О.Л. НЕДАШКІВСЬКИЙ

# Технології та протоколи Інфокомунікаційних мереж (частина перша)

Київ 2017

# ВСТУП

Курс "Технології та протоколи інфокомунікаційних мереж" є однією з фундаментальних теоретичних та практичних дисциплін базової підготовки фахівців-зв'язківців і відноситься до дисциплін загально-професійної підготовки за вибором вищого навчального закладу за напрямом 0924 "Телекомунікації". Дисципліна забезпечує професійне спрямування процесу навчання студентів. Складається з двох частин і вивчається протягом двох семестрів.

В навчальній дисципліні розглядаються Технології та протоколи інфокомунікаційних мереж мереж на базі стеку протоколів TCP/IP та технології IP/MPLS. Для стеку протоколів TCP/IP розглядаються протоколи канального, мережного та транспортного рівнів. Особлива увага приділяється алгоритму функціонування протоколу TCP. Для технології IP/MPLS визначаються особливості побудови архітектури мережі та створення віртуальних приватних мереж. Для обох технологій розглядаються архітектура, алгоритми та механізми забезпечення необхідної якості надання послуг.

***Мета дисципліни « Технології та протоколи інфокомунікаційних мереж »: навчання студентів основним закономірностям, пов'язаних з принципами функціонування цифрових мереж, їх протоколів та алгоритмів; опанування основними термінами, категоріями, базовими знаннями із су¬часної організації цифрових мереж, використання і оцінювання у своїй практичній діяльності математичних моделей забезпечення якості послуг та планів передавання відповідного типу трафіку за різними політиками; здатність застосовувати правила, методи, принципи, закони у конкретних ситуаціях, своєчасно адаптуватися до зростаючого потоку інформації, впроваджувати новітні науково-технічні досягнення в інфокомунікаційних технологіях в галузь телекомунікацій.***

***Тому предметом дисципліни є: загальна характеристика цифрових мереж, архітектура мережі на базі стеку протоколів TCP/IP та технології IP/MPLS, протоколи і алгоритми функціонування цифрових мереж за даними технологіями; визначення методів забезпечення необхідної якості обслуговування (QoS) в мережах на базі відповідних технологій; математичний опис характеристик продуктивності протоколів TCP/IP та IP/MPLS, в тому числі з урахуванням політик QoS; алгоритми вирівнювання трафіку, механізми розподілу ресурсів та запо-бігання перевантаження мережі..***

Процес створення ІС багато в чому ще не формалізовано. Вміння правильно створити систему чи окрему задачу, виявити і коректно сформулювати критерії і обмеження приходять з досвідом. Існуючі стандарти, керівні документи і методичні матеріали визначають організаційні питання і регламентують склад і зміст проектної документації, але не містять рекомендацій і вказівок, які розкривають суть процесу створення ІС. Це зумовило певні складнощі в ході підготовки навчального матеріалу, який складено з урахуванням окремих питань дисципліни, висвітлених у вітчизняній і зарубіжній літературі, а також досвіду щодо наукових основ створення ІС, практичних розробок ІС різного призначення.

У перших лекціях ви розглянете загальні поняття про архітектуру стеку протоколів TCP/IP, протоколи канального, мережного, транспортного та прикладного

рівнівм, алгоритми функціонування протоколів TCP та UDP, принципи побудови архітектури MPLS, структуру мережного елементу середовища MPLS, принципи комутації в середовищі MPLS, організацію віртуальних приватних мереж в середовищі MPLS. І закінчимо розглядом: архітектура QoS в мережах IP, архітектуру диференційних послуг, класифікацію пакетів, маркування пакетів та управління інтенсивністю трафіку, політикою розподілу ресурсів, політикою попередження перевантаження і політикою відкидання пакетів, якістю обслуговування в мережах MPLS, перерозподілои потоків в мережах MPLS, класами обслуговування IntServ, протоколом RSVP, механізмом обслуговування DiffServ, підтримкою механізмів QoS в віртуальних приватних мережах MPLS.

**ЗМІСТ**

## ПЛАН НАВЧАННЯ

Кількість лекцій: 12.
Кількість практичних занять: 5
Кількість семінарський занять: 5
Модулів: 2
Підсумковий контроль: дифзалік.

# ЧАСТИНА 1
# МОДУЛЬ 1
# ТЕМА 1
# АРХІТЕКТУРА ТА ПРИНЦИПИ ФУНКЦІОНУВАННЯ СТЕКУ ПРОТОКОЛІВ TCP/IP

## ЛЕКЦІЯ 1
## ЗАНЯТТЯ 1
## ЗАГАЛЬНІ ПОНЯТТЯ ПРО АРХІТЕКТУРУ СТЕКУ ПРОТОКОЛІВ TCP/IP. ПРОТОКОЛИ КАНАЛЬНОГО, МЕРЕЖНОГО, ТРАНСПОРТНОГО ТА ПРИКЛАДНОГО РІВНІВ

### §1. Мета та задачі курсу

Технічною базою розвитку сучасного інформаційного суспільства є телекомунікаційні системи, які забезпечують неперервний обіг інформації. Тільки накопичувати інформацію недостатньо — треба ще й доносити її до користувачів, причому так, щоб користувачі не відчували дискомфорту в процесі приймання і сприйняття інформації. Саме забезпечення функціонування інфраструктури всебічного обміну інформацією і є головною задачею телекомунікаційних систем. Звісно, інтенсивний розвиток інформаційних технологій потребує не менш значного розвитку телекомунікаційних систем, але при цьому телекомунікації повинні випереджати у своєму розвитку техніку накопичення інформаційних ресурсів, щоб не гальмувати доступ до них користувачів. З метою забезпечення якісної передачі інформації із врахуванням всіх можливих умов розміщення користувачів, телекомунікаційні системи постійно вдосконалюються і трансформуються (адаптуються), підвищуючи свою пропускну спроможність. В результаті цього з'явилася велика кількість принципово нових видів телекомунікаційних систем і спостерігається подальший процес їх конвергенції та інтелектуалізації. Темпи розвитку телекомунікаційних технологій нині можна порівняти із сходом гірської снігової лавини, що невпинно набирає швидкість і силу. Метою даного курсу є з єдиних позицій показати сучасний стан телекомунікаційних систем і перспективи їх подальшого розвитку, що базуються на широкому використанні комп'ютерних технологій.

Отже, предметом навчальної дисципліни є:

- ✓ загальна характеристика цифрових мереж, архітектура мережі на базі стеку протоколів TCP/IP та технології IP/MPLS, протоколи і алгоритми функціонування цифрових мереж за даними технологіями; визначення методів забезпечення необхідної якості обслуговування (QoS) в мережах на базі відповідних технологій;
- ✓ математичний опис характеристик продуктивності протоколів TCP/IP та IP/MPLS, в тому числі з урахуванням політик QoS;
- ✓ алгоритми вирівнювання трафіку, механізми розподілу ресурсів та запобігання перевантаження мережі.

Метою вивчення навчальної дисципліни є:

- ✓ навчання студентів основним закономірностям, пов'язаних з принципами функціонування цифрових мереж, їх протоколів та алгоритмів;
- ✓ опанування основними термінами, категоріями, базовими знаннями із сучасної

організації цифрових мереж, використання і оцінювання у своїй практичній діяльності математичних моделей забезпечення якості послуг та планів передавання відповідного типу трафіку за різними політиками;

- ✓ здатність застосовувати правила, методи, принципи, закони у конкретних ситуаціях, своєчасно адаптуватися до зростаючого потоку інформації, впроваджувати новітні науково-технічні досягнення в інфокомунікаційних технологіях в галузь телекомунікацій.

Задачею навчальної дисципліни є формування таких навичок:

- ✓ розуміти архітектуру цифрових мереж на базі стеку протоколів TCP/IP та технології IP/MPLS;
- ✓ вміти використовувати в практичній діяльності протоколи та алгоритми стеку TCP/IP та технології IP/MPLS;
- ✓ володіти методами розрахунку продуктивності мережі на на базі стеку протоколів TCP/IP та технології IP/MPLS;
- ✓ проводити класифікацію типів трафіку на основі вимог до забезпечення необхідної якості обслуговування;
- ✓ використовувати механізми розподілу ресурсів мережі для забезпечення необхідного рівня якості надання послуг;

використовувати механізми запобігання перевантаження мережі для забезпечення необхідного рівня якості надання послуг; вміти створювати віртуальні приватні мережі з підтримкою методів забезпечення необхідного рівня якості надання послуг.

## §2. Загальні поняття про архітектуру стеку протоколів TCP/IP

Протокол TCP/IP був створений у результаті досліджень мереж з комутацією пакетів (packet-switching networks), проведених агентством DARPA (U.S. Department of Defence Advanced Research Projects Agency) наприкінці 60-х - початку 70-х років.

Термін "TCP/IP" звичайно позначає усе, що зв'язано з протоколами TCP і IP. Він охоплює ціле сімейство протоколів, прикладні програми і навіть саму мережу. До складу сімейства входять протоколи UDP, ARP, ICMP, TELNET, FTP і багато інших. TCP/IP - це технологія міжмережної взаємодії, технологія Internet.

Архітектура протоколів TCP/IP призначена для об'єднаної мережі, що складається з з'єднаних один з одним шлюзів окремих різнорідних пакетних підмереж, до яких підключаються різнорідні машини. Кожна з підмереж працює у відповідності зі своїми специфічними вимогами і має свою природу засобів зв'язку. Однак передбачається, що кожна підмережа може прийняти пакет інформації (дані з відповідним мережним заголовком) і доставити його по зазначеній адресі в цій конкретній підмережі. Не потрібно, щоб підмережа гарантувала обов'язкову доставку пакетів і мала надійний наскрізний протокол.

Коли необхідно передати пакет між машинами, що підключені до різних підмереж, то машина-відправник посилає пакет у відповідний шлюз (шлюз підключений до підмережі також як звичайний вузол). Після цього пакет направляється по визначеному маршруту через систему шлюзів і підмереж, поки не досягне шлюзу, підключеного до тієї ж підмережі, що і машина-одержувач; після чого пакет направляється до одержувача. Об'єднана мережа забезпечує датаграмний сервіс.

Проблема доставки пакетів у такій системі вирішується шляхом реалізації у всіх

вузлах і шлюзах міжмережного протоколу IP. Міжмережний рівень є, власне кажучи, базовим елементом у всій архітектурі протоколів, забезпечуючи можливість стандартизації протоколів верхніх рівнів.

Міжнародна громадська організація, іменована Співтовариством Інтернету (Internet Society, ISOC), керує розвитком сімейства протоколів TCP/IP. Стандарти для TCP/IP публікуються в серіях документів, названих RFC (Request for Comments).

Протоколи сімейства TCP/IP можна представити у виді спрощеної моделі взаємодії відкритих систем, що складається з чотирьох рівнів: прикладного (Application layer), транспортного (Transport layer), міжмережного, чи рівня Інтернету (Internet layer) і мережного інтерфейсу (Network Interface layer). Основні протоколи TCP/IP - це набір стандартів, що необхідні для з'єднання комп'ютерів і міжмережної взаємодії. В табл. 1 приведено перелік основних протоколів у відповідності до рівня на якому вони працюють.

Таблиця 1

Стек протоколів TCP/IP

| | | |
|---|---|---|
| ***HTTP, FTP, SSH, TELNET, SMTP, DNS …*** | | Прикладний рівень |
| TCP | UDP | Транспортний рівень |
| IP, ARP, ICMP,RIP,OSPF | | Міжмережний рівень |
| Ethernet, PPP, SLIP, HDLC, FrameRelay | | Мережний рівень |

В основі цієї моделі лежить рівень мережного інтерфейсу. Відповідні компоненти відповідають за відправлення в мережу і прийом з мережі кадрів, що містять пакети інформації. Кадри передаються по мережі як єдине ціле. Мережний рівень стеку TCP/IP відповідає фізичному і канальному рівням моделі взаємодії відкритих систем OSI (Open System Interconnection). Цей рівень у протоколах TCP/IP не регламентується, але підтримує всі популярні стандарти фізичного і канального рівня. Розроблена також спеціальна специфікація, що передбачає використання технології ATM як транспорт канального рівня.

Протоколи міжмережного рівня інкапсулюють пакети даних у датаграми Інтернету і проводять необхідну маршрутизацію. До протоколів міжмережного рівня відносяться:

IP (Internet Protocol) - в основному для відправлення і маршрутизації пакетів між мережами і вузлами.

ARP (Address Resolution Protocol) - для одержання адрес мережних адаптерів вузлів у рамках однієї фізичної мережі.

ICMP (Internet Control Message Protocol) - для відправлення повідомлень та інформації про помилки, що пов'язані з доставкою пакетів.

RIP (Routing Internet Protocol) і OSPF (Open Shortest Path First) - складання і модифікація таблиць маршрутизації, збір маршрутної інформації

Транспортний рівень забезпечує сеанси зв'язку між комп'ютерами. Існує два транспортних протоколи: TCP (Transmission Control Protocol) і UDP (User Datagram Protocol). Використання одного з них залежить від обраного методу доставки даних.

TCP орієнтований на з'єднання і використовується прикладеннями, що зазвичай

передають великі обсяги даних за одну операцію, тому що забезпечує надійне з'єднання, а також прикладеннями, яким необхідне підтвердження прийому даних.

Протокол UDP забезпечує не орієнтовану на з'єднання передачу даних і не гарантує доставку пакетів. Прикладення, що використовують протокол UDP, звичайно передають невеликі обсяги даних за одну операцію. Відповідальність за надійну доставку даних несе прикладення.

Завершує модель TCP/IP рівень, на якому прикладення одержують доступ до мережних компонентів. Тут працює безліч стандартних утиліт і сервісів протоколу TCP/IP, наприклад FTP, Telnet, SNMP і DNS

## §3. Протоколи канального, мережного, транспортного та прикладного рівнів

### *Протокол ARP*

Локальна адреса використовується в протоколі IP тільки в межах локальної мережі при обміні даними між маршрутизатором і вузлом цієї мережі. Маршрутизатор, одержавши пакет для вузла однієї з мереж, безпосередньо підключених до його портів, повинний для передачі пакета сформувати кадр відповідно до вимог прийнятої в цій мережі технології і вказати в ньому локальну адресу вузла, наприклад його Mac-адрес. У пакеті, що прийшов, ця адреса не зазначена, тому перед маршрутизатором устає задача пошуку його по відомій IP-адресі, що зазначена в пакеті як адреса призначення. З аналогічною задачею стикається і кінцевий вузол, коли він хоче відправити пакет у виділену мережу через маршрутизатор, підключений до тієї ж локальної мережі, що і даний вузол.

Для того щоб установити з'єднання, вузлам повинні бути відомі адреси мережних адаптерів інших вузлів. Дозвіл адреси (address resolution) - це процес визначення апаратної адреси мережного адаптера по його IP-адресі. Протокол ARP (Address Resolution Protocol) - частина рівня Інтернету моделі TCP/IP - дозволяє визначати адреси мережних адаптерів вузлів, розташованих в одній фізичній мережі.

Протокол ARP працює різним чином, у залежності від того, який протокол канального рівня працює в даній мережі - протокол локальної мережі (Ethernet, Token Ring, FDDI) з можливістю широкомовного доступу одночасно до усіх вузлів мережі, чи ж протокол глобальної мережі (X.25, frame relay), як правило не підтримуючий широкомовний доступ. Існує також протокол, що вирішує зворотну задачу - визначення IP-адреси по відомій локальній адресі. Він називається реверсивний ARP - RARP (Reverse Address Resolution Protocol) і використовується при старті бездискових станцій, що не знають у початковий момент своєї IP-адреси, але знають адресу свого мережного адаптера.

### *Протокол RARP*

Протокол RARP виконує зворотною функцію, тобто визначення IP-адреси по відомій локальній адресі.

### *Адресація в TCP/IP*

IP-адреса - це логічна 32-розрядна адреса, що однозначно визначає вузол TCP/IP. Кожна IP-адреса складається з двох частин: ідентифікатора мережі й ідентифікатора вузла. Перший служить для позначення усіх вузлів в одній фізичній мережі. Другий позначає конкретний вузол у мережі. Кожному вузлу TCP/IP, потрібна унікальна IP-адреса.

IP-адреса привласнюється не самому вузлу а мережному інтерфейсу цього вузла. Вузол може мати декілька мережних інтерфейсів з різними IP-адресами. Одному мережному інтерфейсу може бути привласнено кілька IP-адрес (аліаси). Декільком мережним інтерфейсам одного і того ж вузла можуть бути привласнені однакові IP-адреси.

Старші біти 4-х байтної IP-адреси визначають номер IP-мережі. Інша частина IP-адреси – номер вузла (хостномер).

Звичайно IP-адреси записують у виді чотирьох десяткових чисел розділених крапкою. Значення кожного десяткового числа відповідає значенню одного байта IP-адреси.

Мережний інтерфейс крім IP-адреси характеризується наступними параметрами [7]:

*Маска підмережі* – число де всі біти, що відносяться до номера мережі встановлені в 1, а всі біти, що відносяться до номера хоста встановлені в 0. Маска підмережі виділяє частину IP-адреси і дозволяє відрізнити ідентифікатор мережі від ідентифікатора вузла.

*Шлюз по умовчанню* – IP-адреса вузла даної підмережі, на який потрібно направляти пакети адресовані вузлам з інших підмереж.

Існують 5 класів IP-адрес, що відрізняються кількістю біт у номері мережі та хост-номері (рис. 1). Клас адреси визначається значенням його першого октету (рис. 2).

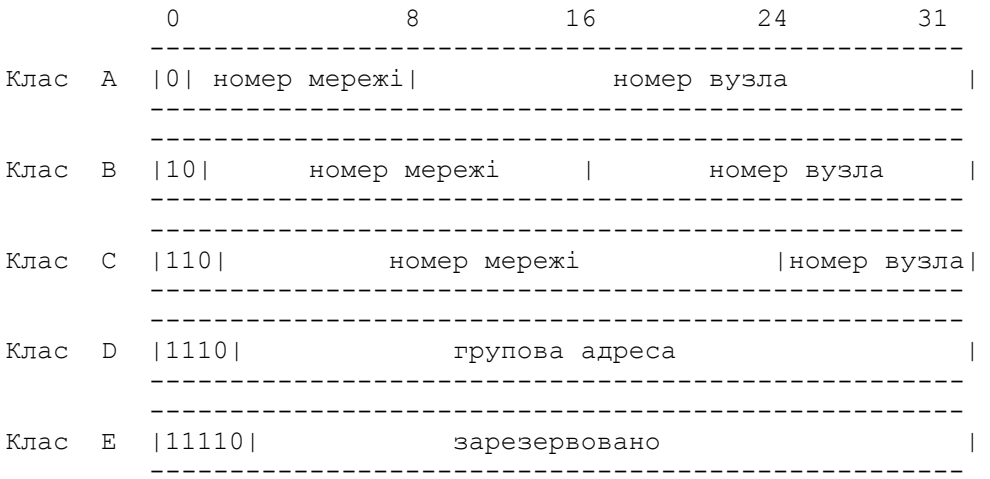



Рисунок 1 – Структура IP-адресів

| Клас | Діапазон значень першого октету | Можлива кіл-ть мереж | Можлива кіл-ть вузлів |
|---|---|---|---|
| A | 1 - 126 | 126 | 16777214 |
| B | 128-191 | 16382 | 65534 |
| C | 192-223 | 2097150 | 254 |
| D | 224-239 | - | 2**28 |
| E | 240-247 | - | 2**27 |

Рисунок 2 – Характеристики класів адресів

Адреси класу A призначені для використання у великих мережах загального користування. Вони допускають велику кількість номерів вузлів. Адреси класу B використовуються в мережах середнього розміру. Адреси класу C використовуються в мережах з невеликим числом комп'ютерів. Адреси класу D використовуються при зверненні до груп машин, а адреси класу E зарезервовані на майбутнє.

Деякі IP-адреси є виділеними і трактуються по-особливому (рис. 3).

| | |
|---|---|
| усі нулі | Даний вузол |
| номер мережі \| усі нулі | Дана IP-мережа |
| усі нулі \| номер вузла | вузел в даній (локальній) IP-мережі |
| усі одиниці | Всі вузли в даній (локальній) IP-мережі |
| номер мережі \| усі одиниці | Всі вузли у вказаній IP-мережі (multicast) |
| 127.0.0.1 \| "Петля" | |

Рисунок 3 – Виділені IP-адреси

Адреса 127.0.0.1 є зарезервованою для організації зворотного зв'язку при тестуванні роботи програмного забезпечення вузла без реального відправлення пакета до мережі. Ця адреса має назву loopback.

Форма групової IP-адреси – multicast – означає, що даний пакет має бути доставлений відразу декільком вузлам, що утворюють групу з номером, зазначеним у полі адреси. Вузли самі ідентифікують себе, тобто визначають, до якій із груп вони відносяться. Один вузол може входити у кілька груп. Такі повідомлення на відміну від широкомовних називаються мультимовними.

У протоколі IP немає поняття широкомовності у тому смислі, в якому воно використовується в протоколах канального рівня локальних мереж, коли дані повинні бути доставлені абсолютно усім вузлам. Так само, як обмежена широкомовна IP-адреса, так і широкомовна IP-адреса має межі поширення в інтермережі - вона обмежені або мережею, до якої належить вузол - джерело пакета, або мережею, номер якої зазначений в адресі призначення. Тому розподіл мережі за допомогою маршрутизаторів на частини локаліз

### *Протокол RIP*

*Протокол RIP (Routing Information Protocol)* є одним з найстарших протоколів обміну маршрутною інформацією, однак він дотепер надзвичайно розповсюджений в обчислювальних мережах. Крім версії RIP для мереж TCP/IP, існує також версія RIP для мереж IPX/SPX компанії Novell.

У цьому протоколі всі мережі мають номери (спосіб утворення номеру залежить від використовуваного в мережі протоколу мережного рівня), а всі маршрутизатори - ідентифікатори. Протокол RIP широко використовує поняття "вектор відстаней". Вектор відстаней являє собою набір пара чисел, що є номерами мереж і відстанями до них у хопах.

### *Протокол OSPF*

*Протокол OSPF (Open Shortest Path Firs)* є досить сучасною реалізацією алгоритму стану зв'язків (він прийнятий у 1991 році) і має багато особливостей, що орієнтовані на застосування у великих гетерогенних мережах.

Протокол OSPF обчислює маршрути в IP-мережах, зберігаючи при цьому інші протоколи обміну маршрутною інформацією.

### *Протокол TCP*

Транспортний рівень забезпечує сеанси зв'язку між комп'ютерами. Існує два основних транспортних протоколи: TCP (Transmission Control Protocol) і UDP (User Datagram Protocol). Використання одного з них залежить від обраного методу доставки даних.

TCP орієнтований на з'єднання і використовується прикладеннями, що зазвичай передають великі обсяги даних за одну операцію, тому що забезпечує надійне з'єднання, а також прикладеннями, яким необхідне підтвердження прийому даних.

### *Протокол UDP*

Протокол UDP забезпечує не орієнтовану на з'єднання передачу даних і не гарантує доставку пакетів. Прикладення, що використовують протокол UDP, звичайно передають невеликі обсяги даних за одну операцію. Відповідальність за надійну доставку даних несе прикладення.

### *Протокол TELNET*

Протокол TELNET дозволяє обслуговуючій машині розглядати усі визначені термінали як стандартні "мережні віртуальні термінали" рядкового типу, що працюють у коді ASCII, а також забезпечує можливість узгодження більш складних функцій (наприклад, локальний чи визначений луна-контроль, сторінковий режим, висота і ширина екрану і т.д.). TELNET працює на базі протоколу TCP. На прикладному рівні над TELNET знаходиться або програма підтримки реального терміналу (на стороні користувача), або прикладний процес в обслуговуючій машині, до якого здійснюється доступ з терміналу.

Протокол TELNET працює у діалоговий спосіб. Користувач вводить необхідні команди, на що протокол дає відповіді. Для того, щоб зв'язатись із машиною всередині мережі іноді замало знати її доменне ім'я, як правило необхідно вказати IP-адресу машини до якої звертається користувач, а також порт по якому знаходиться сервіс, який йому необхідний. За допомогою цього протоколу можна не тільки тестувати машини, що знаходяться всередині мережі, а й перевірити свою власну машину на можливість формування та відсилання пакетів (про що було розказано в розділі 4) та інш.

Протокол TELNET існує вже давно. Він добре випробуваний і широко розповсюджений. Створено безліч реалізацій для самих різних операційних систем.

### *Протокол FTP*

Протокол FTP (File Transfer Protocol - протокол передачі файлів) розповсюджений також широко як TELNET. Він є одним з найстарших протоколів сімейства TCP/IP. Також як TELNET він користується транспортними послугами TCP. Існує безліч реалізацій для різних операційних систем, що добре взаємодіють між собою.

FTP - стандартна програма, що поставляється з операційною системою. Її

основне призначення - передача файлів між різними комп'ютерами, що працюють у мережах tcp/ip: на одному з комп'ютерів працює програма-сервер, на другому користувач запускає програму-клієнта, що з'єднується із сервером і передає або одержує файли. Сервер FTP будується таким чином, що з'єднатися з ним можна не тільки під своїм ім'ям, але і під умовним ім'ям anonymous - анонім. Тоді стає доступна не уся файлова система комп'ютера, але деякий набір файлів на сервері, що складають вміст сервера anonymous ftp - публічного файлового архіву. Отже, якщо хтось хоче надати в публічне користування файли з інформацією, програмами й іншим, то достатньо організувати на комп'ютері, включеному в Інтернет, сервер anonymous ftp. Зробити це досить просто, програми-клієнти ftp є практично на будь-якому комп'ютері - тому сьогодні публічні файлові архіви організовані в основному як сервери anonymous ftp. На серверах доступна велика кількість інформації і програмного забезпечення.

### *Протокол SMTP*

Протокол SMTP (Simple Mail Transfer Protocol - простий протокол передачі пошти) підтримує передачу повідомлень (електронної пошти) між довільними вузлами мережі internet. Маючи механізми проміжного збереження пошти і механізми підвищення надійності доставки, протокол SMTP допускає використання різних транспортних служб. Він може працювати навіть у мережах, що не використовують протоколи сімейства TCP/IP. Протокол SMTP забезпечує як групування повідомлень на адресу одного одержувача, так і розмноження декількох копій повідомлення для передачі в різні адреси. Над модулем SMTP розташовується поштова служба конкретних обчислювальних систем.

Взаємодія в рамках SMTP будується за принципом двостороннього зв'язку, що встановлюється між відправником і одержувачем поштового повідомлення (рис. 10). При цьому відправник ініціює з'єднання і посилає запити на обслуговування, а одержувач - відповідає на ці запити. Фактично відправник виступає в ролі клієнта, а одержувач - сервера.

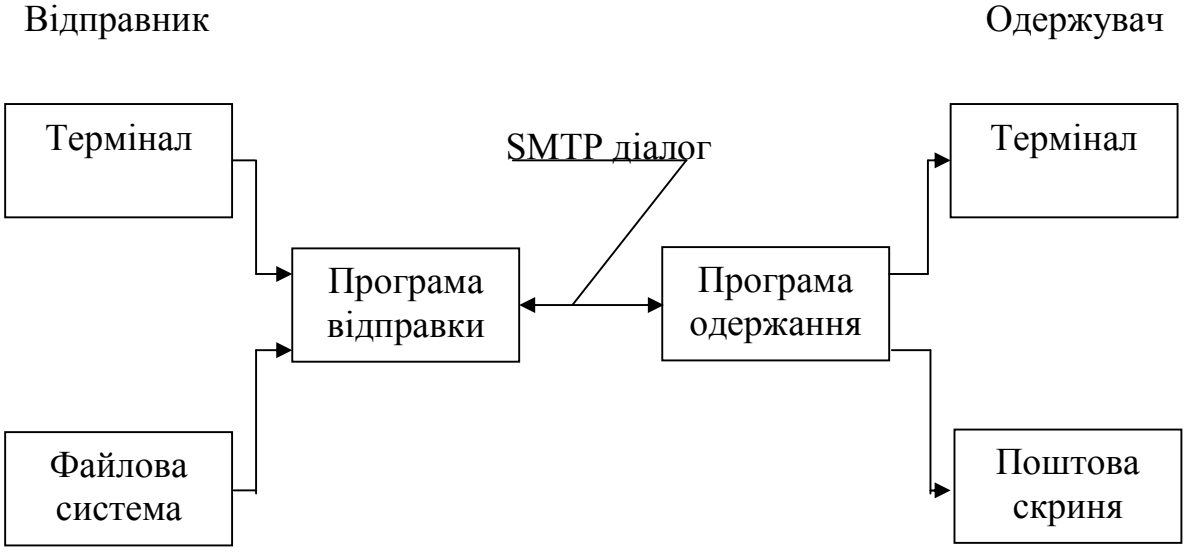



Рисунок 10 – Схема взаємодії протоколу SMTP

Канал зв'язку встановлюється безпосередньо між відправником і одержувачем повідомлення. При такій взаємодії пошта досягає абонента протягом декількох секунд

після відправлення.

Простий протокол передачі пошти забезпечує двосторонній обмін повідомленнями між локальним клієнтом і сервером. Обмін командами і відповідями на них називається поштовою транзакцією (mail transaction). Команди простого протоколу передачі пошти (SMTP) представлені в табл. 12.

У відповідності до специфікації команди, позначені хрестиком (X) у табл. 12, зобов'язані бути присутнім у будь-якій реалізації SMTP. Інші команди SMTP можуть бути реалізовані додатково. Кожна SMTP-команда повинна закінчуватися або пробілом (якщо в неї є аргумент), або комбінацією CRLF. В описі команд вживалося слово “дані”, а не <повідомлення>. Цим підкреслювалося, що, крім тексту, SMTP дозволяє передавати і двійкову інформацію, наприклад графічні чи звукові файли. Іншими словами, SMTP здатний передавати дані будь-якого змісту, а не тільки текстові повідомлення.

Таблиця 12

Специфікації команд

| Команда | Обов'язкова | Опис |
|---|---|---|
| **HELO** | X | Ідентифікує модуль-передавач для модуля-приймача (hello). |
| **MAIL** | X | Починає поштову транзакцію, що завершується передачею даних в одну чи кілька поштових скринь (mail). |
| **RCPT** | X | Ідентифікує одержувача поштового повідомлення (recipient). |
| **DATA** | | Рядки, що слідують за цією командою, розглядаються одержувачем як дані поштового повідомлення. У випадку SMTP, поштове повідомлення закінчується комбінацією символів: CRLF-точка-CRLF. |
| **RSET** | | Перериває поточну поштову транзакцію (reset). |
| **NOOP** | | Вимагає від одержувача не починати ніяких дій, а тільки видати відповідь ОК. Використовується головним чином для тестування.(No operation). |
| **QUIT** | | Вимагає видати відповідь ОК і закрити поточне з'єднання. |
| **VRFY** | | Вимагає від приймача підтвердження, що її аргумент є дійсним ім'ям користувача. (Див. Примітка.). |
| **SEND** | | Починає поштову транзакцію, що доставляє дані на один чи кілька терміналів (а не в поштову скриньку). |
| **SOML** | | Починає транзакцію MAIL чи SEND, що доставляє дані на один чи кілька терміналів або у поштові скриньки. |
| **SAML** | | Починає транзакцію MAIL і SEND, що доставляють дані на один чи кілька терміналів і в поштові скриньки. |
| **EXPN** | | Команда вимагає від SMTP-приймача підтвердження, чи дійсно аргумент є адресою поштового розсилання і якщо так, повернути адресу одержувача повідомлення (expand). |
| **HELP** | | Команда вимагає від SMTP-приймача повернути повідомлення-довідку про його команди. |

| | | |
|---|---|---|
| **TURN** | | Команда вимагає від SMTP-приймача або сказати ОК і помінятися ролями, тобто стати STMP- передавачем, або послати повідомлення-відмовлення і залишитися в ролі SMTP-приймача. |

У будь-який момент під час транзакції клієнт може використовувати команди NOOP, HELP, EXPN і VRFY. У відповідь на кожну команду сервер висилає клієнту визначену інформацію. Звичайно, у залежності від відповіді клієнт може почати конкретні дії, однак специфікація SMTP нічого не говорить з цього приводу. Наприклад, клієнт може передати команду VRFY для того, щоб переконатися, що ім'я користувача дійсно. Якщо сервер відповість, що даного імені не існує, клієнт може не передавати пошту для цього користувача. У специфікації SMTP, однак, немає ніяких указівок з цього приводу - клієнт може нічого не робити у відповідь на команду VRFY.

*Коди відповідей SMTP.* У специфікації SMTP потрібно, щоб сервер відповідав на кожну команду SMTP-клієнта. Сервер відповідає тризначною комбінацією цифр, що називається кодом відповіді. Разом з кодом відповіді, як правило, передається один чи кілька рядків текстової інформації.

Кілька рядків тексту, як правило, супроводжують тільки команди EXPN і HELP.У специфікації SMTP, однак, відповідь на будь-яку команду може складатися з декількох рядків тексту.

Кожна цифра в коді відповіді має визначений сенс. Перша цифра означає, чи було виконання команди успішним (2), неуспішним (5), чи ще не закінчилося (3). Як зазначено в додатку Е документа RFC 821, простий SMTP-клієнт може аналізувати тільки першу цифру у відповіді сервера, і на підставі її продовжувати свої дії. Друга і третя цифри коду відповіді роз'ясняють значення першої.

*Проміжні агенти.* Термін "маршрут доставки" (forward-path) служить для того, щоб відрізняти поштову скринька (mailbox), ім'я якої абсолютно, від шляху (він може бути різним), по якому слідує пошта. Припустимо, що ми хочемо доставити два поштових повідомлення за однією адресою. Обидва повідомлення мають один пункт призначення, однак не обов'язково будуть слідувати ідентичному маршруту. Як правило, конкретний маршрут для пошти вибирається системним адміністратором. Щоб направити пошту за потрібним шляхом, використовуються значення маршруту доставки і зворотного маршруту, у яких указуються проміжні агенти (relay agents). Проміжний агент доставки – це так званий поштовий хаб (mail hub), який настроєний на передачу транзитної пошти.

У наш час проміжні агенти присутні практично у всіх мережах, що входять у Internet. Щоб спростити процес конфігурації поштової системи, у локальній мережі встановлюється один комп'ютер, що служить проміжним агентом (relay host). Уся пошта користувачів попадає спочатку на нього. Потім цей комп'ютер розсилає повідомлення по Internet. Крім того, такий комп'ютер може служити захистом фірми від злодіїв-хакерів. Обмежуючи спілкування локальної мережі з зовнішнім світом до рівня пошти, організація зводить до мінімуму ризик небажаного вторгнення у свої власні системи.

Крім того, адмініструвати і захищати в цьому випадку приходиться єдиний комп'ютер. SMTP у змозі послати повідомлення безпосередньо з комп'ютера користувача на комп'ютер адресата в тому випадку, якщо між ними існує пряме

поштове з'єднання. Як правило, між двома комп'ютерами знаходиться один чи кілька проміжних агентів. Щоб забезпечити доставку, у поштовому повідомленні потрібно вказати ім'я комп'ютера-одержувача і точне найменування поштової скриньки. Аргументом команди MAIL є зворотний маршрут, що включає ім'я джерела повідомлення й імена всіх проміжних агентів. Аргумент команди RCPT - маршрут доставки, що містить ім'я одержувача повідомлення. Зворотний маршрут описує шлях, який пройшло повідомлення, тоді як маршрут доставки ідентифікує місце призначення. Зворотний маршрут використовується SMTP, коли потрібно передати повідомлення про помилку або про неможливість доставки повідомлення, коли воно вже пройшло через проміжний агент. В міру просування повідомлення по Internet записи про його маршрут змінюються. До обов'язків системних адміністраторів входить правильна побудова місцевих локальних агентів на передачу повідомлень проміжному агенту, і навпаки, проміжні агенти на доставку повідомлень місцевим агентам. Якщо в проміжного агента зміниться ім'я, усе, що потрібно зробити в конфігурації місцевого агента - змінити ім'я комп'ютера в системі DNS. Інші параметри конфігурації не змінюються.

Зворотні маршрути і маршрути доставки будуються агентами передачі пошти в міру проходження повідомлення від одного агента до наступного. Якщо черговий на шляху сполучення SMTP-агент не вміє обслуговувати проміжну доставку, він має відповісти кодом, який передбачений на випадок відсутності місцевої поштової

## §4. Основи та структура міжмережної взаємодії

Усі протоколи обміну маршрутною інформацією стеку TCP/IP відносяться до класу адаптивних протоколів, що у свою чергу поділяються на дві групи, кожна з яких зв'язана з одним із наступних типів алгоритмів:

дистанційно-векторний алгоритм (Distance Vector Algorithms, DVA),

алгоритм стану зв'язків (Link State Algorithms, LSA).

В алгоритмах дистанційно-векторного типу кожен маршрутизатор періодично і широкомовно розсилає по мережі вектор відстаней від себе до усіх відомих йому мереж. Під відстанню звичайно розуміється число проміжних маршрутизаторів через які пакет має пройти перш, ніж потрапить у відповідну мережу. Може використовуватися й інша метрика, що враховує не тільки число перехідних пунктів, але і час проходження пакетів між сусідніми маршрутизаторами. Одержавши вектор від сусіднього маршрутизатора, кожен маршрутизатор додає до нього інформацію про відомі йому інші мережі, про які він довідався безпосередньо (якщо вони підключені до його портів) чи з аналогічних оголошень інших маршрутизаторів, а потім знову розсилає нове значення вектора по мережі. У кінців-кінців, кожен маршрутизатор отримує інформацію про існуючі в мережі маршрутизатори і про відстань до них через сусідні маршрутизатори.

Дистанційно-векторні алгоритми добре працюють тільки в невеликих мережах. У великих мережах вони завантажують лінії зв'язку інтенсивним широкомовним трафіком, до того ж зміни конфігурації можуть відпрацьовуватися за цим алгоритмом не завжди коректно, тому що маршрутизатори не мають точного представлення про топологію зв'язків у мережі, а мають у своєму розпорядженні тільки узагальнену інформацію - вектор дистанції, до того ж отриманий через посередників. Робота маршрутизатора відповідно до дистанційно-векторного протоколу нагадує роботу

моста, тому що точної топологічної картини мережі такий маршрутизатор не має.

Найбільш розповсюдженим протоколом, заснованим на дистанційно-векторному алгоритмі, є протокол RIP.

Алгоритми стану зв'язків забезпечують кожен маршрутизатор інформацією, достатньою для побудови точного графа зв'язків мережі. Усі маршрутизатори працюють на підставі однакових графів, що робить процес маршрутизації більш стійким до змін конфігурації. Широкомовне розсилання використовується тут тільки при змінах стану зв'язків, що відбувається в надійних мережах не так часто.

Для того, щоб зрозуміти, у якому стані знаходяться лінії зв'язку, підключені до його портів, маршрутизатор періодично обмінюється короткими пакетами зі своїми найближчими сусідами. Цей трафік також широкомовний, але він циркулює тільки між сусідами і тому не сильно завантажує мережу.

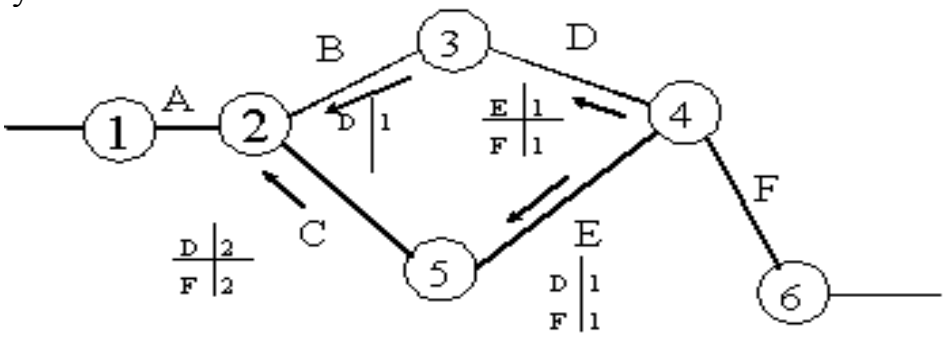
Протоколом, заснованим на алгоритмі стану зв'язків, у стеці TCP/IP є протокол OSPF.

*Протокол RIP (Routing Information Protocol)* є одним з найстарших протоколів обміну маршрутною інформацією, однак він дотепер надзвичайно розповсюджений в обчислювальних мережах. Крім версії RIP для мереж TCP/IP, існує також версія RIP для мереж IPX/SPX компанії Novell.

У цьому протоколі всі мережі мають номери (спосіб утворення номеру залежить від використовуваного в мережі протоколу мережного рівня), а всі маршрутизатори - ідентифікатори. Протокол RIP широко використовує поняття "вектор відстаней". Вектор відстаней являє собою набір пара чисел, що є номерами мереж і відстанями до них у хопах.

Вектора відстаней ітераційно поширюються маршрутизаторами по мережі, і через кілька кроків кожен маршрутизатор має дані про досяжні для нього мережі і про відстані до них. Якщо зв'язок з якою-небудь мережею обривається, то маршрутизатор відзначає цей факт тим, що привласнює елементу вектора, що відповідає відстані до цієї мережі, максимально можливе значення, що має спеціальний сенс - "зв'язку немає". Таким значенням у протоколі RIP є число 16.

На рис. 4 приведений приклад мережі, що складається із шести маршрутизаторів, що мають ідентифікатори від 1 до 6, і із шести мереж від A до F, утворених прямими зв'язками типу "точка-точка".




Початкова інформація у вузлі 2:

| Мережа | Відстань | Сусід |
|---|---|---|
| A | 1 | - |
| B | 1 | - |
| C | 1 | - |

Після 2-х кроків

| | | |
|---|---|---|
| A | 1 | - |
| B | 1 | - |
| C | 1 | - |
| D | 2 | 3 |
| E | 2 | 5 |
| D | 3 | 5 |
| F | 3 | 5 |

Рисунок 4 – Обмін маршрутною інформацією в протоколі RIP

На малюнку приведена початкова інформація, що міститься в топологічній базі маршрутизатора 2, а також інформація в цій же базі після двох ітерацій обміну маршрутними пакетами протоколу RIP. Після визначеного числа ітерацій маршрутизатор 2 буде знати про відстані до всіх мереж інтермережі, причому в нього може бути кілька альтернативних варіантів відправлення пакета до мережі призначення. Нехай у нашому прикладі мережею призначення є мережа D.

При необхідності відправити пакет у мережу D маршрутизатор переглядає свою базу даних маршрутів і вибирає порт, що має найменшу відстань до мережі призначення (у даному випадку порт, що зв'язує його з маршрутизатором 3).

Для адаптації до зміни стану зв'язків, з кожним записом таблиці маршрутизації пов'язують таймер. Якщо за час тайм-ауту не прийде нове повідомлення, що підтверджує цей маршрут, то він видаляється з маршрутної таблиці.

При використанні протоколу RIP працює евристичний алгоритм динамічного програмування Беллмана-Форда, і рішення, знайдене з його допомогою є не оптимальним, а близьким до оптимального. Перевагою протоколу RIP є його обчислювальна простота, а недоліками - збільшення трафіку при періодичному розсиланні широкомовних пакетів і неоптимальність знайденого маршруту.

На рис. 5 показано випадок помилкової роботи мережі, що обслуговується протоколом RIP при зміні конфігурації - відмовленні лінії зв'язку маршрутизатора М1 з мережею 1. При працездатному стані цього зв'язку в таблиці маршрутів кожного маршрутизатора є запис про мережу з номером 1 і відповідною відстанню до неї.

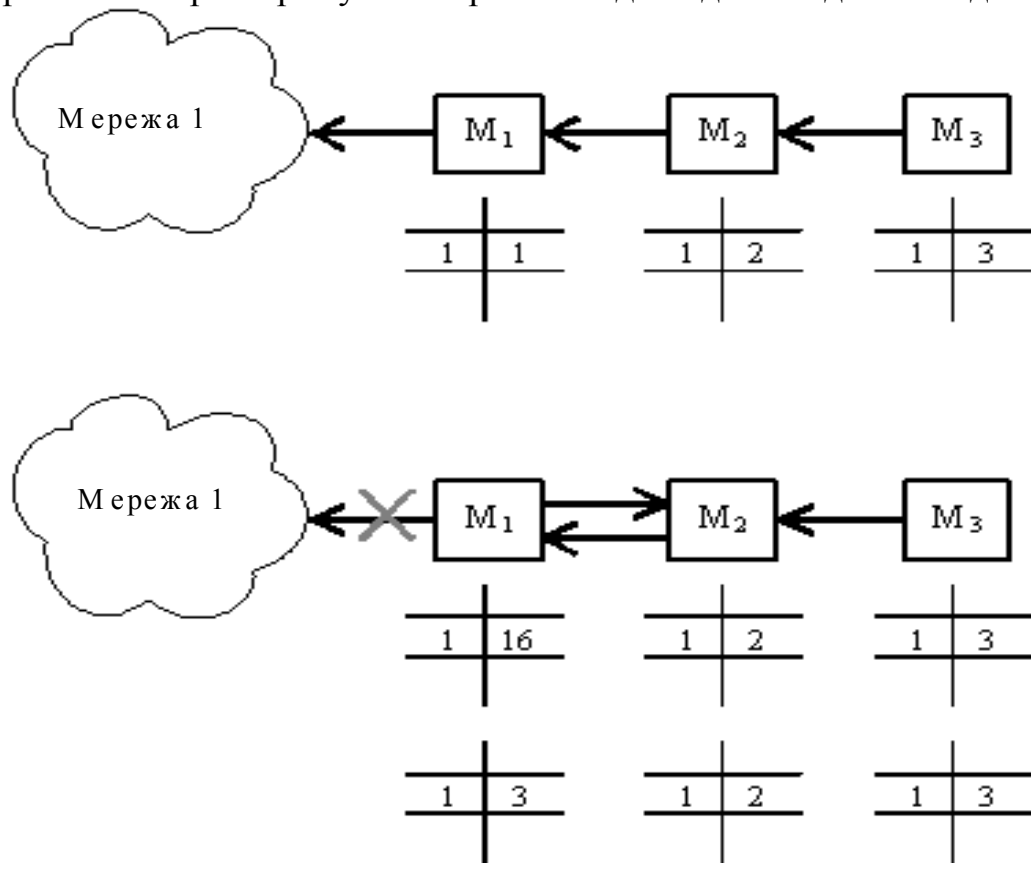



Рисунок 5 – Приклад невірної роботи мережі при використанні протоколу RIP

При обриві зв'язку з мережею 1 маршрутизатор М1 відзначає, що відстань до цієї мережі прийнялася значення 16. Однак одержавши через якийсь час від маршрутизатора М2 маршрутне повідомлення про те, що від нього до мережі 1 відстань складає 2 хопа, маршрутизатор М1 нарощує цю відстань на 1 і відзначає, що мережа 1 досяжна через маршрутизатор 2. У результаті пакет, призначений для мережі 1, буде циркулювати між маршрутизаторами М1 і М2 доти, поки не мине час збереження запису про мережу 1 у маршрутизаторі 2, і він не передасть цю інформацію маршрутизатору М1.

Для виключення подібних ситуацій маршрутна інформація про відому маршрутизатору мережу не передається тому маршрутизатору, від якого вона прийшла.

*Протокол OSPF (Open Shortest Path Firs)* є досить сучасною реалізацією алгоритму стану зв'язків (він прийнятий у 1991 році) і має багато особливостей, що орієнтовані на застосування у великих гетерогенних мережах.

Протокол OSPF обчислює маршрути в IP-мережах, зберігаючи при цьому інші протоколи обміну маршрутною інформацією.

Безпосередньо зв'язані (тобто досяжні без використання проміжних маршрутизаторів) маршрутизатори називаються "сусідами". Кожен маршрутизатор зберігає інформацію про те, у якому стані на його думку знаходиться сусід. Маршрутизатор покладається на сусідні маршрутизатори і передає їм пакети даних тільки в тому випадку, якщо він упевнений, що вони цілком працездатні. Для з'ясування стану зв'язків маршрутизатори-сусіди досить часто обмінюються короткими повідомленнями HELLO.

Для поширення по мережі даних про стан зв'язків маршрутизатори обмінюються повідомленнями іншого типу. Ці повідомлення називаються router links advertisement - оголошення про зв'язки маршрутизатора (точніше, про стан зв'язків). OSPF-маршрутизатори обмінюються не тільки своїми, але і чужими оголошеннями про зв'язки, одержуючи в кінців-кінців інформацію про стан усіх зв'язків мережі. Ця інформація й утворить граф зв'язків мережі, що є однаковим для всіх маршрутизаторів мережі.

Крім інформації про сусідів, маршрутизатор у своєму оголошенні перелічує IP-підмережі, з якими він зв'язаний безпосередньо, тому після одержання інформації про граф зв'язків мережі, обчислення маршруту до кожної мережі здійснюється безпосередньо по цьому графу по алгоритму Дейкстри. Більш точно, маршрутизатор обчислює шлях не до конкретної мережі, а до маршрутизатора, до якого ця мережа підключена. Кожен маршрутизатор має унікальний ідентифікатор, що передається в оголошенні про стани зв'язків. Такий підхід дає можливість не витрачати IP-адреси на зв'язку типу "точка-точка" між маршрутизаторами, до яких не підключені робочі станції.

Маршрутизатор обчислює оптимальний маршрут до кожної адресованої мережі, але запам'ятовує тільки перший проміжний маршрутизатор з кожного маршруту. Таким чином, результатом обчислень оптимальних маршрутів є список рядків, у яких указується номер мережі й ідентифікатор маршрутизатора, якому потрібно переслати пакет для цієї мережі. Зазначений список маршрутів і є маршрутною таблицею, але обчислений він на підставі повної інформації про граф зв'язків мережі, а не часткової

інформації, як у протоколі RIP.

Описаний підхід приводить до результату, що не може бути досягнуть при використанні протоколу RIP чи інших дистанційно-векторних алгоритмів. RIP припускає, що всі підмережі визначеної IP-мережі мають той самий розмір, тобто, що усі вони можуть потенційно мати однакове число IP-вузлів, адреси яких не перекриваються. Більш того, класична реалізація RIP вимагає, щоб виділені лінії "точка-точка" мали IP-адресу, що приводить до додаткових витрат IP-адрес.

У OSPF такі вимоги відсутні: мережі можуть мати різне число хостів і можуть перекриватися. Під перекриттям розуміється наявність декількох маршрутів до однієї і тієї ж мережі. У цьому випадку адреса мережі в пакеті, що прийшов, може збігтися з адресою мережі, що привласнена декільком портам.

Якщо адреса належить декільком підмережам у базі даних маршрутів, то пакет маршрутизується по найбільш специфічному маршруту, тобто за адресою підмережі, що має більш довгу маску.

Наприклад, якщо робоча група відгалужується від головної мережі, то вона має адресу головної мережі поряд з більш специфічною адресою, що обумовлена маскою підмережі. При виборі маршруту до хосту в підмережі цієї робочої групи маршрутизатор знайде два шляхи, один для головної мережі й один для робочої групи. Тому що останній більш специфічний, то він і буде обраний. Цей механізм є узагальненням поняття "маршрут за замовчуванням", що використовується в багатьох мережах.

Використання підмереж з різною кількістю хостів є цілком природним. Наприклад, якщо в будинку на кожнім поверсі є локальні мережі, і на деяких поверхах комп'ютерів більше, ніж на інших, то адміністратор може вибрати розміри підмереж такими, що необхідні для кожного поверху, а не у відповідності до розміру найбільшої підмережі.

У протоколі OSPF підмережі поділяються на три категорії:

"хост-мережа", що представляє собою підмережу з однієї адреси,

"тупікова мережа", що являє собою підмережу, підключену тільки до одного маршрутизатора,

"транзитна мережа", що являє собою підмережу, підключену до більш ніж одного маршрутизатора.

Транзитна мережа є особливим випадком для протоколу OSPF. У транзитній мережі кілька маршрутизаторів є взаємно й одночасно досяжними. У широкомовних локальних мережах, таких як Ethernet чи Token Ring, маршрутизатор може послати одне повідомлення, яке одержать усі його сусіди. Це зменшує навантаження на маршрутизатор, коли він посилає повідомлення для визначення існування чи зв'язків між сусідами. Однак, якщо кожен маршрутизатор буде перелічувати усіх своїх сусідів у своїх оголошеннях, то вони займуть багато місця в пам'яті маршрутизатора. При визначенні шляху за адресами транзитної підмережі може виявитися багато надлишкових маршрутів. На обчислення, перевірку й відкидання цих маршрутів піде багато часу.

Коли маршрутизатор починає працювати в перший раз (тобто інсталюється), він намагається синхронізувати свою базу даних із усіма маршрутизаторами транзитної локальної мережі, що по визначенню мають ідентичні бази даних. Для спрощення й оптимізації цього процесу в протоколі OSPF використовується поняття "виділеного"

маршрутизатора, що виконує дві функцій.

По-перше, маршрутизатор буде синхронізувати свою базу тільки з виділеним маршрутизатором і його резервний "напарником. Синхронізувавши базу з виділеним маршрутизатором, новий маршрутизатор буде синхронізований із усіма маршрутизаторами даної локальної мережі.

По-друге, виділений маршрутизатор оголошує про мережні зв'язки, перелічуючи своїх сусідів по підмережі. Інші маршрутизатори просто повідомляють про свій зв'язок з виділеним маршрутизатором. Це робить оголошення про зв'язки (яких багато) більш короткими, розміром з оголошення про зв'язки окремої мережі.

Для початку роботи маршрутизатору OSPF потрібно мінімум інформації - IP-конфігурація (IP-адреси і маски підмереж), деяка інформація по умовчанню (default) і команда на включення. Для багатьох мереж інформація по умовчанню дуже схожа. У той же час протокол OSPF передбачає високий ступінь програмованості.

Інтерфейс OSPF (порт маршрутизатора, що підтримує протокол OSPF) є узагальненням підмережі IP. Подібно підмережі IP, інтерфейс OSPF має IP-адресу і маску підмережі. Якщо один порт OSPF підтримує більш, ніж одну підмережу, протокол OSPF розглядає ці підмережі так, ніби вони були на різних фізичних інтерфейсах, і обчислює маршрути відповідно.

Інтерфейси, до яких підключені локальні мережі, називаються широкомовними (broadcast) інтерфейсами, тому що вони можуть використовувати широкомовні можливості локальних мереж для обміну сигнальною інформацією між маршрутизаторами. Інтерфейси, до яких підключені глобальні мережі, що не підтримують широкомовлення, але забезпечують доступ до багатьох вузлів через одну точку входу, наприклад мережі X.25 чи Frame Relay, називаються нешироковомовними інтерфейсами з множинним доступом NBMA (non-broadcast multi-access). Вони розглядаються аналогічно широкомовним інтерфейсам за винятком того, що широкомовне розсилання емулюється шляхом посилки повідомлення кожному сусіду. Так як виявлення сусідів не є автоматичним, NBMA-сусіди повинні задаватися при конфігуруванні вручну. Як на широкомовних, так і на NBMA-інтерфейсах можуть бути задані пріоритети маршрутизаторів для того, щоб вони могли вибрати виділений маршрутизатор.

Інтерфейси "точка-точка", подібні PPP, трохи відрізняються від традиційної IP-моделі. Хоча вони і можуть мати IP-адреси і підмаски, але необхідності в цьому немає.

У простих мережах досить визначити, що пункт призначення досяжний і знайти маршрут. У складних мережах звичайно є кілька можливих маршрутів. Іноді було б добре мати більше можливостей по встановленню додаткових критеріїв для вибору шляху: наприклад, найменша затримка, максимальна пропускна здатність чи найменша вартість. З цих причин протокол OSPF дозволяє мережному адміністратору призначати кожному інтерфейсу визначене число, що називається метрикою, щоб уплинути на вибір маршруту.

Число, що використовується як метрика шляху, може бути призначено довільним чином за бажанням адміністратора. Але по умовчанню як метрика використовується час передачі біта в 10-ти наносекундних одиницях (10 Мб/с Ethernet призначається значення 10, а лінії 56 Кб/с - число 1785). Метрика, що обчислюється протоколом OSPF для шляху, являє собою суму метрик усіх прохідних зв'язків; це дуже груба оцінка затримки шляху. Якщо маршрутизатор виявляє більш, ніж один

шлях до зазначеної підмережі, то буде обрано той, що має найменшу вартістю шляху.

У протоколі OSPF використовується кілька тимчасових параметрів, і серед них найбільш важливими є інтервал повідомлення HELLO і інтервал відмовлення маршрутизатора (router dead interval).

HELLO - це повідомлення, яким обмінюються сусідні, тобто безпосередньо зв'язані маршрутизатори підмережі, з метою установити стан лінії зв'язку і стан маршрутизатора-сусіда. У повідомленні HELLO маршрутизатор передає свої робочі параметри для тих, кого він розглядає як своїх найближчих сусідів. Маршрутизатори з різними робочими параметрами ігнорують повідомлення HELLO один одного, тому невірно сконфігуровані маршрутизатори не будуть впливати на роботу мережі. Кожен маршрутизатор шле повідомлення HELLO кожному своєму сусіду принаймні один раз протягом інтервалу HELLO. Якщо інтервал відмовлення маршрутизатора минає без одержання повідомлення HELLO від сусіда, то вважається, що сусід непрацездатний, і поширюється нове оголошення про мережні зв'язки, щоб у мережі відбулася перебудова маршрутів.

## §5. Протоколи ARP та RARP

### *Протокол ARP*

Локальна адреса використовується в протоколі IP тільки в межах локальної мережі при обміні даними між маршрутизатором і вузлом цієї мережі. Маршрутизатор, одержавши пакет для вузла однієї з мереж, безпосередньо підключених до його портів, повинний для передачі пакета сформувати кадр відповідно до вимог прийнятої в цій мережі технології і вказати в ньому локальну адресу вузла, наприклад його Mac-адрес. У пакеті, що прийшов, ця адреса не зазначена, тому перед маршрутизатором устає задача пошуку його по відомій IP-адресі, що зазначена в пакеті як адреса призначення. З аналогічною задачею стикається і кінцевий вузол, коли він хоче відправити пакет у виділену мережу через маршрутизатор, підключений до тієї ж локальної мережі, що і даний вузол.

Для того щоб установити з'єднання, вузлам повинні бути відомі адреси мережних адаптерів інших вузлів. Дозвіл адреси (address resolution) - це процес визначення апаратної адреси мережного адаптера по його IP-адресі. Протокол ARP (Address Resolution Protocol) - частина рівня Інтернету моделі TCP/IP - дозволяє визначати адреси мережних адаптерів вузлів, розташованих в одній фізичній мережі.

Протокол ARP працює різним чином, у залежності від того, який протокол канального рівня працює в даній мережі - протокол локальної мережі (Ethernet, Token Ring, FDDI) з можливістю широкомовного доступу одночасно до усіх вузлів мережі, чи ж протокол глобальної мережі (X.25, frame relay), як правило не підтримуючий широкомовний доступ. Існує також протокол, що вирішує зворотну задачу - визначення IP-адреси по відомій локальній адресі. Він називається реверсивний ARP - RARP (Reverse Address Resolution Protocol) і використовується при старті бездискових станцій, що не знають у початковий момент своєї IP-адреси, але знають адресу свого мережного адаптера.

У локальних мережах протокол ARP використовує широкомовні кадри протоколу канального рівня для пошуку в мережі вузла з заданою IP-адресою.

Вузол, якому потрібно виконати відображення IP-адреси на локальну адресу, формує ARP запит, вкладає його в кадр протоколу канального рівня, вказуючи в ньому

відомий IP-адрес, і розсилає запит широкомовно. Усі вузли локальної мережі одержують ARP запит і порівнюють зазначений там IP-адреса з власним. У випадку їхнього збігу вузол формує ARP-відповідь, у якому вказує свій IP-адрес і свою локальну адресу, і відправляє його вже направлено, тому що в ARP запиті відправник указує свою локальну адресу. ARP-запити і відповіді використовують однаковий формат пакета. Локальні адреси можуть у різних типах мереж мати різну довжину, тому формат пакета протоколу ARP залежить від типу мережі. В табл. 2 показано формат пакета протоколу ARP для передачі по мережі Ethernet.

Таблиця 2

Формат пакету

<table>
<tr><td colspan="2">Тип мережі</td><td>Тип протоколу</td></tr>
<tr><td>Довжина локальної адреси</td><td>Довжина мережевої адреси</td><td>Операція</td></tr>
<tr><td colspan="2">Локальна адреса відправника (байти 0 - 3)</td><td></td></tr>
<tr><td colspan="2">Локальна адреса відправника (байти 4 - 5)</td><td>IP-адреса відправника (байти 0-1)</td></tr>
<tr><td colspan="2">IP-адреса відправника (байти 2-3)</td><td>Шукана локальна адреса (байти 0 - 1)</td></tr>
<tr><td colspan="2">Шукана локальна адреса (байти 2-5)</td><td></td></tr>
<tr><td colspan="2">Шукана IP-адреса (байти 0 - 3)</td><td></td></tr>
</table>

Вузол, що відправляє ARP-запит, заповнює в пакеті всі поля, крім поля шуканої локальної адреси (для RARP-запиту не вказується шукана IP-адреса). Значення цього поля заповнюється вузлом, що пізнав свою IP-адресу.

У глобальних мережах адміністратору мережі найчастіше приходиться вручну формувати ARP-таблиці, у яких він задає, наприклад, відповідність IP-адреси адресі вузла мережі X.25. Останнім часом намітилася тенденція автоматизації роботи протоколу ARP і в глобальних мережах. Для цієї мети серед усіх маршрутизаторів, підключених до якої-небудь глобальної мережі, виділяється спеціальний маршрутизатор, що веде ARP-таблицю для всіх інших вузлів і маршрутизаторів цієї мережі. При такому централізованому підході для усіх вузлів і маршрутизаторів вручну потрібно задати тільки IP-адресу і локальну адресу виділеного маршрутизатора. Потім кожен вузол і маршрутизатор реєструє свої адреси у виділеному маршрутизаторі, а при необхідності встановлення відповідності між IP-адресою і локальною адресою вузол звертається до виділеного маршрутизатора з запитом і автоматично одержує відповідь без участі адміністратора.

### *Протокол RARP*

Протокол RARP виконує зворотною функцію, тобто визначення IP-адреси по відомій локальній адресі.

## Завдання на СРС

### *1. Вивчення стеку протоколів TCP/IP*

### *2. Поглиблене вивчення протоколів канального рівня*

# ЛЕКЦІЯ 2
# ЗАНЯТТЯ 2
# ПРОТОКОЛИ ТРАНСПОРТНОГО РІВНЯ

Транспортний рівень забезпечує сеанси зв'язку між комп'ютерами. Існує два основних транспортних протоколи: TCP (Transmission Control Protocol) і UDP (User Datagram Protocol). Використання одного з них залежить від обраного методу доставки даних.

TCP орієнтований на з'єднання і використовується прикладеннями, що зазвичай передають великі обсяги даних за одну операцію, тому що забезпечує надійне з'єднання, а також прикладеннями, яким необхідне підтвердження прийому даних.

Протокол UDP забезпечує не орієнтовану на з'єднання передачу даних і не гарантує доставку пакетів. Прикладення, що використовують протокол UDP, звичайно передають невеликі обсяги даних за одну операцію. Відповідальність за надійну доставку даних несе прикладення.

## §1. Алгоритм функціонування протоколу UDP. Формат UDP-повідомлення

Задачею протоколу транспортного рівня UDP (User Datagram Protocol) є передача даних між прикладними процесами без гарантії доставки, тому його пакети можуть бути загублені, продубльовані або можуть прийти не в тому порядку, у якому вони були відправлені.

У той час, як задачею мережного рівня є передача даних між довільними вузлами мережі, задача транспортного рівня полягає в передачі даних між будь-якими прикладними процесами, що виконуються на будь-яких вузлах мережі. Дійсно, після того, як пакет засобами протоколу IP доставлений у комп'ютер-одержувач, дані необхідно направити конкретному процесу-одержувачу. Кожен комп'ютер може виконувати кілька процесів, більш того, прикладний процес теж може мати кілька точок входу, що виступають як адреси призначення для пакетів даних.

Пакети, що надходять на транспортний рівень, організуються операційною системою у виді безлічі черг до точок входу різних прикладних процесів. У термінології TCP/IP такі системні черги називаються портами. Таким чином, адресою призначення, що використовується на транспортному рівні, є ідентифікатор (номер) порту прикладного сервісу. Номер порту, що задається транспортним рівнем, у сукупності з номером мережі і номером комп'ютера, що задаються мережним рівнем, однозначно визначають прикладний процес у мережі.

Призначення номерів портів прикладним процесам здійснюється або централізовано, якщо ці процеси являють собою популярні загальнодоступні сервіси, наприклад сервіс доступу до файлів TFTP (Trivial FTP) чи сервіс визначеного керування telnet, або локально для тих сервісів, що ще не стали настільки розповсюдженими, щоб за ними закріплювати стандартні (зарезервовані) номери.

Локальне присвоєння номера порту полягає в тому, що розроблювач деякого прикладення просто зв'язує з ним будь-який доступний, довільно обраний числовий ідентифікатор, звертаючи увагу на те, щоб він не входив у число зарезервованих номерів портів. Надалі усі виділені запити до даного прикладення від інших прикладень повинні адресуватися з указівкою призначеного йому номера порту.

Протокол UDP веде для кожного порту дві черги: черга пакетів, що надходять у

даний порт із мережі, і черга пакетів, що відправляються даним портом у мережу.

Процедура обслуговування протоколом UDP запитів, що надходять від декількох різних прикладних сервісів, називається мультиплексуванням.

Розподіл протоколом UDP пакетів, що надходять від мережного рівня, між набором високорівневих сервісів, ідентифікованих номерами портів, називається демультиплексуванням (рис. 9).

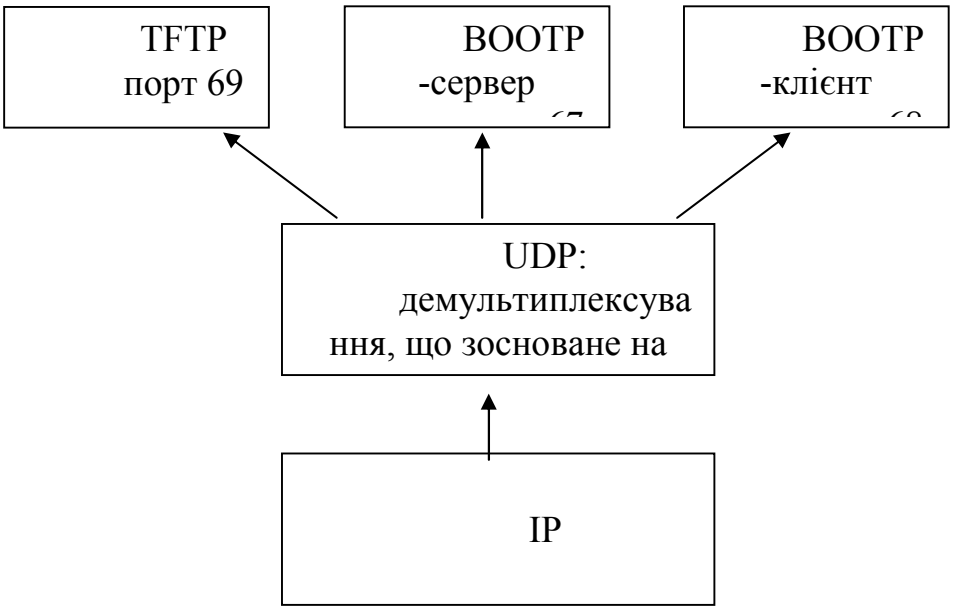



Рисунок 9 – Зображення процесу демультиплексування протоколу UDP

Протокол UDP виступає простим посередником між мережним рівнем і прикладними сервісами, і, на відміну від TCP, не бере на себе ніяких функцій по забезпеченню надійності передачі. UDP є дейтаграмним протоколом, тобто він не встановлює логічного з'єднання, не нумерує і не упорядковує пакети даних.

З іншого боку, функціональна простота протоколу UDP обумовлює простоту його алгоритму, компактність і високу швидкодія. Тому ті прикладення, у яких реалізований власний, досить надійний, механізм обміну повідомленнями, заснований на встановленні з'єднання, віддають перевагу при безпосередній передачі даних по мережі менш надійному, але більш швидкому засобу транспортування, у якості якого стосовно протоколу TCP і виступає протокол UDP. Звичайно такі прикладення передають дані невеликого обсягу за один раз. Приклади служб і додатків, що використовують UDP: сервіс імен NetBIOS, сервіс датаграм NetBIOS і сервіс SNMP. Протокол UDP може бути використаний і в тому випадку, коли гарна якість каналів зв'язку забезпечує достатній рівень надійності і без застосування додаткових прийомів типу встановлення логічного з'єднання і квитирування переданих пакетів.

Одиниця даних протоколу UDP називається UDP-пакетом або дейтаграмою користувача (user datagram). UDP-пакет складається з заголовка і поля даних, у якому розміщається пакет прикладного рівня. Заголовок має простий формат і складається з чотирьох двобайтових полів. Поля 8-байтного заголовку UDP-пакету перераховані в табл. 10:

Таблиця 10

Формат UDP-повідомлення

| Поле | Опис |
|---|---|
| Source Port (Порт відправника) | UDP-порт вузла-відправника. |
| Destination Port (Порт одержувача) | UDP-порт вузла-одержувача. Указує кінцеву точку з'єднання. |
| Message Length (Довжина повідомлення) | Розмір повідомлення. Мінімальний UPD-пакет містить тыльки інформацію заголовку (8 байт) |
| Checksum (Контрольна торба) | Перевіряє заголовок на предмет ушкодження |

Не всі поля UDP-пакета обов'язково повинні бути заповнені. Якщо що посилається дейтаграма, що не потребує відповіді, то на місці адреси відправника можуть міститися нулі. Можна відмовитися і від підрахунку контрольної суми, однак варто врахувати, що протокол IP підраховує контрольну суму тільки для заголовку IP-пакета, ігноруючи поле даних.

Для використання протоколу UDP додаток повинний знати IP-адреса і номер порту одержувача. Порт - місце призначення при доставці повідомлень - діє як мультиплексор черги повідомлень, тобто він може одержувати кілька повідомлень одночасно. Важливо відзначити, що порти протоколу UDP, перераховані в табл. 11, відрізняються від портів TCP, незважаючи на використання тих же значень номерів.

Таблиця 11

Порти протоколу UDP

| Номер порта | Ключове слово | Опис |
|---|---|---|
| 15 | NETSTAT | Стан мережі |
| 53 | DOMAIN | Сервер імен домену |
| 69 | TFTP | Протокол TFTP |
| 137 | NETBIOS-NS | Сервіс імен NetBIOS |
| 138 | NETBIOS-DGM | Сервіс датаграм NetBIOS |

## §2. Інкапсуляція і розділ за рівнями

Концепції мережних протоколів розвивалися протягом останніх двадцяти років і були спрямовані на забезпечення:

-логічної декомпозиції складної мережі на менші, зрозуміліші частини (рівні);

-стандартних інтерфейсів між мережними функціями, наприклад стандартних інтерфейсів між модулями програмного забезпечення;

-симетрії стосовно функцій, реалізованих у кожному вузлі мережі.

Кожний рівень у деякому вузлі мережі виконує ті ж самі функції, що й аналогічний рівень в іншому вузлі:

-забезпечує засоби передбачення змін і керування змінами, які можуть бути внесені в мережну логіку (програмне забезпечення або мікропрограми);

-реалізує просту стандартну мову комунікації розробників мереж,

адміністраторів, фірм-постачальників і користувачів, використовувану під час обговорення мережних функцій.

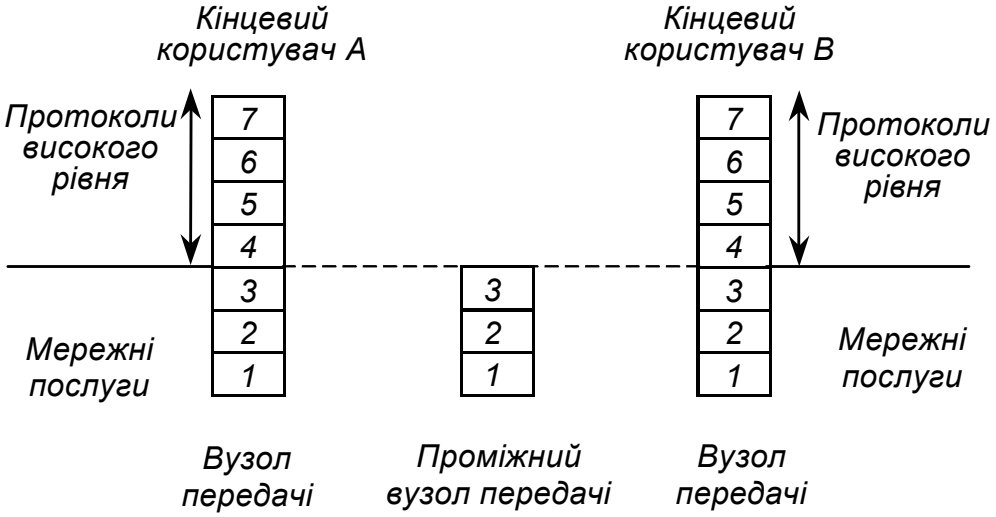



Рисунок 3.1. Архітектура багаторівневого зв’язку

Можна зробити висновок, що загальна проблема зв’язку, яка полягає в забезпеченні своєчасної, правильної та розпізнаваної доставки даних кінцевому користувачеві, зайнятому в сеансі зв’язку в мережі або в декількох мережах, поділяється на дві частини. Перша стосується мережі зв’язку: дані, що передаються кінцевому користувачеві з мережі, повинні надійти за призначенням в правильному вигляді та своєчасно. Другу частину проблеми – дані, що надійшли зрештою за призначенням кінцевому користувачеві, повинні розпізнаватись і мати належну форму для їх правильного використання, – можна вирішити введенням протоколів високого рівня. Повна архітектура, орієнтована на кінцевого користувача, містить у собі і мережні протоколи і протоколи високого рівня. Як приклад, на рис. 3.1 показано схему зв’язку між користувачами *А* і *В* через проміжний вузол мережі. До цього вузла можуть бути приєднані кінцеві користувачі, а з ними можуть бути зв’язані протоколи високого рівня, але функцією проміжного вузла є тільки надання відповідних мережних послуг.

У свою чергу, дві групи протоколів – ті, що надають мережні послуги, і протоколи високого рівня – звичайно поділяються далі на окремі рівні. Кожний рівень вибирається для надання визначеної послуги за змістом названих основних завдань: правильністю і своєчасністю доставлення даних у формі, за якою їх можна розпізнати. Шляхом розробки еталонної моделі взаємодії відкритих систем була побудована концепція, при якій кожний рівень надає послуги вищестоящому рівню.

Модель рівневих протоколів взаємодії відкритих систем є семирівневим стандартом. Рівні в мережній моделі, запропонованій МОС, наведено на рис. 3.2.

Міжнародною організацією стандартизації розроблено базову еталонну модель ВВС для визначення рівневих мереж і рівневих протоколів. Ця модель привернула велику увагу в усьому світі і була реалізована багатьма фірмами – виробниками засобів зв’язку.

Метою моделі ВВС є: стандартизація обміну даними між системами; усунення будь-яких технічних перешкод для зв’язку систем; усунення труднощів “внутрішнього” опису функціонування окремої системи; визначення точок

взаємосполучення для обміну інформацією між системами; звуження діапазону можливостей послуг для того, щоб підвищити здатність обміну даними між користувачами без зайвих накладних витрат на переговори і переклад; забезпечення розумної відправної точки відходу від стандартів, якщо вони не задовольняють усіх вимог.

Зупинимося детально на характеристиці цих стандартів.

***Фізичний рівень*** найнижчий рівень. Функції цього рівня забезпечують активізацію, підтримку і дезактивізацію фізичного ланцюга між кінцевим устаткуванням даних (КУД) і апаратурою каналу даних (АКД). Для фізичного рівня опубліковано велику кількість стандартів. Найбільш відомими є RS-232C і рекоменда ції МСЕ.

***Канальний рівень*** (рівень ланки даних) відповідає за передачу даних по каналу. Він забезпечує:

-синхронізацію даних для розмежування потоку бітів фізичного рівня;

-вид подання бітів; визначені гарантії прибуття даних в приймальне КУД;

-керування потоком даних, щоб КУД не перевантажувалися в будь-який момент часу занадто великою кількістю даних.

Одна з найважливіших функцій цього рівня полягає у виявленні помилок передачі і забезпеченні механізму відновлення даних у випадку їх втрати, дублювання за наявності помилок у даних.

***Мережний рівень*** визначає інтерфейс КУД користувача з мережею пакетної комутації, інтерфейс двох пристроїв КУД один з одним у мережі пакетної комутації, а також маршрутизацію в мережі і зв’язок між мережами (інтермережний протокол). Цей рівень детально визначений і має велику кількість функцій. Протокол X.25 реалізує цей рівень. ***Транспортний рівень*** забезпечує інтерфейс між мережею передачі даних і верхніми трьома рівнями. Саме цей рівень надає користувачеві факультативні можливості одержання сервісу визначеної якості (і вартості) від самої мережі (тобто мережного рівня). Він проектується таким чином, щоб відокремити користувача від деяких фізичних і функціональних аспектів пакетної мережі, а також забезпечує наскрізну звітність у мережі.

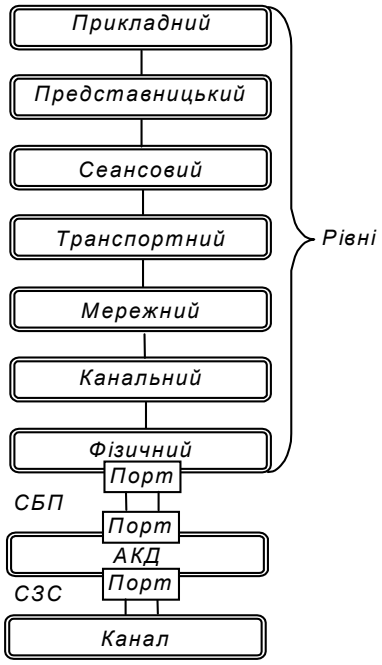
***Сеансовий рівень*** служить інтерфейсом користувача з рівнем транспортних послуг. Цей рівень забезпечує засоби організації обміну даними між користувачами. Користувачі можуть вибрати тип синхронізації і керування, що вимагаються від рівня, такі як:

-почергово або одночасно двоспрямований діалог;

-точки синхронізації для проміжного контролю і відновлення при передачі файлів;

-аварійне закінчення і рестартування;

-нормальна і прискорена передача даних.




Сеансовий рівень має спеціальні послуги, примітивні і протокольні блоки даних, що визначені в документах МОС і МСЕ.

***Представницький рівень даних*** визначає синтаксис даних у моделі. Він не пов'язаний зі значенням або семантикою даних. Його головна роль полягає в тому, щоб приймати типи даних (знак, ціле число) з прикладного рівня і потім узгоджувати з рівнем того ж рангу синтаксичне подання (таке, як телетекс, відеотекс і т. д.). Рівень подання забезпечує відображення даних на віртуальному терміналі, а також надання таких послуг, як дозвіл прийому електронного повідомлення від рівня додаткових програмних продуктів і узгодження з одноранговим рівнем виду подання сторінки (наприклад, для друкарського набору), для прикладного рівня іншого вузла користувача.

***Прикладний рівень*** призначений для підтримки прикладного процесу кінцевого користувача. На відміну від рівня подання даних цей рівень має справу із семантикою даних. Рівень містить сервісні елементи для підтримки прикладних процесів, таких як керування різноманітними процесами, обмін фінансовими даними, діловими і довідковими даними (наприклад, система обробки повідомлень). Прикладний рівень також підтримує концепції віртуального терміналу і віртуального файла.

Багаторівнева організація керування процесами в мережі породжує необхідність модифікувати на кожному рівні повідомлення стосовно функцій, реалізованих на цьому рівні. Модифікація виконується за схемою взаємодії процесів, поданою на рис. 3.3.

Даним, що передаються у формі повідомлення, надаються *заголовок* і *закінчення*, в яких міститься інформація, необхідна для опрацювання повідомлення на відповідному рівні: покажчики типу повідомлення, адреси відправника, одержувача, каналу, порту і т. д. Заголовок і закінчення називаються обрамленням повідомлення (даних).

Повідомлення, сформоване на рівні $n+1$, при обробці на рівні $n$ супроводжується додатковою інформацією у вигляді заголовка $Зn$ і закінчення $Кn$.

Це ж повідомлення, надходячи на нижчий рівень, у черговий раз забезпечується додатковою інформацією – заголовком $Зn–1$ і закінченням $Кn–1$.

У разі передавання від нижчих рівнів до вищих повідомлення звільняється від відповідного обрамлення.

Таким чином, кожний рівень оперує власними заголовком і закінченням, а послідовність символів, що знаходиться між ними, розглядається як дані більш високого рівня. За рахунок цього забезпечується незалежність даних, що стосуються різних рівнів керування передачею повідомлень.

Супроводження повідомлень обрамленням – процедура, аналогічна вкладанню листа в конверт, який використовується у поштовому зв'язку. Всі дані, необхідні для передачі повідомлення, вказуються на конверті. При передачі цього повідомлення на нижчий рівень воно вкладається в новий конверт, забезпечений відповідними даними. Повідомлення, що надходить у систему, проходить від нижніх рівнів до верхнього ( рис.3.3).

Засіб керування нижнього рівня оперує даними, зазначеними в обрамленні, як і з даними на конверті. При передаванні на вищий рівень повідомлення звільняється від "конверта", внаслідок чого на наступному рівні обробляється черговий "конверт".

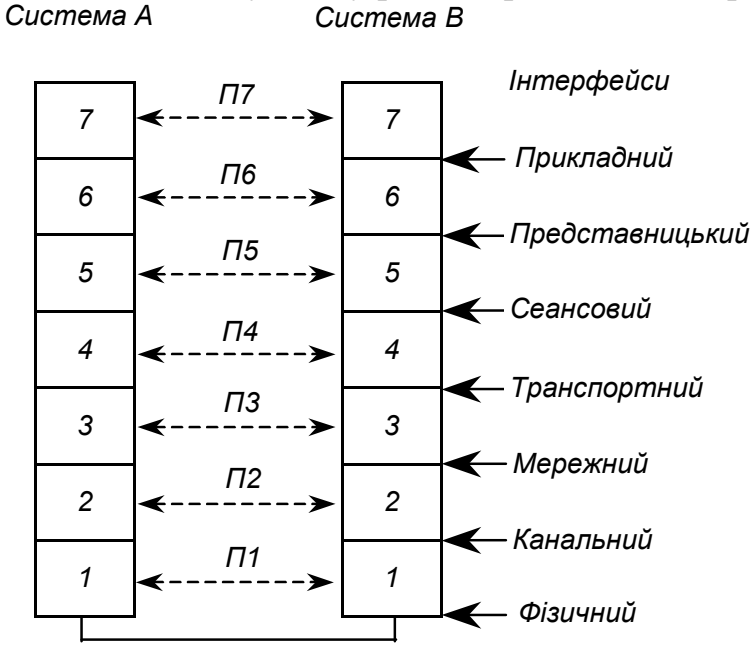



Рисунок 3.3. Схема взаємодії процесів на базі домережних протоколів і інтерфейсів

Таким чином, кожний рівень керування оперує не з самими повідомленнями, а з "конвертами", в які "упаковані" повідомлення. Тому склад повідомлень, що формуються на верхніх рівнях, ніяк не впливає на функціонування нижніх рівнів керування передачею. Процес передавання повідомлення відбувається послідовно з 7-го рівня на 1-й, а процес прийому – з 1-го на 7-й.


Рисунок 3.4. Схема інкапсуляції між двома рівнями

## §3. Алгоритм функціонування протоколу TCP

Протокол TCP реалізує надійну, орієнтовану на з'єднання службу доставки. Дані протоколу TCP передаються сегментами, і з'єднання повинне бути встановлене до того, як вузли почнуть обмінюватися даними. TCP використовує поняття потоків, у яких дані представлені у виді послідовності байт.

TCP забезпечує надійність, привласнюючи номера послідовності (sequence number) кожному переданому сегменту. Якщо сегмент розбивається на дрібні пакети, то вузол-одержувач може довідатись, чи всі частини отримані. Для цього використовуються підтвердження. Для кожного відправленого сегмента вузол-одержувач повинен повернути відправнику підтвердження (acknowledgement, ACK) протягом визначеного часу.

Якщо відправник не одержав ACK, то дані передаються повторно. Якщо сегмент ушкоджений, то вузол-одержувач відкидає його. Оскільки ACK у цьому випадку не посилається, відправник передає сегмент ще раз.

*Порти і сокети.* Прикладення, що використовують TCP, ідентифікують себе на комп'ютері за допомогою номера порту протоколу (protocol port number). Наприклад, FTP-сервер використовує визначений TCP-порт (який саме порт призначено окремому прикладенню в операційній системі Windows можна дізнатись з файлу, який знаходиться: c:\Windows\Services), тому інші додатки можуть з ним зв'язатися.

Порти можуть мати будь-який номер від 0 до 65536. Номера портів для прикладень клієнтів динамічно призначаються операційною системою при обробці запиту на обслуговування. Відомі (well-known) номера портів для прикладень-серверів призначаються групою Internet Assigned Numbers Authority (IANA) і не міняються. Номера відомих портів звичайно розташовані в інтервалі від 1 до 1024. Остаточний список відомих номерів портів задокументований у RFC 1700.

Сокет (socket) багато в чому аналогічний дескриптору файлу (file handler). Він забезпечує кінцеву точку мережного з'єднання. Прикладення, створюючи сокет, указує три параметри: IP-адреса вузла, тип обслуговування (протокол TCP для орієнтованого на з'єднання обслуговування і UDP для не орієнтованого) і порт, використовуваний прикладенням.

Прикладення може створити сокет і використовувати його для відправлення не орієнтованого на з'єднання трафіку відповідним прикладенням чи ж підключити його до сокету іншого прикладення. В іншому випадку дані будуть послані по надійному з'єднанню.

Порт протоколу TCP указує місце доставки повідомлень. Номера портів, менші 256, визначені як широко використовувані. У табл. 8 перераховані деякі з таких портів.

Таблиця 8

Порти протоколу TCP

| Номер порта | Опис |
|---|---|
| 21 | FTP |
| 23 | Telnet |
| 25 | SMTP |
| 80 | HTTP |
| 53 | Доменная система имен (DNS) |

*Встановлення TCP з'єднання.* Ініціалізація TCP-з'єднання відбувається в три етапи. Робиться це для синхронізації відправлення й одержання сегментів повідомлення іншого вузла про кількість даних, які можна послати за один раз, і установки віртуального з'єднання.

Опишемо з яких операцій складається цей процес:

Вузол-відправник відправляє запит на з'єднання, посилаючи сегмент із установленим прапором синхронізації (SYN).

Вузол-адресат підтверджує одержання запиту, відправляючи назад сегмент із:

Установленим прапором синхронізації.

Порядковим номером початкового байта сегменту, що він може послати або номером послідовності (sequence number).

Підтвердженням, що включає порядковий номер наступного сегменту, який він очікує одержати.

Запитуючий вузол посилає назад сегмент із підтвердженням номеру послідовності і номером свого підтвердження (acknowledgement number).

Для завершення з'єднання TCP діє аналогічно. Це гарантує, що обидва вузли закінчать передачу і приймуть усі дані.

Протокол TCP буферизує дані для передачі між двома вузлами, використовуючи ковзні вікна (sliding windows). Кожен TCP/IP вузол підтримує два ковзних вікна: одне для прийому даних, а інше - для відправлення. Розмір вікна визначає обсяг даних, що можуть бути буферизовані на комп'ютері.

*Концепція квитирування.* У рамках з'єднання вірність передачі кожного сегменту повинна підтверджуватись квитанцією одержувача. Квитирування - це один із традиційних методів забезпечення надійного зв'язку.

Для того, щоб можна було організувати повторну передачу спотворених даних, відправник нумерує відправлені одиниці переданих даних (далі для простоти називані кадрами). Для кожного кадру відправник очікує від приймача так називану позитивну квитанцію - службове повідомлення, що сповіщає про те, що вихідний кадр був отриманий і дані в ньому виявилися коректними.

Існують два підходи до організації процесу обміну позитивними і негативними квитанціями: із простоями і з організацією "вікна".

Метод із простоями вимагає, щоб джерело, що надіслало кадр, очікувало одержання квитанції (позитивної чи негативний) від приймача і тільки після цього надсилало наступний кадр (чи повторювало спотворений), цей метод отримав назву передачі кадрів із простоєм джерела

В другому методі для підвищення коефіцієнту використання лінії, джерелу дозволяється передати декілька кадрів у безупинному режимі, тобто в максимально можливому для джерела темпі, без одержання на ці кадри відповідних квитанцій. Кількість кадрів, що дозволяється передавати таким чином, називається розміром вікна. Рис. 8 ілюструє даний метод для розміру вікна в W кадрів. Зазвичай кадри при обміні нумеруються циклічно, від 1 до W. При відправленні кадру з номером 1 джерелу дозволяється передати ще W-1 кадрів до одержання квитанції на кадр 1. Якщо ж за цей час квитанція на кадр 1 так і не прийшла, то процес передачі припиняється, і по проходженні деякого часу кадр 1 вважається загубленим (чи квитанція на нього

загублена) і він передається знову.

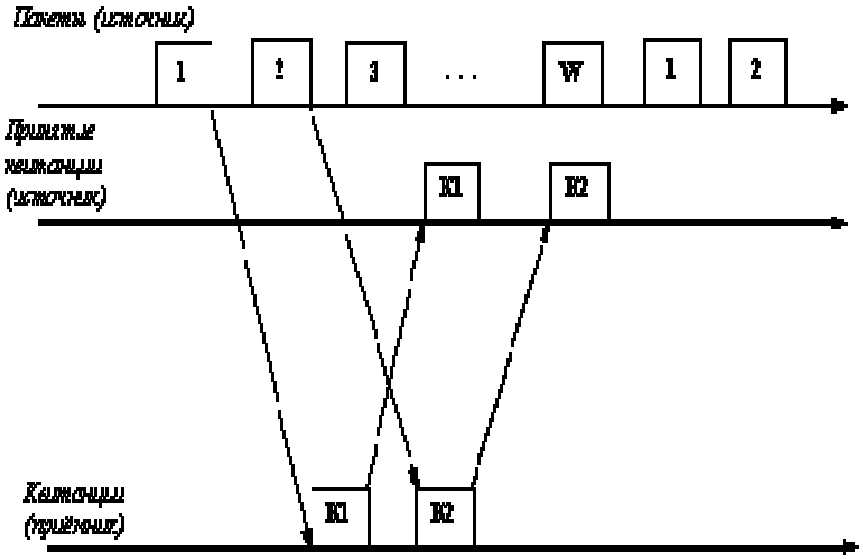



Рисунок 8 – Метод «вікна»- безперервна відправка пакетів

Якщо потік квитанцій надходить більш-менш регулярно, у межах допуску в W кадрів, то швидкість обміну досягає максимально можливої величини для даного каналу і прийнятого протоколу. Цей алгоритм називають алгоритмом ковзного вікна.

*Реалізація ковзного вікна в протоколі TCP*. У протоколі TCP реалізований різновид алгоритму квитирування з використанням вікна. Особливість цього алгоритму полягає в тому, що, хоча одиницею переданих даних є сегмент, вікно визначене на безлічі нумерованих байт неструктурованого потоку даних, що надходять з верхнього рівня і буферизуються протоколом TCP.

Квитанція посилається тільки у випадку правильного прийому даних, негативні квитанції не посилаються. Таким чином, відсутність квитанції означає або прийом спотвореного сегменту, або втрату сегмента, або втрату квитанції.

Як квитанцію одержувач сегмента відсилає відповідне повідомлення (сегмент), у яке поміщає число, на одиницю перевищуюче максимальний номер байта в отриманому сегменті. Якщо розмір вікна дорівнює W, а остання квитанція містила значення N, то відправник може посилати нові сегменти доти, поки в черговий сегмент не потрапить байт із номером N+W. Цей сегмент виходить за рамки вікна, і передачу в такому випадку необхідно призупинити до приходу наступної квитанції.

*Вибір тайм-ауту*. Вибір часу чекання (тайм-ауту) чергової квитанції є важливою задачею, результат рішення якої впливає на продуктивність протоколу TCP.

Тайм-аут не повинен бути занадто коротким, щоб по можливості виключити надлишкові повторні передачі, що знижують корисну пропускну здатність системи. Але він не повинний бути і занадто великим, щоб уникнути тривалих простоїв, зв'язаних з чеканням неіснуючої чи "заблудлої" квитанції.

При виборі величини тайм-ауту повинні враховуватися швидкість і надійність фізичних ліній зв'язку, їхня довжина і багато інших подібних факторів. У протоколі TCP тайм-аут визначається за допомогою досить складного адаптивного алгоритму, ідея якого полягає в наступному. При кожній передачі засікається час від моменту відправлення сегмента до приходу квитанції про його прийом (час обороту).

Одержувані значення часів обороту усереднюються з ваговими коефіцієнтами, що зростають від попереднього виміру до наступного. Це робиться для того, щоб підсилити вплив останніх вимірів. Як тайм-аут вибирається середній час обороту, помножений на деякий коефіцієнт.

*Реакція на перевантаження мережі*. Варіюючи величину вікна, можна вплинути на завантаження мережі. Чим більше вікно, тим більшу порцію непідтверджених даних можна послати в мережу. Якщо мережа не справляється з навантаженням, то виникають черги в проміжних вузлах-маршрутизаторах і в кінцевих вузлах-комп'ютерах.

При переповненні прийомного буфера кінцевого вузла "перевантажений" протокол TCP, відправляючи квитанцію, поміщає в неї новий, зменшений розмір вікна. Якщо він зовсім відмовляється від прийому, то в квитанції указується вікно нульового розміру. Однак навіть після цього прикладення може надсилати повідомлення на порт, що відмовився від прийому. Для цього, повідомлення повинне супроводжуватися позначкою "терміново" (біт URG у запиті встановлений у 1). У такій ситуації порт зобов'язаний прийняти сегмент, навіть якщо для цього прийдеться витиснути з буфера дані, які вже там знаходяться.

Після прийому квитанції з нульовим значенням вікна протокол-відправник час від часу робить контрольні спроби продовжити обмін даними. Якщо протокол-приймач уже готов приймати інформацію, то у відповідь на контрольний запит він посилає квитанцію з указівкою ненульового розміру вікна.

Іншим проявом перевантаження мережі є переповнення буферів у маршрутизаторах. У таких випадках вони можуть централізовано змінити розмір вікна, посилаючи керуючі повідомлення деяким кінцевим вузлам, що дозволяє їм диференційовано керувати інтенсивністю потоку даних у різних частинах мережі.

*Структура TCP-пакету.* Усі пакети протоколу TCP мають дві частини - під дані і заголовок. У табл. 9 приведені поля заголовку TCP-пакету.

Таблиця 9

Структура TCP-пакету

| Поле | Опис |
|---|---|
| Source Port (Порт відправника) | TCP порт вузла-відправника |
| Destination Port (Порт одержувача) | TCP порт вузла-одержувача. Визначає кінцеву точку з'єднання |
| Sequence Number (Порядковий номер) | Номер послідовності пакету. Використовуються для перевірки одержання всіх байт з'єднання |
| Acknowledgement Number (Номер підтвердження) | Порядковий номер байта, який локальний вузол планує одержати наступним |
| Data Length (Довжина даних) | Довжина TCP-пакету |
| Reserved (Зарезервовано) | Зарезервовано для подальшого використання |
| Flags (Прапори) | Це поле описує вміст сегменту |
| Windows (Вікно) | Показує, скільки місця доступно в даний момент у вікні протоколу TCP |
| Checksum (Контрольна торба) | Перевіряє, чи ушкоджений заголовок |
| Urgent Pointer (Вказівник терміновості) | Коли відправляються термінові дані (зазначено в поле Flags), у цьому полі задається кінцева |

| | |
|---|---|
| | границя області термінових даних у пакеті |

## §4. Забезпечення надійності доставки пакетів

Протокол TCP реалізує надійну, орієнтовану на з'єднання службу доставки. Дані протоколу TCP передаються сегментами, і з'єднання повинне бути встановлене до того, як вузли почнуть обмінюватися даними. TCP використовує поняття потоків, у яких дані представлені у виді послідовності байт. TCP забезпечує надійність, привласнюючи номера послідовності (sequence number) кожному переданому сегменту. Якщо сегмент розбивається на дрібні пакети, то вузол-одержувач може довідатись, чи всі частини отримані. Для цього використовуються підтвердження. Для кожного відправленого сегмента вузол-одержувач повинен повернути відправнику підтвердження (acknowledgement, ACK) протягом визначеного часу. Якщо відправник не одержав ACK, то дані передаються повторно. Якщо сегмент ушкоджений, то вузол-одержувач відкидає його. Оскільки ACK у цьому випадку не посилається, відправник передає сегмент ще раз.

Задачею протоколу транспортного рівня UDP (User Datagram Protocol) є передача даних між прикладними процесами без гарантії доставки, тому його пакети можуть бути загублені, продубльовані або можуть прийти не в тому порядку, у якому вони були відправлені.

### Завдання на СРС

***1. Поглиблене вивчення протоколів транспортного рівня***

**ЛЕКЦІЯ 3**
**ЗАНЯТТЯ 3**
**ОСОБЛИВОСТІ ФУНКЦІОНУВАННЯ ПРОТОКОЛУ TCP**

## §1. Механізм ковзаючого вікна

В методі ковзаючого вікна для підвищення коефіцієнту використання лінії, джерелу дозволяється передати декілька кадрів у безупинному режимі, тобто в максимально можливому для джерела темпі, без одержання на ці кадри відповідних квитанцій. Кількість кадрів, що дозволяється передавати таким чином, називається розміром вікна. Рис. 8 ілюструє даний метод для розміру вікна в W кадрів. Зазвичай кадри при обміні нумеруються циклічно, від 1 до W. При відправленні кадру з номером 1 джерелу дозволяється передати ще W-1 кадрів до одержання квитанції на кадр 1. Якщо ж за цей час квитанція на кадр 1 так і не прийшла, то процес передачі припиняється, і по проходженні деякого часу кадр 1 вважається загубленим (чи квитанція на нього загублена) і він передається знову.

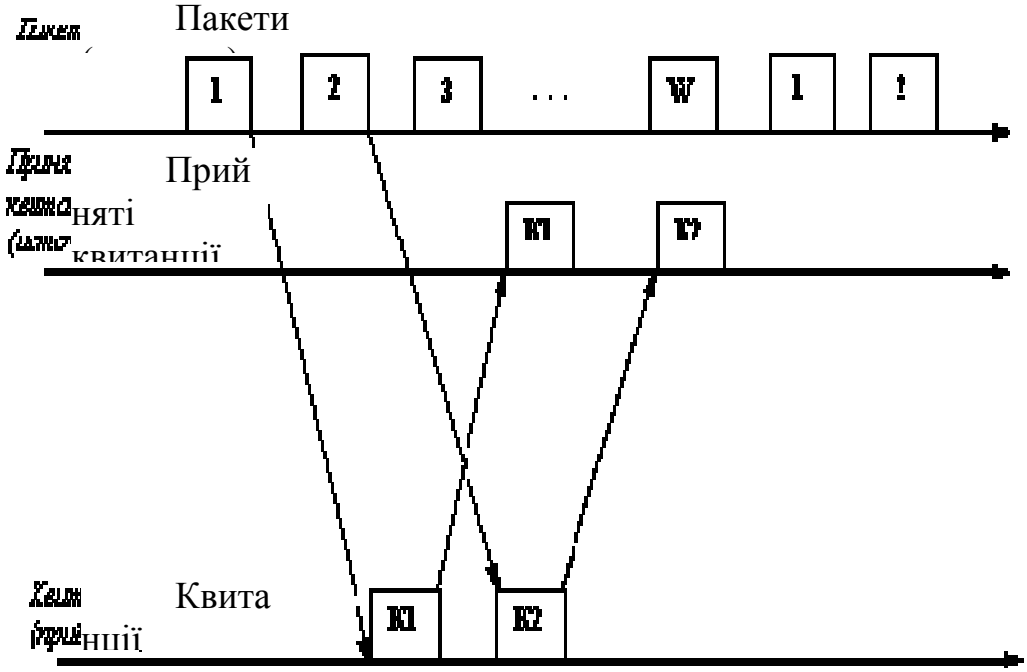



Рисунок 8 – Метод «вікна»- безперервна відправка пакетів

Якщо потік квитанцій надходить більш-менш регулярно, у межах допуску в W кадрів, то швидкість обміну досягає максимально можливої величини для даного каналу і прийнятого протоколу. Цей алгоритм називають алгоритмом ковзного вікна.

## §2. Вікна змінного розміру та управління потоком

*Реалізація ковзного вікна в протоколі TCP*. У протоколі TCP реалізований різновид алгоритму квитирування з використанням вікна. Особливість цього алгоритму полягає в тому, що, хоча одиницею переданих даних є сегмент, вікно визначене на безлічі нумерованих байт неструктурованого потоку даних, що надходять з верхнього рівня і буферизуються протоколом TCP.

Квитанція посилається тільки у випадку правильного прийому даних, негативні квитанції не посилаються. Таким чином, відсутність квитанції означає або прийом спотвореного сегменту, або втрату сегмента, або втрату квитанції.

Як квитанцію одержувач сегмента відсилає відповідне повідомлення (сегмент), у яке поміщає число, на одиницю перевищуюче максимальний номер байта в отриманому сегменті. Якщо розмір вікна дорівнює W, а остання квитанція містила значення N, то відправник може посилати нові сегменти доти, поки в черговий сегмент не потрапить байт із номером N+W. Цей сегмент виходить за рамки вікна, і передачу в такому випадку необхідно призупинити до приходу наступної квитанції.

*Вибір тайм-ауту*. Вибір часу чекання (тайм-ауту) чергової квитанції є важливою задачею, результат рішення якої впливає на продуктивність протоколу TCP.

Тайм-аут не повинен бути занадто коротким, щоб по можливості виключити надлишкові повторні передачі, що знижують корисну пропускну здатність системи. Але він не повинний бути і занадто великим, щоб уникнути тривалих простоїв, зв'язаних з чеканням неіснуючої чи "заблудлої" квитанції.

При виборі величини тайм-ауту повинні враховуватися швидкість і надійність фізичних ліній зв'язку, їхня довжина і багато інших подібних факторів. У протоколі TCP тайм-аут визначається за допомогою досить складного адаптивного алгоритму, ідея якого полягає в наступному. При кожній передачі засікається час від моменту відправлення сегмента до приходу квитанції про його прийом (час обороту). Одержувані значення часів обороту усереднюються з ваговими коефіцієнтами, що зростають від попереднього виміру до наступного. Це робиться для того, щоб підсилити вплив останніх вимірів. Як тайм-аут вибирається середній час обороту, помножений на деякий коефіцієнт.

*Реакція на перевантаження мережі*. Варіюючи величину вікна, можна вплинути на завантаження мережі. Чим більше вікно, тим більшу порцію непідтверджених даних можна послати в мережу. Якщо мережа не справляється з навантаженням, то виникають черги в проміжних вузлах-маршрутизаторах і в кінцевих вузлах-комп'ютерах.

При переповненні прийомного буфера кінцевого вузла "перевантажений" протокол TCP, відправляючи квитанцію, поміщає в неї новий, зменшений розмір вікна. Якщо він зовсім відмовляється від прийому, то в квитанції указується вікно нульового розміру. Однак навіть після цього прикладення може надсилати повідомлення на порт, що відмовився від прийому. Для цього, повідомлення повинне супроводжуватися позначкою "терміново" (біт URG у запиті встановлений у 1). У такій ситуації порт зобов'язаний прийняти сегмент, навіть якщо для цього прийдеться витиснути з буфера дані, які вже там знаходяться.

Після прийому квитанції з нульовим значенням вікна протокол-відправник час від часу робить контрольні спроби продовжити обмін даними. Якщо протокол-приймач уже готов приймати інформацію, то у відповідь на контрольний запит він посилає квитанцію з указівкою ненульового розміру вікна.

Іншим проявом перевантаження мережі є переповнення буферів у маршрутизаторах. У таких випадках вони можуть централізовано змінити розмір вікна, посилаючи керуючі повідомлення деяким кінцевим вузлам, що дозволяє їм диференційовано керувати інтенсивністю потоку даних у різних частинах мережі.

## §3. Формат TCP-сегменту

*Структура TCP-пакету.* Усі пакети протоколу TCP мають дві частини - під дані і заголовок. У табл. 9 приведені поля заголовку TCP-пакету.

Таблиця 9

Призначення полів TCP-сегменту

| Поле | Опис |
|---|---|
| Source Port (Порт відправника) | TCP порт вузла-відправника |
| Destination Port (Порт одержувача) | TCP порт вузла-одержувача. Визначає кінцеву точку з'єднання |
| Sequence Number (Порядковий номер) | Номер послідовності пакету. Використовуються для перевірки одержання всіх байт з'єднання |
| Acknowledgement Number (Номер підтвердження) | Порядковий номер байта, який локальний вузол планує одержати наступним |
| Data Length (Довжина даних) | Довжина TCP-пакету |
| Reserved (Зарезервовано) | Зарезервовано для подальшого використання |
| Flags (Прапори) | Це поле описує вміст сегменту |
| Windows (Вікно) | Показує, скільки місця доступно в даний момент у вікні протоколу TCP |
| Checksum (Контрольна торба) | Перевіряє, чи ушкоджений заголовок |
| Urgent Pointer (Вказівник терміновості) | Коли відправляються термінові дані (зазначено в поле Flags), у цьому полі задається кінцева границя області термінових даних у пакеті |

## §4. Алгоритм Карна і корегування тайм-слоту

### *Час очікування та повторна передача сегментів*

Одно из самых важных и сложных понятий протокола TCP — метод обработки истекшего времени ожидания и выполнения повторной передачи. Как и в других надежных протоколах, в протоколе TCP предполагается, что получатель пришлет сигнал подтверждения приема, успешно получив новые октеты из потока данных. Каждый раз при отправке сегмента в модуле протокола TCP устанавливается значение таймера ожидания получения сигнала подтверждения приема. Если время ожидания истечет, прежде чем будет подтвержден прием отправленного сегмента, модуль протокола TCP считает, что сегмент не достиг получателя или его данные были искажены в процессе передачи, и передает его повторно.

Чтобы понять, почему алгоритм повторной передачи протокола TCP отличается от аналогичных алгоритмов, используемых в других сетевых протоколах, необходимо вспомнить, что протокол TCP изначально предназначался для работы в межсетевом окружении. Передаваемый по объединенной сети между парой машин сегмент может проходить по одной сети с низким временем задержки (т.е. по высокоскоростной локальной сети) либо передаваться через несколько взаимодействующих между собой сетей, объединенных маршрутизаторами. Поэтому заранее невозможно предсказать, насколько быстро отправитель получит сигнал подтверждения приема. Ситуация усугубляется еще и тем, что задержка прохождения каждого маршрутизатора зависит от величины трафика в конкретной сети. Поэтому суммарное время задержки на

передачу сегмента и получение отправителем подтверждения его приема может колебаться в очень широких пределах. Это проиллюстрировано на рис. 13.9, где изображены данные измерений полного (т.е. туда и обратно) времени доставки 100 последовательных пакетов по глобальной сети Internet. Из графика видно, что в программах протокола TCP должны учитываться два фактора: существенные отличия во времени передачи пакетов разным получателям и колебания времени задержки при доставке пакетов одному получателю, вызванные изменением трафика.

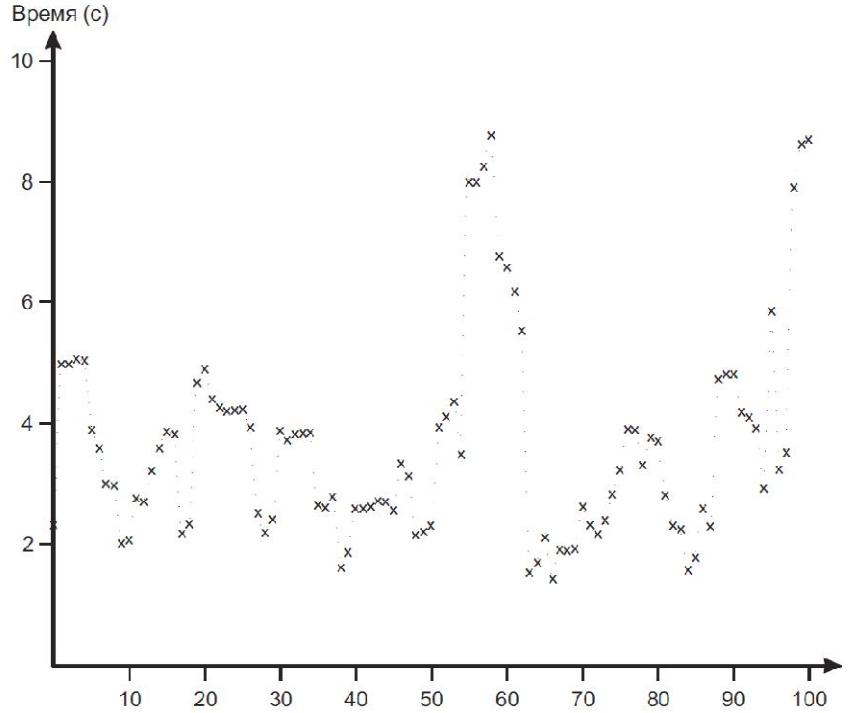



Рисунок 13.9. Данные измерений полного времени доставки 100 последовательных IP-дейтграмм. Хотя в современной сети Internet задержки передачи пакетов намного ниже, разброс остается таким же большим

Для учета изменяющегося в широких пределах времени задержки в объединенной сети в протоколе TCP используется адаптивный алгоритм повторной передачи (adaptive retransmission algorithm). В общих чертах суть его состоит в том, что в протоколе TCP отслеживается производительность каждого соединения и на основе полученных значений выбирается приемлемое время задержки. По мере изменения производительности соединения соответственно изменяется и величина задержки (т.е. величина задержки адаптируется к изменениям в производительности соединения).

При сборе данных, необходимых для работы адаптивного алгоритма, модуль протокола TCP регистрирует время отправки каждого сегмента и время получения сигналов подтверждения приема данных, находящихся в этих сегментах. По разности определяется время, затраченное на доставку каждого пакета. Описанная выше процедура называются замером полного времени доставки пакета (sample round trip time). После очередного замера модуль протокола TCP пересчитывает среднее время доставки для данного соединения. Обычно модуль протокола TCP вычисляет предполагаемую средневзвешенную величину полного времени доставки пакетов RTT (round trip time). При таком подходе вновь полученные результаты замеров мало влияют на среднее время доставки. Например, при вычислении новой средневзвешенной величины по одной из ранних методик использовалось постоянное

значение весового коэффициента а, где 0<a<1. Это позволяло регулировать степень влияния новых результатов измерений на среднее значение полного времени доставки:

$$\text{RTT} = (\alpha \times \text{Старый\_RTT}) + ((1 - \alpha) \times \text{Новый\_RTT})$$

Выбирая величину а близкой к 1, можно добиться того, чтобы новые результаты измерений, полученные за короткий промежуток времени, не влияли на средневзвешенную величину доставки пакетов. Это позволяет отсеять случайные отклонения значений от среднего, например, если один из сегментов доставлялся к получателю слишком долго. Если величину а выбрать близкой к 0, то средневзвешенная величина доставки пакетов будет достаточно быстро корректироваться в соответствии с текущими изменениями времени доставки пакетов.

При отправке пакета, модуль протокола TCP вычисляет величину тайм-аута как функцию от текущего значения предполагаемой величины полного времени доставки. В ранних реализациях протокола TCP использовалась линейная функция с постоянным весовым коэффициентом (3 (P>1). Это позволяло сделать величину тайм-аута большей, чем текущее значение предполагаемой величины полного времени доставки:

$$\text{Тайм-аут} = \beta \times \text{RTT}$$

Выбрать величину (3 не так-то просто. С одной стороны, величина тайм-аута не должна сильно отличаться от текущего времени полной доставки пакетов (т.е. значение коэффициента (3 должно быть близким к 1). Это позволит быстро обнаруживать потери пакетов и благодаря этому увеличить пропускную способность сети, поскольку модулю протокола TCP не нужно будет понапрасну долго ожидать, пока истечет тайм-аут перед повторной передачей пакета. С другой стороны, при (3=1 модуль протокола TCP может генерировать ненужный сетевой трафик, поскольку любая небольшая задержка в доставке пакета будет сразу же вызывать его повторную передачу. В результате пропуская способность сети снизится. В первоначальной спецификации стандарта TCP рекомендуется выбирать значение (3=2. Однако проведенные недавно исследования (о них пойдет речь ниже) позволили выработать более точную методику выбора времени тайм-аута. Таким образом можно подвести некоторые итоги.

Чтобы учесть большой разброс задержек во времени доставки пакетов в реально действующей объединенной сети, в протоколе TCP используется адаптивный алгоритм повторной передачи. Суть его состоит в том, что величина тайм-аута выбирается в соответствии с текущим временем задержки доставки пакета, которое постоянно отслеживается во всех открытых соединениях.

### *Точне вимірювання повного часу доставки пакетк*

Теоретически измерить полное время доставки пакета несложно: нужно из времени получения сигнала подтверждения доставки сегмента вычесть время отправки сегмента. Следует заметить, что при этом могут возникнуть определенные затруднения, поскольку в протоколе TCP используется кумулятивная система подтверждения приема. Суть ее заключается в том, что сигнал подтверждения приема отражает факт успешного получения данных, но не конкретной дейтаграммы, в которой эти данные были переданы. Рассмотрим подробнее процесс повторной передачи. Сначала модуль протокола TCP формирует сегмент, помещает его в дейтаграмму и отправляет ее получателю. После исчерпания значения таймера

выполняется повторная передача сегмента, но уже в другой дейтаграмме. Поскольку в обеих дейтаграммах содержатся одни и те же данные, отправитель не может узнать, к какой из дейтаграмм (оригинальной или повторной) относится полученный сигнал подтверждения приема. Налицо явление, которое было названо неоднозначностью сигналов подтверждения приема (acknowledgement ambiguity). Таким образом, сигналы подтверждения приема данных в протоколе TCP являются неоднозначными.

Проблема заключается в том, к какому из двух моментов передачи модулю протокола TCP следует привязать полученный сигнал подтверждения приема — к более раннему (т.е. оригинальному) или к более позднему (т.е. к тому, который был выполнен совсем недавно). Самое удивительное заключается в том, что нельзя принять ни одно из этих предположений. Если сигнал подтверждения приема соотнести с моментом передачи оригинальной дейтаграммы, то если эта дейтаграмма будет утеряна, предполагаемое полное время доставки пакета будет большим (в худшем случае, когда утеряны все посланные сегменты, значение предполагаемого времени доставки увеличивается до очень больших значений.). Рано или поздно сигнал подтверждения приема будет получен, даже если для этого потребуется выполнить одну или несколько повторных передач сегмента. В результате модуль протокола TCP измерит полное время доставки сегмента относительно времени его первоначальной посылки и на основе этого чрезвычайно большого значения вычислит новое значение RTT. Таким образом, по сравнению со старым, новое значение RTT вырастет незначительно. В следующий раз, когда модуль протокола TCP будет отправлять сегмент получателю, увеличенное значение RTT приведет к заметному увеличению времени ожидания сигнала, подтверждения приема (тайм-аута). Поэтому, если сигнал подтверждения приема будет получен также после одной или нескольких повторных передач, полное время доставки пакета будет еще больше, и т.д.

Нельзя также соотносить сигнал подтверждения приема со временем самой последней повторной передачи сегмента. Давайте рассмотрим, что произойдет, если внезапно возрастет полное время доставки сегмента. При отправке сегмента модулем протокола TCP для вычисления времени тайм-аута будет использоваться старое (небольшое) значение предполагаемого полного времени доставки пакета. Предположим, что, после успешной доставки сегмента получателю, отправителю был послан сигнал подтверждения приема. Однако из-за перегрузок в сети сигнал подтверждения приема не будет получен до истечения значения таймера. В этом случае модуль протокола TCP выполнит повторную передачу сегмента. Вскоре после этого отправитель получит первый сигнал подтверждения приема и соотнесет его с моментом последней повторной передачи. Измеренное полное время доставки пакета будет маленьким, что приведет к небольшому снижению значения предполагаемого полного времени доставки пакета, или RTT. К сожалению, уменьшение значения RTT приведет к тому, что для передачи следующего сегмента модуль протокола TCP выберет малое значение тайм-аута. В конечном счете, значение предполагаемого полного времени доставки пакета стабилизируется на величине T, такой, что корректное значение полного времени доставки будет немного превышать значение T, помноженное на некоторый коэффициент. Проведенные исследования показали, что в тех реализациях протокола TCP, где сигналы подтверждения приема соотносятся с моментом последней повторной передачи сегмента, стабильное значение RTT немного меньше половины корректного значения полного времени доставки. Следовательно,

при отсутствии потерь сегментов в сети, модуль протокола TCP будет два раза посылать один и тот же сегмент получателю.

### *Алгоритм Карна і корекція тайм-аута*

В предыдущем разделе шла речь о том, что независимо от того к какому из моментов времени будет соотнесен полученный сигнал подтверждения приема (к передаче оригинального сегмента или к его повторной передаче), измерение полного времени доставки пакета будет неточным. Как же быть? Традиционный ответ на этот вопрос очень простой: модуль протокола TCP не должен пересчитывать значение предполагаемого полного времени доставки пакета (RTT) на основе данных, полученных при повторной передаче сегмента. Эта идея, называемая также алгоритмом Карна (Karn's Algorithm), позволяет полностью избежать проблем, связанных с неоднозначностью сигналов подтверждения приема. Значение предполагаемого полного времени доставки пакета должно вычисляться только на основании данных, полученных для однозначных сигналов подтверждения приема (т.е. тех сигналов, которые были получены в ответ на однократную передачу пакета).

Естественно, что упрощенный вариант реализации алгоритма Карна (тот, в котором попросту игнорируются замеры, сделанные для повторно переданных сегментов) также не может использоваться для определения точного времени доставки сегмента. Давайте рассмотрим, что произойдет, если в момент отправки сегмента модулем протокола TCP резко возрастет задержка в сети. Напомним, что значение тайм-аута вычисляется на основании текущего значения предполагаемого полного времени доставки пакета. Очевидно, что для больших задержек в сети вычисленное значение тайм-аута будет слишком малым. Это вызовет повторную передачу сегмента. Таким образом, если игнорировать сигналы подтверждения приема для повторно переданных сегментов, новое значение предполагаемого полного времени доставки пакета никогда не изменится, и описанный выше процесс будет продолжаться до тех пор, пока не уменьшится время задержки.

Чтобы устранить подобные недостатки, в алгоритме Карна используется метод коррекции значения тайм-аута (timer backoff). Суть его состоит в том, что начальное значение тайм-аута вычисляется на основании текущих данных (по формуле, подобной приведенной выше). Однако, если в результате истечения тайм-аута произойдет повторная передача сегмента модуль протокола TCP увеличит значение тайм-аута. На практике, каждый раз перед повторной передачей сегмента, модуль протокола TCP увеличивает значение тайм-аута. Чтобы не допустить бесконтрольного увеличения тайм-аута, в большинстве реализаций оно ограничивается сверху заранее заданным значением, которое всегда больше максимально возможной задержки передачи пакета по любому из маршрутов объединенной сети.

Алгоритм вычисления нового значения тайм-аута зависит от конкретной реализации протокола TCP. В большинстве реализаций оно вычисляется путем умножения старого значения тайм-аута на специальный множитель y:

$$\text{Новый\_тайм-аут} = \gamma \times \text{Тайм-аут}$$

Обычно $\gamma$ выбирается равным 2. (Выше уже шла речь о том, что при $\gamma<2$ система будет работать нестабильно.) В некоторых реализациях протокола TCP значение множителя выбирается из таблицы. Таким образом, значение нового тайм-аута будет непосредственно зависеть от номера шага (самой известной операционной

системой, в которой используется таблица множителей, является Berkeley UNIX. Однако в текущих версиях системы в таблице хранятся одинаковые значения, эквивалентные $\gamma$ =2.).

Для того чтобы решить проблему постоянства предполагаемого полного времени доставки пакета, в алгоритме Карна используется методика принудительного изменения тайм-аута.

Суть алгоритма Карна состоит в следующем. При вычислении предполагаемого полного времени доставки пакета игнорируются замеры, сделанные для повторно посланных сегментов. Для определения точного времени доставки пакетов в алгоритме Карна используется методика увеличения значения тайм-аута при повторной передаче сегмента.

Обобщая все сказанное выше можно отметить, что при возникновении больших задержек в объединенной сети, алгоритм Карна позволяет разделить вычисление текущего значения тайм-аута и определение предполагаемого полного времени доставки пакета. Предполагаемое значение полного времени доставки используется только для вычисления начального значения тайм-аута. При каждой повторной передаче сегмента значение тайм-аута увеличивается на некоторую величину до тех пор, пока сегмент не будет успешно доставлен получателю. Таким образом, при отправке последовательности сегментов используется значение тайм-аута, полученное в результате принудительной коррекции таймера. Так происходит до тех пор, пока не будет получен сигнал подтверждения приема, соответствующий однократно посланному сегменту. После этого модуль протокола TCP на основании сделанного замера пересчитывает предполагаемое значение полного времени доставки и соответствующим образом изменяет значение тайм-аута. Эксперименты показывают, что алгоритм Карна хорошо себя зарекомендовал даже в тех сетях, где потери пакетов очень велики (Филипп Карн — радиолюбитель и энтузиаст своего дела. Он придумал этот алгоритм для того, чтобы можно было использовать протокол TCP для передачи данных по радиоканалу в условиях помех, когда происходят большие потери пакетов).

## Завдання на СРС

### *1. Вивчення протоколів прикладного рівня*

# ПРАКТИЧНЕ ЗАНЯТТЯ 1
# ЗАНЯТТЯ 4
# ОЦІНКА ПРОДУКТИВНОСТІ ФУНКЦІОНУВАННЯ ТРАНСПОРТНИХ ПРОТОКОЛІВ

## §1. Формування логічного каналу

Как упоминалось модель производительности может разрабатываться с различными уровнями детализации. В этой части пойдет речь о моделях производительности уровня системы, в отличие от моделей уровня компонентов. С точки зрения модели производительности *уровня системы,* сама система считается "черным ящиком". В этом случае отдельные ее компоненты не моделируются явным образом, а рассматривается только функция производительности. Такая функция, $X_Q(k)$, дает усредненное значение производительности "черного ящика" в зависимости от количества имеющихся в системе запросов, к. На рис. 8.1 показан сервер в виде "черного ящика". Модель производительности уровня системы представляется в виде диаграммы переходов, отражающей состояния, в которых система может пребывать , а также переходы из одного состояния в другое.

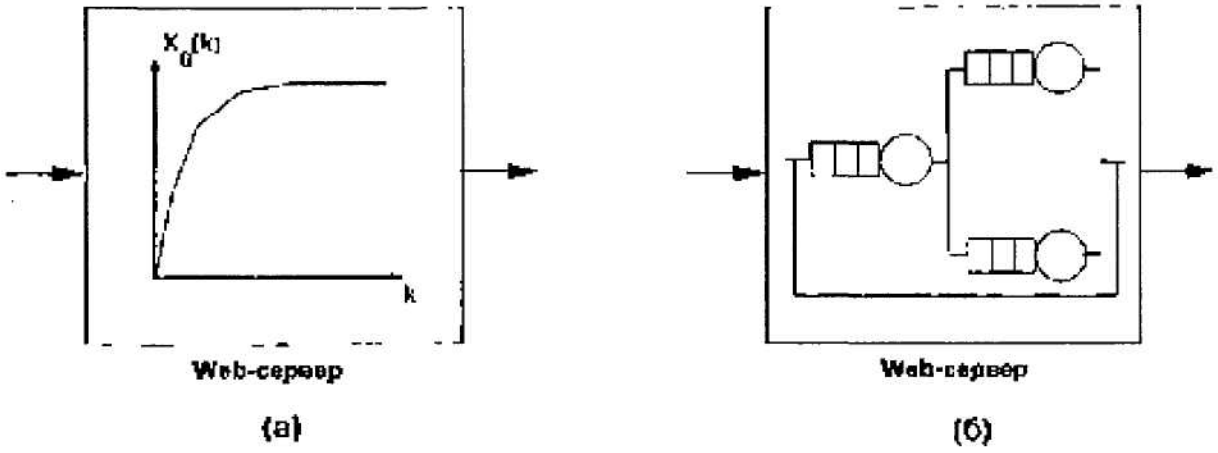



Рисунок В,1. *(а) Модель уровня системы (б) Модель уровня компонентов*

В этой части приводится подробное описание диаграмм переходов и их использования для моделирования функционирования на уровне системы.

Модель ***уровня компонентов*** учитывает отдельные ресурсы системы и то, как они используются различными запросами. В модели такого типа явным образом представлены процессоры, диски и сеть. На рис. 16) показана модель уровня компонентов того же сервера, что и на рис. 1а). Модели уровня компонентов используют сети массового обслуживания.

В начале этой части описываются очень простые модели, что без труда позволяет понять принцип их построения. Сложность моделей постепенно возрастает, однако решения для каждой модели представляются с использованием простых принципов и интуитивных понятий. После нескольких первых моделей подход к построению моделей обобщается. Следующих несколько примеров служат для иллюстрации более общего подхода.

### *Проста модель сервера I – безкінечна черга*

Рассмотрим сервер, доступ к которому получает очень большое число

пользователей. Можно, например, считать, что такой сервер используется для организации онлайнового книжного магазина или же как сервер новостей Как правило, сайты подобного типа используют несколько серверов, но для простоты будем рассматривать только один из них. Кроме того, предположим, что балансировщик нагрузки распределяет нагрузку сайта между всеми серверами. Количество пользователей неизвестно, однако достаточно велико. Под "достаточно велико" понимается, что на частоту прихода запросов не влияет количество запросов, которые уже поступили и обрабатываются. Мы будем называть такой случай случаем ***бесконечной совокупности (infinite population).*** Процесс поступления запросов на сервер характеризуется запросами, пребывающими со средней частотой *X* запросов/с. Более того, предположим, что все запросы статистически неразличимы. Это подразумевает, что сами запросы не важны для сервера, а имеет значение только их количество. Такой случай будем называть случаем ***одного класса*** или случаем ***однородной нагрузки.***

Поскольку это наш первый и самый простой случай, будем считать, что функция усредненной производительности достаточно проста. Она принята константной, т.е. не зависит от количества запросов в системе. Таким образом, усредненная производительность сервера равна $X_o$ *(к)* = // запросов

в секунду. Следует отметить, что скорость обслуживания запросов сервера является функцией не только его физических характеристик (например, скоростей процессора или диска, или количества процессоров), но и требований нагрузки в обслуживании (например, требованиями запроса к процессорам и дискам).

Кроме того, в этом примере будем считать, что сервер не отклоняет запросы. Все прибывающие запросы ставятся в очередь на обслуживание. Подобное предположение известно под названием ***бесконечной очереди*** **В** последующих разделах будет показано, что без особою труда моделировать и ситуации конечной очереди с целью иллюстрации случая, когда сервер может одновременно обнаруживать только определенное количество запросов случай ***конечной очереди.***

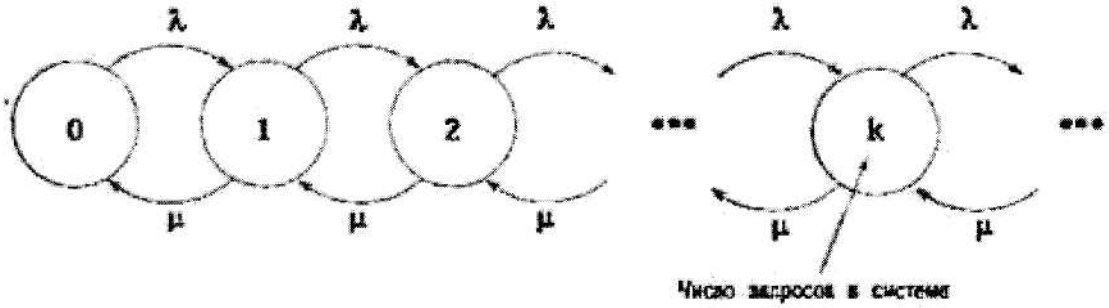
Анализы, приведенные в этом разделе и всех главах этой книги, предполагают, что рассматриваемые системы находятся в ***операционном равновесии.*** Это означает, что количество запросов, обрабатываемых системой на начало рассматриваемою периода, равно количеству запросов, обрабатываемых системой на копец рассматриваемого периода. В пределах самого периода времени количество запросов может изменяться. Для достаточно больших временных интервалов количество поступающих запросов стремится к числу обработанных запросов. Следовательно, предположение об операционном равновесии соблюдается с пренебрежительно малой ошибкой

Запросы поступают на сервер с частотой *X* запросов/с, ставятся в очередь на обработку, обслуживаются со скоростью */л* запросов/с и затем удаляются. Нам необходимо вычислить относительный интервал времени, $p_к$, когда в очереди на обслуживание сервером находятся *к (к* = 0. 1, ...) запросов, среднее количество запросов в очереди, среднее время обработки запроса, коэффициент использования сервера и его производительность.

А теперь давайте решим, как будет описываться ***состояние*** сервера. Учитывая приведенные выше допущения, описание состояния будет представлять собой всего ***один*** параметр — число ожидающих обслуживания или обслуживаемых запросов на

сервере. Оказывается, что выбрав такой простой пример, мы неявно вводим еще одно допущение, история прибытия запросов не оказывает никакого влияния на систему. Это означает, что не имеет значения ни то, как система вошла в состояние *к,* ни то, как долго она находится в этом состоянии. Единственным значимым для системы параметром является состояние *к.* Такое допущение называют допущением системы *без последействия* или *марковским* допущением.

В этом случае возможные состояния системы определяются целыми числами 0,1, 2, .... *к,* ... Учитывая принятые в рассматриваемом примере допущения о бесконечной совокупности и бесконечной очереди, мы получим бесконечное, однако перечислимое число состояний. Теперь нарисуем диаграмму переходов, в которой каждое состояние будет представлено в виде кружочка (рис.2). Переходы между состояниями соответствуют происшедшим в системе физическим событиям и представляются в виде линий со стрелками между состояниями. Например, получение нового запроса в момент времени, когда система находится в состоянии *к,* порождает переход в состояние к + 1. Такой тип перехода происходит при получении нового запроса, а, следовательно, с частотой поступления запросов, *К* переходов/с.




Аналогично, если на сервере находятся к запросов и один из них выполняется, сервер перейдет в состояние *к-1.* Упомянутые переходы осуществляются с частотой *ju* - частотой обработки запросов.

Начнем с определения значений параметра $p_к$ *{к =0, 1, 2* ...). Поскольку мы допустили, что система находится в операционном равновесии, то поток переходов в состояние *к* должен быть равен потоку переходов из этого состояния. (Более подробное описание можно найти в [1].) Такое состояние называется ***уравнением равновесия потоков*** или принципом ***равенства входящего и исходящего потоков (входящий поток = исходящий поток}*** **и** может применяться к любому набору состояний. Рассмотрим рис. 3, на котором приведена последовательность границ (пунктирные линии) вокруг состояний. Каждая следующая граница содержит на одно состояние больше, нежели предыдущая. Первая граница содержит только состояние 0. следующая - состояния 0 и 1, далее - состояния 0, 1 и 2 и т.д. Принцип равенства входящего и исходящего потоков соблюдается для любой из этих границ.

Выходной поток из любой границы определяется как количество переходов из состоянии, которые находятся внутри границы, в состояния, находящиеся вне границы. Ниже приведена система уравнений, описывающих принцип равенства входящего и исходящего потоков:

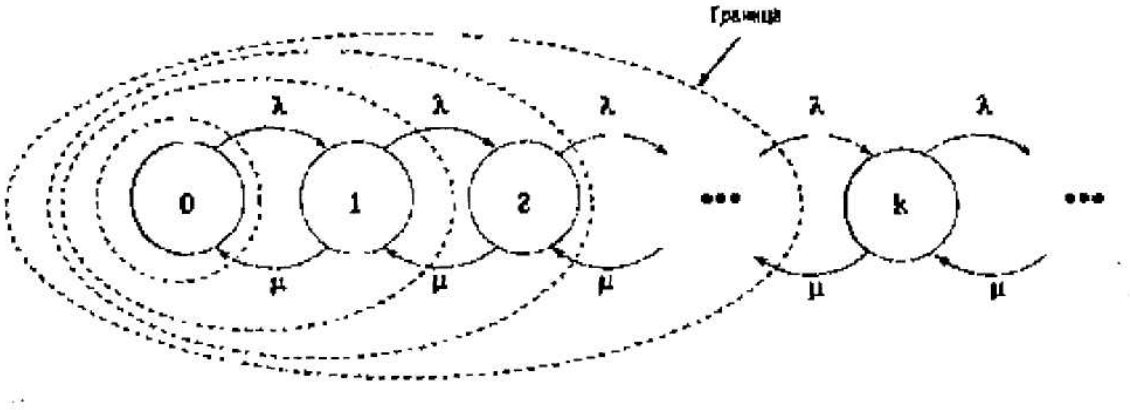



Рисунок В.3 - .Диагамма переходов с границами

ВходящийПоток = ИсходящийПоток

$$\mu \cdot p_1 = \lambda \cdot p_0$$
$$\mu \cdot p_2 = \lambda \cdot p_0$$
$$\cdot$$
$$\cdot$$
$$\mu \cdot p_k = \lambda \cdot p_{k-1}$$

Обратите внимание, что если объединить уравнения то получается:

$$p_k = \frac{\lambda}{\mu} \cdot p_{k-1} = \frac{\lambda}{\mu} \cdot \left( \frac{\lambda}{\mu} \cdot p_{k-2} \right) = \ldots = p_0 \left( \frac{\lambda}{\mu} \right)^k, k = 1,2\ldots$$

Теперь мы имеем параметр *рк* в форме функции от *pO* для значений к = 1, 2, ...Осталось только найти *pO.* Сервер в любой момент времени должен находиться в одном из возможных состояний. Следовательно, сумма относительных интервалов времени, в течение которых сервер находится в состояниях от 0 до оо, равна единице. Таким образом:

$$p_0 + p_1 + \ldots + p_k + \ldots = \sum_{k=0}^{\infty} p_k = \sum_{k=0}^{\infty} p_0 \cdot \left( \frac{\lambda}{\mu} \right)^k = 1$$

Отсюда следует:

$$p_0 = \left[ \sum_{k=0}^{\infty} \left( \frac{\lambda}{\mu} \right)^k \right]^{-1} = 1 - \frac{\lambda}{\mu}$$

Обратите внимание, что бесконечная сумма в уравнении представляет собой суммирование геометрической прогрессии, которая сходится (т.е. имеет конечную сумму) только при *Л//л* < 1. Это означает, что равновесное решение системы может быть найдено только для случая, когда средняя частота поступления запросов ниже, чем скорость их обработки. Это имеет очень большой смысл.

## §2. Визначення характеристик продуктивності елементу мережі

### *Приклад 1*

Запросы поступают на Web-сервер с частотой 30 запросов/с. Обработка каждого запроса в среднем длится 0.02 с. Требуется определить относительный интервал

времени пребывания сервера в состоянии к (где *к* — 0,1,...).

Если сервер за 1 секунду может обрабатывать */л* запросов, то в среднем обработка каждого запроса занимает 1/// секунд. Следовательно, частота обработки запросов р, является обратной величиной для значения среднего времени обработки запроса. Таким образом, р. = 1/0.02 = 50 запросов/с. Средняя частота поступления запросов составляет *X = 30* запросов/с. Отсюда находим, что относительный интервал времени бездействия Web-сервера, т.е. рО, равен 1 — *{XI* р.) = 1 - (30 / 50) =- 1 - 0.6 = 40%. Тогда относительный интервал времени функционирования сервера составляет 1 - рО = *X/ /и* = 60%. Относительный интервал времени, в течение которого на сервере находятся к запросов определяется по формуле:

$$p_k = (1-\lambda/\mu)\cdot(\lambda/\mu)^k = 0{,}4\cdot 0{,}6^k, \qquad k = 0,1,2...$$

Как видно из рис. 4, значение р при увеличении к быстро падает. Это геометрическое распределение

$$U = 1 - p_0 = \lambda/\mu$$

Это означает, что $p_k$ =(\-*U*)-$U^k$ для *к* = 0. 1, ... Функция состояний

зависит только от отношения частоты поступления запросов и скорости их обработки, т.е. от использования сервера, а не от самих значений частоты и скорости!

Теперь, когда нам известно значение $p_к$ то, используя определение среднего значения, можно легко найти среднее количество запросов *N.* находящихся на сервере:

$$\bar{N} = \sum_{k=0}^{\infty} k\cdot p_k$$

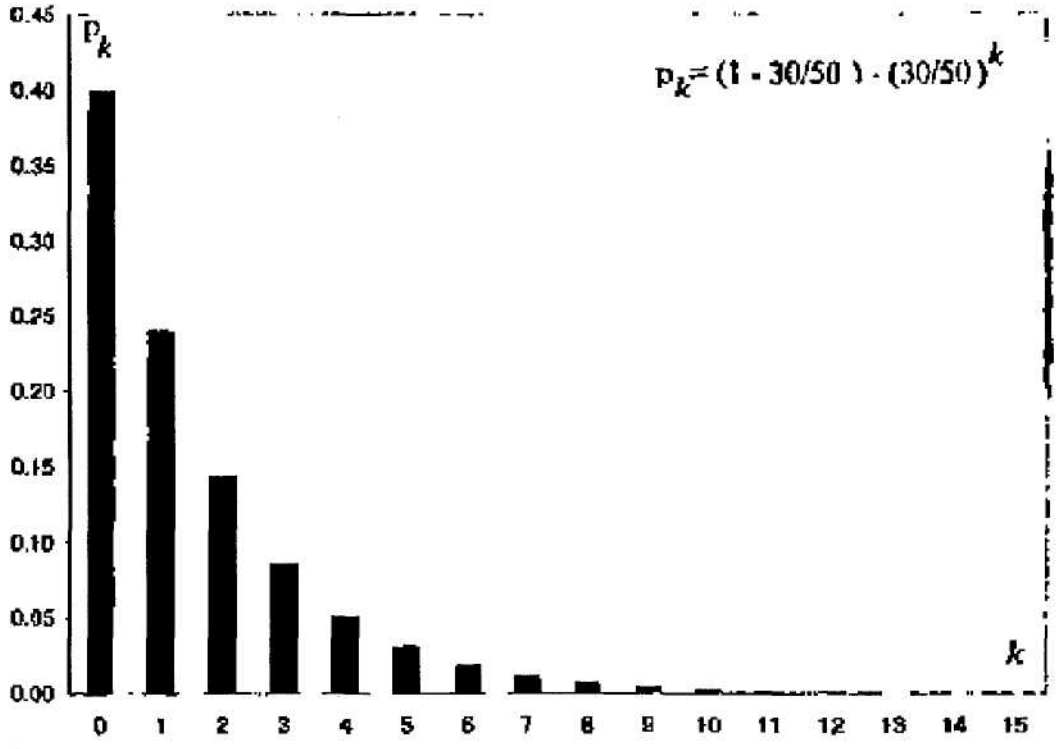



**Рисунок 8.4.** *Относительный интервал времени ($p_k$) в зависимости от k для уравнения 8.1.*

Для *U*< 1 сумма выглядит следующим образом.

$$\sum_{k=0}^{\infty} k\cdot U^k = U/(1-U)^k$$

Вводя подстановки, получаем

$$\bar{N} = U/(1-U)$$

Таким образом, подставляя параметры, приведенные в условии примера, получаем среднее количество запросов, находящихся на сервере в очереди на обработку: 0.6/(1 —0.6) =1.5.

Производительность сервера при наличии в очереди хотя бы одного запроса равна р. Такое состояние сохраняется на протяжении относительного интервала времени *U.* Если сервер находится в состоянии бездействия, его производительность равна нулю. Таким образом, средняя производительность сервера, *X* равна:

$$X = (\lambda/\mu)\cdot\mu = \lambda$$

Это уравнение дает вполне ожидаемый результат, поскольку на сервере запросы не теряются. Следовательно, в состоянии равновесия средняя частота поступления запросов будет равна средней скорости их обработки.

А теперь давайте определим среднее время отклика сервера, *R,* используя закон Литтла. "Черным ящиком" в этом случае будет сервер. При заданной протводительности *X,* вычисляемой с помощью уравнения, и среднем количестве запросов на сервере *N* получаем:

$$\hat{R} = \hat{N}/X = S/(1-\hat{U})$$

где *S — XI/л —* среднее время обслуживания запроса на сервере. Давайте подумаем, о чем нам говорит уравнение. Во-первых, при очень низких значениях коэффициента использования сервера, т.е. когда значение *U* близко к нулю, среднее время отклика сервера равно среднему времени обработки запросов. Это ожидаемый результат, поскольку запросы не теряют время в очереди на обслуживание. При больших значениях коэффициента использования сервера, т.е. когда значение *U* близко к 1. знаменатель в уравнении становится равным нулю, а значение *R* стремится к бесконечности. Фактически *R* быстро стремится к бесконечности при приближении *U* к 100%. На рис.5 приведено отношение среднего времени отклика сервера к среднему времени обработки запросов в зависимости от коэффициента использования сервера.

***Приклад 2***

Снова рассмотрим параметры из примера 1. Как определяется среднее время отклика сервера? Как оно изменится, если заменить имеющийся сервер сервером производительность которого в два раза выше? Как изменится среднее время отклика, для сервера с более высокой производительностью, если частота прибытия запросов увеличится в два раза?

Используя уравнение, найдем среднее время отклика сервера: R = (1/50) / (1-0.6) = 0.05 с. Если производительность сервера увеличится вдвое, т.е. до р. = 100 запросов/с, значение его коэффициента использования станет равным U = 30 / 100 = 0.3. Таким образом, R = (1 / 100) / (1 -0.3) = 0.014 с. Следовательно, при использовании вдвое более мощного сервера среднее время отклика уменьшится примерно до 28% от его начального значения. Если удвоить и частоту поступления запросов, и скорость их обработки, коэффициент использования сервера ие изменится, т.е. U = 0.6. Воспользовавшись уравнением (8.2.12), получим, что R = (1 / 100) / (1 -0.6) = 0.025.

Давайте коротко перечислим шаги, которые были выполнены для решения поставленной задачи. Те же шаги будут использоваться и в других примерах.

Относительный период времени, на протяжении которого на сервере находится к запросов:

$$p_k = (1 - \lambda/\mu) \cdot (\lambda/\mu)^k$$

Коэффициент использования сервера:

$$U = 1 - p_0 = \lambda/\mu$$

Средняя производительность сервера:

$$X = (\lambda/\mu) \cdot \mu = \lambda$$

Среднее количество запросов на сервере:

$$\bar{N} = U/(1-U)$$

Среднее время отклика:

$$R = N/X = S/(1-U)$$

1. Выбрать правильное представление моделируемой системы;
2. Определить совокупность возможных состояний;
3. Путем рассмотрения возможных событий, которые происходят в моделируемой системе, определить совокупность возможных переходов между состояниями. Примерами событий могут служить поступление запроса и завершение обработки запроса;
4. Для каждого возможного перехода между состояниями путем рассмотрения события, вызвавшего переход, определить частоту переходов. Например, если переход из состояния к в состояние к + 1 вызван получением запроса, то частотой перехода будет частота получения запросов в момент нахождения системы в состоянии к;
5. Для вывода уравнений определения параметра рк (относительного интервала времени, на протяжении которого система находится в состоянии к) используется принцип равновесия потоков (входящий поток — исходящий поток). Помните, что сумма всех рк должна быть равна 1.

## Завдання на СРС:

### *1. Провести розрахунок для заданих начальних умов*

Завдання 1. Запросы поступают на Web-сервер с частотой X запросов/с. Обработка каждого запроса в среднем длится μ с. Требуется определить относительный интервал времени пребывания сервера в состоянии k (где k — 0,1,...).

Завдання 2. Сассмотрите параметры из Примера 1 и определите: как определяется среднее время отклика сервера; как оно изменится, если заменить имеющийся сервер сервером производительность которого в два раза выше; как изменится среднее время отклика, для сервера с более высокой производительностью, если частота прибытия запросов увеличится в два раза?

Таблица 1

Таблица для выбора варианта

| № п/п | X, запросов/с | μ, с |
|---|---|---|
| 1 | 16 | 0,35 |

| | | |
|---|---|---|
| 2 | 17 | 0,34 |
| 3 | 18 | 0,33 |
| 4 | 19 | 0,32 |
| 5 | 20 | 0,31 |
| 6 | 21 | 0,3 |
| 7 | 22 | 0,29 |
| 8 | 23 | 0,28 |
| 9 | 24 | 0,27 |
| 10 | 25 | 0,26 |
| 11 | 26 | 0,25 |
| 12 | 27 | 0,24 |
| 13 | 28 | 0,23 |
| 14 | 29 | 0,22 |
| 15 | 30 | 0,21 |
| 16 | 31 | 0,2 |
| 17 | 32 | 0,19 |
| 18 | 33 | 0,18 |
| 19 | 34 | 0,17 |

| № п/п | X, запросов/с | μ, с |
|---|---|---|
| 20 | 35 | 0,16 |
| 21 | 36 | 0,15 |
| 22 | 37 | 0,14 |
| 23 | 38 | 0,13 |
| 24 | 39 | 0,12 |
| 25 | 40 | 0,11 |
| 26 | 41 | 0,1 |
| 27 | 42 | 0,09 |
| 28 | 43 | 0,08 |
| 29 | 44 | 0,07 |
| 30 | 45 | 0,06 |
| 31 | 46 | 0,05 |
| 32 | 47 | 0,04 |
| 33 | 48 | 0,03 |
| 34 | 49 | 0,02 |
| 35 | 50 | 0,01 |

**ЧАСТИНА 2**
**ТЕМА 2**
**АРХІТЕКТУРА ТА ПРИНЦИПИ ФУНКЦІОНУВАННЯ ТЕХНОЛОГІЇ IP/MPLS.**

**ЛЕКЦІЯ 4**
**ЗАНЯТТЯ 5**
**ПРИНЦИПИ ПОБУДОВИ АРХІТЕКТУРИ MPLS**

**§1. Принципи функціонування середовища MPLS**

**§2. Структура мережного елементу середовища MPLS**

**§3. Площина передавання пакетів**

**§4. Площина управління**

**Завдання на СРС**

***1. Порівняння технології MPLS з технологіями IP та ATM***

# ЛЕКЦІЯ 5
# ЗАНЯТТЯ 6
# ПРИНЦИПИ КОМУТАЦІЇ В СЕРЕДОВИЩІ MPLS

## §1. Маршрут з комутацією за мітками

## §2. Протокол розповсюдження міток LDP

## §3. Умови виникнення петель маршрутизації в середовищі MPLS

## §4. Контроль петель маршрутизації в середовищі MPLS

## Завдання на СРС

***1. Приклади комутації в фрагменті мережі з кільцевою та повнозв'язною структурою***

# ЛЕКЦІЯ 6
# ЗАНЯТТЯ 7
# ОРГАНІЗАЦІЯ ВІРТУАЛЬНИХ ПРИВАТНИХ МЕРЕЖ В СЕРЕДОВИЩІ MPLS

## §1. Мережі VPN другого рівня з встановленням з'єднання

## §2. Мережі VPN третього рівня з встановленням з'єднання

## §3 Мережі VPN на основі комутації MPLS

## Завдання на СРС

***1. Переваги VPN мереж на базі технології MPLS***

**МК1**
**(ПРАКТИЧНЕ ЗАНЯТТЯ 2)**
**ЗАНЯТТЯ 8**
**ВИКОНАННЯ КВАЛІФІКАЦІЙНИХ ЗАВДАНЬ ЗГІДНО ФОНДУ КВАЛІФІ¬КАЦІЙНИХ ЗАВДАНЬ ЗА "МОДУЛЬ 1"**

# ЧАСТИНА 3
# МОДУЛЬ 2
# ТЕМА 3
# АРХІТЕКТУРА QOS В МЕРЕЖАХ НА БАЗІ СТЕКУ ПРОТОКОЛІВ TCP/IP

## ЛЕКЦІЯ 7
## ЗАНЯТТЯ 9
## АРХІТЕКТУРА QOS В МЕРЕЖАХ IP

**§1. Архітектура диференційних послуг**

**§2. Класифікація пакетів**

**§3. Маркування пакетів на основі IP- пріоритету, DSCP та створення QoS-групи**

**§4. Управління інтенсивністю трафіку. Політики обмеження інтенсивності трафіку. Політики вирівнювання інтенсивності трафіку**

**Завдання на СРС**

***1. Розглянути класифікацію трафіка, запропонованого Cisco***

# ЛЕКЦІЯ 8
# ЗАНЯТТЯ 10
# ПОЛІТИКА РОЗПОДІЛУ РЕСУРСІВ

## §1. Максимінна схема рівномірного розподілу ресурсів

## §2. Алгоритм зваженого рівномірного обслуговування черг

## §3. Алгоритм розподіленого зваженого рівномірного обслуговування черг

## §4. Модифікований алгоритм зваженого кругового обслуговування черг

## §5. Модифікований алгоритм зваженого кругового обслуговування черг з дефіцитом

## Завдання на СРС

***1. Розглянути інші алгоритми розподілу ресурсів***

# ЛЕКЦІЯ 9
# ЗАНЯТТЯ 11
# ПОЛІТИКА ПОПЕРЕДЖЕННЯ ПЕРЕВАНТАЖЕННЯ І ПОЛІТИКА ВІД¬КИДАННЯ ПАКЕТІВ

**§1. Алгоритм довільного раннього виявлення перевантаження мережі**

**§2. Алгоритм зваженого довільного раннього виявлення перевантаження мережі**

**§3. Механізм явного повідомлення про перевантаження мережі**

**§4. Механізм вибіркового відкидання пакетів**

**Завдання на СРС**

***1. Розглянути інші політики відкидання пакетів***

# ПРАКТИЧНЕ ЗАНЯТТЯ 3
# ЗАНЯТТЯ 12
# ОЦІНКА ПРОДУКТИВНОСТІ ФУНКЦІОНУВАННЯ ТРАНСПОРТНИХ ПРОТОКОЛІВ З УРАХУВАННЯМ МЕХАНІЗМІВ ЗАБЕЗПЕЧЕННЯ QOS

## §1. Формування логічного каналу

## §2. Формування моделі для заданого алгоритму обробки черги та відкидання пакетів

## §3. Визначення характеристик продуктивності

## Завдання на СРС:

***1. Провести розрахунок для заданих начальних умов***

# ЧАСТИНА 4
# ТЕМА 4
# АРХІТЕКТУРА QOS В МЕРЕЖАХ НА БАЗІ ТЕХНОЛОГІЇ IP/MPLS

## ЛЕКЦІЯ 10
## ЗАНЯТТЯ 13
## АРХІТЕКТУРА QOS В МЕРЕЖАХ IP/MPLS

**§1. Перерозподіл потоків в мережах MPLS.**

**§2. Класи обслуговування IntServ. Протокол RSVP в мережах IP/MPLS. IP-пріоритет**

**§3. Механізм обслуговування DiffServ**

**§4. MPLS-реалізація функцій DiffServ**

**Завдання на СРС**

***1. Поглиблене вивчення протоколу RSVP для мереж IP/MPLS***

# ЛЕКЦІЯ 11
# ЗАНЯТТЯ 14
# ПІДТРИМКА МЕХАНІЗМІВ QOS В ВІРТУАЛЬНИХ ПРИВАТНИХ МЕРЕЖАХ MPLS

## §1. Модель з використанням ізольованого каналу для засобів забезпечення якості обслуговування в віртуальних приватних мережах MPLS

## §2. Розподілена модель QoS в віртуальних приватних мережах MPLS

## §3. Пріоритезація пакетів. Експериментальне поле MPLS

## Завдання на СРС

***1. Поглиблене вивчення структури кадру MPLS***

# ЛЕКЦІЯ 12
# ЗАНЯТТЯ 15
# УПРАВЛІННЯ ТРАФІКОМ В МЕРЕЖАХ IP/MPLS

## §1. Маршрутизація на основі резервування ресурсів

## §2. Створення та встановлення TE-тунелю

## §3. Атрибути тунелю

## §4. Атрибути ресурсів каналу

## Завдання на СРС

***1. Вивчення реалізації механізмів управління трафіком***

**МК2**
**(ПРАКТИЧНЕ ЗАНЯТТЯ 4)**
**ЗАНЯТТЯ 16**
**ВИКОНАННЯ КВАЛІФІКАЦІЙНИХ ЗАВДАНЬ ЗГІДНО ФОНДУ КВАЛІФІКАЦІЙНИХ ЗАВДАНЬ ЗА "МОДУЛЬ 2"**

## СПИСОК ЛІТЕРАТУРИ

Основні джерела


1. Вегешна, Шринивас. Качество обслуживания в сетях IP. : Пер. с англ. — М. : Издательский дом "Вильямс", 2003. — 368 с.
2. Дуглас 3. Камер. Сети TCP/IP. Принципьі, протоколы и структура. Том 1. Четвертое Издание.- М. : Издательский дом "Вильямс", 2003. — 851 с.
3. Снейдер Й. Эффективное программирование TCP/IP. Библиотека программиста. — СПб.: Питер, 2002. — 320 с.
4. Вевек Олвейн. Структура и реализация современной технологии MPLS. : Пер. с англ. — М. : Издательский дом "Вильямс", 2004. — 480 с.
5. Вентцель Е.С., Овчаров Л.А. Теория случайных процессов и ее инженернаые приложения - М.: Наука, 1991. - 384 с.
6. Гнеденко Б.В., Беляев Ю.К., Соловьев А.Д. Математические методы в теории надежности. - М.: Наука, 1965. - 526 с.
7. Стеклов В. К., Беркман Л. Н. Проектування телекомунікаційних мереж.: Підручник - К.: Техніка, 2002. - 792 с.
8. Барковський В.В., Беркман Л.Н., Кривуца В.Г. Математичне моделювання телекомунікаційних систем. Навчальний посібник. ДУІКТ.-к.:-2007, 467с.


Додаткова друковані джерела


9. Барлоу Р., Прошан Ф. Математическая теория надежности. - М.: Советское радио, 1969. - 487 с.
10. Баскер Р., Саати Т. Конечные графы и сети. - М.: Наука, 1974. - 368 с. RFC768 J. Postel, User Datagram Protocol, August 1980 RFC793 J. Postel, Transmission Control Protocol, September 1981 RFC791 J. Postel, Internet Protocol, September 1981.


Інтернет джерела


11. RFC2210 J. Wroclawski, The Use of RSVP with IETF Integrated Services, September 1997.
12. RFC2212 S. Shenker et al., Specification of Guaranteed Quality of Service, September 1997.
13. RFC2330 V. Paxson et al., Framework for IP Performance Metrics, May 1998.