A protocol for packet network intercommunication. // IEEE Trans. on communications. vol. COM-22, N. 5. -1974.- p. 637-648.138

10 Tanenbaum A.S. Computer Networks. Third edition, Prentice-Hall, New Jersey, 1996.

11 Nemeth E., Snyder G., Seebass S., Hein T. UNIX System Administration Handbook. Prentice Hall PTR. NJ. Second edition. 1995.

12 Selecting Sequence Numbers. // SIGCOMM/SIGOPS Interprocess Commun. Workshop. ACM. -1975.- p. 11-23.

13 unshine C., Dalal Y. Connection Management in Transport Protocols. // Computer Networks. vol. 2. -1978.- p. 454-473.

14 Watson R. Timer-Based Mechanisms in Reliable Transport Protocol Connection Management. // Computer Networks. vol. 5. -1981.- p. 47-56.

15 Braden R. T. Requirements for Internet Hosts - Communication Layers. // RFC1122. 1989.

16 Jacobson V., Braden R., Borman D. TCP Extensions for High Performance. // RFC1323. 1992.

17 agle J. Congestion Control in IP/TCP Networks. // ARPANET Working Group Requests for Comment (RFC-896), DDN Network Information Center, SRI International, Menlo Park, CA.1984.

18 Jacobson V. Congestion Avoidance and Control. // ACM SIGCOMM'88. 1988.

19 Karn P., Partridge C. Estimating Round-trip Times in Reliable Transport Protocols. // ACM SIGCOMM'87. 1987.

20 George F., Young G. SNA Flow Control: Architecture and Implementation. // IBM System Journal, vol. 21, no. 2. - 1982. - pp. 179-210.

21 Digital Equipment Corporation. DECnet Digital Network Architecture [phase IV] General Description. // Order AA-N149A-TC, Digital Equipment Corporation. 1982.

22 Yang C., Reddy A. A Taxonomy for Congestion Control Algorithms in Packet Switching Networks. // IEEE Network Magazine. Vol. 9, Number 5. - 1995.

23 Lefelhocz C., Lyles B., Shenker S., Zhang L. Congestion Control for Best-Effort Service: Why We Need a New Paradigm. // IEEE Network, Vol. 10, N. 1. -1996.

24 Floyd S., Jacobson V. Random Early Detection gateways for Congestion Avoidance. // IEEE/ACM Transactions on Networking. Vol.1, N.4. -1993.- p. 397-413.

25 Brakmo L., O'Malley S., Peterson L. TCP Vegas: New Techniques for Congestion Detection and Avoidance. // ACM SIGCOMM. -1994.- pp. 24-35.

26 Brakmo L., Peterson L. TCP Vegas: End to End Congestion Avoidance on a Global Internet. // IEEE Journal on Selected Areas in Communications, 13(8). -1995.

27 Brakmo L., Peterson L. Performance Problems in BSD4.4 TCP. // ACM Computer Communications Review. 25(5). -1995.- p. 69-86.

28 Caceres R., Danzig P., Jamin S., Mitzel D. Characteristics of Wide-Area TCP/IP Conversations. // ACM SIGCOMM'91. -1991.

29 Fall K., Floyd S. Simulation-based Comparisons of Tahoe, Reno, and SACK TCP. // Computer Communications Review. July -1996.

30 Tomey C. Rate-Based Congestion Control Framework for Connectionless Packet-Switched Networks. // Doctor of Philosophy Thesis. University of New South Wales Australian Defence Force Academy. -1997.

31 Benmohamed L., Meerkov S. M. Feedback Control of Congestion in Packet Switched Networks: The Case of a Single Congested Node. // IEEE Transactions on Networking. Vol. 1, No. 6. -1993.- p. 693-707.

32 Charney A. An Algorithm for Rate Allocation in a Packet-Switched Network with Feedback. // M.Sc. thesis. Department of EECS. MIT. -1994.

33 Ramakrishnan K., Jain R. A Binary Feedback Scheme for Congestion Avoidance in Computer Networks. // ACM Transactions on Computer Systems. Vol. 8, No. 2. -1990.- p. 158-181.

34 Keshav S. The Packet Pair Flow Control Protocol. // ICSI Tech. Rept. TR-91-028. Computer Science Division, Department of EECS, University of California, Berkeley and International Computer Science Institute. Berkeley, CA. May 1991.

35 Bennett J., Zhang H. Hierarchical Packet Fair Queueing Algorithms. // ACM SIGCOMM'96. Aug 1996.

36 Demers A., Keshav S., Shenker S. Analysis and Simulation of a Fair Queueing Algorithm. // ACM SIGCOMM'89. -1989.- p. 3-12.

37 Floyd S., Jacobson V. Link Sharing and Resource Management Models for Packet Networks. // IEEE/ACM Transactions on Networking. 3(4). -1995.

38 Алексеев И.В. Интегрированные услуги нового поколения Internet. // Сети. № 10. -1999.-с. 102-108.

39 Clark D., Lambert M., Zhang L. NETBLT: A High Throughput Transport Protocol. // ACM SIGCOMM '88. -1988.- p. 306-312.

40 Clark D., Lambert M., Zhang L. NETBLT: A Bulk Data Tranfer Protocol. // RFC 969. -1987.

41 Wang Z., Crowcroft J. A New Congestion Control Scheme: Slow Start and Search (Tri-S). // ACM Computer Communications Review. vol. 21. -1991.- p. 32-34.

42 Wang Z., Crowcroft J. Eliminating periodic packet losses in the 4.3-Tahoe BSD TCP congestion control algorithm. // ACM computer communications review. vol. 22. -1992.- p. 916.

43 Padhye J., Firoiu V., Towsley D., Kurose J. Modelling TCP throughput: a simple model and its empirical validation. // ACM SIGCOMM'98. -1998.

44 Johnson D., Maltz D. Protocols for Adaptive Wireless and Mobile Networking. // IEEE Personal Communications. -February 1996. p. 34-42.

45 Eckhardt D., Steenkiste P. Measurement and Analysis of the Error Characteristics of an In-Building Wireless Network. // Proc. of ACM SIGCOMM '96. -1996.- p. 243-254.

46 Cäceres R., Iftode L. Improving the Performance of Reliable Transport Protocols in Mobile Computing Environments. // IEEE Journal of Selected Areas in Communication. 13(5). -1995.-p. 850-857.

47 Balakrishnan H., Padmanabhan V.N., Seshan S., Katz R.H. A Comparison of Mechanisms for Improving TCP Performance over Wireless Links. // Proc. of ACM SIGCOMM. -1996.- p. 256269.

48 Balakrishnan H., Padmanabhan V.N., Katz R.H. The Effects of Asymmetry on TCP Performance. // Proc. Of ACM/IEEE International Conf. on Mobile Computing and Networking. -September 1997.

49 Chiu D., Jain R. Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks. // Computer Networks and ISDN Systems. v. 17. -1989.- p. 1-14.

50 Alekseev I.V., Sokolov V.A. Compensation Mechanism for Adaptive Rate TCP. // 1-St International IEEE/Popov Seminar "Internet: Technologies A and Services". P. 68-75, October 1999.

51 Алексеев И.В., Соколов В.А. Протокол TCP с адаптацией скорости. // Моделирование и анализ информационных систем. Т.6, №1. - 1999.- С. 4-12.

52 Алексеев И.В. Математическая модель протокола TCP с адаптацией скорости. // Моделирование и анализ информационных систем. Т.6, №2. - 1999.- С. 51-53.

53 Алексеев И. В. Модель и анализ транспортного протокола ARTCP. // Электронный журнал "Исследовано в России". № 27. - 2000.- С.395-404. http://zhurnal.mipt.rssi.ru/articles/2000/027.pdf

ДОДАТОК А. Відомість матеріалів дипломної роботи

№	Формат	Найменування	Кількість листів	Примітка
1	А4	Реферат	3	
2	А4	Список умовних скорочень	1	
3	А4	Зміст	2	
4	А4	Вступ	4	
5	А4	1 Розділ	55	
6	А4	2 Розділ	22	
7	А4	3 Розділ	10	
8	А4	Висновки	1	
9	А4	Список літератури	4	
10	А4	Додаток А	1	
11	А4	Додаток Б	1	
12	А4	Додаток В	1	
13	А4	Додаток Г	1	

ДОДАТОК Б. Перелік документів на оптичному носії

1 Титульна сторінка.doc

2 Завдання.doc

3 Реферат.doc

4 Список умовних скорочень.doc

5 Зміст.doc

6 Вступ.doc

7 Розділ 1.doc

8 Розділ 2.doc

9 Розділ 3.doc

10 Висновки.doc

11 Перелік посилань.doc

12 Додаток А.doc

13 Додаток Б.doc

14 Додаток В.doc

15 Додаток Г.doc

Презентація.pptx

ДОДАТОК В. Відгуки керівників розділів

Відгук керівника економічного розділу:

Керівник розділу _____ _____

(підпис) (ініціали, прізвище)

ДОДАТОК Г. ВІДГУК

на дипломну роботу бакалавра на тему:

Вдосконалення способу адаптивного управління пакетним трафіком протоколу транспортного рівня

студента групи 172-16ск-1

Данильченко Роман Володимирович

Пояснювальна записка складається з титульного аркуша, завдання, реферату, списку умовних скорочень, змісту, вступу, трьох розділів, висновків, переліку посилань та додатків, розташованих на __ сторінках та містить __ рисунків, __ таблиць, __ джерел та __ додатка.

Об’єкт розробки: транспортний протокол ARTCP.

Мета дипломної роботи: розробити програмну імітаційну модель, відтворюючу основні характеристики мережі, які і визначають функціонування в ній транспортного протоколу.

У спеціальній частині визначено найважливіші характеристики ARTCP, а також показано наявність властивості самоподібності у трафіку ARTCP. Результати експерименту дозволяють стверджувати, що ARTCP перевершує TCP по основним критеріям.

У роботі наведені: керування швидкістю потоку в різних режимах; топологічна схема з одним потоком і парою кінцевих систем; ілюстрація функціонування протоколу ARTCP в даних умовах.

Зміст та структура дипломної роботи дозволяють розкрити поставлену тему повністю.

Студент показав достатній рівень володіння теоретичними положеннями з обраної теми, показав здатність формувати власну точку зору (теоретичну позицію).

Робота оформлена та написана грамотною мовою. Містить необхідний ілюстрований матеріал. Автор добре знає проблему, уміє формулювати наукові та практичні завдання і знаходить адекватні засоби для їх вирішення.

В цілому дипломна робота задовольняє усім вимогам і може бути допущена до захисту, а його автор заслуговує на оцінку «_____».

Керівник дипломної роботи,

к.ф.-м.н., проф. Гусєв О.Ю.