Зміст

1 Глобальна мережа Інтернет 5
1.1 Визначення Інтернет 5
1.2 Мережі. Класифікація мереж 6
1.3 Історія розвитку Інтернет 8
1.4 Конфігурація апаратного забезпечення для доступу в Інтернет 12
1.5 Загальна структура мережі. 13
2 Сім'я протоколів TCP/IP 24
2.1 Будова мереж 24
2.2 Стек протоколів TCP/IP 25
2.3 Принципи адресації в Інтернет 28
2.4 Протокол ARP 32
2.5 Міжмережевий протокол IP 38
2.5.1 Формат IP-датаграми 38
2.5.2 Маршрутизація: пряма й непряма. Правила маршрутизації 42
2.5.3 Фрагментація й дефрагментація IP-датаграми 47
2.6 Транспортний протокол TCP 57
2.6.1 Загальні поняття протоколу TCP 57
2.6.2 Формат TCP-заголовка 61
2.6.3 Модель дії протоколу TCP 64
2.7 Протокол датаграм користувача UDP 69
2.7.1 Загальні поняття UDP-протоколу 69
2.7.2 Формат UDP-сегментів 70
2.7.3 Інкапсуляція пакетів 72
2.7.4 Поділ на рівні та обчислення контрольної суми UDP 74
2.7.5 Мультиплексування, демультиплексування та порти UDP 75
2.7.6 Зарезервовані й вільні номери портів UDP 77
2.7.7 Команди контролю з'єднань та маршрутизації 78
2.8 Протоколи прикладного рівня 83
2.9 Word Wide Web 87
2.9.1 Мова гіпертекстової розмітки документів HTML 90
2.9.2 Універсальний спосіб адресації ресурсів у мережі (URL) 92
2.9.3 Протокол обміну гіпертекстовою інформацією HTTP 92
2.9.4 Універсальний інтерфейс шлюзів *CGI.* Типи web-документів 93
3 HTML- мова опису WWW-сторінок 97
3.1 Типи редакторів HTML документів 97
3.2 Команди й атрибути команд в HTML 98
3.3 Структура HTML документа 99
3.4 Команди заголовка HTML документа 100
108

3.5 Команди тіла HTML документа
3.5.1 Робота з текстом 109
3.5.2 Списки 114
3.5.3 Робота з таблицями в HTML 119
3.5.4 Елементи на рівні тексту.
Фізичний і логічний стилі форматування 129
3.5. 5 Посилання 134
3.5. 6 Робота з графічними зображеннями 139
4 Розміщення й реклама сайтів в Інтернеті 144
4.1 Розміщення сторінки в Інтернеті 144
4.2 Банери 146
4.3 Реклама в Інтернеті 147
5 Електронні гроші 156
5.1 Кредитні картки 156
5.2 Електронні гроші 158
5.3 WebMoney Transfer 161
Список літератури 167

1 Глобальна мережа Інтернет

1.1 Визначення Інтернет

Відповідь на запитання: «Що таке Інтернет?» є неоднозначною: вона істотно залежить від тієї точки зору, з якої ви намагаєтесь її одержати (рис. 1).

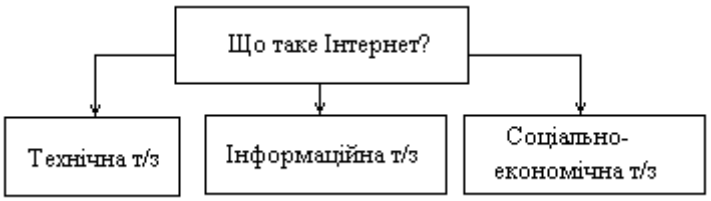

Рисунок 1

З ***технічної точки зору*** Інтернет – це сукупність десятків тисяч незалежних мереж і мільйонів різних комп'ютерів, об'єднаних загальним набором протоколів, тобто угод про взаємодію комп'ютерних і мережних компонентів. Сукупність протоколів Інтернет досить різноманітна, однак основною є сім'я протоколів TCP/IP – Transmission Control Protocol/Internet Protocol.

Із ***інформаційної точки зору***, Інтернет – це сукупність мільйонів інформаційних центрів, які, як правило, називають web-сайтами, вони містять різноманітну структуровану та неструктуровану інформацію, пронизану безліччю взаємозв'язків, що утворюють "інформаційну супермагістраль" або "всесвітню павутину".

Із ***соціально-економічної точки зору***, Інтернет – це єдине середовище спілкування, розваг та ведення бізнесу й реклами, комунікацій, сучасний засіб обміну ідеями та "віртуальні збори".

Найбільш ***важливою характерною деталлю*** Інтернету є ***відсутність єдиного центру і єдиного власника***– це дійсно розподілена система, керована в основному самими користувачами й постачальниками послуг (***провайдерами***) через різні суспільні або спеціальні організації.

Наприклад, в Україні багато аспектів розвитку Інтернету вирішуються спільно державними (наприклад, ВАТ "Укртелеком") й приватними (наприклад, провайдери, інтернет-кафе і т.п.) структурами з урахуванням потенційно високої значущості Інтернету для комерційної діяльності.

1.2 Мережі. Класифікація мереж

Мережа (у типовому визначенні) являє собою групу комп'ютерів, пов'язаних спеціальними технічними з'єднаннями, які використовують те або інше однакове технічне та програмне забезпечення для спільної роботи та поділу ресурсів [1].

Передавальним середовищем для мереж можуть бути

- телефонні й захищені виділені кабелі;
- радіо- й супутникові канали;
- спеціальні засоби зв'язку.

Мережі розбиваються на ***2 типи***:

- *з комутацією пакетів.* Прикладом може бути звичайна пошта. Листи доставляються окремому користувачеві не окремим літаком або поїздом, а разом з такими самими іншими листами. Лист увесь час переміщується з іншими листами. Це зручно, але за це доводиться платити додатково – служби зв'язку, які сортують листи й визначають, за якою адресою направляти кореспонденцію);
- *з комутацією каналів.* Приклад – звичайний телефон –якщо абонент А зв’язується з абонентом Б за допомогою телефонної лінії, то канал зв’язку належить лише абонентам А та Б).

Головне, що забезпечує роботу комп'ютерів, об'єднаних у мережу, - набір спеціальних угод, названих, як було зазначено вище, ***протоколами***. Протоколи описують як технічні аспекти з'єднання, так і прикладні та мають ***багаторівневу структуру*** відповідно до прийнятої в ISO (Міжнародна організація із стандартизації) стандартної схеми.

Комп’ютерні мережі поділяються на ***локальні*** **та** ***глобальні.***

Локальні мережі поєднують комп'ютери, що знаходяться в одному будинку або групі будинків якоїсь однієї установи (університет, банк, інститут та ін.) і найчастіше вирішують завдання спільного використання ресурсів: принтерів, комп'ютерів-сховищ даних (файлових серверів), потужних комп'ютерів для вирішення прикладних завдань (сервера додатків), комунікаційного устаткування.

Глобальні мережі поєднують безліч локальних мереж та окремих комп'ютерів, розташованих на значній відстані один від одного, з'єднаних швидкісними каналами та спеціальними програмно-технічними комплексами. Зміст ресурсів глобальної мережі найрізноманітніший.

Сукупність глобальних (формально незалежних) мереж, локальних мереж і окремих комп'ютерів, об'єднаних протоколами TCP/IP, і становить Інтернет. Інтернет надає своїм користувачам безліч ресурсів і можливостей: від послуг електронної пошти до мультимедійних інтерактивних сеансів.

Таким чином, коли йдеться про Інтернет, то мають на увазі мережу, що з'єднує та поєднує окремі мережі (у буквальному перекладі термін Інтернет означає “міжмережний”; лат. inter – “між”, net – “мережа”). При цьому виникає нове віртуальне об'єднання, що, у свою чергу, означає свій новий інформаційний простір.

1.3 Історія розвитку Інтернет

У 60-х роках XX сторіччя Міністерство оборони США створило мережу **ARPAnet**, що стала витоком Інтернет. ARPAnet була експериментальною мережею, вона створювалася для підтримки наукових досліджень у військово-промисловій сфері, – зокрема, для дослідження методів побудови мереж, стійких до часткових ушкоджень, отриманих, наприклад, при бомбардуванні авіацією та здатних у таких умовах продовжувати нормальне функціонування. Ця вимога дає тлумачення розуміння принципів побудови й структури Інтернет. У моделі ARPAnet завжди був зв'язок між комп'ютером-джерелом і комп'ютером-приймачем (станцією призначення). Мережа a priori передбачалася ненадійної: *будь-яка частина мережі може зникнути в будь-який момент* [3].

На комп'ютери, що зв'язуються, – не тільки на саму мережу – також покладена відповідальність забезпечувати налагодження й підтримку зв'язку. *Основний принцип мережі полягав у тому, що будь-який комп'ютер міг з'єднатися як рівний з рівним з будь-яким іншим комп'ютером*.

Передавання даних у мережі було організоване на основі протоколу IP, який містить правила налагодження й підтримки зв'язку в мережі, правила обігу IP-пакетів і їх обробки, правила опису мережевих пакетів сімейства IP. Мережа замислювалася та проектувалася так, щоб *користувач не міг мати ніякої інформації про конкретну структуру мережі*.

Приблизно 10 років після появи ARPAnet з'явилися локальні обчислювальні мережі (LAN), наприклад, такі, як Ethernet. Одночасно з'явилися комп'ютери, які стали називати робочими станціями. На більшості робочих станцій була встановлена операційна система UNIX. Ця ОС мала можливість роботи в мережі із IP-протоколом. У зв'язку з виникненням принципово нових завдань і методів рішення цих завдань з'явилася нова потреба: організації бажали підключитися до ARPAnet. Приблизно в той самий час з'явилися інші організації, які почали створювати свої власні мережі, що використовували близькі до IP комунікаційні протоколи. Стало зрозуміло, що всі тільки б виграли, якби ці мережі могли спілкуватися між собою, адже тоді користувачі однієї мережі змогли мати зв’язок з користувачами іншої мережі.

Найважливішою серед вищезгадуваних мереж була NSFNET, розроблена з ініціативи Національного наукового фонду (National Science Foundation – NSF), аналога Міністерства освіти і науки. Наприкінці 80-х років фондом було створено п'ять суперкомп'ютерних центрів, зробивши їх доступними для використання в будь-яких наукових установах. Створені центри потребували значних капіталовкладень, саме тому використовувати їх могли лише кооперативно. Виникла проблема зв'язку: був потрібен спосіб з'єднати ці центри й надати доступ до них різним користувачам. Спочатку була зроблена спроба використати комунікації ARPAnet, але це рішення зазнало краху, зіштовхнувшись із бюрократією оборонної галузі й проблемою забезпечення персоналом.

Тоді NSF вирішив побудувати свою власну мережу, засновану на IP-технології ARPAnet. Центри були з'єднані спеціальними телефонними лініями із пропускною здатністю 56 Kbps . Однак було очевидно, що не слід навіть і намагатися з'єднати всі університети й дослідницькі організації безпосередньо із центрами, тому що прокласти таку кількість кабелю – не тільки дуже дорого, але практично неможливо. Тому вирішено було створювати мережі за регіональним принципом. У кожній частині країни зацікавлені установи повинні були з'єднатися зі своїми найближчими сусідами. Ланцюжки, що утворилися, приєднувалися до суперкомп'ютера в одній зі своїх точок, у такий спосіб суперкомп'ютерні центри були з'єднані разом. У такій топології будь-який комп'ютер міг мати зв'язок з будь-яким іншим, передаючи повідомлення через сусідні.

Це рішення було успішним, але настав час, коли мережа вже не справлялася із потребами, які постійно зростали. Спільне використання суперкомп'ютерів дозволяло використовувати безліч інших речей, які не стосувалися суперкомп'ютерів. Несподівано університети, школи й інші організації усвідомили, що мають безліч даних і світ користувачів. Потік повідомлень у мережі (трафік) збільшувався швидше й швидше поки, зрештою, не перевантажив комп'ютери, які керували мережею, та телефонні лінії, які об’єднували ці комп’ютери. В 1987 р. контракт на керування та розвиток мережі був переданий компанії Merit Network Inc., що займалася освітньою мережею Мічигану разом з IBM й MCI. Фізично застаріла мережа була замінена на більш швидкі (приблизно у 20 разів) телефонні лінії. Були замінені на більш потужні й машини, які керували мережею.

Процес удосконалювання мережі відбувається безупинно. Однак більшість цих перебудов відбувається непомітно для користувачів. Якщо користувач ввімкне комп'ютер, він не побачить оголошення про те, що найближчі півроку Інтернет не буде доступний через модернізацію. Можливо, навіть більш важливе те, що перевантаження мережі і її вдосконалення створили зрілу й практичну технологію. Проблеми були вирішені, а ідеї розвитку перевірені в процесі використання.

Важливо відмітити те, що зусилля NSF щодо розвитку мережі привели до того, що будь-хто може одержати доступ до мережі. Колись мережа Інтернет була доступна тільки для дослідників у сфері інформатики, державним службовцям і підрядникам. NSF сприяв загальній доступності Інтернету по лінії освіти, вкладаючи гроші в приєднання навчального закладу до мережі, тільки якщо той, у свою чергу, мав плани поширювати доступ далі. Таким чином, кожен студент коледжу міг стати користувачем Інтернету.

Потреби продовжують зростати. Більшість таких коледжів на Заході вже приєднано до Інтернет, приймається рішення підключити до цього процесу середні й початкові школи. Випускники коледжів чудово інформовані про переваги Інтернет і розповідають про них своїм роботодавцям. Вся ця діяльність приводить до безперервного збільшення мережі, до виникнення й вирішення проблем збільшення мережі, розвитку технологій і системи безпеки мережі.

Історію Інтернету в країнах СНД відраховують з початку 80-х років XX століття, коли Курчатовський інститут першим одержав доступ до світових мереж. Інтернет у СНД, як і в усьому світі, усе більше стає елементом життя суспільства, зрозуміло, стаючи усе більш схожим на це суспільство. Зараз в Інтернет можна потрапити з мільйонів комп'ютерів СНД, і число їх постійно зростає. В Україні та Росії на сьогодні представлена більшість різновидів Інтернет-сервісів. Найвідоміші Web-сервери можуть похизуватися декількома сотнями тисяч постійних читачів у день. Це непогано в порівнянні, наприклад, з діловою паперовою пресою. А якщо порівняти якісні показники аудиторії Інтернет і телеаудиторії, то перевага в багатьох випадках може бути віддана першій [2].

1.4 Конфігурація апаратного забезпечення для доступу в Інтернет

Підключити до Інтернету можливо практично будь-який комп'ютер, але від потужності машини й від швидкості зв'язку (передачі інформації) залежить, які послуги Мережі можна буде використати.

У таблиці 1 наведені вимоги до ресурсів комп'ютера для перегляду WWW.

Таблиця 1

	Стандарт можливостей Інтернет	*Максимум можливостей Інтернет*
Процесор	486DX	Pentium
Оперативна пам'ять	16 Mb	32 Mb
Швидкість модема	28 800	33 600
Кількість кольорів	HiColour	TrueColour
Розподільна здатність екрана	800×600	1024×768
Розмір монітора	15"	17"
Операційна система	Windows 95	Windows NT
Можливості системи	текст, графіка Java, Active	текст, графіка Java, Active, VRLM

VRML (Virtual Reality Modelling Language) – мова, призначена для опису тривимірних зображень, яка оперує об'єктами, що описують геометричні фігури і їх розташування в просторі. Використовуючи VRML, можна показати з усіх боків тривимірне зображення, використовуючи клавіатуру або мишу. Звичайно VRML застосовують в індустрії розваг, фінансах і статистиці (організація й візуалізація даних), у науковій візуалізації, торгівлі та маркетингу, у спілкуванні та освіті.

Active – технологія Microsoft, призначена для написання мережних додатків. Стандарт Active дозволяє програмним компонентам взаємодіяти один з одним по мережі незалежно від мови програмування, на якому вони написані. Програмні елементи Active – це компоненти, що працюють на комп'ютері-клієнті, але завантажуються вони в перший раз із Web-сервера. З їх допомогою можна демонструвати різнорідну інформацію, що включає відео й звук без запуску додаткових програм. Більше того, ці програмні компоненти можуть використовуватися у додатках, написаних на будь-яких популярних мовах програмування, включаючи Java (Visual J++), Visual Basic, Visual C++.

Якщо в комп'ютері є звукова плата й швидкісний модем (не нижче *26 600* бод), то, придбавши програму *Інтернет Phone* або аналогічну, можна спілкуватися в Інтернет як по звичайному телефону. Хоча якість звуку може бути нижче, ніж при звичайному телефонному зв’язку, проте не потрібно оплачувати міжнародні розмови.

Для відеоконференції необхідний сучасний комп'ютер із процесором *Pentium,* оперативною пам'яттю не менше *16Mb,* обладнаний *мікрофоном* й *web-камерою*. Крім того, для гарної якості зображення потрібен канал зв'язку *ISDN* зі швидкістю передачі 64 Кбод. Можливо встановити відеозв'язок з нижчою якістю, використовуючи найшвидші з аналогових модемів (зі швидкістю *33 600* або 56 000 бод).

1.5 Загальна структура мережі

Для організації зв'язку двох комп'ютерів потрібно спочатку створити сукупність правил взаємодії цих комп’ютерів, визначити мову спілкування між ними, тобто визначити, що означають сигнали, які циркулюють між ними, та ін. Ці правила й визначення називаються ***протоколом***.

Сучасні мережі побудовані за багаторівневим принципом.

Багаторівнева структура спроектована з метою впорядкування безлічі протоколів і відносин. У структурі Інтернет прийнята семирівнева структура організації мережної взаємодії (рис. 2). Ця модель відома як «Еталонна модель ISO OSI» (Open System Interconnection – зв'язок відкритих систем) [1]. Вона дозволяє створювати мережні системи з модулів програмного забезпечення, які випускаються різними виробниками.

Розрізняють ***два види взаємодії***:

- реальна;
- віртуальна.

Під ***реальною*** взаємодією розуміється безпосередня взаємодія, передача інформації, наприклад, пересилання даних в оперативній пам'яті з області, відведеної одній програмі, в область іншої програми. При безпосередній передачі дані залишаються незмінними увесь час.

Під ***віртуальною*** взаємодією ми розуміємо опосередковану взаємодію й передачу даних: тут дані в процесі передачі можуть уже певним, заздалегідь обумовленим чином, видозмінюватися.

Така взаємодія аналогічна схемі посилання листа одним директором фірми іншому. Наприклад,

1 Директор деякої фірми пише листа редактору газети. Директор пише лист на своєму фірмовому бланку й віддає цей лист секретареві.

2 Секретар запечатує лист у конверт, підписує конверт, наклеює марку й відносить на пошту.

3 Пошта доставляє лист до відповідного поштового відділення. Це відділення зв'язку безпосередньо доставляє лист одержувачеві – секретареві редактора газети.

4 Секретар розкриває конверт й, за потреби, подає редактору.

Жодна з ланок ланцюга не може бути пропущена, інакше ланцюг розірветься: якщо відсутній, наприклад, секретар, то листок з текстом директора так і буде лежати на столі у секретаря.

Тут ми бачимо, як інформація (аркуш паперу з текстом) передається з верхнього рівня вниз, проходячи безліч необхідних щабелів – ***стадій обробки***. Обростає службовою інформацією (пакет, адреса на конверті, поштовий індекс; контейнер з кореспонденцією; поштовий вагон, станція призначення поштового вагона і т. ін.), змінюється на кожній стадії обробки й поступово доходить до найнижчого рівня – рівня поштового транспорту (автомобільного, залізничного, повітряного і т.п.), яким реально перевозиться в пункт призначення.

У пункті призначення відбувається зворотний процес: розкривається контейнер і витягується кореспонденція, зчитується адреса на конверті й листоноша несе його адресатові

(секретареві), що відновлює інформацію в первісному вигляді, – дістає лист із конверта, читає його й визначає терміновість, важливість і залежно від цього передає інформацію вище.

Директор і редактор, таким чином, віртуально мають прямий зв'язок. Адже редактор газети одержує точно таку інформацію, що відправив директор, а саме – аркуш паперу з текстом листа. Керівництво зовсім не піклується про проблеми пересилання цієї інформації. Секретарі також мають віртуально прямий зв'язок: секретар редактора одержить у точності те саме, що відправив секретар директора, а саме – конверт із листом. Секретарів зовсім не хвилюють проблеми пошти, яка пересилає листи. І так далі.

Аналогічні зв'язки й процеси мають місце і в *еталонній моделі ISO OSI*. Фізичний зв'язок *реально* має місце тільки на найнижчому рівні (аналог поштових поїздів, літаків, автомобілів). Горизонтальні зв'язки між всіма іншими рівнями є *віртуальними*, реально вони здійснюються передачею інформації спочатку вниз, послідовно до самого нижнього рівня, де відбувається реальна передача, а потім, на іншому кінці, зворотна передача нагору послідовно до відповідного рівня.

Модель ISO OSI пропонує дуже жорстоку стандартизацію вертикальних міжрівневих взаємодій. Така стандартизація гарантує сумісність продуктів, що працюють за стандартом якого-небудь рівня, із продуктами, що працюють за стандартами сусідніх рівнів, навіть у тому випадку, якщо вони випущені різними виробниками. Кількість рівнів може здатися надлишковою, однак таке розбиття необхідне для досить чіткого поділу необхідних функцій, щоб уникнути зайвої складності й створення структури, що може налаштовуватись під потреби конкретного користувача, залишаючись у рамках стандарту.

Виконаємо короткий огляд рівнів:

Рівень 0 (*Physical media*) пов'язаний з фізичним середовищем – передавачем сигналу – і насправді не включається в цю схему, але досить корисний для розуміння. Цей почесний рівень представляє посередників, що з'єднують кінцеві пристрої: кабелі, радіолінії і т.д.

Кабелі можуть бути:

- екрановані й неекрановані кручені пари;
- коаксіальні, на основі оптичних волокон і т.д.

Оскільки цей рівень не включено до схеми, він нічого й не описує, тільки вказує на середовище.

Рівень 1 (*Physical protocol*) – фізичний. Включає фізичні аспекти передачі двійкової інформації по лінії зв'язку. Детально описує, наприклад, напругу, частоти, природу середовища, що передає дані. Цьому рівню ставиться в обов'язок підтримка зв'язку та прийняття-передача бітового потоку. Безпомилковість бажана, але не потрібна.

Рівень 2 (*DataLink protocol*) – канальний. Забезпечує безпомилкову передачу блоків даних (називаних *кадрами,* фреймами (*frame),* або датаграмами) через перший рівень, який при передачі може спотворювати дані. Цей рівень повинен:

- визначати початок і закінчення *датаграми* в бітовому потоці;
- формувати з даних, переданих фізичним рівнем, *кадри* або послідовності;
- включати процедуру перевірки наявності помилок й їх виправлення.

Цей рівень (і тільки він) оперує такими елементами, як бітові послідовності, методи кодування, маркери.

Отже, канальний рівень:

- відповідає ***за правильну передачу даних*** (*пакетів*) на ділянках між безпосередньо зв'язаними елементами мережі;
- ***забезпечує керування доступом*** до середовища передачі даних.

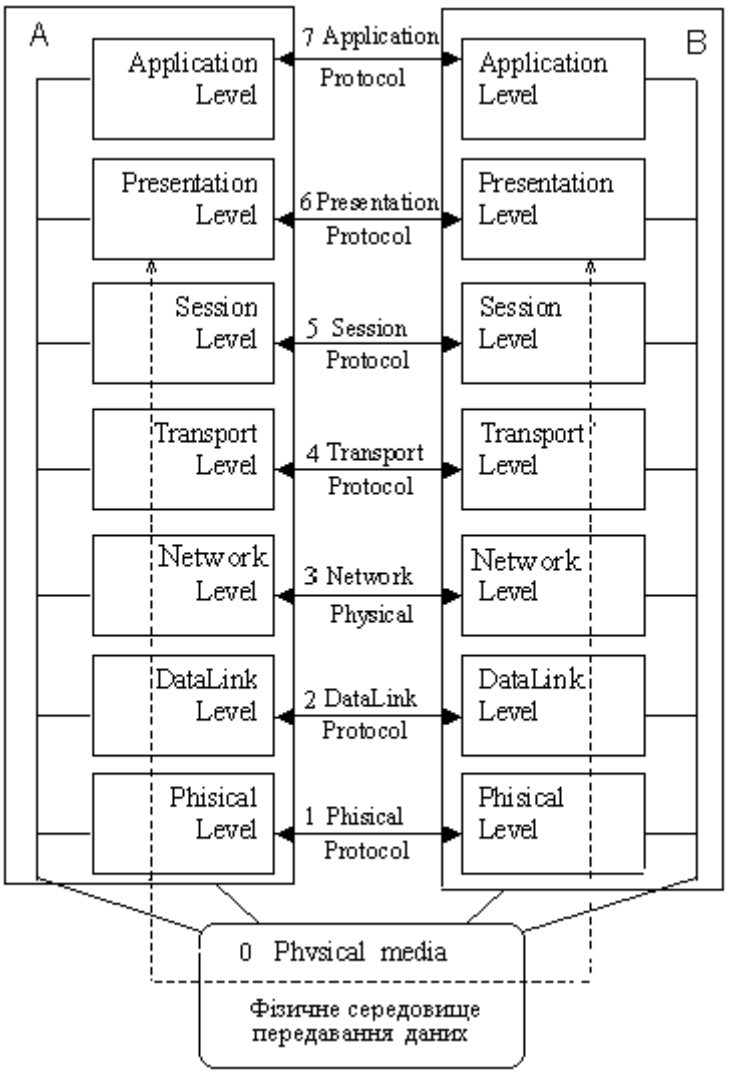

Рисунок 2 – Багаторівнева реалізація мережної взаємодії.

За складністю канальний рівень поділяють на 2 підрівні:

- *Керування логічним зв'язком чи каналом* (LLC – Logical Link Control), який посилає й одержує повідомлення з даними;
- *Керування доступом до середовища (*MAC – Medium Access Control), який керує доступом до мережі (з передаванням маркера в мережах *Token Ring* або розпізнаванням конфліктів (зіткнень передач) у мережах *Ethernet*).

Рівень 3 *(Network protocol)* – мережний. Основними функціями програмного забезпечення на цьому рівні є:

- вибірка інформації із джерела;
- перетворення інформації в пакети;
- правильна передача інформації в пункт призначення;
- обробка адрес і маршрутизація.

Цей рівень користується можливостями, які надаються йому канальним рівнем, для забезпечення зв'язку двох будь-яких точок у мережі. Він здійснює проведення повідомлень мережею, яка може мати багато ліній зв'язку, або безліччу мереж, які спільно працюють, що вимагає *маршрутизації*, тобто визначення шляху, яким варто пересилати дані. *Маршрутизація* виконується на цьому самому рівні.

Є два принципово різних способи роботи мережевого рівня. Перший – це метод ***віртуальних каналів***. Він полягає у тому, що канал зв'язку встановлюється при виклику (початку *сеансу* (*session*) зв'язку), по каналу передається інформація, і по закінченні передачі канал закривається (знищується). Передача *пакетів* відбувається зі збереженням вихідної послідовності, навіть якщо *пакети* пересилаються різними фізичними *маршрутами*, тобто *віртуальний канал* динамічно перенаправляється. При цьому пакети даних не включають адреси пункта призначення, тому що він визначається під час установлення зв'язку.

Другий – метод ***датаграм***. *Датаграми* – незалежні порції інформації, які містять усю необхідну для їх пересилання інформацію.

У той час як перший метод надає наступному рівню – транспортному – надійний канал передачі даних, вільний від перекручувань (помилок), і правильно доставляє *пакети* в пункт призначення, другий метод потребує від наступного рівня роботи над помилками й перевірки доставки потрібному адресатові.

Рівень 4 (*Transport protocol*) – транспортний. Завершує організацію передачі даних. Контролює на наскрізній основі потік даних, що проходить по *маршруту*, який був визначений мережним рівнем, а саме контролює:

- правильність передачі блоків даних;
- правильність доставки в потрібний пункт призначення;
- комплектність, схоронність і порядок проходження даних;
- збирає інформацію із блоків у її колишній вигляд або ж оперує з *датаграмами*, тобто очікує відгуку-підтвердження прийняття з пункту призначення, перевіряє правильність доставки й адресації, повторює посилання *датаграми*, якщо не надійшов відгук.

Цей рівень повинен включати ***розвинену*** та ***надійну схему адресації*** для забезпечення зв'язку через безліч мереж і *шлюзів*. Інакше кажучи, завданням даного рівня є "довести до розуму" передачу інформації з будь-якої точки мережі в будь-яку.

Рівень 5 (*Session protocol*) – сеансовий. Координує взаємодію користувачів, що з'єднуються:

- установлює їхній зв'язок;
- оперує ним;
- відновлює аварійно перервані сеанси.

Цей самий рівень відповідальний за картографію мережі – він перетворює регіональні (DNS – доменні) комп'ютерні імена в числові адреси, і навпаки. Він координує не комп'ютери та пристрої, а процеси в мережі, підтримує їх взаємодію – керує *сеансами* зв'язку між процесами прикладного рівня.

Рівень 6 (*Presentation protocol*) – рівень подання даних. Цей рівень має справу із синтаксисом і семантикою інформації, яка передається, тобто тут установлюється взаєморозуміння двох з'єднаних комп'ютерів щодо того, як вони уявляють собі і розуміють передану інформацію. Тут вирішуються, наприклад, такі завдання, як :

- перекодування текстової інформації й зображень;

- стискання і розпаковування;
- підтримка мережевих файлових систем (*NFS*), абстрактних структур даних і т. ін.

Рівень 7 (*Application protocol*) – прикладний. Забезпечує інтерфейс між користувачем і мережею, робить доступними для людини всілякі послуги Інтернет. На цьому рівні реалізується, принаймні, п'ять прикладних служб: передача файлів, віддалений термінальний доступ, передача електронних повідомлень, служба довідки та керування мережею.

Висновки

Таким чином, кажучи про ***Інтернет***, ми маємо на увазі, що мова йде про мережу, що з'єднує й поєднує окремі мережі (***локальні*** та ***глобальні***), безліч інформаційних ресурсів, які постійно зростають, а також середовище віртуального спілкування мільйонів людей. Для злагодженої роботи мережі Інтернет розроблена сукупність правил, яку називають ***протоколами***. Для узгодження різноманітних протоколів була спроектована ***багаторівнева структура протоколів і відносин*** – "Еталонна модель ISO OSI".

Контрольні запитання

1 Що таке Інтернет?

2 Які основні завдання ставилися перед розроблювачами й творцями Мережі? Чи вирішені ці завдання в мережі Інтернет?

3 Чи існує обмеження на розвиток і розширення мережі Інтернет?

4 Наскільки корисною або шкідливою можна вважати технологію *Active*?

5 У чому складається відмінність мереж з комутацією каналів і комутацією пакетів?

6 Що забезпечує узгоджену роботу комп'ютерів об'єднаних у мережу?

7 Які завдання вирішують локальні мережі?

8 Що являють собою глобальні мережі?

9 Чи можливе об'єднання різнорідних локальних мереж у єдину глобальну мережу?

10 На якому принципі побудовані сучасні мережі?

11 У чому полягає багаторівнева структура організації мережної взаємодії?

12 Дайте визначення віртуальної і реальної взаємодій?

13 Назвіть основні завдання:
   - фізичного рівня;
   - канального рівня;
   - мережного рівня;
   - транспортного рівня;
   - сеансового рівня;
   - рівня подання даних;
   - прикладного рівня.

14 Чи можлива реальна взаємодія між протоколами, наприклад, транспортного й прикладного рівнів у еталонній моделі ISO OSI? Чому?

2 Сім’я протоколів TCP/IP

2.1 Будова мереж

Архітектура протоколів TCP/IP призначена для об'єднаної мережі, яка складається із окремих різнорідних пакетних підмереж, до яких підключаються різнорідні машини, з'єднані за допомогою *шлюзів* [3].

Розглянемо докладніше. Нехай існує 2 мережі – **A** і **B**, кожна зі своєю специфікацією і природою. Передбачається, що усередині себе кожна з них може передавати, одержувати й розподіляти інформацію.

А якщо необхідно здійснити зв'язок (передати інформацію) між мережами **A** і **B**? Тоді допомагають *шлюзи* (Shl) (рис. 3).

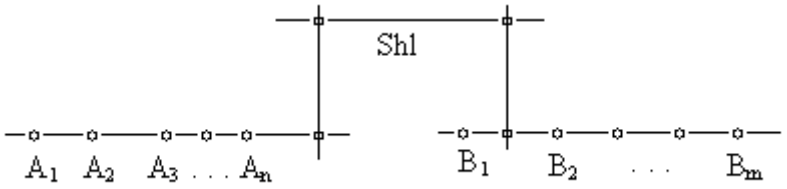

Рисунок 3

Можна дати кілька визначень поняття «шлюз».

Шлюз – це:

- комп'ютер, під'єднаний з одного боку в локальну мережу, а з іншого – в Інтернет;
- комп'ютер, що здійснює об'єднання локальних мереж у глобальні;
- вузол, що поєднує кілька різнорідних незалежних мереж у єдине ціле.

У разі необхідності передати пакет між машинами, підключеними до різних мереж, машина-відправник посилає пакет у відповідний *шлюз* (шлюз під'єднаний до мережі як звичайний вузол). Звідти пакет направляється певним маршрутом через систему шлюзів і мереж, поки не досягне шлюзу, під'єднаного до тієї самої мережі, що й машина-одержувач; там пакет направляється до одержувача.

2.2 Стек протоколів TCP/IP

В основу роботи мережі Інтернет покладено сім'ю протоколів TCP/IP (Transmission Control Protocol / Інтернет protocol). Це стандартний промисловий набір протоколів, розроблений для глобальних обчислювальних мереж. TCP/IP– це технологія міжмережної взаємодії.

Назва сім'ї протоколів утворилася від найбільш часто використовуваних протоколів TCP та IP. До складу сім'ї протоколів TCP/IP входять протоколи: ARP, TCP, UDP, IP, ICMP, FTP, TELNET, SMPT й ін.

Взаємозалежність протоколів сім'ї TCP/IP подана на рисунку 4.

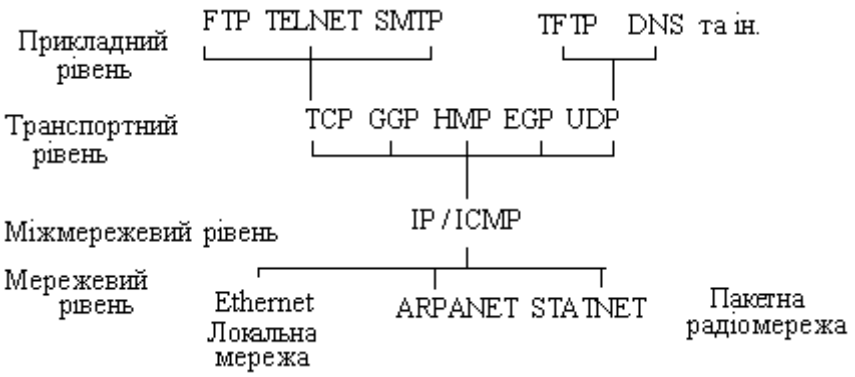

Рисунок 4 – Взаємозалежність протоколів сім'ї TCP/IP

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 та забезпечує можливість стандартизації протоколів верхніх рівнів.

Логічна структура мережного програмного забезпечення, що реалізує протоколи сім'ї TCP/IP у кожному вузлі мережі Інтернет, зображена на рисунку 5. Розуміння цієї логічної структури є основою для розуміння всієї технології Інтернет.

У схемі введені такі позначення:

☐	обробка даних,
\	шляхи передачі даних,
o	трансивер,
*	IP-адреса,
@	адреса вузла в мережі Ethernet (Ethernet-адреса).

Горизонтальна лінія внизу рисунка позначає кабель мережі Ethernet, що використовується як приклад фізичного середовища.

Розглянемо потоки даних, що проходять через стек протоколів (рис. 5).

У випадку використання протоколу TCP (Transmission Control Protocol – протокол керування передачею) дані передаються між прикладним процесом і модулем TCP. Типовим прикладним процесом, що використовує протокол TCP, є модуль FTP (File Transfer Protocol – протокол передачі файлів). *Стек протоколів* у цьому випадку буде таким: FTP/TCP/IP/ENET.

При використанні протоколу UDP (User Datagram Protocol – протокол датаграм користувача) дані передаються між прикладним процесом і модулем UDP. Наприклад, SNMP (Simple Network Management Protocol – простий протокол керування мережею) користується транспортними послугами UDP. Його *стек протоколів* має такий вигляд: SNMP/UDP/IP/ENET.

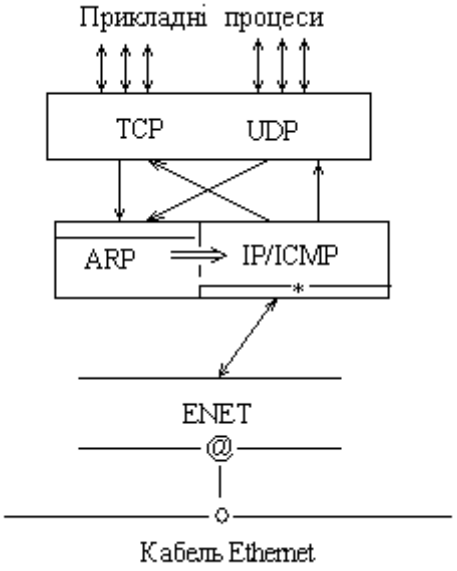

Рисунок 5 – Структура протокольних модулів у вузлі мережі TCP/IP

Модулі TCP, UDP і драйвер ENET є ***мультиплексорами***. Діючи як *мультиплексори*, вони *перемикають кілька входів на один вихід*. Вони також є ***демультиплексорами***. Як *демультиплексори* вони *перемикають один вхід на один з багатьох виходів відповідно до поля типу в заголовку протокольного блоку даних*.

Коли Ethernet-кадр потрапляє в драйвер мережного інтерфейсу ENET, він може бути спрямований або в модуль ARP (Address Resolution Protocol – адресний протокол), або в модуль IP (Internet Protocol – міжмережевий протокол). Про те, куди повинен бути спрямований Ethernet-кадр, свідчить значення поля типу в заголовку кадру.

Якщо IP-пакет потрапляє в модуль IP, то дані, які містяться в ньому, можуть бути передані або модулю TCP, або модулю UDP, це визначається полем "протокол" у заголовку IP-пакета.

Якщо UDP-датаграма потрапляє в модуль UDP, то на підставі значення поля "порт" у заголовку UDP-датаграми визначається прикладна програма, якій повинне бути передане прикладне повідомлення. Якщо TCP-повідомлення попадає в модуль TCP, то вибір прикладної програми, якій повинне бути передане повідомлення, здійснюється на основі значення поля "порт" у заголовку TCP-повідомлення.

Мультиплексування даних у зворотний бік здійснюється досить просто, тому що з кожного модуля існує тільки один шлях униз. Кожен протокольний модуль додає до пакета свій заголовок, на підставі якого машина, що прийняла пакет, виконує *демультиплексування*.

Дані від прикладного процесу проходять через модулі TCP або UDP, після чого потрапляють у модуль IP і звідти – на рівень мережного інтерфейсу.

2.3 Принципи адресації в Інтернет

Відомо, що Інтернет являє собою сукупність мереж, які співіснують за двома правилами:

- всі мережі використовують єдині умовні позначення для здійснення обміну й передачі даних;
- всі мережі використовують загальний принцип адресації.

Кожна машина, під'єднана до мережі, повинна мати своє унікальне ім'я, що прийнято називати *IP-адресою*. ***IP-адреса – це унікальне 32-бітне число (4-байтне), що має вигляд сукупності чотирьох чисел від 0 до 255, розділених між собою крапками.***

Наприклад:

134.30.0.17
192.168.1.56

IP-адресація комп'ютерів у мережі Інтернет побудована на концепції мережі, що складаєтья з *хостів* та інших мереж. ***Хост являє собою об'єкт мережі, що може передавати й приймати IP-пакети***, наприклад, комп'ютер робочої станції або маршрутизатор. Зазвичай помилково розуміють під хостом який-небудь сервер, однак у рамках концепції IP-мережі і робочі станції, і сервери – всі є хостами.

Хости з'єднані один з одним через одну або кілька мереж. ***IP-адреса*** кожного з хостів складається з адреси мережі й адреси хоста в цій мережі. IP-адресація, на відміну, наприклад, від IPX-адресації, використовує один ідентифікатор, що поєднує адресу мережі і адресу хоста. Як відзначалося вище, адреса подана чотирма десятковими числами, розділеними крапками. Кожне із цих чисел не може перевищувати 255 і являє собою один байт з 4-байтної IP-адреси. Довжина IP-адреси становить 32 біти, розділених на дві або три частини. Перша частина позначає *адресу мережі*, друга (якщо вона є) – *адреса підмережі*, третя – *адреса вузла*. Нагадаємо, по-перше, що а*дреса підмережі* наявна тільки у тому випадку, коли адміністратор мережі ухвалив рішення щодо поділу мережі на підмережі; по-друге, *IP-адреса вузла* ідентифікує точку доступу модуля IP до мережного інтерфейсу, а не всю машину.

Довжина полів адреси мережі, підмережі та хоста є змінними величинами. Менеджер мережі (адміністратор або програма) привласнює IP-адреси машинам відповідно до того, до яких IP-мереж вони під'єднані.

Для того, щоб показати, яка частина IP-адреси є ідентифікатором мережі (Network ID), а яка – ідентифікатором хоста (Host ID), вводять поняття "*маска мережі"*.

Маска мережі *– це шаблон, що накладаєтья на IP-адресу для встановлення, яка частина IP-адреси є ідентифікатором мережі (Network ID), а яка – ідентифікатором хоста (Host ID).* Наприклад, для адреси 197.200.12.5 і маски підмережі 255.255.255.0 Network ID буде 197.200.12, а Host ID – 5.

Маска мережі використовується при обміні даними між двома хостами. Якщо хост-комп'ютери **A** і **B** належать одній мережі, хост **A** безпосередньо звертається до хосту **B**, однак якщо хост **B** належить іншій мережі, хост **A** буде звертатися через шлюз, і спосіб, яким хост **A** може повідомити про свою належність даної мережі, – це використання маски мережі. Наприклад, нехай маємо три хости

вузол **A** з IP-адресою	200.163.10.5
вузол **B**	200.163.10.9
вузол **C**	200.163.199.11

і маску мережі 255.255.255.0

Припустимо хост **A** з'єднується з хостом **B**, оскільки обидва вони мають Network ID 200.163.10, то хост **A** звертається до хосту **B** безпосередньо. Якщо ж хост **A** з'єднується з хостом **C**, а вони перебувають у різних мережах (200.163.10 й 200.163.199 відповідно), то хост **A** буде надсилати запит через шлюз.

Існують 5 класів IP-адрес, що відрізняються кількістю бітів у мережному номері й хост-номері. Клас адреси визначається значенням його першого октету.

У таблиці 2 наведена відповідність класів адрес значенням першого октету й зазначена кількість можливих IP-адрес кожного класу.

Таблиця 2 – Характеристики класів адрес

Клас	Діапазон значень першого октету	Маска мережі за замовчуванням	Можлива кількість мереж	Можлива кількість вузлів
A	1 - 126	255.0.0.0	126	16777214
B	128 - 191	255.255.0.0	16382	65534
C	192 - 223	255.255.255.0	2097150	254
D	224 - 239		-	2^{28}
E	240 - 247		-	2^{27}

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, наприклад, в мережах університетів і великих компаній. Адреси класу C використовуються в мережах з невеликим числом комп'ютерів. Адреси класу D – при звертаннях до груп машин, а адреси класу E зарезервовані на майбутнє.

Розглядають *глобальні* (IP-) і *локальні* (Ethernet-) адреси.

Ethernet-адреса – це унікальне 6-байтне число, що має вигляд сукупності шести чисел в 16-й системі числення, розділених між собою двокрапкою або тире.

Якщо локальна адреса привласнюється машині постачальником послуг Інтернет, то Ethernet-адреса є унікальним для кожного мережного адаптера й розпізнається драйвером. Іншими словами, Ethernet-адреса відома машині, коли тільки у неї вставили мережну плату або інший мережний адаптер. Ethernet-адреса «вшита» у мережний адаптер виробником.

Машина, яка працює в глобальній мережі, завжди знає свої IP- та Ethernet- адреси.

2.4 Протокол ARP

Розглянемо те, як при пересиланні IP-пакета визначається Ethernet-адреса призначення. Для відображення IP-адрес в Ethernet-адреси використовується протокол ARP (Address Resolution Protocol – адресний протокол) [3]. Відображення виконується *тільки для IP-пакетів*, які відправляються, тому що тільки в момент відправлення створюються IP- і Ethernet-заголовки.

Перетворення адрес виконується шляхом пошуку відповідності в таблиці. Ця таблиця, називана ARP-таблицею, зберігається в пам'яті комп'ютера й містить рядки для кожного вузла мережі. У двох стовпцях утримуються IP- і Ethernet-адреси. Якщо потрібно перетворити IP-адресу на Ethernet-адресу, то відшукується запис у ARP-таблиці з відповідною IP-адресою. У таблиці 3 наведений приклад спрощеної ARP-таблиці.

IP-адреса	**Ethernet-адреса**
223.1. 2.1	08:00:39:00:2F:C3
223.1. 2.3	08:00:5A:21:A7:22
223.1. 2.4	08:00:10:99:AC:54

ARP-таблиця необхідна тому, що IP-адреси й Ethernet-адреси вибираються незалежно, і немає якого-небудь алгоритму для перетворення однієї адреси в іншу. IP-адресу призначає

менеджер мережі з урахуванням положення машини в мережі Інтернет. Якщо машину переміщають в іншу частину мережі Інтернет, то її IP-адреса повинна бути змінена. Ethernet-адресу призначає виробник мережного інтерфейсного обладнання з виділеного для нього за ліцензією адресного простору. Коли у машини замінюють мережне обладнання, наприклад, мережну плату, то змінюється і її Ethernet-адреса.

У ході звичайної роботи мережна програма, така, як TELNET, відправляє прикладне повідомлення, користуючись транспортними послугами TCP. Модуль TCP посилає відповідне транспортне повідомлення через модуль IP. У результаті складається IP-пакет, що повинен бути переданий драйверу Ethernet. IP-адреса місця призначення відома прикладній програмі, модулю TCP і модулю IP. Необхідно на її основі знайти Ethernet-адресу місця призначення. Для визначення необхідної Ethernet-адреси використовується ARP-таблиця.

Як же заповнюється ARP-таблиця? Вона заповнюється автоматично модулем ARP, у міру необхідності. Коли за допомогою існуючої ARP-таблиці не вдається перетворити IP-адресу, то відбувається таке:

1) по мережі передається широкомовний ARP-запит;

2) вихідний IP-пакет ставиться в чергу.

Кожен мережний адаптер приймає широкомовні передачі. Всі драйвери Ethernet перевіряють поле типу в прийнятому Ethernet-кадрі й передають ARP-пакети модулю ARP. ARP-запит можна інтерпретувати так: "Якщо ваша IP-адреса збігається із зазначеною, то повідомте мені вашу Ethernet-адресу". Пакет ARP-запиту має приблизно такий вигляд:

IP-адреса відправника	223.1. 2.1
Ethernet-адреса відправника	08:00:39:00:2F:C3
Шукана IP-адреса	223.1. 2.2
Шукана Ethernet-адреса	<порожньо>

Кожен модуль ARP перевіряє поле шуканої IP-адреси в отриманому ARP-пакеті, і якщо адреса збігається з його власною IP-адресою, то посилає відповідь прямо по Ethernet-адресі відправника запиту. ARP-відповідь можна інтерпретувати так: "Так, це моя IP-адреса, їй відповідає така-то Ethernet-адреса". Пакет з ARP-відповіддю має приблизно такий вигляд:

IP-адреса відправника	223.1. 2.1
Ethernet-адреса відправника	08:00:39:00:2F:C3
Шукана IP-адреса	223.1. 2.2
Шукана Ethernet-адреса	08:00:39:00:2F:C3

Цю відповідь одержує машина, що зробила ARP-запит. Драйвер цієї машини перевіряє поле типу в Ethernet-кадрі й передає ARP-пакет модулю ARP. Модуль ARP аналізує ARP-пакет і додає запис у свою ARP-таблицю. Оновлена таблиця має такий вигляд:

IP-адреса	**Ethernet-адреса**
223.1. 2.1	08:00:39:00:2F:C3
223.1. 2.2	08:00:28:00:38:A9
223.1. 2.3	08:00:5A:21:A7:22
223.1. 2.4	08:00:10:99:AC:54

Новий запис в ARP-таблиці з'являється автоматично, через декілька мілісекунд після того, як вона була потрібна. Як вже зазначалося, раніше на кроці 2 вихідний IP-пакет був поставлений у чергу. Тепер з використанням оновленої ARP-таблиці виконується перетворення IP-адреси в Ethernet-адресу, після чого Ethernet-кадр передається по мережі. Повністю порядок перетворення адрес має приблизно такий вигляд:

1 По мережі передається широкомовний ARP-запит.

2 Вихідний IP-пакет ставиться в чергу.

3 Повертається ARP-відповідь, що містить інформацію про відповідність IP- і Ethernet-адрес. Ця інформація заноситься в ARP-таблицю.

4 Для перетворення IP-адреси в Ethernet-адресу IP-пакета, поставленого в чергу, використовується ARP-таблиця.

5 Ethernet-кадр передається по мережі Ethernet.

Якщо за допомогою ARP-таблиці не вдається відразу здійснити перетворення адрес, то IP-пакет ставиться в чергу, а необхідна для перетворення інформація надходить за допомогою запитів і відповідей протоколу ARP, після чого IP-пакет передається за призначенням.

Якщо в мережі немає машини із шуканою IP-адресою, то ARP-відповіді не буде й не буде запису в ARP-таблиці. Протокол IP буде знищувати IP-пакети, що направляються за зазначеною IP-адресою. Зазначимо, що протоколи верхнього рівня не можуть відрізнити випадок ушкодження мережі Ethernet від випадку відсутності машини із шуканою IP-адресою.

Деякі реалізації IP- й ARP- протоколів не ставлять у чергу IP-пакети на той час, поки вони чекають ARP-відповідей. Замість цього IP-пакет просто знищується, а його відновлення покладається на модуль TCP або прикладний процес, що працює через UDP. Таке відновлення виконується за допомогою таймаутів і повторних передач. Повторна передача повідомлення відбувається успішно, тому що перша спроба вже викликала заповнення ARP-таблиці.

Слід зазначити, що кожна машина має окрему ARP-таблицю для кожного свого мережного інтерфейсу.

Висновки:

Інтернет являє собою сукупність мереж, які співіснують за певними правилами: всі мережі використовують *єдині умовні позначення* для здійснення обміну (передачі) даних і *загальний принцип адресації*.

Використання єдиних позначень забезпечує сім'я протоколів TCP/IP, покладена в основу роботи мережі Інтернет. У сім'ї TCP/IP виділяють 4 рівні протоколів:

- прикладний рівень;
- транспортний рівень;
- міжмережний рівень;
- мережний рівень.

Кожен вузол, підключений до мережі Інтернет, має *глобальну* IP-адресу та локальну Ethernet-адресу. Варто звернути особливу увагу на такі положення:

IP-адреса вузла ідентифікує точку доступу модуля IP до мережного інтерфейсу, а не всю машину.

IP- та Ethernet- адреси привласнюються вузлу, що входить в Інтернет, незалежно один від одного. IP- адреса призначається провайдером при підключенні до мережі Інтернет, а Ethernet-адреса привласнюється виробником мережного інтерфейсу.

З огляду на незалежність IP- і Ethernet- адрес і той факт, що всі вихідні пакети відправляються на IP-адресу, а надходять на Ethernet-адресу, був розроблений ARP-протокол.

Протокол ARP використовується для відображення IP-адрес в Ethernet-адреси шляхом пошуку відповідності в ARP-таблиці. Якщо вузол, підключений до мережі Інтернет, має два або

декілька мережних інтерфейсів, то в його пам'яті обов'язково зберігаються два або стільки, скільки мережних інтерфейсів, незалежних ARP-таблиць.

Контрольні запитання

1 Що таке шлюз?

2 Дайте коротку характеристику сім'ї протоколів TCP/IP.

3 Як узгоджується між собою семирівнева еталонна модель організації мережної взаємодії та чотирирівнева диференціація сім'ї протоколів TCP/IP?

4 Поясніть логічну структуру мережного програмного забезпечення, що реалізує сім'ю протоколів TCP/IP

5 Що являє собою стек протоколів?

6 Які протоколи є мультиплексорами, які демультиплексорами?

7 Що собою являє IP-адреса?

8 Що таке Ethernet-адреса?

9 У чому полягає різниця між IP- і Ethernet- адресацією?

10 Дати визначення хост-комп'ютера.

11 Що таке маска мережі?

12 Назвіть основне призначення протоколу ARP.

13 Що являє собою ARP-таблиця?

14 Як та у який момент здійснюється відображення IP-адреси на Ethernet- адресу?

2.5 Міжмережний протокол IP

2.5.1 Формат IP-датаграми

Internet Protocol (IP) створений для використання в об'єднаних системах комп'ютерних комунікаційних мереж з комутацією пакетів. Він *забезпечує передачу блоків даних*, ***датаграм*** *від відправника до одержувачів*, де відправник та одержувач є хост-комп'ютерами, які ідентифікуються адресами фіксованої довжини. IP-протокол *забезпечує за необхідності також фрагментацію та дефрагментацію датаграм для передачі даних через мережі з малим розміром пакетів* [3].

Таким чином, IP-протокол виконує дві ***головні функції: адресацію і фрагментацію***.

Як зазначалося раніше, вся інформація передається в мережі Інтернет за допомогою IP-датаграм (Internet-датаграм), причому кожна датаграма має заздалегідь певну структуру.

Розглянемо повну схему IP-датаграми (рис.6).

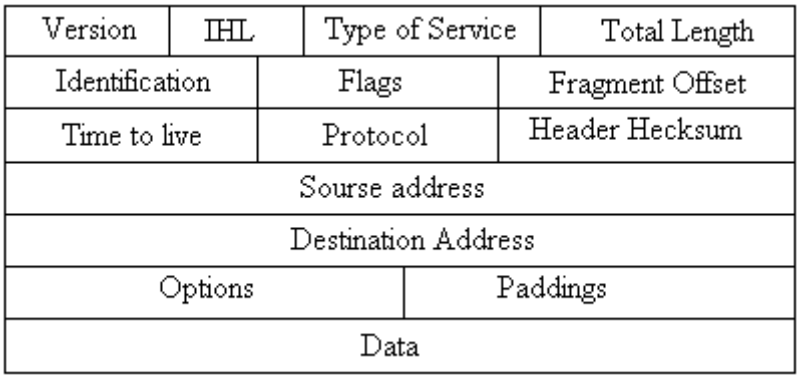

Рисунок 6 – Схема IP-датаграми

Тут

Version (версія) – поле, призначене для називання версії, що визначає формат IP-заголовка. Розглянутий заголовок описує версію 4.

IHL – визначає довжину IP-заголовка, виміряється в словах по 32 біти кожне і вказує на початок поля даних. Коректний заголовок може мати мінімальний розмір 5 слів.

Type of Service – визначає тип необхідного сервісного обслуговування. Ці параметри повинні використовуватися для керування вибором реальних робочих характеристик при передачі датаграми через конкретну мережу. Реально вибір здійснюється між трьома альтернативами:

- малою затримкою;
- високою вірогідністю;
- високою пропускною здатністю.

Для даного поля виділяється 8 бітів:

біти 0-2 визначають пріоритет:

111 – керування мережею;
110 – міжмережне керування;
101 – CRITIC/ECP;
100 – більш ніж миттєво;
011 – миттєво;
010 – негайно;

001 – пріоритетно;
000 – звичайний маршрут;

біт 3 визначає затримку:
0 – нормальна затримка;
1 – мала затримка;

біт 4 визначає пропускну здатність:
0 – нормальна пропускна здатність;
1 – висока пропускна здатність;

біт 5 визначає достовірність:
0 – звичайна достовірність;
1 – висока достовірність;

біти 6-7 зарезервовані.

Використання індикації *затримки*, *пропускної здатності* й *вірогідності* може, у деякому розумінні, збільшити вартість обслуговування. У багатьох мережах поліпшення одного із цих параметрів пов'язане з погіршенням іншого. Винятки, коли існує сенс встановлювати два із цих трьох параметрів, дуже рідкі.

Total Length – визначає загальну довжину IP-датаграми. Загальна довжина – це довжина датаграми, яка вимірюється в октетах (байтах), включаючи IP-заголовок і поле даних. Це поле може задавати довжину датаграми аж до 65535 октетів.

У більшості хост-комп'ютерів і мереж настільки великі датаграми не використовуються. Всі хости повинні бути готові приймати датаграми довжиною 576 октетів, незалежно від того, чи надходять вони єдиним цілим або фрагментами. Хостам рекомендується відправляти датаграми розміром більше 576 октетів, тільки в тому випадку, якщо вони впевнені, що хост, який приймає, готовий обслуговувати датаграми підвищеного розміру.

Значення 576 вибране для того, щоб відповідним чином обмежений блок даних передавався разом з необхідною інформацією в заголовку.

Identification (ідентифікатор) – визначає ідентифікатор, що встановлюється відправником для збирання фрагментів якої-небудь датаграми.

Flags – визначає різні керуючі прапорці. Під поле відводяться 3 біти, які можна подати у такому вигляді:

0	1	2
0	DF	MF

Біт 0 – зарезервований, повинен бути нулем.
Біт 1 (DF) – відповідає за фрагментацію датаграми:
0 – можлива фрагментація;
1 – заборона фрагментації.
Біт 2 (MF) – прапорець появи додаткових фрагментів:
0 – останній фрагмент;
1 – будуть ще фрагменти.

Fragment Offset – визначає зсув фрагмента й показує, де в датаграмі перебуває цей фрагмент. Зсув фрагмента змінюється порціями по 8 октетів (64 біти). Перший фрагмент має зсув нуль.

Time to Live (TTL) – визначає максимальний час, протягом якого датаграмі дозволено перебувати в системі Інтернет. Якщо це поле має значення нуль, то датаграма повинна бути зруйнована. Значення цього поля змінюється при обробці IP-заголовка. Час виміряється в

секундах. Однак оскільки кожен модуль, що обробляє датаграму, повинен зменшувати значення поля TTL принаймні на одиницю навіть якщо він обробляє цю датаграму менше, ніж за секунду, то *поле TTL варто розуміти як максимальний інтервал часу, протягом якого датаграма може існувати* в Інтернеті.

Protocol – визначає протокол більш високого (наступного) рівня, що використовує дані з IP-датаграми.

Header Checksum – визначає контрольну суму заголовка. Оскільки деякі поля заголовка змінюють своє значення (наприклад, TTL), це значення перевіряється й повторно перераховується при кожній обробці IP-заголовка.

Source Address – визначає адресу відправника.

Destination Address – визначає адресу одержувача.

Options – поле змінної довжини для службових записів. Опції можуть з'явитися у датаграмах, а можуть і не з'являтися. Вони повинні підтримуватися всіма Інтернет- модулями (хостами й шлюзами). Необов'язково кожна конкретна датаграма має опції, але мати їх все-таки може.

Padding – забезпечує вирівнювання. Вирівнювання Інтернет-заголовка використовується для того, щоб переконатися в тому, чи закінчується Інтернет-заголовок на 32-бітній границі. Вирівнювання здійснюється нулями.

Data – поле, що містить дані, які передаються по мережі Інтернет.

Наприклад, заголовок мінімальної IP-датаграми, що несе дані (рис. 7).

<table>
<tr><td>Version =4</td><td>IHL =5</td><td>Type of Service</td><td>Total Length =21</td></tr>
<tr><td colspan="2">Identification =111</td><td>Flags =0</td><td>Fragment Offset =0</td></tr>
<tr><td colspan="2">Time to live =123</td><td>Protocol=1</td><td>Header Hecksum</td></tr>
<tr><td colspan="4">Sourse address</td></tr>
<tr><td colspan="4">Destination Address</td></tr>
<tr><td colspan="3">Options</td><td>Paddings</td></tr>
<tr><td colspan="4">Data</td></tr>
</table>

Рисунок 7 – Заголовок мінімальної IP-датаграми, що містить дані

2.5.2 Маршрутизація пряма й непряма. Правила маршрутизації

Раніше зазначалося, що однією з основних функцій IP-протоколу є адресація. IP-протокол використовує адреси, розміщені в IP-заголовку, для передачі IP-датаграм їх одержувачам. *Вибір шляху передачі даних називається* ***маршрутизацією***.

Центральною частиною протоколу IP є його ***таблиця маршрутів***. Протокол використовує цю таблицю при прийнятті всіх рішень про маршрутизацію IP-пакетів. Зміст таблиці маршрутів визначається адміністратором мережі. Помилки при встановленні маршрутів можуть заблокувати передачу даних.

Щоб зрозуміти техніку міжмережної взаємодії, потрібно зрозуміти те, як використовується таблиця маршрутів.

Пряма маршрутизація

Розглянемо невелику IP-мережу (рис. 8), що складається з 3 машин: **A**, **B** і **C**. Кожна машина має такий самий стек протоколів TCP/IP, як показано на рисунку 4. Кожен мережний адаптер цих машин має свою Ethernet-адресу. Менеджер мережі повинен привласнити машинам унікальні IP-адреси.

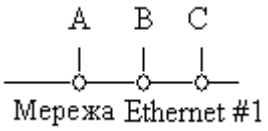

Рисунок 8 – Проста IP-мережа

Коли **A** посилає IP-пакет **B**, то заголовок IP-пакета містить у полі відправника IP-адресу вузла **A**, а заголовок Ethernet-кадру містить у полі відправника Ethernet-адресу **A**. Крім цього, IP-заголовок містить у полі одержувача IP-адресу вузла **B**, а Ethernet-заголовок містить у полі одержувача Ethernet-адресу **B**. Наприклад, адреси в Ethernet-кадрі, що передає IP-пакет від **A** до **B**, мають такий вигляд:

Адреса	**Відправник**	**Одержувач**
IP-заголовок	A	B
Ethernet-заголовок	A	B

У цьому простому прикладі протокол IP є надмірністю, що мало чого додає до послуг, які надаються мережею Ethernet. Однак протокол IP вимагає додаткових витрат на створення, передачу й обробку IP-заголовка. Коли в машині **B** модуль IP одержує IP-пакет від машини **A**, він порівнює IP-адресу місця призначення зі своєю і, якщо адреси збігаються, передає датаграму протоколу верхнього рівня. У цьому випадку при взаємодії **A** з **B** використовується ***пряма маршрутизація***.

Непряма маршрутизація

На рисунку 9 представлена більш реалістична картина мережі Інтернет. У цьому випадку мережа Інтернет складається із трьох мереж Ethernet, на базі яких працюють три IP-мережі, об'єднані шлюзом **D**. Кожна IP-мережа включає чотири машини; кожна машина має свої власні IP- і Ethernet-адреси.

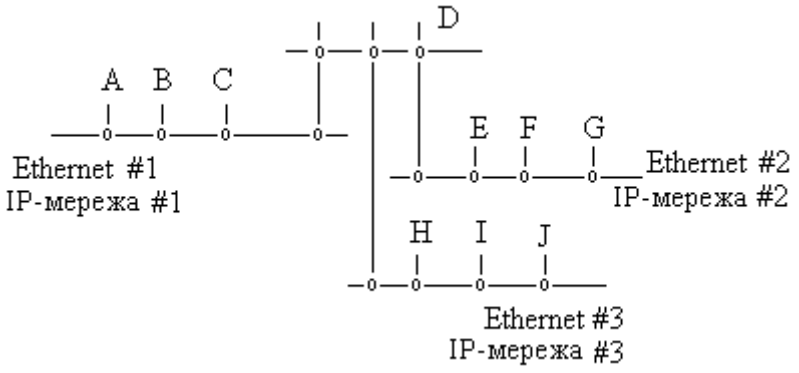

Рисунок 9 – Мережа Інтернет, що складається із трьох IP-мереж

За винятком **D**, всі машини мають стек протоколів, аналогічний показаному на рисунку 4. Шлюз **D** з'єднує всі три мережі, отже, має три IP-адреси та три Ethernet-адреси. Машина **D** має стек протоколів TCP/IP, що містить три модулі ARP і три драйвери Ethernet. Звернемо увагу на те, що машина **D** має тільки один модуль IP.

Менеджер мережі надає кожній мережі Ethernet унікальний номер, названий IP-номером мережі. На рисунку 9 IP-номери не зазначені, замість них використовуються імена мереж (IP-мережа #1, IP-мережа #2, IP-мережа #3).

Коли машина **A** посилає IP-пакет машині **B**, то процес передачі відбувається в межах однієї мережі. При всіх взаємодіях між машинами, підключеними до однієї IP-мережі, використовується пряма маршрутизація, що обговорювалася в попередньому прикладі.

Коли машина **D** взаємодіє з машиною **A**, то це *пряма взаємодія*. Коли машина **D** взаємодіє з машиною **E**, то це також пряма взаємодія. Коли машина **D** взаємодіє з машиною **H** – знову пряма взаємодія. Це здійснюється саме так, оскільки кожна пара цих машин належить одній IP-мережі.

Однак коли машина **A** взаємодіє з машинами, які об'єднані іншою IP-мережею, то взаємодія вже не буде прямою. Машина **A** повинна використати шлюз **D** для ретрансляції IP-пакетів в іншу IP-мережу. Така взаємодія називається ***непрямою***.

Маршрутизація IP-пакетів виконується модулями IP та є прозорою для модулів TCP, UDP і прикладних процесів.

Якщо машина **A** посилає машині **E** IP-пакет, то IP-адреса й Ethernet-адреса відправника відповідають адресам **A**. IP-адреса місця призначення є адресою **E**, але оскільки модуль IP машини **A** посилає IP-пакет через **D**, Ethernet-адреса місця призначення є адресою **D**:

Адреса	**Відправник**	**Одержувач**
IP-заголовок	A	E
Ethernet-заголовок	A	D

Модуль IP у машині **D** одержує IP-пакет і перевіряє IP-адресу місця призначення. Визначивши, що це не його IP-адреса, шлюз **D** посилає цей IP-пакет прямо до **E**, тобто Ethernet-кадр після шлюзу **D** буде мати такий вигляд:

Адреса	**Відправник**	**Одержувач**
IP-заголовок	A	E
Ethernet-заголовок	A	E

Отже, *при **прямій маршрутизації** IP- і Ethernet-адреси відправника відповідають адресам того вузла, що послав IP-пакет, а IP- і Ethernet-адреси місця призначення відповідають адресам одержувача. При **непрямій маршрутизації** IP- і Ethernet-адреси не утворять таких пар.*

У даному прикладі мережа Інтернет є дуже простою. Реальні мережі можуть бути набагато складнішими, тому що можуть містити кілька шлюзів і кілька типів фізичних середовищ передачі. У наведеному прикладі кілька мереж Ethernet поєднуються шлюзом для того, щоб локалізувати широкомовний трафік у кожній мережі.

Правила маршрутизації

1 Для IP-пакетів, які надійшли від модулів верхнього рівня для відправлення, модуль IP повинен визначити спосіб доставки – прямий або непрямий – і вибрати мережний інтерфейс. Цей вибір робиться на підставі результатів пошуку в таблиці маршрутів.

2 Для прийнятих IP-пакетів, що надходять від мережних драйверів, модуль IP повинен вирішити, чи потрібно ретранслювати IP-пакет по іншій мережі або передати його модулям верхнього рівня. Якщо модуль IP вирішить, що IP-пакет необхідно ретранслювати, то подальша робота з ним здійснюється так само, як з IP-пакетами, що відправляються (див. пункт 1).

3 Вхідний IP-пакет ніколи не ретранслюється через той самий мережний інтерфейс, яким він був прийнятий.

4 Рішення про маршрутизацію приймається до того, як IP-пакет передається мережному драйверу, і до того, як відбувається звертання до ARP-таблиці.

2.5.3 Фрагментація та дефрагментація датаграм

При переході даних з мереж, які характеризуються великою пропускною здатністю, в мережі з малою пропускною здатністю та навпаки виникає необхідність фрагментації й дефрагментації датаграм відповідно.

Тому другою *основною функцією* IP-протоколу *є фрагментація й дефрагментація датаграм* для передачі даних через мережі з малим розміром пакетів.

IP-протокол використовує спеціальне поле *Flags* в IP-заголовку для фрагментації та відновлення IP-датаграм, коли це необхідно для їх передачі через мережі з малим розміром пакетів.

Ще раз звернемо увагу на те, що кожний IP-модуль повинен бути здатний передати датаграму з 68 октетів без подальшої фрагментації. Це відбувається тому, що IP-заголовок може включати до 60 октетів, а мінімальний фрагмент – 8 октетів. Кожен IP-одержувач повинен приймати датаграму розміром в 576 октетів як єдиний шматок, або у вигляді фрагментів, що підлягають збиранню.

Розглянемо, як реалізуються процедури фрагментації й дефрагментації на прикладах.

У наступних псевдопрограмах введемо такі позначення:

≤ менше або дорівнює;

≠ не дорівнює;

= дорівнює;

← встановлюється в;

вираз «від *x до y*» варто розуміти як інтервал [x,y).

Наприклад, вираз "від 0 до 5" означав би включення 0, 1, 2, 3 й 4, але не включало б 5;

FO – зсув фрагмента;
IHL – довжина IP-заголовка;
DF – прапорець заборони фрагментації;
MF – прапорець появи додаткових фрагментів;
TTL – час існування;
NFB – кількість фрагментів;
TL – загальна довжина;
TDL – загальна довжина даних;
OFO – старий зсув фрагмента;
OIHL – стара довжина IP-заголовка;
OMF – старе значення прапора MF;
OTL – старе значення загальної довжини;
NFB – кількість блоків фрагментації;
MTU – максимальна довжина переносу;
BUFID – ідентифікатор буфера;

RCVBT – бітова таблиця фрагментації;

TLB – нижня границя для значення таймера.

Процедура фрагментації

Датаграма найбільшого розміру, що ще може бути передана через чергову локальну мережу, називається ***максимальною одиницею, що передається*** (maximum transmission unit – MTU).

Якщо загальна довжина датаграми менше або дорівнює максимальній одиниці, що передається, то датаграма передається наступним процедурам обробки. У протилежному разі спочатку вона розбивається на два фрагменти, причому перший з них буде мати розмір максимальної одиниці, а в другий фрагмент буде поміщено залишок вихідної датаграми. Перший фрагмент відправляється на подальшу обробку, а другий повторно піддається тільки що розглянутій процедурі, якщо його розміри виявляться занадто великими.

Процедура:

BEGIN
IF TL ≤ MTU
THEN відправити датаграму на наступні процедури обробки
ELSE
IF DF =1
THEN зруйнувати датаграму
ELSE
BEGIN
(1) Створити перший фрагмент:
- скопіювати вихідний IP-заголовок;
- OIHL ← IHL; OTL ← TL; OMF ← MF;
- NFB ← (MTU - IHL×4)/8;
- взяти перші NFB×8 октетів даних;
- скорегувати заголовок:
  - MF←1;TL← (IHL×4)+(NFB×8);
  - перерахувати контрольну суму;
- направити даний фрагмент на наступні процедури обробки.

(2) Створити другий фрагмент:
- вибірково скопіювати IP-заголовок (деякі опції не копіюються);
- додати дані, що залишилися;
- корегування заголовка:
  - IHL←(((OIHL*4)-(довжина нескопійованих опцій))+3)/4;
  - TL←OTL -NFB×8 -(OIHL-IHL)×4;
  - FO ← OFO + NFB;
  - MF ←OMF;
  - перерахувати контрольну суму;
- приготувати цей фрагмент до повторного тесту на необхідність фрагментації. Виконати тест.

END
END.

У попередній процедурі кожен фрагмент (за винятком останнього) одержує максимально дозволену довжину. Альтернатива може полягати у створенні датаграм, які не досягають максимального розміру. Для прикладу, вона може включати процедуру фрагментації, що

повторно ділить більші датаграми навпіл доти, поки фрагменти, що утворюються, не стануть коротші, ніж максимальний припустимий розмір одиниці, що передається.

Процедура дефрагментації

Для кожної датаграми ідентифікатор буфера визначається як об'єднання полів адреси відправника, адреси одержувача, протоколу й ідентифікації. Якщо це ціла датаграма (поля fragment offset та more fragments нульові), то всі ресурси, пов'язані із цим ідентифікатором буфера, звільняються, а сама датаграма направляється на наступні процедури обробки.

Якщо наступний фрагмент не пов'язаний із цим ідентифікатором буфера, то виділяються ресурси для збирання. Вони включають буфер даних, буфер заголовка, бітову таблицю фрагментації, поле загальної довжини даних, а також таймер. Дані із фрагмента розміщують в буфері даних у відповідності до значень полів fragment offset і totla lenght, а також установлюються біти в бітовій таблиці фрагментації відповідно до отриманих блоків фрагментів.

Якщо це перший фрагмент (поле fragment offset нульове), то його заголовок міститься в буфер заголовка. Якщо це останній фрагмент (поле more fragments нульове), то обчислюється загальна довжина даних. Якщо цей фрагмент завершує датаграму (перевіряється по установці бітів у таблиці фрагментації), то датаграма направляється на наступний етап обробки. У протилежному разі таймер установлюється на максимальне із двох: поточне значення таймера та час існування для даного фрагмента. Виконання процедури збирання припиняється.

Якщо таймер відрахував заданий час, то всі ресурси збирання, пов'язані з даним ідентифікатором буфера, звільняються. Первісна установка таймера є нижньою границею для часу очікування при збиранні. Це відбувається тому, що час очікування буде збільшено, якщо час життя фрагмента, який надходить, виявиться більше, але не може бути менше. Установка таймера може досягати максимального часу існування (приблизно 4,25 хвилини). У цей час рекомендується спочатку встановлювати таймер на 15 секунд. Це значення можна змінити при одержанні достатнього практичного досвіду. Помітимо, що вибір значення для цього параметра зв'язку пов'язаний з ємністю буфера і швидкістю одержання даних з комунікаційних мереж. Наприклад, швидкість одержання даних слід множити на розмір буфера (тобто 10 кбайт/с × 15 с = 150 кбайт).

Процедура:

BEGIN
BUFID ← відправник | отримувач | протокол | ідентификація;
IF (FO = 0) ***AND*** (MF = 0)
THEN
IF буфер з ідентифікатором BUFID виділені
THEN
BEGIN
- завершити збирання для цього ідентифікатора BUFID;
- приготувати датаграму для подальшої обробки;
- запустити обробку.

END
ELSE
BEGIN
IF буфер для ідентифікатора BUFID не виділений
THEN
BEGIN
- виділити ресурси для збирання з ідентифікатором BUFID;

- TIMER ← TLB;
- TDL ← 0;
- перенести дані із фрагмента в буфер даних з ідентифікатором BUFID, дані з октету FO*8 по октет (TL-(IHL×4))+FO×8;
- встановити біти RCVBT з FO по FO+((TL-(IHL×4)+7)/8);

END

IF MF = 0

THEN TDL ← TL-(IHL×4)+(FO×8)

IF FO = 0

THEN помістити заголовок у буфер заголовка

IF TDL ≠ 0 AND всі біти RCVBT від 0 до (TDL+7)/8 виставлені

THEN

BEGIN

- TL ← TDL+(IHL×4);
- приготувати датаграму до подальшої обробки;
- звільнити всі ресурси збирання для цього ідентифікатора BUFID. Запустити обробку.

END

TIMER ← MAX(TIMER,TTL);

Призупинити роботу до одержання наступного фрагмента або сигналу від таймера

END

Сигнал від таймера: Звільнити всі ресурси, пов'язані із цим ідентифікатором BUFID.

END.

У випадку, якщо два або більше фрагменти містять або ті самі дані, або ідентичні, або частково перекриваються, то ця процедура буде використовувати останню отриману копію при створенні буфера даних і відтворенні датаграми.

Таким чином, IP-протокол використовує ***чотири ключових механізми*** для формування своїх послуг: *задання типу сервісу, часу життя датаграми, опцій і контрольної суми заголовка*.

Тип обслуговування або *тип сервісу* використовується для позначення необхідної послуги. ***Тип обслуговування – це абстрактний або узагальнений набір параметрів, що характеризує набір послуг, які надаються мережами, і складають, власне кажучи, IP-протокол.*** Цей спосіб позначення послуг повинен використовуватися шлюзами для вибору робочих параметрів передачі в конкретній мережі, для вибору мережі, використовуваної при подальшому переході датаграми, для вибору наступного шлюзу при маршрутизації мережної IP-датаграми.

Механізм часу життя датаграми використовується для зазначення верхньої межі часу існування IP-датаграми. Цей параметр здається відправником датаграми й зменшується на кожному вузлі маршруту, по якому проходить датаграма. Якщо параметр часу життя стане нульовим до того, як IP-датаграма досягне одержувача, то датаграма буде знищена. Час життя можна розглядати як годинниковий механізм самознищення.

Механізм опцій надає різні функції керування, які є необхідними або просто корисними при певних ситуаціях, однак він не потрібний при звичайних комунікаціях. Механізм опцій надає такі можливості, як тимчасові штампи, безпека, спеціальна маршрутизація та ін.

Контрольна сума заголовка забезпечує перевірку того, що інформація, яка використовувалась при обробці IP-датаграм, передана правильно. Якщо контрольна сума виявиться неправильною, IP-датаграма буде зруйнована, коли помилка буде виявлена.

До ***недоліків IP-протоколу*** можна віднести таке:

1 Протокол не забезпечує надійності комунікації.

2 Не має механізму підтверджень ні між відправником й одержувачем, ні між хост-комп'ютерами .

3 Не має контролю помилок для поля даних, тільки контрольна сума для заголовка.

4 Не підтримується повторна передача, відсутнє керування потоком даних.

Висновки

IP-протокол – це ключовий протокол сім'ї TCP/IP, який забезпечує роботу всієї мережі Інтернет. Інформація, що оброблюється протоколом, розбита на датаграми, які часто називають IP-датаграмами.

IP-протокол виконує дві головні функції: адресацію і фрагментацію.

IP-протокол використовує адреси, розміщені в IP-заголовоку, для передачі IP-датаграм одержувачам. Вибір шляху передачі IP-датаграми називається маршрутизацією.

IP-протокол використовує поля в IP-заголовку для фрагментації й відновлення датаграм, коли це необхідно для передачі даних через мережі з малим розміром пакетів.

Сценарій дії IP-протоколу полягає в тому, що модуль IP змінює розмір IP-датаграми на кожному з хостів, які задіяні в IP-комунікації, і на кожному зі шлюзів, що забезпечують взаємодію між мережами. Ці модулі дотримуються загальних правил для інтерпретації полів адрес, для фрагментації й збирання IP-датаграм. Крім цього, дані модулі (і особливо шлюзи) мають процедури для прийняття рішень щодо маршрутизації пакетів, а також інші функції.

IP-протокол працює з кожною датаграмою як з незалежною одиницею, що не має зв'язку ні з якими іншими датаграмами.

IP-протокол використовує чотири ключових механізми для формування своїх послуг: тип сервісу, час життя, опції і контрольні суми заголовка.

Контрольні запитання

1 Зробіть перелік основних функцій протоколу IP.

2 Назвіть поля IP-заголовка.

3 Дайте характеристику для кожного поля IP-заголовка.

4 Який розмір датаграми повинні приймати всі хост-комп'ютери в Інтернеті? Чим це обумовлено?

5 Поясніть, як працює механізм запобігання "засмічення" мережі від застарілих датаграм.

6 Що таке маршрутизація?

7 Яким чином визначається маршрут проходження IP-датаграми в мережі Інтернет?

8 Дайте визначення прямої маршрутизації.

9 Дайте визначення непрямої маршрутизації.

10 Який тип маршрутизації використовується у випадку, коли комп'ютер з IP-адресою 192.168.10.2 взаємодіє з комп'ютером, IP-адреса якого 192.168.10.5 (маска мережі 255.255.255.0).

11 Що необхідно мати шлюзу, що поєднує, наприклад, чотири локальні мережі Ethernet?

12 Назвіть правила маршрутизації пакетів у мережі Інтернет.

13 Що називається максимальною одиницею, що передається?

14 У яких випадках відбувається фрагментація (дефрагментація) датаграм?

15 Які механізми використовує IP-протокол для формування своїх послуг.

16 Перелічіть недоліки в роботі IP-протоколу.

2.6 Транспортні протоколи. Протокол TCP

2.6.1 Загальні поняття протоколу TCP

Протокол керування передачею (Transmission Control Protocol або TCP) призначений для використання як надійного протоколу спілкування між хост-комп'ютерами в комунікаційних комп'ютерних мережах з комутацією пакетів, а також у системах, що поєднують такі мережі [3].

Протокол TCP забезпечує надійність комунікацій між парами процесів на хост-комп'ютерах, які включені у різні комп'ютерні комунікаційні мережі й об'єднані в єдину систему.

TCP займає в багаторівневій архітектурі протоколів нішу безпосередньо над IP-протоколом, що дозволяє протоколу TCP відправляти й одержувати сегменти інформації змінної довжини, укладені в оболонку IP-датаграм. Датаграма надає дані для адресації відправника й одержувача TCP-сегментів у різних мережах. IP-протокол також здійснює будь-яку фрагментацію й збирання сегментів TCP, які необхідні для здійснення передачі й доставки через безліч мереж і проміжних шлюзів. IP-протокол також обробляє інформацію про пріоритет, класифікацію безпеки, а також здійснює розмежування TCP сегментів. Отже, дана інформація може бути передана прямо через безліч мереж.

Протокол TCP припускає, що він може одержати простий, потенційно ненадійний сервіс для своїх датаграм із боку протоколів нижнього рівня. Отже, він:

- повинен забезпечити надійний сервіс для комунікацій між процесами в багатомережній системі;
- повинен бути загальним протоколом для комунікацій між хост-комп’ютерами в безлічі мереж.

Ще раз зазначимо, що протокол TCP взаємодіє, з одного боку, з користувачем (або прикладною програмою), а з іншого боку - із протоколом більш низького рівня, таким, як IP-протокол.

Головною ***метою протоколу TCP*** є *забезпечення надійного, безпечного сервісу* для логічних з'єднань між парами процесів. Щоб забезпечити такий сервіс, ґрунтуючись на менш надійних комунікаціях Інтернет, система повинна мати можливості для роботи в таких областях:

- базова передача даних;
- достовірність;
- керування потоком даних;
- поділ каналів;
- робота із з'єднаннями;
- пріоритет і безпека.

Розглянемо дії протоколу TCP у кожній із цих областей.

Базова передача даних

Протокол TCP здатний передавати безперервні потоки октетів між своїми клієнтами в обох напрямках, пакуючи якусь кількість октетів у сегменти для передачі через системи Інтернету. У загальному випадку протоколи TCP вирішують на свій розсуд, коли робити блокування й передачу даних.

Іноді користувачам необхідно переконатися в тому, що всі дані, передані ними протоколу TCP, вже відправлені. Для цього визначена функція ***проштовхування*** (push). Щоб переконатися

в тому, що дані, які були передані протоколу TCP, дійсно відправлені, користувач зазначає, що їх варто проштовхнути до одержувача. Проштовхування приводить до того, що програми протоколу TCP відразу здійснюють відправлення й відповідно одержання даних.

Достовірність

Протокол TCP повинен мати захист від руйнування даних, втрати, дублювання й порушення черговості одержання, що виникають під час проходження даних мережами Інтернету. Це досягається присвоєнням чергового номера кожному переданому октету, а також вимогою підтвердження від програми TCP, що приймає дані. Якщо підтвердження не отримане протягом контрольного інтервалу часу, то дані посилаються повторно.

З боку одержувача номери черги використовуються для відновлення черговості сегментів, які можуть бути отримані в неправильному порядку, а також для обмеження можливості появи дублікатів. Ушкодження фіксуються за допомогою додавання до кожного переданого сегмента контрольної суми, перевірки її при одержанні й наступній ліквідації дефектних сегментів.

Таким чином, протокол TCP захищає від помилок комунікаційної системи Інтернет.

Керування потоком

Протокол TCP дає можливість одержувачеві керувати кількістю даних, що посилаються йому відправником. Це досягається поверненням так званого "вікна". Вікно визначає кількість октетів, що відправник може послати до одержання подальших вказівок. Іншими словами, ***вікно*** (іноді кажуть ***розмір вікна) –*** *це кількість байтів, які можна передавати без підтвердження.*

Поділ каналів

Щоб дозволити на окремо взятому комп'ютері багатьом процесам одночасно використовувати комунікаційні можливості транспортного рівня, протокол TCP надає на кожному хост-комп’ютері набір адрес або портів. Разом з IP-адресами на комунікаційному рівні Інтернету вони утворюють ***сокет*** (socket – гніздо). Кожне з'єднання унікальним чином ідентифікується *парою сокетів*. Так, будь-який сокет може одночасно використовуватися в багатьох з'єднаннях.

Робота із з'єднаннями

Механізми керування потоком і забезпечення достовірності вимагають, щоб програми протоколу TCP ініціалізовували і підтримували певну інформацію про стан кожного потоку даних. Набір такої інформації, що включає сокети, номери черги, розміри вікон, називається ***з'єднанням***, або ***віртуальним каналом***. Кожне з'єднання унікальним чином ідентифікується *парою сокетів*. Описаний віртуальний канал є ***дуплексним:*** дані можуть одночасно передаватися в обох напрямках.

Якщо два процеси хочуть обмінюватися інформацією, відповідні програми протоколу TCP повинні спершу встановити з'єднання (на кожному боці ініціалізувати інформацію про статус), потім зробити передачу даних. Після завершення обміну інформацією з'єднання повинне бути закрите. Оскільки з'єднання повинні встановлюватися між ненадійними хост-комп'ютерами та через ненадійну комунікаційну систему Інтернет, використовується механізм підтвердження зв'язку із хронометрованими номерами черги, щоб уникнути помилкової ініціалізації з'єднань.

Пріоритет і безпека

Користувачі протоколу TCP можуть зажадати для свого з'єднання пріоритет і безпеку. Передбачені прийняті за замовчуванням характеристики з'єднань, коли такі параметри не потрібні.

2.6.2 Формат TCP-заголовка

Передавання TCP сегментів здійснюється у вигляді IP-датаграм. Заголовок TCP-сегмента (рис.10) в Інтернет-протоколі має кілька інформаційних полів, включаючи адреси хост-комп'ютерів, що відправляють і приймають дані. TCP-заголовок іде за IP-заголовком і доповнює його інформацією, яка є специфічною для TCP-протоколу. Такий розподіл допускає використання на рівні хост-комп'ютерів протоколів, інших, ніж TCP.

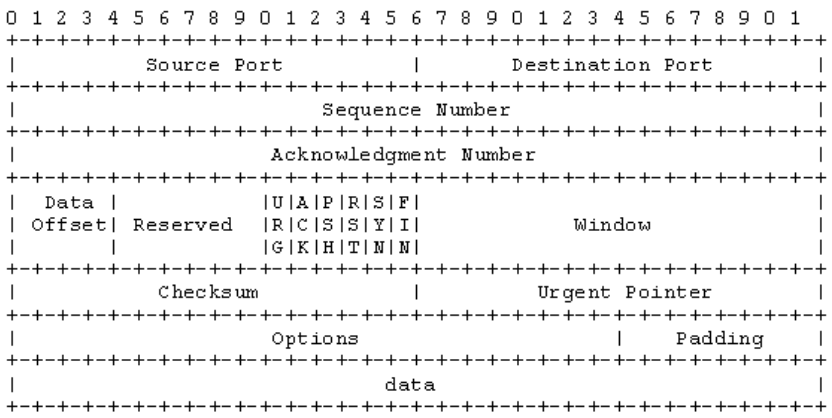

Рисунок 10 – Формат TCP-заголовка

Зазначимо, що кожна мітка (рис.10) вказує місце для відповідного біта.

Source Port – номер порту відправника –16 бітів.

Destination Port – номер порту одержувача – 16 бітів.

Sequence Number – номер черги для першого октету даних у даному сегменті (за винятком тих випадків, коли наявний прапорець синхронізації SYN). Якщо ж прапор SYN наявний, то номер черги є ініціалізованим (ISN), а номер першого октету даних – ISN+1.

Acknowledgment Number (номер підтвердження) 32 біти. Якщо встановлено контрольний біт ACK, то це поле містить наступний номер черги, який відправник даної датаграми бажає отримати у зворотному напрямку. Номери підтвердження посилають постійно, як тільки з'єднання буде встановлено.

Data Offset – поле, що свідчить про початок поля даних, тобто вказує кількість 32-бітних слів в TCP-заголовку– 4 біти. TCP-заголовок завжди закінчується на 32-бітній межі, навіть якщо він містить опції.

Reserved – резервне поле, яке повинне бути заповнене нулями – 6 бітів.

Control Bits – поле контрольних бітів – 6 бітів. Кожен біт цього поля несе певне смислове навантаження:

URG – поле термінового покажчика;

ACK – поле підтвердження;

PSH – функція проштовхування;

RST – поле, що показує на необхідність перезавантаження даного з'єднання;

SYN – поле, що відповідає за синхронізацію номерів черги;

FIN – поле, що свідчить про кінець потоку даних (є чи немає даних для передавання).

Window – поле, яке визначає кількість октетів даних, одержання яких чекає відправник дійсного сегмента.

Checksum – поле контрольної суми – 16 бітів.

Поле контрольної суми – це 16-бітне доповнення суми всіх 16-бітних слів заголовка й тексту. Якщо сегмент містить у заголовку й тексті непарну кількість октетів, що підлягають обліку в контрольній сумі, то останній октет буде доповнений нулями праворуч для того, щоб утворити для надання контрольній сумі 16-бітне слово. Утворений при такому вирівнюванні октет не передається разом із сегментом по мережі. Перед обчисленням контрольної суми поле цієї суми заповнюється нулями.

Контрольна сума, крім усього іншого, враховує 96 бітів псевдозаголовка (рис.11), що для внутрішнього використання ставиться перед TCP-заголовком. Цей псевдозаголовок містить адреси відправника та одержувача, протокол (PRTL) і довжину TCP-сегмента. Такий підхід забезпечує захист протоколу TCP від сегментів, які помилилися на маршруті. Цю інформацію обробляє IP-протокол.

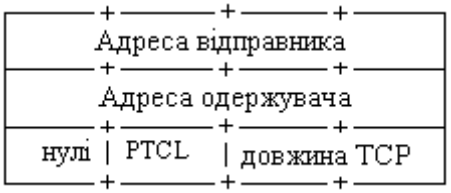

Рисунок 11 – Формат псевдозаголовка

Довжина TCP-сегмента – це довжина TCP-заголовка й поля даних, яка вимірюється в октетах. Це не є точною цифрою кількості переданих по мережі октетів, вона не враховує 12 октетів псевдозаголовка, але розрахунок цього параметра все-таки відбувається.

Urgent Pointer (терміновий покажчик) – поле, що повідомляє поточне значення термінового покажчика – 16 бітів.

Терміновий покажчик є додатною величиною – зсувом щодо номера черги даного сегмента. Її значення повідомляє номер черги для октету, що йде за терміновими даними. Це поле інтерпретується тільки в тому випадку, коли в сегменті виставлений контрольний біт URG.

Options – поле опцій. Опції можуть розміщуватися наприкінці TCP-заголовка, а їх довжина завжди кратна 8 бітам. Всі опції враховуються при розрахунку контрольної суми.

Padding – поле вирівнювання, довжина поля змінна.

Вирівнювання TCP-заголовка здійснюється для того, щоб переконатися в тому, що TCP-заголовок закінчується, а поле сегмента даних починається на 32-бітній межі. Вирівнювання виконується нулями.

2.6.3 Модель дії протоколу TCP

Комп'ютерні мережі можуть бути або локальними (наприклад, Ethernet), або глобальними мережами (наприклад, ARPANET), але в кожному разі вони ґрунтуються на технології комутації пакетів. Реальними агентами, що створюють і споживають повідомлення, які циркулюють у мережі, є ***процеси***. Протоколи різних рівнів у мережах на шлюзах і на хост-комп’ютерах

підтримують *систему комунікацій між процесами*, що забезпечує двонаправлений потік даних логічними з'єднаннями між портами процесів.

Процес пересилає дані, викликаючи програму протоколу TCP і передаючи їй як аргументи буфера з даними. Протокол TCP пакує дані із цих буферів у сегменти, а потім викликає модуль IP для передачі кожного сегмента на програму протоколу TCP, що є одержувачем. Одержувач, у свою чергу, розміщує дані із сегмента в буфери одержувача та сповіщає свого клієнта про надходження призначених йому даних.

Модель Інтернет-комунікацій полягає в тому, що з кожною програмою протоколу TCP зв'язаний модуль протоколу IP, що забезпечує їй інтерфейс із локальною мережею. Даний модуль поміщає сегменти TCP в IP-датаграми, а потім направляє їх на інший IP-модуль або ж проміжний шлюз. Для передачі датаграми по локальній мережі вона, у свою чергу, поміщається в пакет відповідного типу.

Комутатори пакетів можуть здійснювати подальше пакування, фрагментацію або інші операції для того, щоб у локальній мережі здійснити передачу пакетів за призначенням на IP-модуль.

На шлюзах між локальними мережами датаграма звільняється від пакета локальної мережі й досліджується для того, щоб визначити, по якій мережі вона повинна далі рухатися. Потім датаграма пакується в пакет, що відповідає обраній локальній мережі, і надсилається на наступний шлюз або ж прямо до кінцевого одержувача.

Шлюз має можливість розбивати датаграму на більш дрібні датаграми-фрагменти, якщо це необхідно для передачі по черговій локальній мережі. Щоб здійснити це, шлюз сам створює набір датаграм-фрагментів. Надалі фрагменти можуть бути знову розбиті наступними шлюзами на ще більш дрібні частини. Формат фрагмента датаграми спроектований так, щоб адресат, IP-модуль, зміг зібрати фрагменти знову у вихідні IP-датаграми.

IP-модуль, що є адресатом, виділяє TCP-сегмент із датаграми і потім передає його за призначенням на програму протоколу TCP.

Дана проста модель дії протоколу найчастіше замовчує безліч деталей. Однією з важливих характеристик є тип сервісу. Ця ознака дає вказівку шлюзу (або IP-модулю) про вибір параметрів сервісу, які повинні використовуватися при передачі датаграми в черговій локальній мережі. Пріоритет датаграми зазначається серед інформації про тип сервісу. Датаграми також можуть нести інформацію про безпеку для того, щоб дозволити хост-комп’ютерам і шлюзам, що діють у багаторівневій системі безпеки, давати на перевірку відповідні датаграми.

Потік даних, що посилає на TCP з'єднання, *приймається одержувачем надійно й у відповідності до черги.*

Передача здійснюється надійно завдяки використанню підтверджень і номерів черги. Концептуально кожному октету даних привласнюється номер черги. Номер черги для першого октету даних у сегменті передається разом із цим сегментом і називається ***номером черги для сегмента***. Сегменти також несуть номер підтвердження, що є номером для наступного очікуваного октету даних, переданого у зворотному напрямку. Коли протокол TCP передає сегмент із даними, він поміщає його копію в чергу повторної передачі й запускає таймер. Коли надходить підтвердження для цих даних, сегмент видаляється із черги. Якщо підтвердження не надходить до кінця терміну, то сегмент посилається повторно.

Підтвердження протоколу TCP не гарантує, що дані досягли кінцевого одержувача, а тільки те, що програма протоколу TCP на комп'ютері одержувача бере на себе відповідальність за це.

Висновки

Протокол TCP є протоколом транспортного рівня, головним завданням якого є *забезпечення надійного, безпечного сервісу* для логічних ланцюгів або з'єднань між парами процесів.

Коротко модель роботи протоколу TCP можна подати так:

1 Прикладний процес пересилає дані, викликаючи програму протоколу TCP і передаючи їй аргументи буфера з даними.
2 TCP-модуль відправника:
  2.1 Розбиває дані із цих буферів на TCP-сегменти.
  2.2 Викликає протокол IP, що, у свою чергу, перетворить TCP-сегменти в IP-датаграми, й передає кожну датаграму на програму протоколу TCP, що є одержувачем.
3 TCP-модуль одержувача:
  3.1 Аналізує прийняті дані.
  3.2 Збирає сегменти, перетворені датаграми, у єдине ціле.
  3.3 Розташовує прийняті й укомплектовані дані в буфери.
  3.4 Сповіщає свого клієнта про прибуття призначених йому даних.

Коли прикладний процес починає використовувати TCP, то модуль TCP на машині клієнта й модуль TCP на машині сервера починають спілкуватися. Ці два кінцевих TCP-модулі підтримують інформацію про стан з'єднання, який називають ***віртуальним каналом***. Цей віртуальний канал споживає ресурси обох TCP-модулів. Канал є ***дуплексним:*** дані можуть одночасно передаватися в обох напрямках. Один прикладний процес пише дані в TCP-порт, вони проходять по мережі, інший прикладний процес читає їх зі свого TCP-порту.

Протокол *TCP розбиває потік байтів на сегменти:* він не зберігає границь між записами. Тому *не існує залежності між числом і розміром записуваних повідомлень – із одного боку, й числом і розміром повідомлень, які зчитуються – із іншого.*

Протокол TCP вимагає, щоб всі відправлені дані були підтверджені стороною, що прийняла дані. Він використовує таймаути та повторні передачі для забезпечення надійної доставки. Відправникові дозволяється передавати деяку кількість даних, не чекаючи підтвердження прийому раніше відправлених даних. Таким чином, між відправленими й підтвердженими даними існує вікно вже відправлених, але ще непідтверджених даних. Кількість байтів, які можна передавати без підтвердження, називається ***розміром вікна***. Як правило, розмір вікна встановлюється в стартових файлах мережного програмного забезпечення. Тому що TCP-канал є дуплексним, підтвердження для даних, що рухаються в одному напрямку, можуть передаватися разом з даними, що рухаються у протилежному напрямку.

Контрольні запитання

1 Перелічити основні обов'язки протоколу TCP.

2 Хто є користувачами протоколів транспортного рівня?

3 У чому полягає функція проштовхування TCP-протоколу?

4 Яким чином у протоколі TCP досягається захист від дублювання, втрати й руйнування даних?

5 Що таке сокет?

6 Що таке віртуальний канал?

7 Поясніть поняття «дуплексний віртуальний канал»?

8 Що таке розмір вікна?

9 Чим ідентифікується TCP-з'єднання?

10 Перелічіть основні поля, їх призначення в заголовку TCP-сегмента.

11 Опишіть модель роботи TCP-протоколу.

2.7 Протокол датаграм користувача (UDP)

2.7.1 Загальні поняття UDP-протоколу

Протокол UDP (протокол датаграм користувача) забезпечує основний механізм, який використовується прикладними програмами для передачі датаграм іншим додаткам. UDP надає протокольні порти, які використовуються для розрізнення декількох процесів, що виконуються на одному комп'ютері. Крім даних, що пересилаються, кожне UDP-повідомлення містить номер порту-приймача й номер порту-відправника, уможливлюючи таким чином доставлення повідомлення відповідному реципієнтові, а для одержувача посилання відповіді відповідному відправникові.

UDP використовує IP-протокол для передачі повідомлення від однієї машини до іншої та забезпечує ту саму ***ненадійну*** доставку повідомлень, що й IP. Протокол UDP:

- не використовує підтвердження надходження повідомлень;
- не впорядковує повідомлення, які надходять отримувачу;
- не забезпечує зворотного зв'язку для керування швидкістю передачі інформації між машинами.

Тому UDP-повідомлення можуть бути загублені, розмножені або надходити в неустановленому порядку.

Крім того, пакети можуть надходити раніше**,** ніж одержувач зможе обробити їх. Взагалі можна сказати, що *UDP забезпечує ненадійну службу без установлення з'єднання й використовує IP-протокол для транспортування повідомлень між машинами. Він надає можливість називати кілька місць доставки на одному комп'ютері* [3].

Прикладні програми, що використають UDP, несуть повну відповідальність за проблеми надійності, включаючи втрату повідомлень, дублювання, затримку, невпорядкованість або втрату зв'язку. На жаль, програмісти часто ігнорують ці проблеми при розробленні програм. Крім того, оскільки програмісти тестують свої програми, використовуючи надійні високошвидкісні локальні мережі, тестування може не виявити можливі помилки. Таким чином, програми, що використовують UDP й успішно працюють в локальній мережі, будуть аварійно завершуватися в глобальних мережах TCP/IP.

2.7. 2 Формат UDP-сегмента

Кожен UDP-сегмент називається *датаграмою користувача*. Концептуально датаграма складається із двох частин, UDP-заголовка й області даних. Як показано на рисунку 12, заголовок складається із чотирьох 16-бітних полів, які визначають порт, з якого було послане повідомлення, порт, у який повідомлення надходить, а також довжину повідомлення й контрольну суму повідомлення.

Рисунок 12 – Формат UDP-сегмента

Поля ***порт відправника*** й ***порт одержувача*** містять 16-бітні номери портів, які використовуються для розподілу повідомлень, одержання яких очікують процеси. Поле *порт відправника* необов'язкове. Коли воно використовується, то позначає порт-джерело повідомлення, на який потрібно посилати відповіді, якщо ж не використовується, то повинно містити нуль.

Поле ***довжина*** містить число байтів датаграми, включаючи UDP-заголовок і дані.

Поле ***контрольна сума*** UDP необов'язкова, значення дорівнює нулю в полі *контрольної суми* означає, що сума не обчислюється. Розроблювачі вирішили зробити контрольну суму необов'язковою, щоб зменшити об’єм обчислень при використанні UDP у високонадійній локальній мережі. Помітимо, однак, що IP не обчислює контрольну суму поля даних в IP-датаграмах. Таким чином, контрольна сума UDP забезпечує єдину гарантію того, що цілісність даних збережена й ними можна користуватися.

Трапляються випадки, коли значення контрольної суми, що обчислювалась, також дорівнює нулю. У такому випадку нуль не є тим нулем, що маємо у випадку, коли контрольні суми не обчислюються. У цьому випадку арифметика має два подання нуля: всі біти містять або нуль, або одиницю. Коли контрольна сума, що обчислює, дорівнює нулю, UDP використовують подання з установкою всіх бітів в одиницю.

Слід зазначити, що для розрахунку контрольної суми в UDP-протоколі потрібно більше інформації, ніж представлено в UDP-повідомленні (в UDP-повідомленні відсутні IP-адреси відправника й одержувача, які входять у контрольну суму). Щоб уникнути такого нестикування, UDP-протокол приписує ***псевдозаголовок*** (рис. 13).

Метою використання псевдозаголовка є перевірка того, що UDP-датаграма досягла свого дійсного місця призначення. Ключем до розуміння псевдозаголовка є розуміння того, що *правильне місце призначення складається з конкретного комп'ютера й конкретного порту в комп'ютері*. UDP-заголовок сам по собі визначає тільки номер протокольного порту. Таким чином, щоб перевірити місце призначення, UDP на комп'ютері-джерелі обчислює контрольну суму, що враховує IP-адресу призначення, а так само саму UDP-датаграму. При одержанні датаграми в місці призначення UDP-модуль перевіряє контрольну суму, використовуючи IP-адресу призначення, отриману із заголовка IP-датаграми, що містила UDP-повідомлення. Якщо контрольні суми однакові, датаграма дійсно досягла потрібного хост-комп’ютера й потрібного порту в ньому.

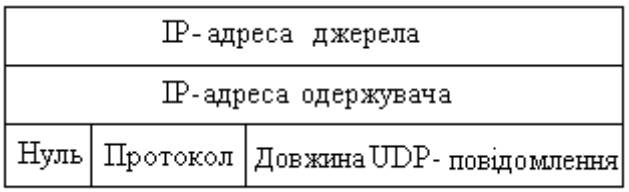

Рисунок 13 – Формат псевдозаголовка

Поля псевдозаголовка – ***IP-адреса джерела*** й ***IP-адреса одержувача*** – містять IP-адреси джерела й одержувача, які будуть використані при посиланні повідомлення. Поле ***протокол*** містить код типу протоколу IP (наприклад, 17 для UDP). Поле ***довжина UDP*** містить довжину UDP-датаграми (не включаючи псевдозаголовка). Для перевірки контрольної суми одержувач повинен спочатку витягти ці поля з IP-заголовка, помістити їх у відповідні поля псевдозаголовка й знову обчислити контрольну суму.

2.7. 3 Інкапсуляція пакетів

У багаторівневій сім'ї протоколів TCP/IP протокол UDP належить транспортному рівню й займає нішу між міжмережними (IP, ICMP) і прикладними протоколами. Прикладні програми звертаються до UDP, що використовує IP-протокол для пересилання й одержання датаграм (рис. 14)

Рисунок 14 – Взаємозв'язок рівнів при відправленні (прийомі) повідомлень

Знаходження UDP-протоколу над IP означає, що повні UDP-повідомлення, що включають UDP-заголовок і дані, інкапсулюються в IP-датаграмах при передаванні мережею (рис.15).

Для протоколів IP та UDP інкапсуляція означає, що UDP приписує заголовок на початку даних, які передав користувач, і передає все це нижньому рівню протоколів (протоколу IP). Міжмережний рівень – рівень протоколу IP – приписує свій заголовок до UDP-сегменту. І, нарешті, рівень взаємодії з мережею вставляє датаграми в ***кадри*** перед передачею їх від однієї машини до іншої.

Формат кадру залежить від використовуваної мережної технології. Звичайно мережні кадри включають додатковий заголовок. Після передачі пакета на хост-одержувач пакет приймається нижчим рівнем мережного програмного забезпечення, а потім починає передаватися наверх через наступні рівні. Кожен рівень видаляє один заголовок перед передачею повідомлення наступному рівню, і коли верхній рівень передає дані процесу-приймачу, всі заголовки вже вилучені. Таким чином, зовнішній заголовок відповідає протоколу нижчого рівня, у той час як внутрішній заголовок відповідає протоколу верхнього рівня.

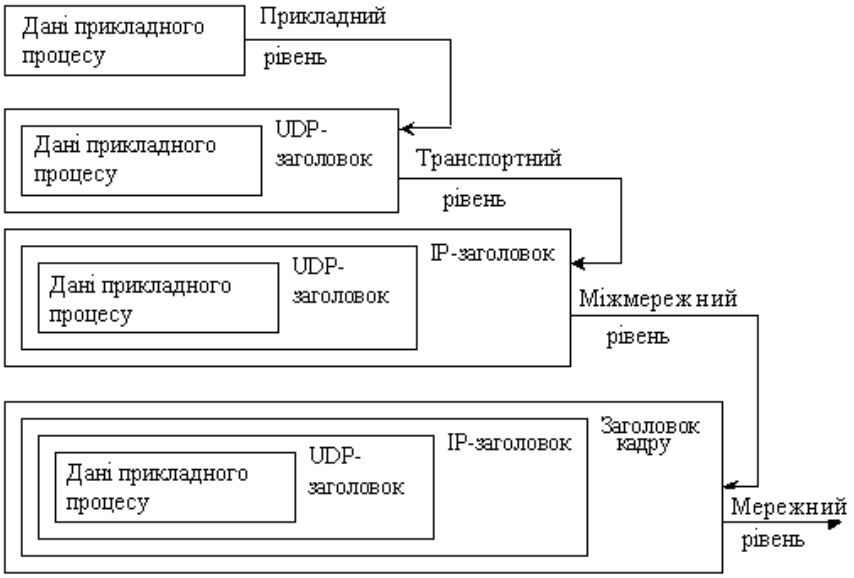

Рисунок 15 – Інкапсуляція UDP-датаграми

2.7.4 Поділ на рівні й обчислення контрольної суми UDP

Аналізуючи все вищезазначене, неважко помітити протиріччя між правилом поділу на рівні й обчисленням контрольної суми. Нагадаємо, що контрольна сума UDP включає псевдозаголовок, що містить поля для IP-адрес відправника й одержувача. Можна довести, що IP-адреса одержувача повинна бути відомою користувачеві при відправленні UDP-датаграми і що користувач повинен передати її на рівень UDP. Тому рівень UDP може одержати IP-адресу, не взаємодіючи з рівнем IP. Однак IP-адреса джерела залежить від обраного шляху для датаграми, тому що IP-адреса джерела визначає мережний інтерфейс, через який буде передаватися датаграма. Таким чином, UDP не може знати IP-адресу джерела без контакту з рівнем протоколу IP.

Припустимо, що UDP просить рівень IP визначити IP-адресу відправника та, можливо, одержувача, використовує їх для формування псевдозаголовка, обчислює контрольну суму, відкидає псевдозаголовок і передає UDP-датаграму IP-модулю для пересилання мережею. Альтернативний варіант, що дає більшу ефективність, полягає в інкапсуляції UDP-датаграми рівнем UDP в IP-датаграму, заповненні полів IP-адрес відправника й одержувача в IP-заголовку, обчисленні контрольної суми UDP і передачі IP-датаграми міжмережному рівню (рівень IP-протоколу), що заповнить поля, що залишилися, IP-заголовка.

Чи порушить явна взаємодія між UDP й IP головну передумову про те, що поділ на рівні відбиває поділ функцій? Так. UDP-протокол тісно пов'язаний з IP- протоколом. У цьому випадку відхилення від принципів повного поділу зроблено з практичних причин. Розроблювачі змушені порушити принцип поділу на рівні, тому що неможливо повністю ідентифікувати процес-одержувача, не назвавши комп'ютер одержувача.

2.7.5 Мультиплексування, демультиплексування та порти UDP

Програмний модуль UDP являє приклад мультиплексування й демультиплексування. Він приймає UDP-датаграми від багатьох прикладних програм і посилає їх до модуля протоколу IP для передачі, а також він приймає UDP-датаграми, які надходять від IP-модуля, та передає їх відповідним прикладним програмам.

Концептуально всі процеси мультиплексування й демультиплексування між UDP та прикладними програмами здійснюються за допомогою механізму портів. На практиці кожна прикладна програма повинна домовитися з операційною системою про одержання протокольного порту та пов'язаного з ним номера перед посиланням UDP-датаграми. Коли порт виділений, прикладна програма посилає будь-яку датаграму через порт, номер якого зазначений у полі ***порт відправника UDP***. У ході обробки вхідних даних UDP приймає датаграми, які надходять від IP, та демультиплексує їх по портах призначення (рис. 16).

Рисунок 16 – Демультиплексування датаграм по портах

Порт UDP найлегше представити у вигляді черги. У більшості реалізацій, коли прикладна програма домовляється з операційною системою про використання даного порту, операційна система створює внутрішню чергу, що зберігає повідомлення, що надходять. Часто додаток може вказати або змінити розміри черги. Коли UDP одержує датаграму, він перевіряє, чи немає порту призначення з таким номером серед використовуваних портів. Якщо ні, він посилає ICMP-повідомлення про помилку “порт недоступний” і знищує датаграму. Якщо є, UDP додає нову датаграму в чергу порту, де прикладна програма може її одержати. Якщо черга порту вже переповнена, то UDP-модуль знищує нову датаграму.

2.7.6 Зарезервовані й вільні номери портів UDP

Коли комп'ютер А хоче одержати файл від комп'ютера В, він повинен знати, який порт у комп'ютері В використовується програмою передачі файлу. Тому два комп'ютери повинні домовитися про номери портів, перш ніж вони зможуть взаємодіяти.

Існують два фундаментальних підходи до призначення портів:

1 ***Універсальне призначення***. Використовує централізоване керування призначенням. Усі домовляються дозволити центральному органу призначати номери всім необхідним портам і потім опублікувати список призначень. Тоді всі програми створюються відповідно до цього списку. Цей підхід іноді називають “універсальним призначенням”, а такі призначення портів називають “широковідомими призначеннями портів”.

2 ***Динамічне призначення***. При такому підході номери портів нікому не відомі, а мережне забезпечення самостійно призначає порт, коли це необхідно прикладному процесу. Щоб довідатися про поточне призначення портів на іншому комп'ютері, потрібно зробити запит, у якому ставитья приблизно таке питання: “Як мені викликати службу передачі файлів?” Комп'ютер-одержувач відповість, який порт необхідно використати.

Розроблювачі TCP/IP прийняли ***змішаний підхід***, у якому призначається група портів апріорно, але більшість може вільно використовуватися для будь-яких цілей прикладними програмами в локальній мережі. Апріорно призначені номери портів починаються з маленьких

значень і потім збільшуються, а порти з більшими значеннями використовуються для динамічного призначення.

2.7.7 Команди контролю з'єднань та маршрутизації

PING

Для перевірки з'єднання з віддаленим комп'ютером або комп'ютерами використовують команду ***ping***. Ця команда доступна тільки після встановлення підтримки протоколу TCP/IP.

Команда ***ping*** виконує такі дії:

- перевірку стану з'єднання з іншим комп'ютером або комп'ютерами відправленням echo-пакетів ICMP і аналіз отриманих відповідних пакетів;
- очікування до однієї секунди для кожного переданого пакета;
- виведення числа відправлених і прийнятих пакетів;
- кожен отриманий пакет порівнюється з відповідним відправленим.

Формат команди:

ping [-t] [-a] [-n лічильник] [-l довжина] [-f] [-i ttl] [-v тип]
[-r лічильник] [-s число] [[-j список_хостів] |
[-k список_хостів] [-w інтервал] список_призначень

Параметри

-t – повторює запити до віддаленого комп'ютера до того часу, поки програма не буде зупинена.

-a – дозволяє використовувати ім'я комп'ютера як адресу.

-n лічильник – задається число echo-пакетів. За замовчуванням – 4.

-l довжина – відправляються echo-пакети, що містять порцію даних заданої довжини. За замовчуванням – 32 байти, максимум – 65527 байтів.

-f – відправляє пакети із прапорцем заборони фрагментації. Пакети не будуть розриватися при проходженні шлюзів на своєму маршруті.

-i ttl – встановлює час життя пакетів TTL (Time To Live).

-v тип – встановлює тип служби (Type Of Service) пакетів.

-r лічильник – записує маршрут відправлених і повернених пакетів у поле записів маршруту Record Route. Параметр *лічильник* задає число комп'ютерів в інтервалі від 1 до 9.

-s число – задає число ретрансляцій на маршруті, де буде робитися оцінка часу.

-j список_хостів – направляє пакети по маршруту, що задається параметром *список_хостів*. Хост-комп'ютери в списку можуть бути розділені проміжними шлюзами (вільна маршрутизація). Максимальна кількість хостів, що дозволена протоколом IP, дорівнює 9.

-k список_хостів – направляє пакети по маршруту, що задається параметром *список_хостів*. Хост-комп'ютери в списку не можуть бути розділені проміжними шлюзами (обмежена маршрутизація). Максимальна кількість, що дозволена протоколом IP, дорівнює 9.

-w інтервал – вказує проміжок часу очікування (в мілісекундах).

список_призначень – вказує список комп'ютерів, яким направляються запити.

За замовчуванням передаються чотири echo-пакети, що містять 32 байти даних (періодична послідовність великих букв).

Команда ***ping*** дозволяє перевірити як ім'я, так й IP-адресу комп'ютера. Якщо IP-адреса пройшла перевірку, а ім'я не пройшло, відбулася помилка дозволу імені на адресу. У цьому випадку необхідно впевнитися, що ім'я запитуваного комп'ютера міститься в локальному файлі Hosts або в базі даних служби DNS.

Наведений нижче приклад містить результати роботи команди ping:

C:\>ping ds.internic.net
Обмін пакетами з ds.internic.net [192.20.239.132] по 32 байти:
Відповідь від 192.20.239.132: число байтів=32 час=101мс TTL=243
Відповідь від 192.20.239.132: число байтів=32 час=100мс TTL=243
Відповідь від 192.20.239.132: число байтів=32 час=120мс TTL=243
Відповідь від 192.20.239.132: число байтів=32 час=120мс TTL=243

Traceroute (Tracert)

Відслідкування маршруту, яким проходить пакет (або виявлення шлюзу, що відкидає пакети), може бути непростою справою.

Для цього використовують команду ***tracert,*** що визначає маршрут, який проходять пакети до точки призначення в мережі.

Ця програма намагається відстежити маршрут, яким IP-пакет швидше за все буде йти до деякої машини в мережі, запускаючи пробні UDP-пакети з коротким ttl (time to live -і часом життя) і, потім, слухаючи ICMP відповіді “час перевищений” від шлюзу.

Формат команди

tracert [-d] [-h лічильник] [-j host-list] [-w timeout] target_name

Ключі:

-d – при перевірках маршруту використовувати тільки IP-адресацію без доменних імен.

-h лічильник – називається максимальна кількість переходів для визначення шляху проходження по маршруту. За замовчуванням дорівнює 30.

-j host-list – список хостів передбачуваного маршруту.

-w timeout – максимальний час очікування відповіді.

Увага. *Ці програми призначені для тестування, вимірів і менеджменту в мережі. Вони збільшують навантаження мережі, тому нерозумно використовувати їх занадто часто або занадто довго.*

Висновки

Протокол UDP забезпечує основний механізм, який використовується прикладними програмами для передачі датаграм іншим додаткам. UDP використовує IP-протокол для передачі повідомлення від однієї машини до іншої і забезпечує *ненадійну* доставку повідомлень.

Протокол UDP не використовує підтвердження надходження повідомлень, не впорядковує повідомлення, що надходять, не забезпечує зворотного зв'язку для керування швидкістю передачі інформації між машинами.

Протокол UDP у своїй роботі використовує *порти* (цілі позитивні числа, які ідентифікують прикладний процес), що дозволяє розрізняти кілька процесів в одному комп'ютері.

Існує два фундаментальних підходи до призначення портів: *універсальне* й *динамічне* призначення. Деякі номери портів, називані широковідомими, закріплені постійно й відомі всім користувачам Інтернету (наприклад, порт 69 зарезервований для використання протоколом передачі файлів TFTP). Інші порти призначені для довільного використання прикладними програмами.

У схемі рівнів протоколів UDP належить транспортному рівню, що розміщений вище міжмережного рівня й нижче прикладного рівня. Загалом транспортний протокол незалежний від міжмережного рівня, але на практиці вони тісно взаємодіють. Контрольна сума UDP включає IP-адреси відправника й одержувача, що означає, що модуль протоколу UDP повинен взаємодіяти з модулем протоколу IP для знаходження потрібних адрес перед посиланням датаграми.

Контрольні запитання

1 Які завдання вирішує UDP-протокол?

2 У чому полягає відмінність TCP- та UDP- протоколів?

3 Назвіть поля UDP-заголовка і їх призначення.

4 Для чого введене поле контрольної суми в UDP-заголовок?

5 Яким чином відбувається обчислення контрольних сум?

6 У чому полягає необхідність використання псевдозаголовка UDP-сегмента?

7 Поясніть суть інкапсуляції UDP-датаграми.

8 Яким чином відбувається призначення портів?

9 Які дії виконує команда PING?

10 Призначення команди TRACERT.

2.8 Протоколи прикладного рівня

Які ж прикладні програми доступні в мережах з TCP/IP? Їх загальна кількість велика й продовжує постійно збільшуватися. Деякі додатки існують із самого початку розвитку Інтернет, наприклад, TELNET й FTP. Інші, наприклад, X-Window, SNMP з'явилися недавно.

Протоколи прикладного рівня орієнтовані на конкретні прикладні задачі. Вони визначають як процедури з організації взаємодії певного типу між прикладними процесами, так і форму подання інформації при такій взаємодії. У цьому розділі ми коротко опишемо деякі із протоколів прикладного рівня. Вичерпну інформацію про прикладні протоколи та сервіси Інтернету можна знайти в літературі [3,4].

Протокол TELNET

Протокол TELNET дозволяє машині, що обслуговує, розглядати всі віддалені термінали як стандартні "мережні віртуальні термінали" рядкового типу, що працюють в ASCII- кодах, а також забезпечують можливість узгодження більш складних функцій (наприклад, локальний або віддалений echo-контроль, сторінковий режим, висота й ширина екрана й т. ін.) TELNET працює на базі протоколу TCP. На прикладному рівні над TELNET перебуває або програма підтримки реального термінала (на боці користувача), або прикладний процес в машині обслуговування, до якої здійснюється доступ з термінала [4].

Робота з TELNET схожа на набір телефонного номера. Користувач набирає на клавіатурі, наприклад,

telnet astra

і одержує на екрані запрошення на вхід у машину з ім'ям *astra*.

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

Протокол FTP

Протокол FTP (File Transfer Protocol – протокол передачі файлів) розповсюджений так, як TELNET. Він є одним з найстарших протоколів сім'ї TCP/IP. Також, як TELNET, він користується транспортними послугами TCP. Існує безліч реалізацій для різних операційних систем, які добре взаємодіють між собою. Користувач FTP може викликати кілька команд, які дозволяють йому подивитися каталог вилученої машини, перейти з одного каталогу в інший, а також скопіювати один або декілька файлів [3,4].

Протокол SMTP

Протокол SMTP (Simple Mail Transfer Protocol – простий протокол передачі пошти) підтримує передачу повідомлень (електронної пошти) між довільними вузлами мережі Інтернет, механізми проміжного зберігання пошти й механізми підвищення надійності доставки. Протокол SMTP допускає використання різних транспортних служб. Він може працювати навіть у мережах, що не використовують протоколи сім'ї 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 розміщується поштова служба конкретних обчислювальних систем [3,4].

Протокол SNMP

Протокол SNMP (Simple Network Management Protocol– простий протокол керування мережею) працює на базі UDP і призначений для використання мережними керуючими станціями. Він дозволяє керуючим станціям збирати інформацію про стан справ у мережі Інтернет. Протокол визначає формат даних, їх обробка та інтерпретація залишаються на розсуд керуючих станцій або менеджера мережі [3].

Поштовий протокол POP

Post Office Protocol (POP) – протокол доставки пошти користувача з поштової скриньки *поштового POP-сервера*. Багато концепцій, принципи й поняття протоколу POP мають вигляд і функціонують подібно до SMTP. Команди POP практично ідентичні командам SMTP, відрізняючись у деяких деталях.

У протоколі POP3 обговорені три стадії процесу одержання пошти: *авторизація*, *транзакція* й *відновлення*. Після того як сервер і клієнт POP3 установили з'єднання, починається стадія *авторизації*. На стадії авторизації клієнт ідентифікує себе для сервера. Якщо авторизація пройшла успішно, сервер відкриває поштову скриньку клієнта й починається стадія *транзакції*. У ній клієнт або запитує у сервера інформацію (наприклад, список поштових повідомлень), або просить його зробити певну дію (наприклад, видати поштове повідомлення). Нарешті, на стадії відновлення сеанс зв'язку закінчується [3,4].

Поштовий протокол IMAP

Істотною відмінністю протоколу IMAP від протоколу POP є те, що IMAP підтримує роботу із системою каталогів (або папок) повідомлень. IMAP дозволяє керувати віддаленими каталогами (папками) повідомлень так само, як якби вони розміщувалися на локальному комп'ютері. *IMAP дозволяє клієнтові створювати, видаляти й перейменовувати поштові скриньки, перевіряти наявність нових повідомлень і видаляти старі*. Завдяки тому, що IMAP підтримує механізм унікальної ідентифікації кожного повідомлення в поштовій папці клієнта, він *дозволяє читати з поштової скриньки тільки повідомлення, що задовольняють певні умови, або частини таких повідомлень, змінювати атрибути повідомлень, переміщати окремі повідомлення та ін.* [3,4].

Висновки

Комплект протоколів прикладного рівня містить у собі велику кількість різноманітних протоколів, які широко використовуються, у тому числі керування мережею, передача файлів, розподіл послуги використання файлами, емуляція терміналів, електронна пошта й т.п.

Контрольні запитання

1 Назвіть відомі вам протоколи прикладного рівня. Для яких цілей вони використовуються?

2 Який протокол забезпечує можливість передавати файли через Інтернет?

3 Назвіть відомі вам поштові протоколи.

4 У чому полягає різниця між протоколами IMAP й POP?

5 Назвіть сервіси Інтернет, які дозволяють працювати в консольному режимі.

2.9 Word Wide Web

World Wide Web (скорочено *WWW*', або Web, або W3, у перекладі на українську мову означає *Всесвітнє павутиння),* спочатку замислювалася для обміну дослідницькою інформацією, але тепер стала частиною повсякденного життя великої кількості людей. Кожний може підключитися до неї, щоб попрацювати над темою наукових досліджень або подивитися, що роблять конкуренти.

WWW – одна із найпопулярніших служб Інтернету, оскільки доступ туди може одержати кожний, хто має *комп'ютер* і *модем або інший мережний інтерфейс.*

Історія WWW

Проект WWW виник на початку 1989 року в Європейському центрі ядерних досліджень в Женеві (European Laboratory for Particle Physics (CERN) in Geneva, Switzerland http://user.web.cern.ch/user/cern.html). Основне призначення проекту – надати користувачам-непрофесіоналам "on-line" доступ до інформаційних ресурсів. Результатом проекту World Wide Web (WWW або W3) є надання користувачам мережних комп'ютерів досить простого доступу до найрізноманітнішої інформації.

Європейський центр ядерних досліджень (CERN) створив прототип першого WWW-сервера, що згодом докорінно змінив вигляд Інтернет.

Перший такий сервер був організований в CERN'е (листопад 1990 р., Тім Бернерс-Лі) [5], там же з метою розвитку й підтримки стандартів WWW технологій створений The World Wide Web Consortium, або W3C [6]. WWW-сервер є інтегруючим сервером з підтримки WEB-технологій Інтернету [7]. Останню інформацію про стан WWW проекту можна одержати за адресою W3C Project [7].

Пізніше до проекту приєдналися й багато інших організацій. Великий внесок у розвиток WWW-технологій зробив Національний центр суперкомп'ютерних додатків [8].

Але, незважаючи на те, що концепція функціонування сервера WWW була вже створена, практичне застосування цього сервісу було неможливе аж до лютого 1993 р., коли в суперкомп'ютерному центрі Іллінойського університету була створена *альфа* (перша, придатна для публікації) версія NCSA Mosaic – найпершої програми перегляду документів на серверах WWW. Саме із цього моменту визначився досить бурхливий характер зростання Інтернету.

Історію розвитку World Wide Web якнайкраще ілюструє роль програм перегляду документів на серверах WWW або, як зараз часто пишуть, *браузерів* (від англ. – *browser*).

На різних етапах цього шляху серед фаворитів були різні браузери. До 1995 року, не витримавши конкуренції, практично припинився випуск нових версій Mosaic, і світовий ринок браузерів захопила фірма Netscape Communications зі своїм продуктом Netscape Navigator.

Такий місткий ринок, як Інтернет, не міг виявитися поза сферою діяльності корпорації Microsoft. Корпорація до початку 1996 року розробила розгорнуту стратегію інтеграції своїх програмних продуктів й операційних систем з Інтернет. Одним із ключових моментів цієї стратегії є розвиток браузера *MS Internet Explorer*, що на поточний час є найбільш використовуваним браузером в Інтернет-населення (до 95%).

Як уже говорилося, випуск програми Mosaic for Windows став одним із ключових моментів успіху WWW, оскільки звичайний персональний комп'ютер із установленою програмою Mosaic розгорнув двері WWW широким рядам рядових користувачів – великим корпораціям, маленьким фірмам, урядовим закладам, політикам, громадським організаціям, історичним суспільствам і приватним особам.

Багато хто з працюючих у WWW просто розміщують там інформацію, використовуючи WWW як середовище для її передачі, однак більш часто можна натрапити на комерційні організації, що рекламують себе, свою продукцію й свої послуги.

У світовому WWW існують мільйони домашніх сторінок, присвячених тисячам різних тем. Інтернет – воістину безмежний океан інформації й людського самовираження.

Суть WWW

WWW – це глобальна система гіпертексту. ***Гіпертекст*** *– текст із вставленими в нього словами (командами) розмітки, що посилаються на інші місця цього тексту, інші документи, картинки й т.д.*

До 1989 року гіпертекст являв собою нову, багатообіцяючу технологію, що мала відносно велику кількість реалізацій, з одного боку, а з іншого – робилися спроби побудувати формальні моделі гіпертекстових систем, які мали скоріше описовий характер і були навіяні успіхом реляційного підходу опису даних. Ідея Т. Бернерс-Лі полягала в тому, щоб застосувати гіпертекстову модель до інформаційних ресурсів, розподілених у Мережі, і зробити це максимально простим способом. Він заклав три наріжних камені системи із чотирьох, існуючих нині, та розробив:

- мову гіпертекстової розмітки документів *HTML (HyperText Markup Language);*
- універсальний спосіб адресації ресурсів у мережі *URL (Universal Resource Locator);*
- протокол обміну гіпертекстовою інформацією HTTP *(HyperText Transfer Protocol)*.

Пізніше команда NCSA додала до цих трьох компонент четверту:

- універсальний інтерфейс шлюзів *CGI (Common Gateway Interface).*

2.9.1 Мова гіпертекстової розмітки документів *HTML*

Перший наріжний камінь системи WWW – це HTML. Ідея HTML – приклад надзвичайно вдалого вирішення проблеми побудови гіпертекстової системи за допомогою спеціального засобу керування відображенням. На розроблення мови гіпертекстової розмітки істотний вплив зробили два фактори: дослідження в області інтерфейсів гіпертекстових систем і бажання забезпечити простий і швидкий спосіб створення гіпертекстової бази даних, розподіленої в мережі.

У 1989 році активно обговорювалася проблема інтерфейсу гіпертекстових систем, тобто способів відображення гіпертекстової інформації й навігації в гіпертекстовій мережі. Значення гіпертекстової технології порівнювали зі значенням друкарства. Стверджувалося, що аркуш паперу та комп'ютерні засоби відображення й відтворення серйозно відрізняються один від одного, і тому форма подання інформації теж повинна відрізнятися. Найбільш ефективною формою організації гіпертексту були визнані *контекстні гіпертекстові посилання*, а, крім того, був визнаний розподіл на *посилання, асоційовані з усім документом у цілому й окремими його частинами*.

Звичайно гіпертекстові системи мають спеціальні програмні засоби побудови гіпертекстових зв'язків. Самі

гіпертекстові посилання зберігаються в спеціальних форматах або навіть становлять спеціальні файли. Такий підхід зручний для локальної системи, але не для розподіленої на безлічі різних комп'ютерних платформ. В HTML гіпертекстові посилання вбудовані в тіло документа й зберігаються як його частина. Часто в системах застосовують спеціальні формати зберігання даних для підвищення ефективності доступу. В WWW документи – це звичайні *ASCII-файли,* які можна підготувати в будь-якому текстовому редакторі. Таким чином, проблема створення гіпертекстової бази даних була вирішена надзвичайно просто.

Як база для розроблення мови гіпертекстової розмітки був обраний SGML. Дотримуючись академічних традицій, Бернерс-Лі описав HTML у термінах SGML (як описують мову програмування в термінах форми Бекуса-Наура). Природно, що в HTML були реалізовані всі розмітки, пов'язані з виділенням параграфів, шрифтів, стилів і т.п. Важливим компонентом мови став опис вбудованих й асоційованих гіпертекстових посилань, вбудованої графіки й забезпечення можливості пошуку за ключовими словами.

Таким способом мова надавала автору матеріалів, які були розміщені на сторінці, широкі можливості відносно того, як ці матеріали показати користувачеві. Але, на жаль, до 1996-1997 років він мав досить примітивні можливості керування поданням інформації й зовнішнім виглядом сторінки. Це було наслідком великої кількості нестандартизованих програм перегляду (браузерів) і багатоплатформеності Інтернет (UNDO, MaxOS, Windows). Кожен браузер тоді (і навіть зараз) відображав інформацію дещо по-своєму.

В основу синтаксису мови HTML був покладений стандарт ISO 8879:1986 «Information processing. Text and office systems. Standard Generalised Markup Language (SGML)». Щоправда, існує велике розходження між стандартом офіційним і стандартом фактичним.

З моменту розроблення першої версії мови (HTML 1.0) пройшло вже багато років. За цей час відбувся досить серйозний розвиток мови. Майже вдвічі збільшилася кількість елементів розмітки, оформлення документів усе більше наближається до оформлення якісних друкованих видань, розвиваються засоби опису нетекстових інформаційних ресурсів і способи взаємодії із прикладним програмним забезпеченням. Удосконалюється механізм розроблення типових стилів. Фактично, у цей час HTML розвивається убік створення стандартної мови розроблення інтерфейсів як локальних, так і розподілених систем.

2.9.2 Універсальний спосіб адресації ресурсів у мережі (URL)

Другим наріжним каменем WWW стала універсальна форма адресації інформаційних ресурсів. Universal Resource Identification (URI). *Універсальний ідентифікатор ресурсів (URI)* для користувача з'являється у вигляді *місцезнаходження (URL) або імені (URN) web-документа і являє собою досить стрункy систему, що враховує досвід адресації й ідентифікації E-mail, Telnet, FTP* та ін. Але реально із усього, що описано в URI, для організації WWW потрібно тільки Universal Resource Locator (URL). Без наявності цієї специфікації вся міць HTML виявилася б марною. URL використовується в гіпертекстових посиланнях і забезпечує доступ до розподілених ресурсів мережі.

2.9.3 Протокол обміну гіпертекстовою інформацією HTTP

Третім у списку наріжних каменів є протокол обміну даними в WWW – Hyper Text Transfer Protocol. Даний протокол призначений для обміну гіпертекстовими документами й ураховує специфіку такого обміну. Так, у процесі взаємодії клієнт може одержати нову адресу ресурсу мережі (relocation), запросити вбудовану графіку, прийняти й передати параметри і т. д. Керування в HTTP реалізоване у вигляді ASCII-команд. Реально розроблювач гіпертекстової бази даних зіштовхується з елементами протоколу тільки при використанні зовнішніх

розрахункових програм або при доступі до зовнішніх відносно WWW інформаційних ресурсів, наприклад баз даних.

З 1990 року проектом World Wide Web використовується HTTP-протокол прикладного рівня, що забезпечує необхідну швидкість передачі даних, яка вимагається для розподілених інформаційних систем гіпермедіа.

Протокол HTTP надає відкриті методи, які можуть бути використані для вказівки цілей запиту. Вони побудовані на дисципліні посилань, де для вказівки ресурсу, до якого повинен бути застосований даний метод, використовується Універсальний ідентифікатор ресурсів (Universal Resource Identifier – URI) у вигляді місцезнаходження (URL) або імені (URN).

HTTP використовується також для комунікацій між різними користувацькими браузерами й шлюзами, що дають гіпермедіа-доступ до існуючих Інтернет-протоколів, таким як SMTP, NNTP, FTP, Gopher й WAIS.

Більш докладну інформацію про протокол HTTP можна знайти в літературі [3,4].

2.9.4 Універсальний інтерфейс шлюзів CGI. Типи web-документів.

Остання складова технології WWW – це вже плід роботи групи NCSA–специфікація Common Gateway Interface.

CGI була спеціально розроблена для розширення можливостей WWW за рахунок підключення всілякого зовнішнього програмного забезпечення. Такий підхід логічно продовжував принцип публічності та простоти розроблення й нарощування можливостей WWW. Якщо команда CERN запропонувала простий і швидкий спосіб розроблення баз даних, то NCSA розвинула цей принцип на розроблення програмних засобів. Треба помітити, що в загальнодоступній бібліотеці CERN були модулі, що дозволяють програмістам підключати свої програми до сервера HTTP, але це вимагало використання цієї бібліотеки.

Запропонований й описаний в CGI спосіб підключення не вимагав додаткових бібліотек і буквально приголомшував своєю простотою. Сервер взаємодіяв із програмами через стандартні потоки вводу/виводу, що спрощує програмування. При реалізації CGI надзвичайно важливе місце зайняли методи доступу, описані в HTTP. І хоча реально використовуються тільки два з них (GET й POST), досвід розвитку HTML показує, що співтовариство WWW чекає розвитку й CGI у міру ускладнення завдань, у яких буде використовуватися WWW-технологія.

Раніше документи на Web були ***статичними*** у тому розумінні, що користувач міг декілька разів завантажувати ту саму сторінку, але її зміст від цього не змінювався. Однак ми живемо в такому світі, де нова інформація користується підвищеною увагою. Один з методів залучення уваги користувачів – це введення елемента інтерактивності. З появою CGI забезпечити цей елемент стало значно простіше.

З’являються CGI-сценарії. CGI-сценарії становлять сховану від очей користувача частину інтерактивної взаємодії. Вони приймають інформацію, послану серверу через Web, й обробляють її, запитуючи бази даних, виконуючи розпорядження або просто реєструючи отримані відомості. Все це відбувається «за кадром», але саме тут і виконується реальна робота. Потім результати передаються назад. Про таку HTML-сторінку говорять, що вона *динамічно генерується*, а сам документ називають ***динамічним***.

Висновки

WWW – це абревіатура від "World Wide Web" (*"Всесвітня павутина"*). Офіційне визначення World Wide Web звучить як світова віртуальна файлова система –

"широкомасштабне гіпермедіасередовище, орієнтоване на надання універсального доступу до документів".

Основне достоїнство WWW полягає у використанні ***гіпертекстової технології***. ***Гіпертекст*** – документ, що має посилання на інші документи. Перевага гіпертексту в тому, що для одержання потрібної інформації користувачу Інтернет зовсім не потрібно переглядати весь документ, досить натиснути курсором на ключове слово (***гіперпосилання***).

WWW працює за принципом ***клієнт-сервер***: існує безліч серверів, які на запит клієнта повертають йому *гіпермедійний документ* – документ, що складається із частин з різноманітним поданням інформації, у якому кожен елемент може бути посиланням на інший документ або його частину.

Для запису документів у гіпертексті використовується спеціальна, але дуже проста мова HTML (Hypertext Markup Language), що дозволяє управляти шрифтами, відступами, вставляти кольорові ілюстрації, підтримує виведення звуку й анімації. До стандарту мови також входить підтримка математичних формул.

Виділяють два основних *типи web-документів*:

- ***статичні*** (сторінки, які вже існують у форматі HTML на момент надходження запиту);
- ***динамічні*** (сайти, які формуються спеціальними із криптами під час надходження запиту).

При створенні динамічних сторінок користувач задає параметри, що передаються серверу. Сервер одержує запит (з параметрами) на створення web-документа й запускає спеціальну програму – ***скрипт***, що формує й видає необхідний для користувача набір інформації. Цей процес називається *CGI* (Common Gateway Interface), а самі програми йменуються *CGI-скриптами*.

World Wide Web для своєї роботи використовує HTTP-протокол, що забезпечує необхідну швидкість передачі даних.

Контрольні запитання

1 Що таке WWW?

2 Який протокол лежить в основі роботи WWW-сервісу?

3 Що таке гіпертекст?

4 Що таке CGI?

5 У чому полягає різниця між статичними й динамічними web-документами?

3 HTML-мова опису WWW-сторінок

3.1 Типи редакторів HTML документів

Існує два типи редакторів HTML:

- візуальні редактори;
- редактори, що працюють з кодом.

Візуальні редактори реалізують принцип “що бачу, те й маю”. Для створення сторінки в редакторах такого типу достатньо розташувати об’єкти (текст, графіку та ін.) у такому порядку, який має бути на web-сторінці, і зберегти файл у HTML-форматі.

Наведемо кілька таких редакторів:

- FrontPage (http://www.microsoft.com/frontpage);
- FrontPad (у поставці MS Internet Explorer 4);
- Netscape Composer, убудований в Communicator та Netscape Gold;
- Hot Metal (http://softquad.com/products/hotmetal/);
- HomePage Publisher (http://ourworld.compuserve.com/homepages/clerin/);
- DreamWeaver (http://www.macromedia.com/software/dreamweaver/).

Редактори, які працюють прямо з кодом, використовують професіонали, що розробляють супер- проекти в певному стилі.

Наведемо кілька таких редакторів:

- Notepad notepad.exe;
- HTML Pad (http://www.book.ru/snk/);
- 1st Page (http://www.evrsoft.com);
- Hot Dog (http://www.sausage.com/hotdog);
- HTML-Kit (http://www.chami.com/html-kit/);
- HTMLed32 (http://www.ist.ca);
- HomeSite (http://www.allaire.com);
- Bred (http://yurok.da.ru);
- CoffeeCup (http://www.coffeecup.com);
- SiteAid (http://www.siteaid.com);
- FAR manager (http://www.rarsoft.com).

Зазначимо, що до FAR потрібно встановити додаткові плагіни *colorer* й *htmledtior*.

3.2 Команди й атрибути команд в HTML

Команди

Команди або теги (TAG) мови HTML, як правило, мають таку структуру:

<COMMAND> – початок команди

поле дії команди

</COMMAND> – кінець команди.

Хоча існує деяка кількість команд, що складаються тільки з тегів «*початок команди» (у* цих командах теги «*кінець команди»* ігноруються браузером), стандарт мови HTML вимагає обов'язкової наявності тегу «*кінець команди»*.

Зауваження

1 Мова HTML не чутлива до регістра. Команди можуть набиратися як великими, так і малими літерами, тобто команда ***<title>*** еквівалентна команді ***<TITLE>*** або ***<TiTle>***.
2 Не всі команди-теги підтримуються всіма web-браузерами. Якщо браузер не підтримує команду, він її просто ігнорує.

Атрибути команд

Команди можуть мати параметри, які називаються ***атрибутами***. Атрибути модифікують виконання команди або перевизначають її стиль. Атрибути ставляться відразу після команди й мають такий формат:

***АТРИБУТ*=“значення атрибута”**

Зауваження. Якщо *значення атрибута* складається з одного слова, то ставити лапки не обов'язково, але рекомендується. Якщо ж *значення атрибута* містить пробіли або небуквені символи, то лапки обов'язкові. Крім того, лапки потрібно писати, щоб додати своїй сторінці сумісність із XHTML.

3.3 Структура HTML-документа

Кожен документ, що відповідає вимогам HTML, **повинен** починатися з декларації <!DOCTYPE>. Це необхідно для того, щоб відрізнити документ, складений, наприклад, за специфікацією HTML 4.0, від документів, написаних для інших версій мови HTML.

Наприклад:

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">,

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2*FINAL*//EN">,

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 TRANSITIONAL//EN">.

Специфікація HTML не конкретизує об'єкти зберігання. Як наслідок, відсутнє обмеження, щоб декларація для типу документа перебувала в тому самому елементі зберігання, що й сам документ (тобто перебувала в тому самому файлі). Web-сайт може автоматично доповнювати надавані HTML-файли такою декларацією, якщо відомо, що всі наявні на сайті HTML-файли відповідають специфікації HTML 3.2.

Структура HTML документа має такий вигляд:

<***HTML***> – початок документа (обов'язкова команда в HTML 3.0 і вище)
<***HEAD***> – заголовок документа (може бути відсутнім)
…тіло заголовка...
</***HEAD***> – кінець заголовка документа
<***BODY***>
…тіло документа...
</***BODY***>
</***HTML***> – кінець документа.

3.4 Команди заголовка HTML-документа

Заголовок HTML-документа виділяється тегами

<HEAD> тіло заголовка **</HEAD>**.

До складу заголовка можуть входити такі елементи: TITLE, STYLE, SCRIPT, ISINDEX, BASE, META, LINK.

Елемент **TITLE**

Він визначає заголовок документа, що відображається окремо (у заголовку вікна браузера) і використовується, насамперед, для ідентифікації документа (наприклад, при пошуку). Заголовок повинен описувати мету документа й містити не більше 5-6 слів. Для задання заголовка існує команда

<TITLE> Заголовок ***</TITLE>***.

Наприклад,

<TITLE>Головна сторінка СумДУ </TITLE>.

Відповідно до специфікації HTML 4.0 кожен документ **зобов'язаний** мати один елемент TITLE у полі HEAD.

Усередині TITLE не можна використовувати елементи розмітки.

Елемент **BASE**

Елемент ***BASE*** визначає для даного документа базову URL-адресу, що потім буде використовуватися при перевизначенні відносних адрес URL з використанням правил, що задаються відповідною специфікацією URL.

Наприклад, у випадку розмітки

```
<BASE href="http://www.sumdu.edu.ua/">
  ...
<IMG SRC="icons/logo.gif">
```

відповідне зображення буде співвіднесено із джерелом

http://www.sumdu.edu.ua/icons/logo.gif.

За відсутності елемента ***BASE*** для перетворення відносних адрес в абсолютні повинен використовуватися URL самого документа. Помітимо, що не обов'язково це буде та сама адреса URL, яка використовується для виклику документа, оскільки його базовий URL може бути перевизначений заголовком HTTP, що супроводжує в мережі розглянутий документ.

Елемент **ISINDEX**

Використовується при простому пошуку за ключовим словом (наприклад при створенні простих форм) і вказує на те, що браузер повинен виділити окреме текстове поле для ручного введення даних у рядок запиту.

Немає ніяких обмежень на кількість символів, які можна було б увести таким способом.

При використанні елемента **ISINDEX** самостійно (без атрибутів) браузер інтерпретує його в такий спосіб (рис. 17).

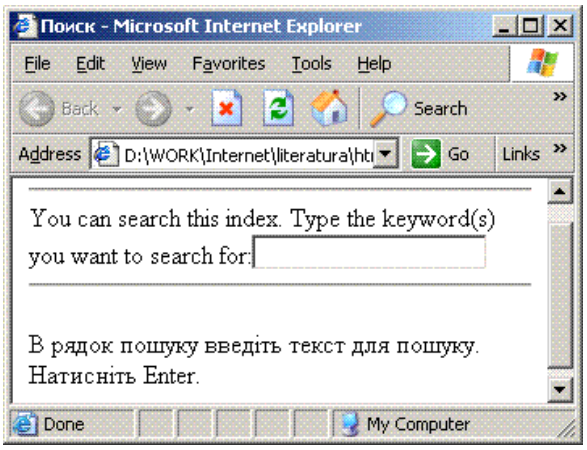

Рисунок 17 – використання елемента ISINDEX за замовчуванням

Щоб замінити повідомлення, яке записується браузером, можна скористатися атрибутом **PROMPT** команди ISINDEX, наприклад

<ISINDEX PROMPT="Рядок пошуку "> (рис. 18).

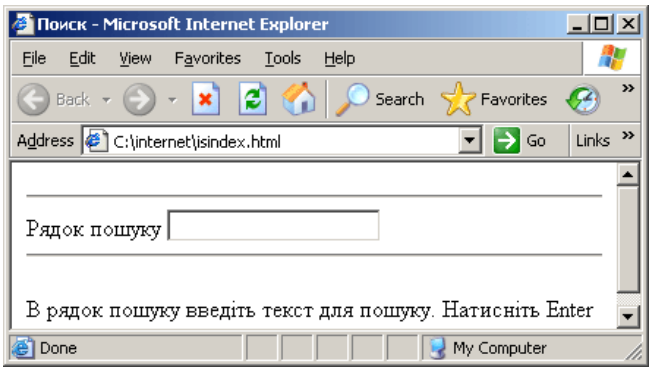

Рисунок 18 – використання атрибуту PROMPT команди ISINDEX

Семантика елементів ISINDEX на сьогодні гарно розроблена тільки для тих випадків, коли базова адреса для поточного документа – URL, побудований на протоколі HTTP. У такому випадку, як правило, як тільки користувач натискає клавішу Enter, серверу посилає рядок запиту, складовою частиною якої є базова адреса URL поточного документа.

Наприклад, якщо в рядок запиту уведена фраза "*кафедра інформатики*" і документ має базовий URL:

http://www.sumdu.edu.ua/,

то запит, що генерується, має вигляд

http://www.sumdu.edu.ua/?Кафедра+інформатики.

Зазначимо, що при цьому символи пробілу відображаються в символи "+" і що в адресі URL використовуються стандартні символи-обмежувачі.

Приклад.
У вікні браузера наступний код буде виглядати так, як показано на рисунку 18.

```
<HTML>
        <HEAD>
                <TITLE>Поиск</TITLE>
                <BASE="http://www.sumdu.edu.ua/">
<ISINDEX prompt = "Рядок пошуку ">
        </HEAD>
        <BODY>
                В рядок пошуку введіть текст для пошуку
                Натисніть Enter
```

ЕЛЕМЕНТ **STYLE**

Елемент **STYLE**

Дані елементів розмітки <**STYLE**> …</**STYLE**> залишають у документі місце під запис стилів (наприклад при використанні каскадних таблиць стилів CSS). Браузери не виводять на екран уміст цього елемента. Наприклад,

```
<STYLE TYPE=”text/css”>
        a:link{text decoration:none}
        a:visited {text decoration:none}
</STYLE>.
```

Елемент **SCRIPT**

Дані команди <**SCRIPT**>…</**SCRIPT**> залишають у документі місце під запис програм, які будуть виконуватися на комп'ютерах кінцевих користувачів або на web-серверах.

Елемент **META**

Використовується для надання метаінформації. ***Метаінформація*** *– парні конструкції типу "назва/значення", що описують властивості даного документа і їх значення.*

Тег <***META***> використовується з парними атрибутами ***NAME*** , що вказує назву певної властивості, та ***CONTENT,*** що надає значення відповідній властивості.

Наприклад,

```
<META NAME= “Author” CONTENT= “Т. Усатенко”>
<META NAME= “Description” CONTENT= “Лекція. HTML”>
<META NAME=“Keywords” CONTENT= “тег, команда, атрибут, посилання, web-документ ”>.
```

Іноді елемент META використовують для зазначення методів протоколу HTTP, наприклад,

```
<META HTTP-EQUIV="Refresh" content="5; URL= http://www.sumdu.edu.ua/">,
```

де

цифра – час очікування;
URL – адреса переходу.

При такому використанні елемента META відбувається перехід на сторінку із зазначеним URL через 5 с після повного завантаження у вікно браузера.

Елемент **LINK**

Використовується для створення зв'язків з іншими документами та надає документу незалежний від середовища метод створення взаємозв'язків з іншими документами й ресурсами середовища. Елемент ***LINK*** був складовою частиною мови HTML, починаючи із самих перших днів, однак дотепер лише деякі браузери використовують його переваги (більша ж частина продовжує ігнорувати елементи LINK).

Елементи <**LINK>** можуть бути використані для:

- створення в документі спеціальних навігаційних кнопок;
- керування процесом відображення набору HTML-файлів у друковані документи;
- прив'язки таблиць стилів (css) і скриптів;
- надання альтернативних форм для даного документа.

Розглянемо кілька рекомендованих типів взаємозв'язку, що задають атрибутами :

HREF – задає URL-адресу, що вказує на асоційований ресурс.

TITLE – визначає заголовок асоційованого ресурсу, що зазначається для довідки.

REL – вказує прямий зв'язок, тип відомий також як "тип зв'язку". Вводить певне взаємовідношення між поточним документом і ресурсом, на який вказує атрибут **HREF**.

REV – визначає зворотні відношення. Наприклад, якщо є прив'язка документа А до документа В, виражена параметром REV=*відношення*, то в документі В те ж саме відношення фіксується за допомогою атрибута REL=*відношення*.

Типи зв'язків у HTML поки що нестандартизовані, але існують наступні рекомендовані типи взаємозв'язків:

REL=TOP – зв'язок, що вказує на вершину в якійсь ієрархічній структурі, наприклад на першу, або титульну сторінку в деякому наборі HTML-документів.

REL=CONTENTS – зв'язок, що вказує на якийсь файл, де наводиться зміст до даного документа.

REL=INDEX – зв'язок, що вказує на інший документ, який можна використати з метою індексного пошуку в поточному документі.

REL=GLOSSARY – зв'язок, що вказує на деякий документ, де міститься глосарій термінів, що належать до поточного документа.

REL=COPYRIGHT – зв'язок, що посилається на текст, у якому зазначені авторські права на даний документ.

REL=NEXT – зв'язок, що вказує на наступний документ у певному заздалегідь визначеному маршруті перегляду. Наприклад, він може використовуватися для автоматичного завантаження браузером наступної сторінки.

REL=PREVIOUS – даний зв'язок посилається на попередній документ у певному заздалегідь визначеному маршруті перегляду.

REL=HELP – зв'язок, що вказує на документ-допомогу, наприклад, це може бути текст, що дає більш розгорнутий опис і пропонує посилання на інші документи з цієї теми. Призначення цього зв'язку – надання допомоги тим читачам, хто втратив свій шлях у Web.

REL=SEARCH – дане посилання веде до пошукової сторінки, що контролює певний набір сторінок, зв'язаних загальною темою.

Приклади елементів LINK:

<LINK REL=Contents HREF=zmist.html>,

<LINK REL=Previous HREF=doc31.html>,

<LINK REL=Next HREF= doc33.html>.

Пример #2 HTML-документ

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Draft//EN">
<HTML>
<HEAD>
<TITLE>Назва сторінки</TITLE>
... інші елементи заголовка
        <META HTTP-EQUIV="Content-Type"
                CONTENT ="text/html; charset=windows-1251">
        <META NAME="Description"
                CONTENT="опис сторінки>
        <META NAME="Keywords"
                CONTENT =" ключові слова, через кому">
        < LINK HREF="зовнішній файл таблиць стилів"
                REL="stylesheet type=text/css">
        <SCRIPT LANQUAGE="JavaScript"
```

3.5 Команди тіла HTML документа

Тіло HTML-документа укладене в теги:

<BODY> … **</BODY>**.

Усе, що перебуває всередині команди **<BODY>**, це те, що користувач бачить у вікні браузера. Ключовими атрибутами тегу є:

BGCOLOR="color" – встановлює колір, який визначений значенням **color**, тіла документа.

Наприклад, BGCOLOR= "siver".

BACKGROUND= "URL**"** – задає URL-адресу, звідки буде братися зображення тіла поточного документа (якщо використовується картинка). Наприклад,

BACKGROUND= "http://podruga.net/image/blood.jpg".

TEXT="color**"** – встановлює кольори тексту, виведеного на екран. Як правило, використовується в сполученні з тегами BGCOLOR або BACKGROUND.

LINK="color" – визначає кольори посилань на інші web-сторінки, які не були відвідані.

VLINK="color" – визначає кольори відвіданих посилань.

ALINK="color" – установлює кольори *активних* посилань у момент, коли користувач обере посилання (клацне по ньому мишкою).

TOPMARGIN (**LEFTMARGIN**) – установлює верхній (лівий) відступ. Підтримується тільки MS Internet Explorer):

TOPMARGIN | LEFTMARGIN=n(%),

де *n* – значення відступу в пікселах (відсотках від розміру вікна).

У мові HTML кольори задаються за схемою RGB числами, записаними у 16-й системі числення, або одним з 16 загальноприйнятих назв для кольорів.

Назви кольорів і відповідні значення RGB

Black = "#000000"	Green = "#008000"
Silver = "#C0C0C0"	Lime = "#00FF00"
Gray = "#808080"	Olive = "#808000"
White = "#FFFFFF"	Yellow = "#FFFF00"
Maroon = "#800000"	Navy = "#000080"

Red = "#FF0000" Blue = "#0000FF"
Purple = "#800080" Teal = "#008080"
Fuchsia = "#FF00FF" Aqua = "#00FFFF"

Приклади використання атрибутів:

<**BODY BACKGROUND**="bgpicture.gif" **TEXT**=#000000 **LINK**=#ff6600 **VLINK**=#330099>,

<**BODY BGCOLOR**= “aqua” **TEXT**= “black” **LINK**= “navy” **VLINK**= “yellow” **TOPMARGIN**=10% **LEFTMARGIN**=100>.

3.5.1 Робота з текстом

Більшість елементів, які можуть з'являтися в тілі документа, підпадають під одну із двох категорій:

- елементи на рівні блоків, що ініціюють перехід до наступного параграфа (заголовки (від H1 до H6), параграфи (P), елементи списків (LI), горизонтальні лінійки (HR) і таблиці (TABLE));
- елементи на рівні тексту: елементи виділення (EM, I, B й FONT); елементи зв'язку для гіпертексту (A), вкладені об'єкти (IMG й APPLET) і кінці рядків (BR).

Зазначимо, що елементи на рівні блоків, як правило, є контейнерами для матеріалів й елементів на рівні тексту, а також для інших елементів рівня блоків (винятком є заголовки й елементи адресації). У той самий час усередині елементів текстового рівня можуть утримуватися лише елементи того ж рівня.

Розглянемо деякі елементи блокового рівня**.**

Заголовки розділів HTML-документа

Тіло HTML – документа може складатися з декількох розділів. HTML має шість рівнів заголовків розділів, що мають номери з 1 по 6 (заголовок першого рівня є заголовком вищого рівня). Порівняно з нормальним текстом заголовки виділяються шрифтом, розміром і товщиною букв і за замовчуванням центруються по лівому краю вікна.

Синтаксис заголовків:

<H*n*> *Текст заголовка***</H*n* >**,

де *n* – число від 1 до 6, що визначає рівень заголовка

З тегами заголовків, як правило, вживають атрибут **ALIGN= “**LEFT|RIGHT|CENTER**”**.

Наприклад,

<H1 ALIGN=LEFT>Лекція по HTML</H1>

Деякі браузери не сприймають атрибут ALIGN=CENTER для тегів <Hn>. Тому іноді доцільно вживати сполучення тегів <Hn> й <CENTER>

Наприклад,

<CENTER><H1>Лекція по HTML</H1></CENTER>

Абзаци (параграфи)

На відміну від документів у більшості текстових процесорів переривання рядків у HTML-файлах не істотні. Обрив рядка може відбуватися в будь-якому пункті вихідного файла, при перегляді це переривання буде ігноровано браузером.

Для виділення абзацу існує спеціальний тег **<P>** , що вказує на початок абзацу. У діючих версіях HTML тег "*кінець абзацу*" </P> не існує. Але його вживання в розмітці документа не є помилкою, оскільки всі незнайомі символи браузер просто ігнорує.

Тег **<P>** може використовуватися з атрибутом **ALIGN**=, що може набувати таких значень:

JUSTIFY – вирівнювання абзацу по ширині вікна;

RIGHT – вирівнювання абзацу по правому краю вікна;

LEFT – вирівнювання абзацу по лівому краю вікна.

Наприклад,

```
<HTML>
<HEAD>
        <TITLE>Приклад HTML-тексту</TITLE>
</HEAD>
<BODY>
        <H1>Розділ 1</H1>
        <H2>Параграф 1</H2>
        <P ALIGN=CENTER>Ласкаво просимо в HTML!
        <P> Тут ми розповімо, <P> як треба і як не треба писати гіпертексти.
        <H2>Параграф 2</H2>
</BODY>
</HTML>
```

Цей текст у вікні браузера має такий вигляд, як показано на рисунку 19.

Елемент **PRE**

Визначає фрагмент відформатованого тексту. Для даного елемента необхідно вказувати початковий і кінцевий теги. Всередині такого елемента текст друкується шрифтом фіксованої ширини й при цьому зберігається розмітка оригіналу (пробіли й символи кінця рядків).

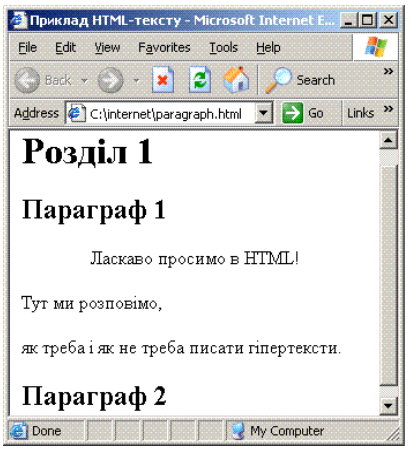

Рисунок 19

Наприклад:

```
<HTML><HEAD>
<TITLE>Приклад HTML-тексту, з командою PRE</TITLE>
</HEAD>
<BODY>
        <P FLIGN=JUSTIFY>Виконаємо виведення таблиць
без використання відповідних тегів
        <PRE>
        Прізвище      Ім'я      Рік народження
        Мілютенко     Олена     1985
        Григоришин    Іван      1984
        Симоненко     Юрій      1980
        </PRE>
```

У вікні браузера вище наведений HTML-код матиме такий вигляд (рис. 20).

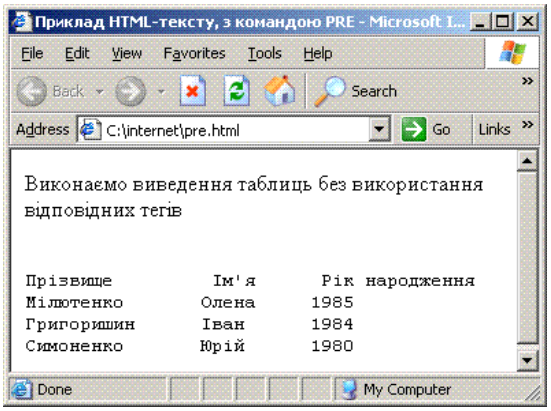

Рисунок 20 – Приклад HTML-тексту з командою PRE

Елемент **DIV**

Забезпечує розподіл документа на окремі блоки. Для даного елемента необхідно вказувати як початковий, так і кінцевий теги. Елемент DIV, як правило, використовується з атрибутом ALIGN = LEFT|CENTER|RIGHT, що вказує спосіб вирівнювання тексту всередині тих блокових елементів, які його містять.

Елемент **CENTER**

Задає вирівнювання тексту по центру вікна. Для цього елемента необхідно вказувати початковий і кінцевий теги. Більш радикальне вирішення проблеми вирівнювання текстів див. в DIV.

Елемент **BLOCKQUOTE**

Забезпечує розмітку цитат. Необхідно вказувати як початковий, так і кінцевий теги. Даний елемент використається при розмітці цитат, текст яких при цьому, як правило, друкується зі зсунутими краями.

Елемент **HR**

Забезпечує відображення горизонтальної лінії. Не є контейнером, так що ***використовувати кінцевий тег не можна***. Може мати атрибути:

ALIGN = LEFT | RIGHT | CENTER;
NOSHADE – затемнення лінії;
SIZE =n – розмір (висота) в *n* пікселах;
WIDTH =m(%) – довжина в m пікселах (відсотків від розміру вікна).

Наприклад, *<HR ALIGN=CENTER SIZE=5 WIDTH=50%>*.

Елемент **ADDRESS**

При внесенні в документ тегу <ADDRESS>, вказуються як початковий, так і кінцевий тег. Тіло тега повинно містити необхідну інформацію для встановлення зв'язку з автором (ім'я, e-mail, icq та ін.). Усередині даного елемента можуть міститись теги текстового рівня.

3.2 Списки

HTML підтримує декілька видів списків: ненумеровані, нумеровані, каскадні й списки визначень.

Для задання ***ненумерованого*** списку вживають сукупність тегів:

<UL>

<LI>список пунктів

</UL>

Наприклад, нижче наведений текст у вікні браузера буде мати такий вигляд (рис.21).

```
<b>Приклад #1</b><br> HTML підтримує:
<UL>
        <LI> ненумеровані списки;
        <LI> нумеровані списки;
        <LI> списки визначень.
</UL>
```

Для тегу <UL> ненумерованого списку додатково може бути використано атрибут ***TYPE***=disc|circle|square***,*** що задає мітку елемента списку диск, коло чи квадрат відповідно (див. приклад #4).

Нумерований список ідентичний ненумерованому списку, тільки замість < UL> вживають < OL>. Наприклад (рис.21),

```
<b>Приклад #2</b><br> HTML підтримує:
<OL>
<LI> неупорядковані списки;
<LI> нумеровані списки;
<LI> каскадні списки.
</OL>
```

За замовчуванням нумерація дається арабськими цифрами, починаючи з одиниці. Використовуючи атрибути команди ***<OL>***, можна змінити стиль оформлення списку. Атрибут ***TYPE*****=** **“**стиль” визначає стиль нумерації (див. приклад #5):

A – використати більші букви (латинські);
a – використати маленькі букви;
I – використати більші римські цифри;
i – використати маленькі римські цифри;
1 – використати арабські цифри.

Наприклад, текст:

```
“…<b>Приклад #3</b><BR>HTML підтримує.
        <OL type= a>
                <LI> неупорядковані списки;
                <LI> нумеровані списки;
                <LI> каскадні списки.
        </OL> ...”
```

у вікні браузера буде мати такий вигляд, як показано на рис.21.

Атрибут ***START=n*** визначає початкове значення списку (n – десяткове число). Приклад використання атрибута див. приклад #5.

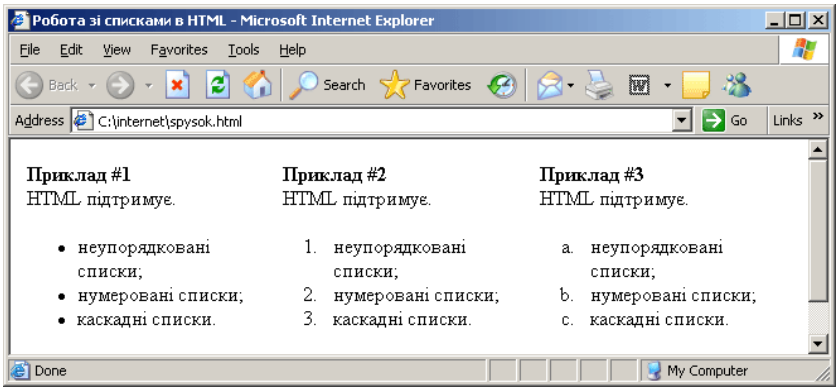

Рисунок 21 – Неупорядковані та нумеровані списки

Вкладені списки

Списки можуть бути довільно вкладені, хоча практично досить обмежитися трьома рівнями вкладених списків.

Розглянемо приклади каскадних списків, у вікні браузера вони будуть мати вигляд, показаний на рисунку 22.

```
<B>Приклад #4:
Каскадні невпорядковані списки, наприклад </B><BR>
<UL>
    <LI> Великі міста Росії:
    <UL type=circle>
        <LI> Москва;
        <LI> Новосибірськ;
        <LI> Санкт-Петербург.
    </UL>
   <LI> Великі міста України:
    <UL type=square>
<LI> Київ;
<LI> Харків;
<LI> Донецьк.
</UL>
</UL>

<B>Приклад #5:</B><BR> Каскадні нумеровані списки, наприклад
<OL>
    <LI> Великі міста Росії:
    <OL type=a>
<LI> Москва;
<LI> Новосибірськ;
<LI> Санкт-Петербург.
    </OL>
    <LI> Великі міста України:
    <OL type=a start=4>
        <LI> Київ;
        <LI> Харків;
        <LI> Донецьк.
```

```
    </OL>
</OL>
```

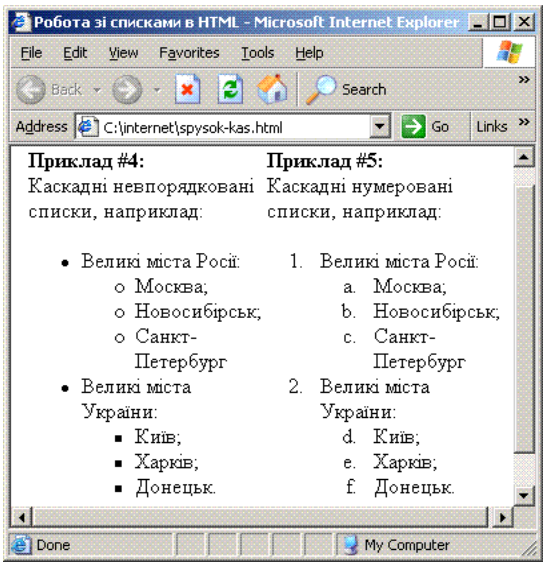

Рисунок 22 – Каскадні списки

Списки визначень задаються в такий спосіб:

```
<DL>
        <DD> назва терміна
        <DT> визначення терміна
</DL>
```

Наприклад (рис. 23),

```
Приклад #6
<b>Приклад #6</b><br>Списки визначень.
<DL>
        <DT> Паралельні прямі
        <DD>Дві прямі називаються
            паралельними, якщо вони на
            площині не перетинаються.
        <DT> Перпендикулярні прямі
        <DD>Дві прямі називаються
            перпендикулярними, якщо вони
            перетинаються під прямим кутом.
</DL>
```

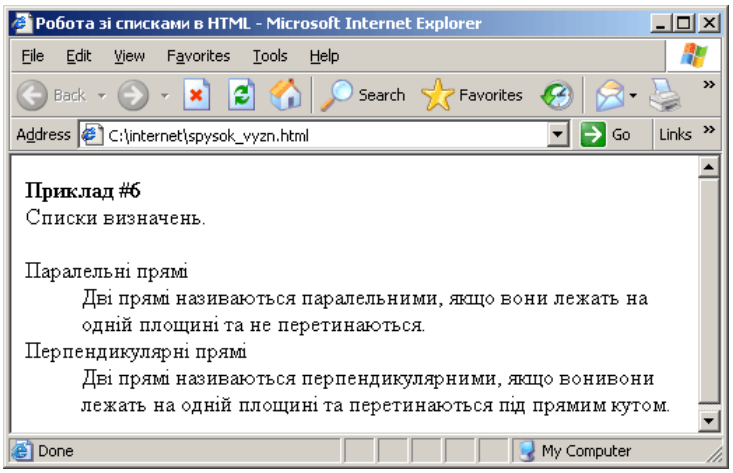

Рисунок 23 – Список визначень

3.5.3 Робота з таблицями в HTML

Елемент **TABLE** забезпечує створення таблиць. Потрібно вказувати початковий і кінцевий теги. Кожна таблиця починається з необов'язкового тегу <**CAPTION**>, за яким ідуть один або кілька елементів **<TR>**, що формують рядки таблиці. Кожен рядок має одну або кілька комірок, що задаються елементами <**TH**> або < **TD**>.

Елемент <**TABLE**> може мати атрибути:

HEIGHT = n (%) – висота таблиці в пікселах (відсотках щодо вікна);

WIDTH = m (%) – ширина таблиці в пікселах (відсотках щодо вікна);

BORDER = n – розмір ліній сітки (*n* – натуральне число);

CELLSPACING = n – відстань між комірками (*n*–натуральне число);

CELLPADDING = n – відстань вмісту (тексту, графіки тощо) комірки від її границь (*n* – натуральне число);

BGCOLOR = color – установлює кольори фону таблиці;

BORDERCOLOR=color – установлює кольори рамки таблиці.

Кожна таблиця складається з рядків і стовпців. Рядок задається парою тегів **<TR>** … **</TR>**. Стовпець –**<TD>**…**</TD>**.

Наприклад, проілюструємо дії команд <TABLE>, <TR>, <TD> (рис.24).

```
<TABLE BORDER=1>
<TR>
<TD> Рядок1. Стовпець1</TD>
<TD>Рядок1. Стовпець2. </TD>
</TR>
<TR>
<TD>Рядок2. Стовпець1</TD>
<TD>Рядок2. Стовпець2</TD>
</TR>
</TABLE>
```

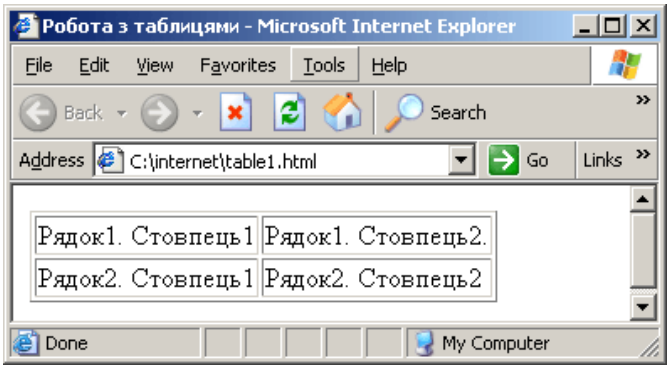

Рисунок 24 – Проста таблиця

Проілюструємо дію атрибутів <CELLSPACING>=10 й <CELLPADDING>=0 команди <BODY> (рис. 25).

```
<TABLE BORDER=1 CELLSPACING=10 CELLPADDING = 0>
<TR>
        <TD>Рядок 1. Стовпець 1</TD>
        <TD>Рядок 1. Стовпець 2 </TD>
</TR>
<TR>
        <TD>Рядок 2. Стовпець 1</TD>
        <TD>Рядок 2. Стовпець 2</TD>
</TR>
</TABLE>
```

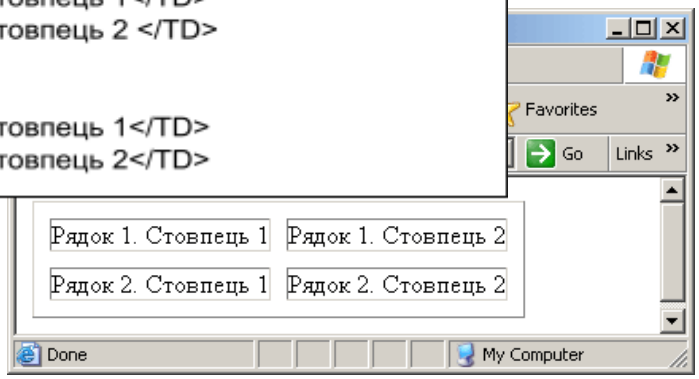

Рисунок 25

Наприклад, проілюструємо дію атрибутів <CELLSPACING>=3 й <CELLPADDING>=10 команди <BODY> (рис. 26).

```
<TABLE BORDER=1 CELLSPACING=3 CELLPADDING = 10>
<TR>
<TD>Рядок 1. Стовпець 1.</TD><TD>Рядок 1. Стовпець 2</TD>
</TR>
<TR><TD>Рядок 2. Стовпець 1.</TD><TD>Рядок 2. Стовпець 2</TD>
</TR>
</TABLE>
```

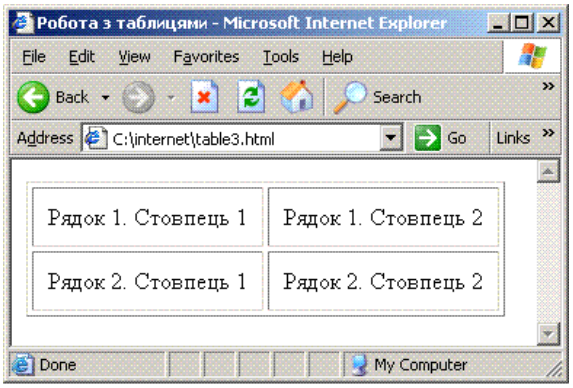

Рисунок 26

Кожен стовпець може мати заголовок, що задається необов'язковим тегом **<TH>**…**</TH>**.

Проілюструємо дії команд <CAPTION> та <TH> (рис.27).

```
<TABLE BORDER=1>
<CAPTION>Заголовок таблиці</CAPTION>
<TH>Заголовок 1</TH><TH>Заголовок 2</TH>
<TR>
<TD>Рядок 1. Стовпець 1.</TD>
<TD>Рядок 1. Стовпець 2.</TD>
</TR>
<TR>
<TD>Рядок 2. Стовпець 1.</TD>
<TD>Рядок 2.Стовпець 2.</TD>
</TR>
</TABLE>
```

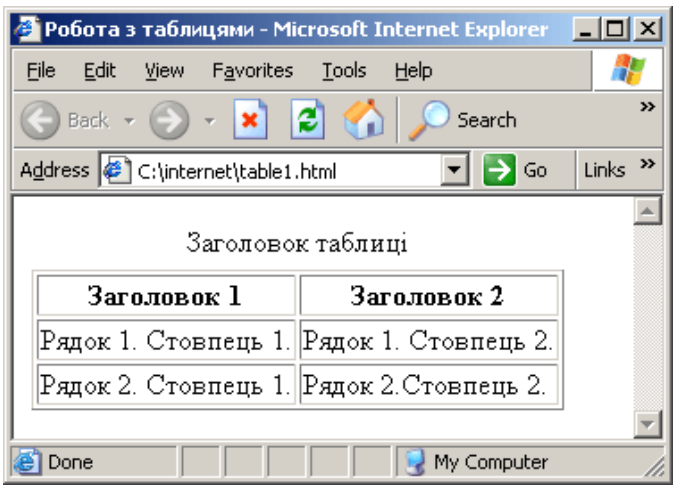

Рисунок 27 – Дії команд <CAPTION> та <TH>

Теги рядків <**TR**> і стовпців <**TD**> можна використовувати в сполученні з такими атрибутами:

ALIGN = LEFT | RIGHT | CENTER вирівнювання вмісту по горизонталі.

VALIGN – вирівнювання вмісту:

=TOP – по вертикалі;
=MIDDLE – по верху;
=BOTTOM – по низу.

BGCOLOR=*color* – заливання кольорами;

COLSPAN=*n* – зробити комірку шириною в *n* стовпців;

ROWSPAN=*m* – зробити комірку висотою в *m* рядків.

Наприклад, проілюструємо дії атрибута BGCOLOR команди <TD>.

```
<TABLE BORDER=0 CELLSPACING=0>
<CAPTION>Протилежності</CAPTION>
<TR>
    <TD BGCOLOR=FFCCFF> Так</TD>
    <TD BGCOLOR=CCFFCC> Ні </TD>
</TR>
<TR>
<TD BGCOLOR=FFCCFF> Гарячий </TD>
<TD BGCOLOR=CCFFCC> Холодний </TD>
</TR>
<TR>
<TD BGCOLOR=FFCCFF> М'який </TD>
<TD BGCOLOR=CCFFCC> Твердий </TD>
</TR>
<TR>
<TD BGCOLOR=FFCCFF> Добрий </TD>
<TD BGCOLOR=CCFFCC> Злий </TD>
</TR>
</TABLE>
```

Як бачимо із прикладу, для того щоб задати кольори в цілому стовпці, необхідно встановити його в кожній комірці цього стовпця.

Інший вигляд має справа при роботі з кольорами рядка – досить вказати кольори тегу рядка <TR>.

Наприклад, проілюструємо дії атрибута BGCOLOR команди <TR>.

```
<TABLE BORDER=0 CELLSPACING=0>
<CAPTION> Протилежності </CAPTION>
<TR GCOLOR=FFCCFF>
        <TD> Так </TD>
        <TD> Ні </TD>
</TR>
<TR BGCOLOR=CCFFCC>
        <TD> Гарячий </TD>
        <TD> Холодний </TD>
</TR>
<TR BGCOLOR=FFCCFF>
        <TD> М'який </TD>
        <TD> Твердий </TD>
</TR>
<TR BGCOLOR=CCFFCC>
        <TD> Добрий </TD>
        <TD> Злий </TD>
</TR>
</TABLE>
```

Зауваження. При оформленні таблиць іноді виникає необхідність задати фонову картинку для комірки таблиці. У таких випадках вдаються до допомоги стилів:

<TD STYLE="BACKGROUND-IMAGE:URI;">

Атрибут BACKGROUND для тегу <TD> не рекомендований стандартами HTML і не підтримується деякими браузерами, наприклад у браузері Opera.

Робота над складними таблицями

Розглянемо це питання на такому прикладі. Необхідно створити складну таблицю вигляду:

Теги HTML таблиць			
Тег	Атрибут	Значення	Примітка
<TABLE>	None	Початок таблиці	
<TR>	None	Початок рядка	
<CAPTION>	None	Заголовок таблиці	
<TH>	ColSpan	Ширина (у стовпцях)	
	RowSpan	Висота (у рядках)	
	Align	Вирівнювання	
<TD>	ColSpan	Ширина (у стовпцях)	
	RowSpan	Висота (у рядках)	
	Align	Вирівнювання	

При створенні таких таблиць (з великою кількістю довільно об'єднаних по вертикалі й горизонталі комірок) корисно дотримуватися такої послідовності кроків:

1 Накреслити схему таблиці.
2 Провести до кінця пунктирними лініями ті перегородки, які не доходять до границь таблиці.
3 Написати HTML-код таблиці, уявивши, що пунктирні лінії – суцільні.
4 Вміст, параметри фону й вирівнювання комірок з пунктирними лініями прописати в тій комірці, що перебуває зверху та ліворуч.
5 Додати в <TD> кожної такої комірки атрибути ROWSPAN й COLSPAN з параметрами, що дорівнюють кількості комірок, які поєднуються по вертикалі й горизонталі відповідно.
6 Видалити порожні пари <TD></TD>.

Розглянемо HTML-код складної таблиці:

```
<TABLE BORDER=1>
<CAPTION>Теги HTML таблиць</CAPTION>
<TH>Тег</TH><TH>Атрибути</TH>
<TH>Значення</TH> <TH>Примітки</Th>
<TR>   <TD>&ltTABLE&gt</TD>
         <TD>NONE</td>
         <TD COLSPAN=2 ALIGN=RIGHT> Початок таблиці </TD>
         <TD></TD>
</TR>
<TR><TD>&ltTR&gt</TD>
       <TD>NONE</TD>
       <TD COLSPAN=2 ALIGN=RIGHT> Початок рядка </TD>
       <TD></TD>
</TR>
<TR><TD>&ltCAPTION&gt</TD>
       <TD>NONE</TD>
       <TD COLSPAN=2 ALIGN=RIGHT> Заголовок таблиці </TD>
       <TD></TD>
</TR>
<TR><TD ROWSPAN=3>&ltTH&gt</TD>
       <TD>Colspan</TD>
       <TDCOLSPAN=2 ALIGN=RIGHT> Ширина в стовпцях</TD>
       <TD></TD>
</TR>
<TR><TD></TD>
       <TD>Rowspan</TD>
       <TD COLSPAN=2 ALIGN=RIGHT> Висота в стовпцях </TD>
       <TD></TD>
</TR>
<TR><TD></TD>
       <TD>Align</TD>
       <TD COLSPAN=2 ALIGN=RIGHT> Вирівнювання в комірці </TD>
       <TD></TD>
</TR>
<TR><TD ROWSPAN=3>&ltTD&gt</TD>
       <TD>Colspan</TD>
  <TD COLSPAN=2 ALIGN=RIGHT>Ширина в стовпцях</TD>
       <TD></TD>
</TR>
<TR><TD></TD>
       <TD>Rowspan</TD>
       <TD COLSPAN=2 ALIGN=RIGHT> Висота в стовпцях </TD>
       <TD></TD>
</TR>
<TR><TD></TD>
       <TD>Align</TD>
       <TD COLSPAN=2 ALIGN=RIGHT> Вирівнювання в комірці </TD>
       <TD></TD>
</TR>
</TABLE>
```

Видаливши всі підкреслені теги з HTML-коду, одержимо підсумковий код складної таблиці, що у вікні браузера буде мати такий вигляд, як показано на рисунку 28.

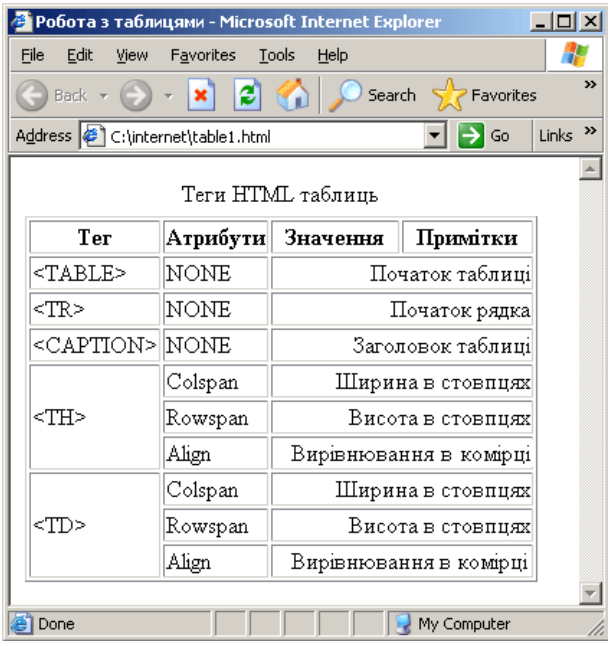

Рисунок 28 – Приклад складної таблиці

Зауваження Атрибут ALIGN (=RIGHT|LEFT) команди <TABLE> призначений для того, щоб задати обтікання таблиці текстом. Наприклад, нижченаведений код у вікні браузера буде мати такий вигляд, як показано на рисунку 29.

```
<BODY>
  <TABLE BORDER=1 ALIGN=RIGHT>
<TR>
      <TD>Це - проста таблиця </TD>
      </TR>
  </TABLE>
  Цей текст повинен обтікати таблицю,
  розміщену праворуч, тому що зазначено
  атрибут ALIGN=RIGHT команди &ltTABLE&gt
</BODY>
```

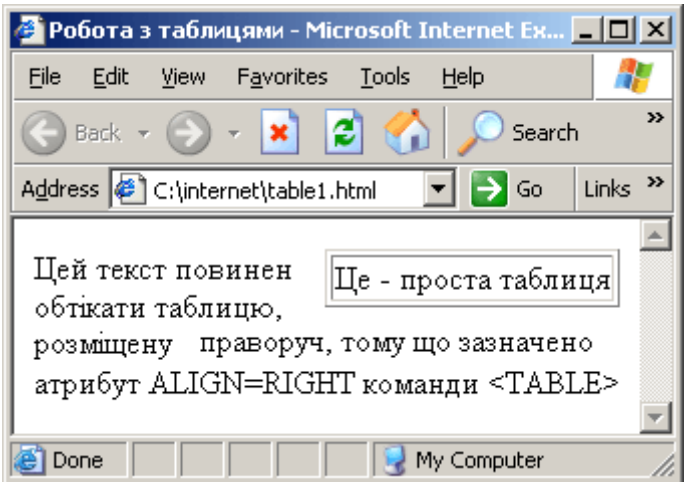

Рисунок 29 – Обтікання таблиці текстом

Щоб вирівняти таблицю по центру вікна, код документа правильно буде написати так:
<***DIV STYLE="ALIGN:CENTER"***>

```
    <TABLE>  ... </TABLE>
</DIV>
```

Щоб розмістити текст, картинку або таблицю в центрі екрана, можна звернутися до таблиць і скористатися таким кодом:

```
<TABLE WIDTH="100%" HEIGHT="100%" BORDER="0" CELLSPACING="0"
       CELLPADDING="0">
   <TR>
      <TD ALIGN="CENTER" VALIGN="MIDDLE">
      Текст, картинка або таблиця в центрі екрана
      </TD>
   </TR>
</TABLE>
```

3.5.4 Елементи на рівні тексту. Фізичний і логічний стилі форматування

Текст HTML-документа (окремі слова або речення) можна описувати спеціальними стилями. Існує два типи стилів: фізичний і логічний.

Фізичні стилі

Існують фізичні способи виділення – автор задає стиль написання тексту, описуючи шрифт у вихідному HTML-документі.

Можливо задати:

<B> *текст* **</B>** – **жирний** шрифт.

<I> *текст* **</I>** – *похилий* шрифт.

<TT>*текст***</TT>** – шрифт заданої ширини (моношрифт).

<U> *текст* **</U>** – підкреслений шрифт.

<STRIKE>*текст***</STRIKE>** – ~~перекресленний~~ шрифт.

Спеціальні символи. Символи <, >, & й " мають в HTML особливе значення як символи форматування. Але іноді виникає необхідність використати їх у тексті за своїм прямим призначенням. Для введення таких символів у текст необхідно використовувати спеціальні сполучення символів:

< – < ліва дужка;
> – > права дужка;
& – & амперсанд;
" – " лапки;
  – пробіл;
¬ – ¬ перенос;
« – « ліві подвійні лапки;
» – » праві подвійні лапки;
© – © знак, який вказує кому належать авторські права;
® – ® знак, який вказує на реєстрацію;
¶ – ;
· – · середня крапка ;
° – ° градус;
± – ± плюс-мінус.

Зауваження. Спеціальні символи чутливі до регістра: НЕ МОЖНА використовувати < замість <

Наприклад, наступний текст до якого застосовані фізичні стилі у вікні браузера буде мати такий вигляд, як показано на рисунку 30.

```
<B>(!!!)</B> Мова HTML <u>не чутлива</u> до регістра. <tt>Команди</tt> мови можуть набиратися як<i> великими, </i>так і малими літерами, тобто команда &lt;title&gt; еквівалентна команді &lt;TITLE&gt; або &lt;TiTle&gt;.
Але <b><i>спеціальні символи</b></i> чутливі до регістра: <b>НЕ МОЖНА</b> використати &amp;LT; замість &amp;lt;
```

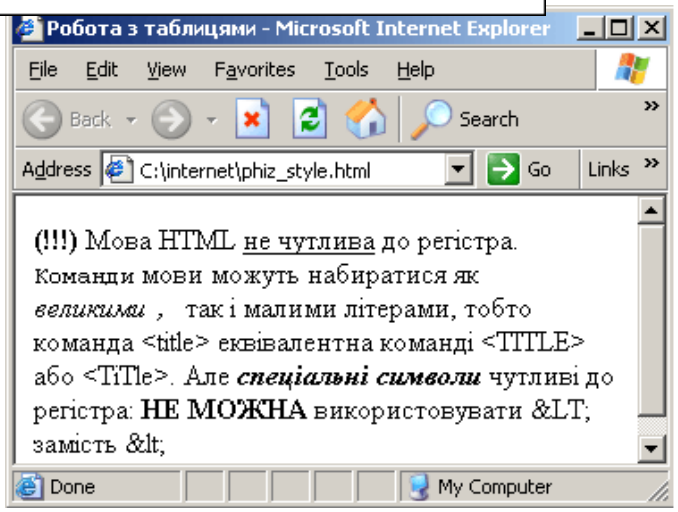

Рисунок 30 – Застосування фізичних стилів форматування та спеціальних символів

Оскільки неможливо передбачити, який буде мати вигляд той або інший тег (фізичного стилю форматування) у різноманітних браузерах, використовують ***логічне форматування***. Теги логічного форматування сприймаються будь-яким браузером й інтерпретуються відповідно до його можливостей.

<DFN> – опис визначень.

<EM> – виділення слів.

<CITE> – виділення заголовків книг, фільмів, цитат і т.п.

<CODE> – виділення програмних кодів, текстів програм і т.п. Зображується шрифтом фіксованої ширини.

<SAMP> – використовується для машинних повідомлень. Зображується шрифтом фіксованої ширини (моношрифт).

<STRONG> – використовується для особливого виділення слів. Як правило, виділяється жирним шрифтом.

<VAR> – вживають для символьних змінних.

Часто виникає питання, в чому полягають відмінності в роботі тегів <B> й <STRONG>, <I> й <EM>.

Як зазначено вище, <B> й <I> – теги фізичного виділення, тобто веб-майстер примусово змушує браузер виділяти текст заданим видом шрифту. Теги <STRONG> й <EM> – теги логічного виділення. Кожен браузер може по-своєму виділити текст усередині цих тегів, так, як зручно його користувачеві.

Таким чином, якщо потрібно виділити текст, варто користуватися тегом <EM>. Якщо ж не потрібно виділяти текст, а зробити його курсивом, використовують тег <I>.

До логічних стилів можна віднести й команди роботи зі шрифтами.

<FONT> – може змінювати як логічний (розмір), так і фізичний стиль шрифту. Ця команда задає кольори, розмір і вигляд сімейства шрифтів, які використовуються в HTML-документі.

Атрибути команди **<FONT>**

COLOR = “color” дозволяє задати колір шрифту.

Наприклад, рядки , що наведені нижче, рівнозначні.

<FONT COLOR= #FF0000> текст червоних кольорів </FONT>

<FONT COLOR= RED> текст червоних кольорів </FONT>

SIZE=(±)n – установлює розмір шрифту. Число *n* при *абсолютному* заданні розміру може набувати значень від 1 до 7. За замовчуванням базовий розмір шрифту в документі вважається таким, що дорівнює 3 в абсолютних значеннях.

При відносному заданні розміру число *n* може набувати значень від 1 до 7 зі знаками плюс (+) – збільшення або мінус (-) – зменшення розміру шрифту на *n* пунктів стосовно базового розміру шрифту в документі.

<***BIG***>*текст*<**/BIG**> – збільшує розмір шрифту на одиницю щодо базового розміру.

<***SMALL***>*текст*<**/SMALL**> – зменшує розмір шрифту на одиницю щодо базового розміру.

<***SUB***> *текст* <**/SUB**> – підрядковий індекс.

<***SUP***> *текст* </***SUP***> – надрядковий індекс.

Сімейства шрифтів

Для команди ***<FONT>*** ім'я сімейства шрифтів визначається атрибутом ***FACE***. Для команди ***<BASEFONT>*** атрибутом ***NAME*** .

Стандартні значення цих **атрибутів** *"Times New Roman"*, *"Courier New"* та *"Arial"*.

Імена сімейств шрифтів у даний момент поки не стандартизовані й у різних системах можуть називатися по-різному. Наприклад, сімейство шрифтів ***"Times New Roman"*** може називатися ***"Times"***, ***"Times Roman"***, ***"Roman"*** і т.д.

Наприклад, нижченаведений фрагмент HTML-коду у вікні браузера буде виглядати так, як показано на рисунку 31.

```
<p><FONT FACE="ARIAL">Ласкаво просимо до СумДУ
</FONT>
<p><FONT FACE="Courier New">Ласкаво просимо до
СумДУ </FONT>
<p><FONT FACE="Times New Roman">Ласкаво просимо до
СумДУ </FONT>
```

Переривання рядка. Команда ***
*** дозволяє перейти на новий рядок, не починаючи нового абзацу (див. приклади #1 – #6).

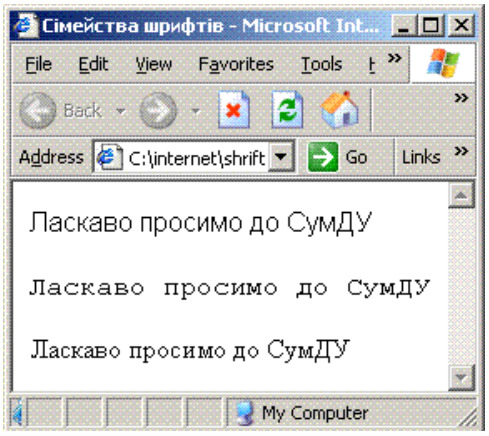

Рисунок 31

3.5.5 Посилання

Припустимо, що в Інтернеті існує хостинг із доменним ім'ям *hosting.net* й IP-адресою 105.32.121.32. На ньому розміщено кілька сайтів :

www.user1.net
www.user2.com
www.user3.sumy.ua

У такий спосіб маємо, що в кожного сайту є унікальне доменне ім'я, але IP-адреса у всіх однакова – 105.32.121.32 – IP-адреса хостинга.

У системі DNS на прохання сисадмінів (власників) web-сайтів USER'ів і за узгодженням із сисадміном *hosting.net* прописується, що слід вважати IP-адресою того або іншого USER'а IP-адресу хостинга.

Коли рядовий користувач на своєму «домашньому комп'ютері» набирає *www.user3.sumy.ua*, запит системою DNS направляється на IP-адресу хостинга. На сервері *hosting.net* запускається програма, що обслуговує роботу всього сервера (найпоширеніша – APACH), що контролює, який із сайтів потрібний конкретному користувачеві.

Нехай сайт *www.user3.sumy.ua* має структуру, яка показана на рисунку 32.

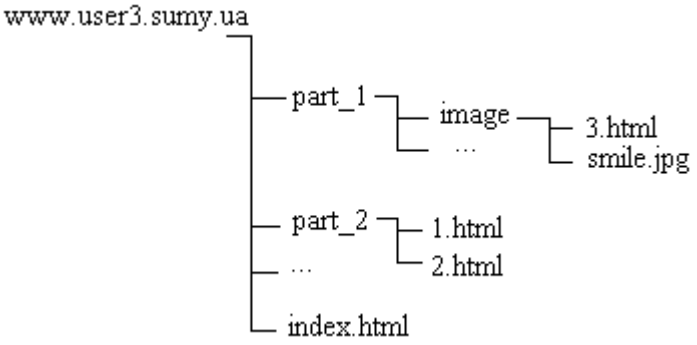

Рисунок 32 – Структура web-сайту

У розрізі одного сайту існує кілька видів посилань:

- посилання на інші сайти;
- внутрішні посилання, тобто посилання усередині сайту;
- посилання усередині файлу або на закладку.

Посилання на інші сайти

Зв'язок між документами й окремими частинами документів задається через відповідні гіперпосилання.

Тег посилання має таку структуру:

<A HREF="*URL***">** *текст посилання* **</A>**,

де

URL – унікальна доменна адреса. Лапки (**" "**) при присвоюванні значення атрибуту HREF необов'язкові, але бажані;

текст посилання – текст, що у вікні браузера буде виділений як посилання.

Наприклад:

1 Необхідно поставити посилання на сайт *www.lenty.ru*:

<A HREF="http://www.lenty.ru"> Найбільш цікаві новини </A>.

2 Поставити посилання на конкретну інформацію, тобто на файл, що належить іншому сайту:

<A HREF="http://www.lenty.ru/ua/01/21.html">Наука в Україні </A>.

Кожний з сайтів (*www.user1.net, www.user2.com, www.user3.sumy.ua*), які розміщені у межах одного сервера (хостинга), є окремим ***віртуальним сервером***. І доступ з одного сайту на інший, розміщений у межах одного хостинг-сервера, здійснюється тільки через Інтернет.

Наприклад, поставити посилання із сайту *www.user3.sumy.ua* на сайт *www.user2.com*. Це можна зробити тільки одним способом:

<A HREF=”http:// www.user2.com”> *www.user2.com* </A>

Поставити посилання із сайту *www.user2.com* на файл 1.html, що належить сайту *www.user3.sumy.ua*:

<A HREF=”http:// www.user3.sumy.ua/razdel2/1.html”>1.html </A>

Посилання усередині одного сайту

1 В *index.html* сайту *www.user3.sumy.ua* зробити посилання на файл *1.html*

<A HREF=”*razdel2/1.html*”> текст </A>

2 У файлі 2.*html* зробити посилання на файл *gabo.jpg*

<A HREF=”**..**/*razdel1*/*image*/*gabo.jpg*”> текст </A>

Символ **“../”** – указує на те, що варто перейти в каталог верхнього рівня.

3 З файлу *3.html* зробити посилання на файл *1.html*

<A HREF=*../../razdel2/1.html*> текст </A>

4 Поставити посилання з будь-якого місця сайту на головну сторінку – *index.html*.

<A HREF=*/index.html*>На головну сторінку</A>

Посилання на закладки

У HTML під закладкою розуміють посилання, але не на весь файл, а на якусь його частину, що заздалегідь була названа, як закладка. Для цього у тега **<A>** існує атрибут **NAME**.

NAME= “*мітка*” – створення мітки усередині HTML документа, де *мітка* – це будь-який символ або набір символів.

Наприклад: <A NAME=”b”> … </A>
<A NAME=b> … </A>
<A NAME=”2”> ... </A>

На певну закладку завжди можна зробити посилання з:

- ***будь-якого місця того ж файлу***:

<A HREF=”#мітка”> Текст посилання </A>.

Наприклад, у файлі *2.html* є текст:

```
«...<H3>Де знайти програмне забезпечення для Macintosh?</H3>

Програмне забезпечення можна отримати різними способами. Наприклад,

  <UL>
<LI><A HREF="#1"> у свого постачальника комп'ютерної
      техніки </A>
<LI><A HREF="#2"> купити в спеціалізованих магазинах
</A>
<LI><A HREF="#3"> знайти в Інтернеті </A>
  </UL> ...
```

```
<A NAME="#1">  У постачальника </A> ...
<A NAME="#2"> Де купити </A> ...
<A NAME="#3">  Як знайти </A> ... »
```

- ***з іншого файлу одного віртуального сервера***.

  Наприклад, з файлу *3.html* поставити посилання на закладку #2 файли *2.html*

  <A HREF=../../*razdel2/2.html#2*> Де купити ПО? </A>.

- ***з іншого віртуального сайту***.

  Наприклад, із сайту *www.user2.com* на закладку #3 файли 2.html сайту *www.user3.sumy.ua*

  <A HREF="http://*www.user3.sumy.ua/razdel2/2.html#3*> Де знайти ПО? </A>.

Посилання на e-mail

Щоб при натисканні на посилання з'являвся бланк створення e-mail повідомлення, використовують команду <A> у такому форматі:

<A HREF="MAILTO:email">посилання.

Наприклад,

<A HREF="MAILTO:student1@id.sumdu.edu.ua">Student

Можна додати автоматичне підставлення теми в бланк створення e-mail повідомлення

посилання.

Наприклад,

<A HREF = "MAILTO:student1@kpm.sumdu.edu.ua ?SUBJECT=Питання заліку"> Student

Зауваження. Цей варіант некоректно обробляється деякими браузерами й мейлерами.

Тег <A>, крім атрибута *HREF,* має атрибут ***TARGET***, що може набувати таких значень:

= **blank** – відкрити посилання в чистому вікні;
= **new** – відкрити посилання в новому вікні;
= **self** – відкрити посилання у тому вікні або кадрі,
де розміщене посилання.

3.5.6 Робота із графічними зображеннями

Графічні файли включаються в HTML-документи за допомогою команди ***<IMG>***. Допускається використання файлів у форматі GIF або JPG/JPEG (для систем, що працюють під MS Windows, допускаються також файли формату BMP).

С тегом ***<IMG>*** можуть бути використані атрибути:

***SRC*=** "URL" – визначає URL-адресу або ім'я графічного файлу.

Наприклад:

<IMG SRC=/img/narjad3.jpg >,

<IMG SRC="http://www/podruga.net/img/narjad3.jpg" >.

ALT= – атрибут, який вказує, що ставити на місце графічного об'єкта, якщо браузер не в змозі показувати графічні файли або внаслідок повільної швидкості перекачування даних не було одержано відповідних даних (рис. 33).

HEIGHT= n – задає висоту контуру (n – кількість пікселів), у якому буде розміщене зображення.

WIDTH= n – задає ширину контуру (n – кількість пікселів), у якому буде розміщене зображення.

Наприклад, на рисунку 33 показано, як буде відображений об'єкт, описаний у нижченаведеному фрагменті HTML-коду.

<IMG SRC="rogd03.jpg" WIDTH=250 HEIGHT=113 ALT="New Year">

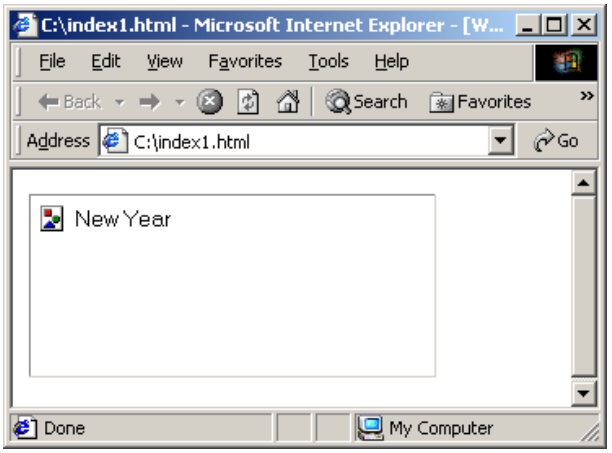

Рисунок 33

Якщо розміри контуру, зазначені атрибутами, не збігаються з розмірами графічного файлу, то останній масштабується. Масштабування може привести до різкого погіршення якості графічного файлу, тому рекомендується задавати розміри, що відповідають розмірам графічного файлу.

Рекомендується для великих графічних файлів (більше 10 Kb) завжди задавати їхні розміри, для збільшення швидкості роботи браузера. Якщо розміри не задані, то, зустрівши графічний файл, браузер припиняє виведення тексту й чекає, поки закачається вся картинка, щоб визначити її розміри.

HSPACE=m – ширина чистих полів у пікселах ліворуч і праворуч зображення.

VSPACE = ***m*** – ширина чистих полів у пікселах зверху й знизу зображення.

BORDER= n – ширина рамки зображення.

Покажемо, як впливає на відображення графічного об'єкта у вікні браузера застосування тегу *IMG* з атрибутами *HSPACE, VSPACE, BORDER.* Припустимо, що в тілі певного HTML-документа присутня команда

<IMG SRC="rogd03.jpg" HSPACE="20" VSPACE="20" BORDER= "2" ALIGN="left">.

Тоді у вікні браузера HTML-документ буде мати такий вигляд, як показано на рисунку 34.

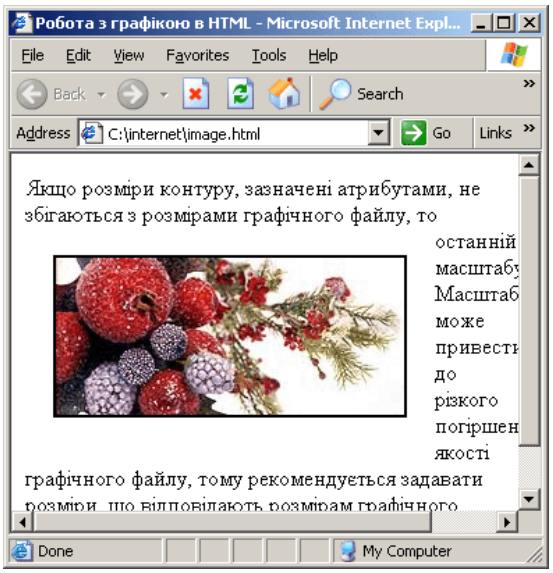

Рисунок 34

Якщо тег *IMG* використовується без атрибутів *HSPACE, VSPACE, BORDER* тоді той самий документ буде мати такий вигляд, як показано на рисунку 35.

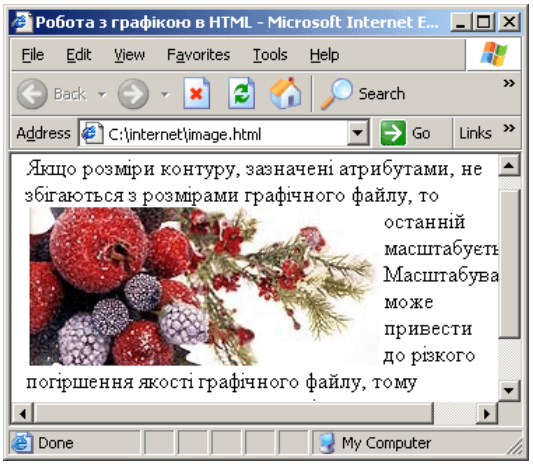

Рисунок 35

Якщо використовувати атрибут ***ALIGN*** зі значеннями ***LEFT*** або ***RIGHT***, можна створити "плаваючу" картинку, яку буде обтікати текст. Наприкінці плаваючого об'єкта обов'язково повинна бути присутня команда <***BR***> з атрибутом ***CLEAR***=, що припиняє обтікання картинки. Після цієї команди текст виводиться нижче графічного файлу.

Наприклад, наступний фрагмент HTML-коду у вікні браузера буде мати такий вигляд, як показано на рисунку 36.

“<IMG SRC="rogd03.jpg" ALIGN=LEFT HSPACE=4> Цей текст, необхідно розмістити праворуч, поруч із картинкою. Текст повинен обтікати картинку. <BR CLEAR=LEFT>А цей текст повинен знаходитись нижче картинки...”

Кілька порад при роботі із графікою:

Завжди використовуйте атрибути задання розміру картинки. У цьому випадку браузер буде продовжувати показувати web-документ, а не чекати доки закачається вся картинка.

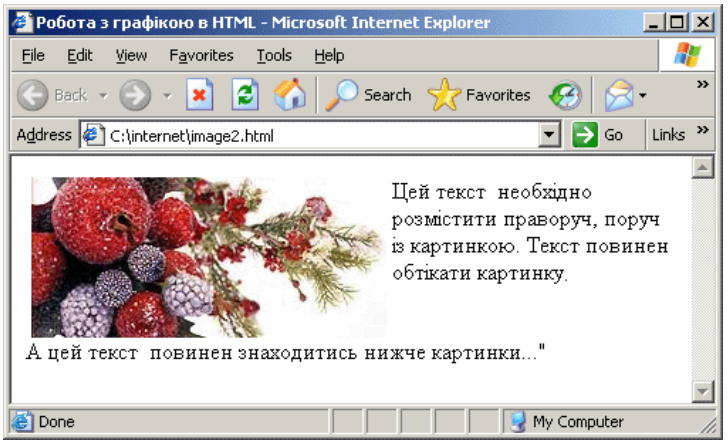

Рисунок 36

Використовуйте картинки, переважно у форматі JPEG/JPG (картинки у форматі GIF мають велику кількість деталей, що повільно відображаються на не дуже потужних машинах).

Якщо бажаєте, щоб користувач із інтересом читав ваші сторінки, то не вставляйте в документ картинки, розмір яких більший 40 Kb. Для об'єктів більшого розміру використовуйте посилання з обов'язковою вказівкою розміру.

Не зловживайте анімаційними GIF'ами – це сильно заважає перегляду сторінки.

4 Розміщення й реклама сайтів в Інтернеті

4.1 Розміщення сторінки в Інтернеті

Кілька рекомендацій для створення «комерційної» web-сторінки:

1 Гарненько продумайте тематику web-сторінок.

2 Перш ніж розпочинати створення web-сторінки, задумайтеся, а чи буде це кому-небудь цікаво.

3 Створюючи HTML, намагайтеся, щоб ваша сторінка мала мінімально можливий обсяг (див. рекомендації до роботи з графікою).

4 Всі посилання в документі повинні бути відносні, з тією метою, щоб «переїжджаючи» з місця на місце в вам не доводилося б щоразу вручну їх змінювати.

5 При виборі назви сайту враховуйте, що воно повинне бути співзвучно з майбутньою URL сайту (див. наступний пункт).

6 Визначитеся з доменним ім'ям сайту. Це можна зробити за допомогою хостингів. Як правило, вони ж і допомагають у реєстрації. Але все можна зробити й самостійно: виберіть і зареєструйте вільне доменне ім'я за допомогою сервісу whois-сервіс.

При реєстрації доменів необхідно знати такі правила**:**

- доменне ім'я повинне починатися й закінчуватися буквою латинського алфавіту або цифрою; проміжними символами в доменному імені можуть бути букви латинського алфавіту, цифри або дефіс;
- доменні імена в зонах .ru, .de реєструються строком на 1 рік;
- домени в зонах .com, .net, .org, .info, .biz, .cc, .tv, .name ви можете зареєструвати на строк від 1 до 10 років;
- доменні імена в зонах .co, .uk, .org., uk реєструються строком від 2 до 10 років;
- домен у зоні *.name* має вигляд: *ім'я.прізвище.name* або *прізвище.ім'я.name*;
- зазначені тільки назви зон, у яких реєструється ваше доменне ім'я, тобто якщо ви реєструєте доменне ім'я "*domain*" у зоні *.ru*, то ім'ям вашого web-сервера буде *www.domain.ru*, а поштові адреси будуть закінчуватися на *@domain.ru*;
- при реєстрації доменного імені усі права на його використання делегуються тільки вам;
- при замовленні реєстрації доменного імені ви погоджуєтеся з відповідним доменній зоні регламентом.

Наведемо деякі розцінки (у середньому):

Доменна зона	Реєстрація	Продовження
.mhost.ru	безкоштовно	безкоштовно
.ru (підтримка RU-CENTER)	22.00 у. о.	17.00 у. о.
.ru (пітримка РосНІІРОС)	Немає	20.00 у. о.
.ru (підтримка "Гарант-Парк-Телеком")	Немає	20.00 у. о.
.su	120 у. о.	120 у. о.
.com.ru, net.ru, org.ru, pp.ru	5 у. о.	безкоштовно
.com , .net , .org , .info , .biz	20 у. о.	20 у. о.
.cc	40 у. о.	40 у. о.
.name	30 у. о.	30 у. о.
.co.uk, .org.uk	50 у. о.	50 у. о.
.de	24 у. о.	24 у. о.

.com.ua	10 у. о.	10 у. о.

7 Зв'яжіться з адміністратором хостинга та домовтеся про умови розміщення вашого сайту (за що будете платити: за трафік або за розмір сторінки; розмір оплати; вимоги до сайту, скриптів, сервера і т.п.).

8 Після домовленості й проплати сисадмін надає: FTP, з логіном і паролем для авторизированого доступу, логін і пароль до поштової скриньки з усіма правами адміністрування, вимоги до сайту, тобто каталог для скриптів, каталог для самого сайту, можливо логін і пароль для telnet, і т.п.

9 Далі ваша турбота – турбота про сайт, все інше, турбота про канали доступу, про зв'язки із системою DNS, підтримка роботи сервера, його апгрейд – це турбота хостинга.

10 Починаємо «розкрутку».

4.2 Банери

Банери – це посилання у вигляді картинок або тексту. Виділяють такі типи банерів:

- графічні;
- текстові.

Графічні бувають ***різних розмірів*** (великі 468x60, *кнопки* 88x31, *квадратні* 100x100, прямокутні 240x120 та ін.) та ***модифікацій*** (*статичні* й *динамічні*). Стандартним вважається розмір 468x60. Всі інші розміри – вибираються індивідуально, на бажання замовника.

Динамічні *банери* задаються так називаними кодами банерів. Наприклад,

<A href="http://www4.reklama.ru/cgi-bin/href/podruga?142851" target=_blank><IMG alt="www.reklama.ru. The Banner Network." border=0 height=60 isMap src="http://www4.reklama.ru/cgi-bin/banner/podruga?142851" width=468></A>

Статичний *банер* – це звичайне посилання у вигляді графічного об’єкта певного розміру. Наприклад,

- статичний банер-***кнопка***:

<A target=_blank href="http://counter.rambler.ru/top100/"><IMG SRC="img/rambler.gif" ALT="Рамблер ТОП100" width=88 height=31 BORDER=0></A>;

- статичний ***текстовий*** *банер*:

<A href=”http://www.podruga.net”>Подруга – жіночий журнал.

4.3 Реклама в Інтернеті

Заходячи на якийсь сайт, відвідувачі, як правило, не замислюються над тим, хто купив сервер, хто платить адміністратору, який його підтримує, хто платить програмісту, хто оплачує трафік, дизайн, контент (вміст) і т.п. Це все "безкоштовно". Але безкоштовно воно для відвідувачів, а для хазяїна сайту, на жаль, далеко не так. І чим більше людей заходить на сайт, тим більше авторові доводиться платити:

- трафік виходить за встановлені хостером обмеження – підвищилася плата за хостинг; скрипти треба оптимізувати– отже, зарплата програмістові (або сам працюй);
- місце на диску скінчилося – доплачуй хостеру або купуй новий диск, якщо перейшов на колокейшн;
- треба розвивати вміст сайту, що теж вимагає роботи і т.д.

Сайти в Мережі можна умовно поділити на 2 типи:

1. Сайти, які *генерують відвідувачів*. До них можна віднести сайти новин, пошукові сайти, архіви програм, літературні та розважальні сайти і т.п.
2. Сайти, які *генерують гроші*: інтернет-магазини, сторінки комерційних компаній, провайдери та ін.

Теоретично два типи сайту можуть зливатися в одному, але однаково їх можна досить чітко розділити.

Перший тип сайтів (які генерують відвідувачів) ***є збитковим***: ресурси сервера, канали, інформація та інше надається відвідувачам безкоштовно, а ось авторам сайту за все це доводиться платити.

Сайти ***другого типу*** приносять гроші, але їхній дохід прямо залежить від популярності, оскільки вони зацікавлені у відвідувачах (причому, бажано, цільових – зацікавлених у запропонованих послугах і товарах, а також платоспроможних).

Виникає цілком природне прагнення до "симбіозу" – перші сайти готові продати частину своїх відвідувачів другим в обмін на деяку частку їхніх доходів. І інструментом такого продажу є ***реклама***.

Найпростішим способом розміщення реклами є пряма взаємодія між рекламодавцем (інтернет-магазином, наприклад) і рекламним майданчиком (сайтом, що генерує відвідувачів).

Такі відносини виникають досить рідко й тільки між сайтами приблизно однакового "розміру".

Часто рекламодавець розміщає рекламу на багатьох майданчиках, а кожний рекламний майданчик "крутить" рекламу різних рекламодавців. Відслідковувати все це вручну досить складно і тим й іншим, тому на ринку з'явилося місце для "координатора" – ***рекламного агентства*** (найбільш розвинені й великі www.rle.ru, www.tx3.ru).

Рекламне агентство займається тим, що розроблює медіа-план для рекламодавця (користуючись своїми знаннями Мережі й домовленостями з власниками рекламних майданчиків) і є великим і більш-менш постійним рекламодавцем для сайтів – рекламних майданчиків.

Рекламні технології

Першим етапом "промислового" розвитку реклами в Мережі були ***банерні мережі*** – вони дозволяли кожному сайту показувати на своїх сторінках чужу рекламу, а рекламу цього сайту крутили на сторінках інших учасників.

Оскільки система була обмінною та безкоштовною, то для того щоб окупити свої витрати, банерні мережі брали комісійний відсоток показів (якщо ви показали на своєму сайті чужий банер 10 разів, то ваш банер показувався на інших сайтах, скажімо, 9 разів, а один показ залишався в розпорядженні баннерної мережі) і ці покази продавали.

Спочатку такий бізнес ішов досить непогано, але досить швидко власникам сайтів теж захотілося одержувати гроші за покази банерів. Виникає *вторинний ринок* банерної реклами (продаж сайтами накопичених показів), де ціни були значно нижчі, ніж на *первинному* (у банерних мережах).

З часом (приблизно до 2002) інтернетівський бум пройшов, користувачі Інтернету звикли до банерів (отже, знизилась їх ефективність), ціни на банерну рекламу різко впали, і такий вид реклами для багатьох популярних сайтів став малоцікавим, тому що не покривав їхніх витрат.

Перевага *банерних мереж* полягає в тому, що рекламодавцеві *не треба укладати договір з кожним із тисяч сайтів*, на яких буде показуватися реклама, а ***недолік*** – у порівняно *низькій ефективності* такої *реклами.*

Ідея банерних мереж була взята на озброєння більшістю великих сайтів – вони створюють щось на кшталт локальної рекламної "крутилки", що за певними правилами вибирає з локального пулу банери й показує їх на сторінках сайту.

Способи показу реклами

Розглянемо тільки "чесну" рекламу, опускаючи різноманітні "нечесні" методи, наприклад, спам й установку спеціалізованих троянів (програм вірусного типу, що може накручувати від вашого імені покази або відвідувачів).

Найпростішим і ***традиційним способом*** є ***виділення*** якоїсь ***частини сторінки*** (верх, низ, бічна колонка) ***для показу реклами***. Цей спосіб має не найвищу ефективність, тому що користувачі швидко звикають просто "проскакувати" повз рекламу, переходячи відразу до вмісту.

Менш поширеним, але помітно ***більш ефективним методом*** є ***вбудовування реклами у вміст сторінок***: це може бути або так називана "прихована реклама" (коли в статті, скажімо, про монітори наводиться як приклад конкретна модель), або виділений рекламний блок.

Іноді реклама показується у вікні, що ***автоматично відкривається*** (це нескладно реалізується за допомогою скриптів). Така реклама вважається вчинком "на грані фола" – більшість користувачів не любить, коли вікна вискакують без їхнього бажання, і в Мережі існує досить багато утиліт, які такі вікна не менш автоматично вбивають. А в альтернативних браузерах навіть передбачене спеціальне настроювання для заборони показу спливаючих вікон. Цей метод реклами застосовується, як правило, *для просування якихось власних сервісів або показу оплачених оголошень – більшість банерних мереж і лічильників забороняють розміщувати свій код на сторінках, які відкриваються без бажання користувача*.

Усілякі лічильники й рейтинги є непоганим маркетинговим інструментом – чим вище рейтинг сайту, тим ближче до початку списку він стоїть у своїй категорії, і тим більше цільових відвідувачів на нього зайде. Тому багато ***сайтів платять*** якісь ***невеликі суми відвідувачам*** за те, щоб ті зайшли на сайт і тим самим підняли рейтинг, що, у свою чергу, викличе приплив цільових відвідувачів, які, щось купивши, покриють видатки на рекламу.

Такий метод можна назвати "чесною накруткою": ніхто не змушує користувача ходити по сайтах – він завжди може відмовитися від одержання грошей або надаваного сервісу, а подібний метод залучення відвідувачів не порушує правил, установлених лічильником. Отже, подібні дії повинні викликати (і викликають) невдоволення самих рейтингів, і те тільки через те, що гроші надходять не до них, а до відвідувачів або сайтам першого типу. А надходять вони туди тому, що для рекламодавця це вигідніше.

Як оплачується реклама

Найпростіший спосіб оплати (який був і найпершим) – це банальна ***фіксована ціна за показ*** (точніше, за 1000 показів. На первинному ринку 1000 показів у середньому коштує від $4-$7, на вторинному реально – до $0.50).

Цей спосіб відрізняє досить низька ефективність – власник рекламного майданчику ніяк не зацікавлений у віддачі від реклами, а тільки в тому, щоб її більше показати. Саме тому був уведений такий показник, як CTR (Click Through Ratio) – відношення числа переходів по рекламі до числа її показів. Якщо рекламодавець збирається оплачувати рекламу, виходячи із числа показів, то йому варто довідатися про середній CTR на рекламному майданчику, щоб хоча б прикинути ефективність. Останнім часом цей метод оплати використається все менше й менше.

Деякою модифікацією є показ за часом, тобто ***рекламодавець викуповує якесь рекламне місце на певний сторінці сайту***. Цей спосіб, як правило, використовується для реклами на "цільових" сайтах.

Більш цікавим для рекламодавця способом є ***оплата переходів***. Тут є багато технічних складностей – як визначати унікальних користувачів, як застрахуватися від "накруток" тощо, але всі вони розв'язувані. Таким способом часто купується не тільки цільовий, але й нецільовий трафік – ті самі випадкові відвідувачі, які потрібні для підняття рейтингу.

Ще один метод – це ***партнерські програми***, які є найбільш вигідними для рекламодавців. Суть у тому, що автор сайту рекламує якийсь товар або послугу, а потім одержує відсоток із сум, які витрачає відвідувач, що прийшов із цього сайту. Тут фактично вся робота із просування реклами звалюється на власника рекламного майданчика – йому треба вибрати найцікавіше місце для реклами, якось відібрати "платоспроможних" відвідувачів і так далі. Але сумарно такий метод реклами може виявитися (хоча й досить рідко) більше вигідним, ніж просто продаж рекламного місця.

Більшість користувачів чомусь дуже люблять рахувати гроші власників web-сайтів. Причому роблять це цікаво: витрати їх не цікавлять, а передбачуваний дохід обчислюється як добуток числа показаних сторінок на максимальну вартість реклами, про яку вони від когось чули.

Насправді все не настільки райдужно.

1 ***Рекламодавців треба шукати.*** Для початку – рекламодавці аж ніяк не юрбляться в черзі. Рекламодавців треба шукати, й далеко не завжди вони знаходяться. Вартість реклами теж трохи відрізняється від того, що користувачі припускають, – на даний момент вона коливається у межах 0.1 – 0.2 цента за перехід (для нецільової реклами) і 2 центи за перехід – для цільової. Це те, що дістається рекламним майданчикам, – частина грошей осідає в агентствах.

2 ***Хостинг.*** Рахуємо далі. На даний момент гарним показником CTR для реклами є показник порядку 0.8-1 % –тобто на 1000 показів припадає 8–10 переходів по рекламі. Середній CTR значно нижчий.

Найдешевший хостинг навряд чи обійдеться дешевше $10 на місяць. Таким чином, якщо вважати, що платять 1 цент за перехід (а це непогано), то тільки для того, щоб цей хостинг окупити, з вашого сайту повинно перейти $\frac{10}{0.01} = 1000$ відвідувачів, що при CTR у 1% означає, що у вас повинно побувати СТО ТИСЯЧ відвідувачів на місяць, або 3333 за день.

Насправді, звичайно ця цифра дещо завищена – сторінка на сайті не одна, та і рекламний блок на сайті теж не один. З іншого боку, за $10 ви й сайт не зможете розмістити, скоріше вже це буде щось ближче до $25-$100 (якщо трафік не дуже великий і обмеження віртуального хостинга вам ще не заважають).

3 ***Доступ в Інтернет.*** Плюс доступ в Інтернет (а як ви сайт обновляти будете?).

4 ***Ваша робота*** із створення сайту, його програмування, рисування кнопочок, підбору кольорів і т. ін., само собою – безкоштовно.

Отже, "об'єктивні витрати" (те, що треба заплатити грошима) складуть як мінімум $50 – $100 на місяць.

Якщо припустити, що середній користувач дивиться 4 сторінки за сесію, а на сторінці розміщено 3 рекламних блоки (це типово), то необхідна відвідуваність, щоб сайт окупити, складе приблизно 83 тисячі відвідувачів (а не показаних сторінок!) на місяць, або 2700 відвідувачів на день.

А з розвитком сайту знадобиться і колокейшн, і більший трафік. І, до речі, створення чогось, що може щодня залучати кілька тисяч чоловік, – це теж справа не п'яти хвилин, але ж вам ще й працювати треба буде, бо ми не враховували вашу зарплату.

Аналізуючи все вищесказане, можна сказати, що блакитна мрія будь-якого сайту – навчитися збирати гроші з користувачів (природно, знайти таких користувачів, які будуть платити).

Дмитро Турецький якось сказав: "...Якби кожен відвідувач мого сайту платив мені один карбованець на рік (я не обмовився – саме карбованець і саме на рік!), то я не тільки прибрав би

всю рекламу із сайту, але й зробив би його найшвидшим і найзручнішим. А замість цього я майже щодня одержую обурені листи, що, я не тільки не хочу розширювати канали, так ще й наживаюся на нещасних відвідувачах, змушуючи їх дивитися рекламу. І в найкращому випадку пару раз на місяць хтось говорить “дякую”...”.

Висновки

При розміщенні власної сторінки в Інтернеті насамперед необхідно подумати про зручність використання інформації вашої сторінки кінцевим користувачем Інтернету. У зв'язку із цим продумайте тематику сайту; задумайтеся, а чи буде це кому-небудь цікаво; зробіть все можливе, щоб сторінка мала мінімально припустимий розмір, щоб користувачам не платити за непотрібні їм кілобайти інформації; постарайтеся зробити максимально співзвучними назву, доменне ім'я та тематику сайту, все це для того, щоб користувач міг без проблем запам'ятати ваш сайт, і , по-друге, для полегшення пошуку інформації.

Всі сайти в Мережі можна умовно поділити на 2 типи: сайти, які генерують відвідувачів, та сайти, які генерують гроші

Тому перші сайти готові продати частину своїх відвідувачів другим в обмін на деяку частку їхніх доходів. І інструментом такого продажу є ***реклама***.

Реклама комерційних проектів і розкручування сайтів в Інтернеті реалізується за допомогою банерів або рекламних статей. Банери – це посилання у вигляді картинок або тексту. Виділяють графічні, текстові, статичні та динамічні баннери.

В Інтернеті існує кілька способів оплати реклами:

- фіксована ціна за 1000 показаних банерів;
- продаж рекламного місця на певний строк;
- оплата переходів;
- партнерські програми.

Контрольні запитання

1 Що таке банер?

2 Які типи банерів використовуються в Інтернеті?

3 У чому полягає різниця між статичними і динамічними банерами?

4 Назвіть які різновиди статичних банерів ви знаєте?

5 Що ви знаєте про рекламні технології в Інтернеті?

6 Назвіть основні способи показу реклами в Інтернеті.

7 Які способи оплати реклами існують в Інтернеті?

5 Електронні гроші

5.1 Кредитні картки

Основним платіжним засобом донедавна в Інтернеті були кредитні картки. Просто тому, що це зручно.

Як же влаштована система електронних платежів за кредитними картками? Механіка цієї справи проілюстрована на рисунку 37.

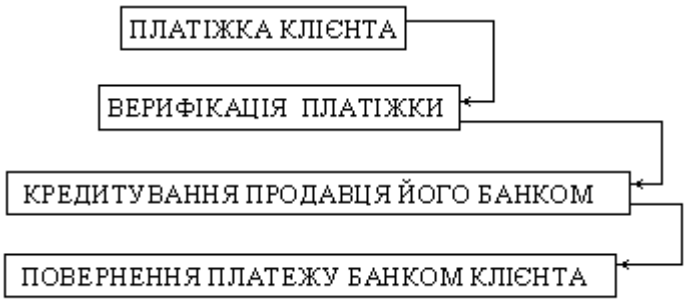

Рисунок 37 – Система електронних платежів за кредитними картками.

Причому так відбуваються не тільки платежі через Інтернет, але й платежі за кредитними картками у звичайному магазині, де вашу картку прокачують в особливому пристрої, що зчитує її номер та ім'я власника.

Але повернемося до Інтернету і розглянемо проведення платежів на прикладі.

1 Інтернет-користувач вирішив купити книжку в подарунок. Зайшов в інтернет-магазин і вибрав книгу.

2 Настав час платити. Заповнив форму, у якій указав своє **ім'я** (власник картки), **адресу, тип картки** (наприклад, Visa), **строк її дії** та **номер**.

3 Форма (спеціальний скрипт) відіслала ці дані на сервер інтернет-магазину.

4 Сервер переслав їх у спеціальну централізовану базу даних для перевірки, чи дійсні картка й чи відповідає ім'я й адреса, які вказав користувач-покупець у формі, імені й адресі власника.

5 Переконавшись, що з карткою все гаразд, сервер відсилає дані проплати у свій банк на спеціальний процесор, що дозволив проплату й (трохи пізніше) прокредитував інтернет-магазин на вартість книжки й доставки.

6 Одержавши підтвердження проплати від свого банку, сервер послав розпорядження у відділ доставки й підтвердження проплати користувачеві.

7 Пізніше банк інтернет-магазину надішле електронний запит у банк користувача й одержить від нього проплату. А банк користувача надішле користувачу рахунок, який той проплатить.

Бурхливий розвиток електронної комерції в Україні просто неминучий. Тому сприяють її незаперечні переваги й специфічні для України умови. От тільки деякі з них:

- широка доступність;
- низькі початкові витрати;
- майже повна автоматизація торговельного процесу;

- відносно низькі поточні витрати;
- висока окупність;
- увесь світ – світ покупців;
- простота й зручність для покупця;
- рівні можливості для столичного та периферійного бізнесу.

5.2 Електронні гроші

Усе, про що говорилося вище, – це все приклад роботи в Інтернеті з реальними грошима, якими оперують банки. Важко сказати, чи є електронні гроші грішми, оскільки не можна сказати точно, чим вони забезпечені або як платити з них податки. Але переваги перед звичайними грішми очевидні.

Розглянемо детальніше, що це таке – *електронні гроші*, навіщо вони можуть знадобитися рядовому користувачеві Інтернету, і в чому полягають їхні переваги й недоліки порівняно з іншими методами платежів.

На сьогоднішній день у Рунеті активно працюють дві платіжні системи – WebMoney (http://www.webmoney.ru/) та "Яндекс.деньги" (http://money.yandex.ru/). Обидві вони:

- безкоштовні;
- анонімні (не жадають від вас ніяких документів при відкриванні рахунку);
- працюють у режимі реального часу;
- мають дуже подібні можливості в плані оплати тих чи інших послуг.

Прямо обміняти WebMoney на “Яндекс.деньги” або іншу будь-яку валюту не можна, але в Мережі є величезна кількість різних обмінних пунктів, які дають таку можливість. Обмін теж відбувається в режимі реального часу й займає декілька хвилин, так що все, що користувачеві треба буде зробити, – це підібрати обмінний пункт із курсом, який його влаштовує.

Електронні гроші за функціональністю дуже схожі на дійсні. Користувач може їх одержувати, передавати знайомим, платити ними за послуги й товари й т.д., причому, як і у випадку “дійсних грошей”, платежі відбуваються в режимі реального часу. Тобто якщо користувачеві заплатили за щось, то вже через хвилину гроші виявляться в нього на рахунку. Крім цього, зазначимо, що для приймання платежу не потрібно знаходження користувача в Мережі – якщо платіж був зроблений у той час, поки користувач ходив обідати, то, запустивши програму-гаманець, він одержить повідомлення, що на рахунок надійшла така-то сума.

Миттєвість платежів дуже корисна при покупці так званих "цифрових товарів" – усіляких пін-кодів і кодів доступу. Наприклад, якщо Ви збиралися зателефонувати за кордон і раптом виявили, що телефонна картка закінчилася, то Ви не захочете чекати тиждень, поки пройде банківський платіж. Набагато зручніше розрахуватися електронною готівкою й уже через пару хвилин одержати новий код для дзвінків. Аналогічно можна оплачувати доступ до різних послуг, які потрібні Вам не через кілька днів, а прямо зараз. Наприклад, платити за Інтернет, мобільний телефон і т.п. Щодо оперативності проходження платежів з електронними грішми можуть посперечатися тільки кредитні карти, але вони є далеко не у всіх та й не дуже вони пристосовані до Інтернету: украдуть номер кредитки – вистачить турбот .

Друга дуже корисна властивість електронних грошей – **можливість не тільки платити, але й одержувати платежі**. У користувача з'являється можливість взяти гроші в борг (що неможливо, наприклад, при використанні кредиток), одержувати оплату за віддалену роботу (скажімо, якщо ви заробляєте перекладами або розробкою банерів), продавати якісь свої послуги, сервіси або навіть товари.

Зрозуміло, тут теж можна обійтися без електронних грошей, наприклад, завести рахунок у банку, приймати поштові перекази або Western Union.

Але, погодьтеся, це все незручно. Треба ходити в банк і на пошту, стояти в чергах (причому не тільки вам, але й потенційним платникам); чекати, поки пройдуть платежі; платити досить високий комісійний збір. Набагато зручніше просто натиснути кнопку на клавіатурі.

Третя корисна властивість – **можливість проведення мікроплатежів** (порядку 1-2 гривень, що виявляється просто невигідним при використанні інших платіжних систем через високу комісію). Це, щоправда, більше потрібно різним интернет-сервісам, які розраховані на велику кількість користувачів. Поки що такі системи не дуже поширені.

Методи обміну дійсних грошей на електронні й навпаки теж досить добре відпрацьовані й докладно розписані на сайтах платіжних систем. Можна перерахувати гроші з банківського рахунку й на рахунок, відправити їх собі поштовим переказом, одержати наявні в обмінному пункті й т.д. Для введення грошей у систему зручно використати платіжні карти – багато фірм дозволяють викликати кур'єра додому, що приїде й продасть вам необхідну картку. Усе, що вам залишиться після цього, – набрати код картки у вашому гаманці, і сума буде зарахована на рахунок.

Тепер про те, **куди можна витратити накопичені вами електронні гроші**. Серед серйозних сервісів можна виділити:

- оплату комунальних послуг;
- покупку телефонних карток, доступу в Інтернет;
- оплату за мобільний телефон;
- покупки в інтернет-магазинах (продукти, квіти, книжки, комп'ютери, програми до них; причому при оплаті електронними грішми в більшості магазинів дадуть знижку в 5-10%).

Серед менш серйозних послуг:

- можна пограти у лотереї, на біржі, купити рекламу на сайтах;
- при деякій наполегливості ви зможете заплатити електронними грошима й у тих магазинах, де офіційно вони не приймаються: треба тільки поспілкуватися з менеджерами – майже в усіх є електронні гаманці.

5.3 WebMoney Transfer

WebMoney Transfer – це облікова система, за допомогою якої всі бажаючі можуть обмінюватися універсальними обліковими одиницями: титульними знаками WebMoney (WM). Система відкрита для вільного використання всіма бажаючими в будь-якій точці земної кулі.

Webmoney Transfer має універсальну гнучку структуру й дає можливість будь-якому користувачеві мережі Інтернет здійснювати безпечні розрахунки готівкою в реальному часі.

Клієнтами системи **є продавці** й **покупці** товарів і послуг. З одного боку, це Web-магазини, з іншого – користувачі Інтернету, що не спроможні або не бажають використати альтернативні методи розрахунків (кредитні карти й т.п.) через тривалість трансакцій, низької безпеки, ризику повернення зроблених платежів і т.п.

За допомогою WebMoney Transfer можна:

- здійснювати миттєві безвідкличні трансакції, пов'язані з передачею майнових прав на будь-які товари й послуги;
- створювати власні web-сервіси й мережеві підприємства;
- проводити операції з іншими учасниками;
- випускати й обслуговувати власні розрахункові інструменти.

Трансакційним засобом у системі є титульні знаки WebMoney (WM) декількох типів, що зберігаються на електронних гаманцях їхніх власників:

WMR – еквівалент RUR – на R-гаманцях,

WME – еквівалент EUR – на E-гаманцях,

WMZ – еквівалент USD – на Z-гаманцях,

WM-C й WM-D – еквівалент USD для кредитних операцій – на C- і D-гаманцях.

При переказуванні коштів використовуються однотипні гаманці, а обмін WM-R на WM-Z здійснюється в ***обмінних пунктах.***

Реєстрація в системі, а також керування коштами здійснюються за допомогою клієнтської програми ***WM KEEPER***.

За допомогою програми WM KEEPER ви можете:

- здійснювати миттєві розрахунки в WM з іншими клієнтами системи;
- оплачувати товари й послуги в Мережі;
- конвертувати WM в активи з переведенням на банківські рахунки;
- обговорювати з партнерами умови торговельної угоди за вбудованою у програму WM KEEPER захищеною системою обміну повідомленнями.

Одержати WebMoney на гаманець ви можете:

- банківським або поштовим переказом на розрахунковий рахунок одного з офіційних агентів системи (сума переказу буде автоматично конвертована в WM і зарахована на зазначений вами гаманець).

Кошти, що зберігаються в гаманці користувача WebMoney, перебувають у його повному розпорядженні й у будь-який момент – цілодобово й щодня – можуть бути використані для розрахунків. За необхідності користувач зможе зняти WebMoney з гаманця й перевести на зазначений ним банківський рахунок з одночасною конвертацією у відповідну валюту.

Безпека трансакцій

При використанні WebMoney Transfer кошти передаються між користувачами через Інтернет. Відкрита архітектура Інтернет вимагає строгих засобів безпеки для того, щоб уникнути спроб перехоплення інформації про торговельну угоду.

Розглянемо декілька засобів безпеки, які застосовані в WebMoney Transfer.

Для входу в програму WM Keeper необхідно

1 Знання унікального 12-значного WM-ідентифікатора користувача. *WM-ідентифікатор* (ім'я користувача в системі) генерується автоматично, унікальний для кожної реєстрації учасника й необхідний для входу в програму WM Keeper і проведення угод у системі WebMoney Transfer.

2 Знання особистого пароля користувача. *Пароль* – призначається користувачем і необхідний для входу в програму WM Keeper і здійснення угод у системі WebMoney Transfer

3 Знання місця розміщення в пам'яті комп'ютера файлів із секретним ключем і гаманцями.

Всі трансакції в системі передаються в закодованому вигляді, з використанням алгоритму захисту інформації подібного RSA з довжиною ключа більше 1024 біт. Для кожного сеансу використовуються унікальні сеансові ключі. Тому протягом сеансу (часу здійснення трансакції) ніхто, крім вас, не має можливості визначити призначення платежу і його суму.

Ніхто не зможе зробити ніяких грошових операцій, ґрунтуючись на реквізитах Ваших минулих угод (цієї можливості позбавлена, наприклад, система оплати за допомогою кредитних

карт). Для кожної угоди використовуються унікальні реквізити, і спроба використати їх удруге негайно відслідковується і вони знищуються.

Система WebMoney Transfer забезпечує стійкість стосовно переривання зв'язку. Якщо будь-яка операція в системі не була успішно завершена через переривання зв'язку, то система не враховує дану операцію.

Таким чином, клієнт WebMoney Transfer, що дотримується елементарних правил щодо схоронності своїх WM-гаманців, WM-ідентифікатора, пароля й секретного ключа, може бути впевнений у безпеці керування своїми коштами.

Реєстрація в системі WebMoney

Для того щоб стати учасником WebMoney Transfer, необхідно встановити на свій комп'ютер клієнтське програмне забезпечення WebMoney Keeper. Цю програму у вигляді інсталяційного архіву, що саморозпаковується, можна безкоштовно одержати на офіційному сайті WebMoney.

WebMoney Keeper дозволяє виконувати такі операції:

- створювати й видаляти WM-гаманці (кількість створюваних гаманців не обмежена);
- виконувати конвертацію валюти;
- здійснювати миттєві розрахунки в WebMoney з іншими учасниками системи;
- одержувати WebMoney, відправлені іншим учасником;
- переводити WebMoney на банківські рахунки;
- обговорювати з передбачуваними партнерами умови торговельної угоди за убудованою в WM Keeper захищеною системою обміну повідомленнями.

WebMoney Keeper є вільно поширеним програмним продуктом, у якому реалізовані універсальні рішення як для покупців, так і для продавців товарів і послуг у Мережі.

У процесі інсталяції клієнтського програмного забезпечення учасникові системи привласнюється унікальний 12-значний ідентифікатор, необхідний для запуску програми й роботи в системі.

Далі користувач самостійно призначає пароль і визначає місце в пам'яті комп'ютера для зберігання файлів із секретним ключем і гаманцями.

Таким чином, для наступного входу в систему необхідно вказувати ***WM-ідентифікатор, особистий пароль і розміщення в пам'яті файлів із ключем і гаманцями***.

Після установки WM Keeper автоматично створює один Z-гаманець. Інші гаманці створюються учасником самостійно за необхідності.

Порада. Зробіть копію, скопіюйте на дискету, а краще на дві, файлів із ключами keys.kwm і гаманцями purse.pwm, які були створені й перебувають у папці «WebMoney». Також скопіюйте й збережіть файл із папки «WebMoney», що закінчується назвою (.pwm.safe). І ще варто зберегти два сертифікати WebMoney – вони заходяться там само.

Не рекомендується губити ці дискети, тому що, по-перше, вони можуть потрапити в чужі руки, які одержать доступ до рахунку власника, а, по-друге, власник сам втратить цей доступ.

Висновки

Можна сказати, що електронні гроші – це дуже зручний засіб для миттєвих платежів, у тому числі мікроплатежів. Якщо користувач в Інтернеті нічого не купує й ніякої діяльності не веде, то вони йому навряд чи знадобляться. Якщо ж він ледача людина і бажає по-справжньому жити в XXI-му столітті, то за допомогою електронних грошей зможе організувати своє життя так, що йому взагалі не знадобиться виходити із квартири. Ну а якщо Ви ведете або плануєте якийсь бізнес у Мережі, то електронні гроші стають майже незамінними – не секрет, що кредитки в Україні не дуже поширені, а тут з'являється можливість миттєво одержати платіж звідки завгодно. Ну й, природно, розплачуватися з Вашими співробітниками теж можна електронними грішми, особливо якщо вони десь далеко, що не рідкість для нашого інформаційного століття.

Контрольні запитання

1 Як влаштована система електронних платежів за кредитними картками?

2 Назвіть переваги використання електронних грошей в Інтернеті.

3 Хто є клієнтами системи WebMoney Transfer?

4 Що є трансакційним засобом у системі WebMoney Transfer?

5 Назвіть типи гаманців, які використовуються в системі WebMoney Transfer.

6 Які засоби безпеки застосовані в системі WebMoney Transfer?

Список літератури

1. Б.Нанс. Компьютерные сети: пер. с англ. – М.: БИНОМ, 1996. – 400 с.
2. История развития Интернет (http://www.actr.ru/pubs/RIIIP/book1/doc3.html).
3. Золотов С. Протоколы Internet –СПб.:BVH – Санкт-Петербург, 1998. –304 с.
4. Божко О.І., Любчак В.О. Сервіси Інтернет: навчальний посібник. – Суми: Сумду, 2000. – 94 с.
5. Г.Реймс. 10 минут на урок. Internet World Wide Web: Пер с англ.. – К.; М.; СПб: Издательский дом “Вильямс” , 1998. –160 с.
6. The World Wide Web Consortium, http://www.w3.org/
7. W3C Project, http://www.w3.org/pub/WWW
8. National Centre for Supercomputing Applications (NCSA), http://www.ncsa.uiuc.edu/
9. Жадаев А.Г. Самоучитель HTML 4. – К.: Юниор, 2001. – 296 с.