МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«УЖГОРОДСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ»
ІНЖЕНЕРНО-ТЕХНІЧНИЙ ФАКУЛЬТЕТ
КАФЕДРА КОМП'ЮТЕРНИХ СИСТЕМ ТА МЕРЕЖ

МЕТОДИЧНІ ВКАЗІВКИ І ЗАВДАННЯ

ДО ЛАБОРАТОРНИХ РОБІТ

МЕРЕЖНІ ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

для студентів спеціальності 123 «Комп'ютерна інженерія»

Ужгород-2021

Методичні вказівки і завдання до лабораторних робіт з курсу «Мережні інформаційні технології для студентів спеціальності 123 – «Комп'ютерна інженерія»

Укладач: Безвершенко Є.І., ст.викладач.

Рецензент: Гузій М.М., к.т.н., доцент.

Відповідальний за випуск: Туряниця І.І., канд. фіз.-мат. наук, доцент,

декан інженерно-технічного факультету.

Методичні вказівки розглянуто та схвалено на засіданні кафедри комп'ютерних систем та мереж, протокол №11 від 11 травня 2021р. та методичної комісії інженерно-технічного факультету, протокол №4 від 24 травня 2021р.

Лабораторна робота №1

Тема: Визначення реальної пропускної здатності мережі

Мета роботи:

Набути навичок визначення реальної пропускної здатності мережі.

Теоретичні відомості

Реальна пропускна спроможність мережі залежить від безлічі чинників і часто є випадковим параметром. Тільки у синхронних мережах передачі даних, використовуваних в цифровій телефонії, пропускна спроможність мережі є фіксованою величиною, проте реально за наявності помилок передачі пропускна спроможність може бути що нижчою.

Розглянемо чинники, що впливають на пропускну спроможність мережі *Ethernet* і подібних мереж з комутацією фреймів.

Чинники, що впливають на пропускну здатність мережі.

Найбільшого поширення в мережах передачі даних набули стандарти, що не підтримують гарантії якості передачі. Яскравим прикладом може бути мережа *Ethernet*, в якій немає ніяких механізмів гарантування швидкості передачі, і, крім того, немає механізмів гарантування часу очікування перед передачею чергового пакету і механізмів уникнення конфліктів при передачі. Всі учасники обміну в мережі працюють у випадковому режимі, отже, і параметри, що описують пропускну спроможність, будуть випадковими величинами. Таким чином, можна привести лише статистично усереднені параметри мережі. Розглянемо основні чинники, що впливають на продуктивність мережі.

Фізична гранична швидкість передачі.

Найбільш поширеною на сьогодні є мережа *Ethernet* 100 Мбіт/с на базі мідної витої пари. Спробуємо приблизно визначити граничну швидкість в цій мережі за даними з використанням протоколу *TCP*.

100000000 біт/с/8 = 12500000 байт/с. - це гранична максимальна швидкість в дуплексному каналі, що не враховує тимчасового зазору між пакетами і службовим байтом заголовків. Втрати на зазори складають приблизно 5%, тобто реальна

швидкість складає 12500000 байт/с * 0.95 = 11875000 байт/с. Крім того, на кожних 1500 байт інформації (стандартний *MTU*), інкапсульованої у фрейм Ethernet, доводиться 16 байтів заголовка *Ethernet*. Далі, в 1500 байтах міститься як мінімум один IP заголовок і, швидше за все, один *TCP* заголовок, тобто 20+20=40 байт, тобто 56/1516*100%=3.7%. Отже, в ідеальному випадку отримуємо 11875000*0.963=11435625 байт/с за даними, тобто 11.4 Мбайт/с. Це, природно, для випадку, коли в мережі «подорожують» тільки пакети максимального розміру. Проте, якщо розмір пакету менший, то на нього доводиться більше в процентному відношенні службових даних і реальна швидкість за даними буде нижче обчисленої нами величини.

Рівень помилок в каналі передачі даних.

При певному рівні помилок працездатність мережі не втрачається, а відбувається зниження реальної швидкості передачі даних, оскільки часто при передачі даних використовується потоковий протокол *TCP*, що забезпечує повторну передачу втрачених даних. Проте, чим вище рівень помилок в каналі, тим більше часу втрачається на повторну передачу і, врешті-решт, пропускна спроможність падає до нуля. Для пакетних протоколів (*UDP*, *ICMP*, і подібні) втрати в каналі є катастрофічними, оскільки не використовуються ніякі механізми управління потоком даних. Для синхронних мереж рівень помилок в каналі 10^{-7} є нормальним, 10^{-6} – аварійним. Для *TCP/IP* мереж вимоги дещо нижчі. За нормальний вважається рівень помилок 10^{-4}, критичний 10^{-3}, аварійний – 10^{-2}.

Завантаження мережі службовими даними.

Англійський термін для даного чинника звучить як *network overhead*. Даний чинник можна розділити на дві складові: перша – довжина службових заголовків пакету даних. Це впливає на пропускну спроможність безпосередньо, як видно з приведених вище розрахунків граничної швидкості передачі даних. Наприклад, при передачі інтерактивних даних по протоколу *telnet* кожна натиснута на терміналі клавіша передається окремим пакетом, унаслідок чого реальна пропускна спроможність за даними складає менше 1%. Друга складова – дані різних службових мережевих протоколів. Наприклад, мережі на основі протоколу *NETBIOS* дуже активно використовують широкомовні пакти (*broadcast*) для анонсування і запиту

доступності мережевих сервісів. Якщо не приймати додаткових заходів по обмеженню розповсюдження *broadcast* пакетів, вони можуть «з'їсти» до 30% смуги пропускання мережі 10 Мбіт/с. Даний чинник визначається як типом сервісів, використовуваним в мережі, так і кількістю робочих станцій в сегменті мережі. Саме тому не рекомендується використовувати більше 200 робочих станцій в комутованому сегменті 100 Мбіт/с і не рекомендується використовувати протокол *NETBIOS* в «чистому вигляді». Рекомендується використовувати стек протоколів *TCP/IP* і до нього прив'язувати служби мереж *Windows*.

У мережах із загальним середовищем передачі, наприклад, в *Ethernet* на коаксіалі або на концентраторах (*hub*) високе завантаження мережі викликає конфлікти доступу до середовища передачі (*collisions*), тобто ситуації, коли одночасно починають передавати декілька пристроїв. При цьому пакет даних не втрачається, а затримується в буфері передачі, що призводить до «м'якого» зниження швидкості передачі до рівня завантаження близько 50% від загальної смуги пропускання. При більшому рівні завантаження може з'явитися істотна втрата пакетів за рахунок переповнення буферів передачі. Природно, це відбувається тільки у разі використання протоколів без механізму управління потоком. Наприклад, механізм потокових сокетів просто блокує процес, дані якого не встигли покинути буфер, запобігаючи таким чином переповнюванню буфера передачі.

Загальне завантаження мережі.

Аналізуючи даний чинник, будь-який канал зв'язку зручно розглядати як деяку «трубу» з певною пропускною спроможністю (діаметр на швидкість розповсюдження), а пакети даних – як деяка зерниста сипка речовина. Природно, що через трубу може прокидатися цілком конкретна кількість речовини за одиницю часу. Така аналогія цілком справедлива для некомутованих мереж *Ethernet*, для синхронних і асинхронних каналів точка-точка, для портів маршрутизаторів. Для комутованих мереж ситуація дещо інша. Якщо продуктивність комутуючої матриці дозволяє працювати всім портам комутатора із «швидкістю дроту», то трафік між двома окремими портами абсолютно не впливає на роботу останніх портів, інакше досягши граничної пропускної здатності матриці швидкість знижуватиметься для всіх портів комутатора.

Для мереж з маршрутизаторами необхідно використовувати хоч би мінімальні механізми управління потоком при передачі по базовому протоколу (наприклад, *IP*) інакше втрата пакетів неминуча. Уявіть собі ситуацію, коли у маршрутизатора один порт 100 Мбіт/с, а інший – 64 Кбіт/с. Дані, що приходять з швидкісного порту ніяк не зможуть в тому ж темпі передаватися в другий порт, отже маршрутизатор повинен передати джерелу пакетів повідомлення, що буфери заповнені і джерело даних повинне тимчасово припинити передачу. Такі мінімальні функції управління потоком *IP* пакетів закладені в протокол *ICMP*. Отже, пропускна спроможність мереж з маршрутом визначається пропускною спроможністю відповідного порту маршрутизатора і його завантаженням.

Затримки в мережі.

Затримки в мережі, на перший погляд, не повинні впливати на пропускну здатність, проте це справедливо тільки для протоколів без управління потоком. У разі, коли необхідно чекати підтвердження отримання певної порції даних, перш ніж передавати наступну порцію, затримки істотно впливають на швидкість передачі даних між парою потокових сокетів. Наприклад, супутниковий канал з пропускною спроможністю 10 Мбіт/с і затримкою 600 мс для пари сокетів дасть всього біля 1 Мбіт/с. Звичайно, якщо канал використовується декількома сокетами, то він може використовуватися на всі 100%.

Крім того, затримки вище 250 мс помітні користувачам інтерактивних застосувань і істотно погіршують суб'єктивну якість голосового зв'язку.

Приведені вище чинники є основними, але список не є вичерпним. Так, часто, маршрутизатори можуть не справлятися з повним об'ємом трафіку за рахунок недостатньої продуктивності шини, пам'яті і процесора.

Вимірювання пропускної спроможності мережі.

Оскільки пропускна спроможність мережі залежить від великої кількості чинників, більшість з яких носить випадковий характер, точний розрахунок пропускної спроможності вельми складний. Тому набагато простіше зміряти реальну пропускну спроможність мережі експериментально. Тут слід враховувати, що пропускна спроможність залежить ще і від типу використовуваного протоколу. У *TCP/IP* мережах необхідно окремо вимірювати пропускну спроможність по двох

транспортних протоколах – *TCP* і *UDP*. Варто так само відзначити, що не завжди доступне управління і навіть оцінка завантаження всіма вузлами мережі по шляху розповсюдження пакетів.

Насамперед необхідно оцінити затримки в каналі зв'язку і рівень втрати пакетів за допомогою команди *ping*.

У другу чергу слід визначитися, який інтервал усереднювання нас цікавить, тобто для якого типу сервісу вимірюватиметься швидкість передачі. В межах даної роботи вважатимемо, що нас цікавить потокова передача великого об'єму даних.

Для оцінки швидкості з'єднання точка-точка по протоколу *TCP* найпростіше використовувати сервіс *FTP*. Для цього на одному хості запускається сервер *ftpd*, а на іншому – будь-який клієнт *ftp*, який оцінює середню швидкість передачі, наприклад ***mc*** *(Midnight Commander)*, за допомогою клієнта необхідно здійснити закачування файлу такого об'єму, щоб встигнути прослідкувати процес закачування близько 1 хвилини. Для точнішого вимірювання пропускної спроможності існують спеціальні програми, наприклад *tcpblast*. Потрібно запустити цю програму на обох кінцях з'єднання.

Для оцінки швидкості пакетної передачі даних можна скористатися командою ***ping***. У цієї команди є ключ **-i**, задаючий інтервал між пакетами і ключ **-s**, задаючий розмір пакету. Маніпулюючи цими ключами можна завантажити канал зв'язку до «затоплення» (*flood*), тобто до ситуації, коли пакети почнуть втрачатися. Це і є гранична пропускна спроможність мережі на даний момент часу. Наприклад, команда ***ping*** **ім'я -i 0.01 -s 125000** завантажить 10Мбіт/с більш ніж на 100%.

По параметру TTL який повертає луну (exo) можна зрозуміти, на якому по рахунку від джерела запитів маршрутизаторі відбувається втрата пакетів.

Шлях проходження пакетів можна подивитися командою *traceroute*.

Способи гарантування заданої пропускній спроможності мережі.

У загальному випадку стек протоколів *TCP/IP v.4*, який використовується на сьогоднішній день, практично не містить засобів гарантування якості сервісу (*Quality of Service, QoS*). Мережі типу *Ethernet* теж не передбачають механізму забезпечення *QoS*. Проте, в *IP* мережах можливо обмежити швидкість передачі, регулюючи швидкість відходу даних з буфера передачі маршрутизатора по якому-

небудь критерію (*IP* адреси, поле *TOS* і тому подібне). Обмежити можна тільки швидкість передачі, тому гарантувати смугу можна тільки між безпосередньо з'єднаними маршрутизаторами. Природно, загальна смуга пропускання повинна дозволяти це зробити, тобто сума смуг, розділених по групі критеріїв не повинна перевищувати граничної швидкості в каналі.

Практична частина:

Робота виконується групою з 3-х чоловік. Двоє займаються відправкою тестових послідовностей, третій – вимірює затримки, втрати пакетів і рівень помилок у фізичному каналі зв'язку.

Для виконання роботи необхідні права суперкористувача на всіх трьох робочих станціях. (для *UNIX*)

Даний варіант роботи повинен виконуватися в мережі з комутатором. В мережі з концентратором або з іншим загальним середовищем передачі результати будуть недостовірними за рахунок взаємного впливу.

Далі по тексту ми посилатимемося на адреси трьох хостів. *Host-1*, *Host-2* – машини, з яких посилаються тестові послідовності, *Host-3* – хост-«жертва», над яким проводиться експеримент.

1. Перевірте затримку в каналі зв'язку і рівень втрат при не завантаженій мережі. Для цього на *Host-1* і *Host-2* запустіть команду:

sh>ping Host-3 і поспостерігайте за її роботою пару хвилин, після чого перервіть її виконання натисненням *CTRL-C*. Запустіть так само команду ***sh>/sbin/ifconfig*** і зафіксуйте кількість колізій і помилок на інтерфейсі *eth0*.

2. Запустіть сервіс *ftpd* на *Host-3* виконанням наступної команди:

sh> /etc/rc.d/init.d/vsftpd start Якщо команда не виконалася, перевірте, чи встановлений пакет *vsftp* командою ***sh>rpm -qa |grep ftp*** і при необхідності встановіть потрібний пакет. У директорії */var/ftp/pub* помістіть файл відповідного об'єму (100Мб). На *Host-1* запустіть в терміналі оболонку ***mc*** і введіть наступну команду:

mc>cd *ftp://Host-3* перейдіть в директорію *pub* і запустіть перекачування файлу. У іншому терміналі запустіть команду ***sh>ping Host-3*** і зупиніть її в момент перед закінченням перекачування файлу. Під час перекачування зафіксуйте

швидкість, яку відображає оболонка *mc*. Для мережі 100Мбіт/с це трохи більше 10 Мбайт/с для повнодуплексної карти і портів комутатора. Зафіксуйте результати виконання команди *ping*. Ви повинні побачити, що втрати даних в каналі не спостерігається, але затримка зросла приблизно в 10 разів. Виконайте команду *sh>/sbin/ifconfig* на *Host-1 і Host-3* і зафіксуйте кількість колізій і помилок на інтерфейсі *eth0*. Кількість колізій може трохи вирости, але помилок і втрат спостерігатися не повинно.

3. Повторіть п. 2 одночасно на *Host-1* і *Host-2*. Зафіксуйте швидкість перекачування файлу на двох хостах і сумарну швидкість. Кількість колізій на *Host-3* може зрости значно, але помилок з'являтися не повинно.

4. Обчисліть необхідний для 100% завантаження порту розмір пакету і інтервал посилки пакетів. Запустіть команду ***ping*** з цими параметрами: ***sh>ping -i інтервал -s розмір Host-3*** одночасно з *Host-1 і Host-2*. У окремому терміналі Запустіть «пробник»: ***sh>ping Host-3***. Очевидно, що ви завантажили порт на *Host-3* на 200%. Після закінчення пари хвилин перервіть «пробник», потім перервіть тестовий ***ping***. Зафіксуйте відсоток втрати пакетів «пробників» і середню затримку в каналі. Зафіксуйте приріст колізій і помилок на всіх трьох хостах. Зрозуміло, що при 200% завантажень катастрофічно виросте кількість втрат пакетів, колізій і помилок.

5. Повторіть п. 4 для 190%, 180% ... 10% завантажень, запускаючи тестовий ***ping*** з відповідними параметрами. Зафіксуйте результати.

6. Побудуйте графічну залежність відсотка втрат пакетів «пробника» і затримки від завантаження мережі. Поясніть отримані результати.

7. Підберіть відсоток завантаження мережі командою ***ping***, при якому перекачування файлу відповідно до п.2 істотно сповільнюється і зупиняється.

Зміст звіту.

Звіт повинен містити роздрук результатів експерименту по п.1-п.7 і висновки по кожному пункту, а так само графіки по п.6 з поясненнями.

Лабораторна робота №2

Тема: Дослідження основних показників мережі *ISDN*

Мета роботи:

1 Вивчити призначення і основні показники мереж *ISDN*.

2 Отримати навики побудови моделі мережі *ISDN* з використанням середовища моделювання *NetCracker*.

3 Досліджувати характеристики мережі *ISDN* за допомогою програми *NetCracker*.

Теоретичні відомості

Цифрові мережі зв'язку з комплексними послугами (*Integrated Services Digital Network, ISDN*) з'явилися в 1970-х роках для передачі в цифровому вигляді мовних сигналів, даних, графіки і відеосигналів. У 1984 і 1988 роках вони були стандартизовані союзом *ITU-T*. Ці стандарти описували вузькосмугові мережі *ISDN (N-ISDN)*.

ISDN – це стандарт цифрових телекомунікацій, який в даний час передбачає передачу призначених для користувача даних на швидкості до 1,536 Мбіт/с і має теоретичну межу в 622 Мбіт/с. Клієнти, яким потрібно отримати послуги *ISDN* для зв'язку з деякою точкою, можуть отримати цифрову *ISDN*-лінію з одноканальним обслуговуванням від своєї регіональної телефонної компанії. Одноканальна служба дозволяє кінцевому користувачеві підключати до лінії декілька пристроїв (наприклад, факс, комп'ютер і цифровий телефон). Деякі компанії дозволяють підключати до восьми пристроїв (максимум для даного типу *ISDN*-служб). Організації, які через глобальну *ISDN*-мережу сполучають між собою локальні мережі, зазвичай використовують для цього *T*-лінії. Мережі *ISDN* надають різні послуги, серед яких наступні:

• забезпечення зв'язку між локальними мережами;

• забезпечення роботи домашніх офісів;

• віддалена архівація і відновлення настільних комп'ютерних систем;

• підключення приватної телефонної системи до регіональної телефонної компанії;

• передача великих файлів зображень і даних;

• забезпечення роботи відео- і мультимедіа-додатків, що працюють в декількох локальних мережах.

Вузькосмугова *ISDN*-мережа (*N-ISDN*) підтримує інтерфейси двох типів: інтерфейс базового рівня (*basic rate interface, BRI*) та інтерфейс основного рівня (*primary rate interface, PRI*).

У *ISDN*-мережі з *інтерфейсом базового рівня* (*BRI*) використовується різновид множинного доступу з ущільненням каналів (який називається мультиплексуванням з розділенням часу). Така мережа має загальну пропускну спроможність, рівну 144 Кбіт/с. Інтерфейс базового рівня складається з трьох каналів: двох несучих каналів для передачі даних, мови і графіки (*bearer, B*), із швидкістю 64 Кбіт/с і третього – *D*-каналу (*Delta*, або *Demand* (запит)), що забезпечує швидкість 16 Кбіт/с і використовується для передачі сигналів управління комунікаціями, комутації пакетів і верифікації кредитних карт. Головне завдання *D*-каналу – забезпечити проходження і зняття *ISDN*-виклику, а також початок і закінчення сеансу передачі даних.

Інтерфейс базового рівня застосовується для виконання наступних завдань:

• забезпечення зв'язку локальних мереж;

• проведення відеоконференцій;

• підключення до постачальника послуг Інтернету;

• високошвидкісний обмін даними з домашніми офісами.

Декілька *BRI*-каналів можна зв'язати між собою (згрупувати) для забезпечення комунікацій з ще вищою швидкістю. Наприклад, два *B*-канали 64-кілобіт в одній *BRI*-лінії можна згрупувати і отримати з'єднання з реальною швидкістю передачі, рівною 128 Кбіт/с. При додаванні *D*-каналу 16-кілобіта плюс 48 Кбіт/с для супроводу і синхронізації можна отримати загальну швидкість в 192 Кбіт/с. Можна згрупувати три *BRI*-лінії, що містять 64-кілобітні *B*-канали, і отримати загальну реальну швидкість передачі даних, рівну 384 Кбіт/с.

Клієнти підключаються до *ISDN*-мереж з інтерфейсом базового рівня (*BRI*) за допомогою 2-парного телефонного кабелю на основі витої пари.

ISDN-мережі з *інтерфейсом основного рівня* (*PRI*) забезпечують вищу в порівнянні з *BRI ISDN* швидкість передачі даних, при цьому сумарна смуга пропускання комутованих даних досягає 1,536 Мбіт/с. У США і Японії інтерфейс основного рівня складається з 23 *B*-каналів по 64-кілобіт і одного *D*-каналу 64-кілобіта для передачі службових сигналів і комутації пакетів. Європейські мережі *PRI ISDN* мають 30 *B*-каналів 64-кілобіт і один канал 64-кілобіт для службових сигналів або комутації. *PRI*-інтерфейси використовуються для зв'язку локальних мереж і постачальників послуг Інтернету, а також для проведення відеоконференцій і (у корпоративних мережах) для підключення «домашніх» працівників, що мають *ISDN*-доступ.

Для підключення клієнтів до *PRI*-інтерфейсу використовується мультиплексор або приватна телефонна система, а також група з 24 каналів, яка називається магістраллю (транком). Мультиплексор зазвичай використовується тоді, коли *PRI ISDN* забезпечує зв'язок між локальними мережами, для постачальника послуг Інтернету він може бути зовнішнім пристроєм або модулем в маршрутизаторі. Приватна телефонна система використовується для організації відеоконференцій і центрів обробки телефонних викликів, що мають бази абонентських номерів, пов'язаних з призначеними для користувача службами. Така телефонна система повинна мати можливість підключення до *PRI ISDN*. У одній точці можна використовувати декілька *PRI*-магістралей, і в цьому випадку кількість *D*-каналів, що використовуються для передачі службових сигналів, можна скоротити. Наприклад, якщо компанія має п'ять *PRI*-магістралей для вирішення комунікаційних завдань, то вона може придбати тільки один або два *D*-канали (другий *D*-канал може використовуватися як резервний у разі відмови першого каналу).

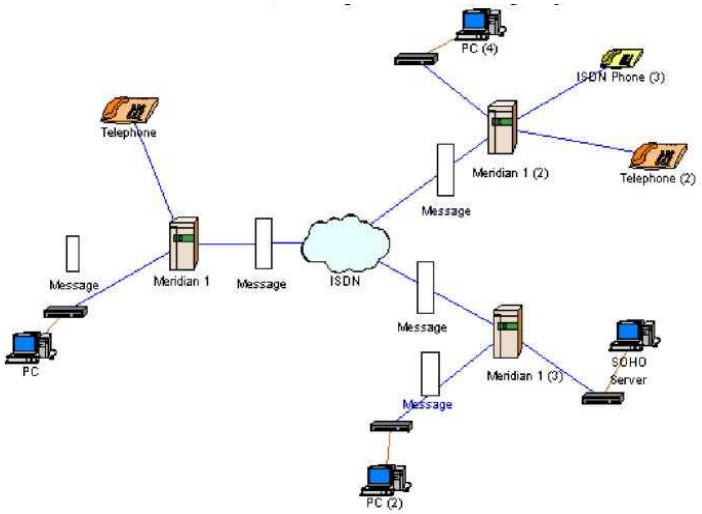
Розвиток високошвидкісних мереж привів до появи *широкосмугових* ISDN-*мереж* (*broadband ISDN, B-ISDN*). Ця технологія, що розвивається, призначена для забезпечення сумісності з мережами *ATM* і *SONET*.

Широкосмугові *ISDN*-мережі призначені для комунікацій з швидкостями від 155 Мбіт/с до 1 Гбіт/с (і вище) по оптоволоконному кабелю. У них застосовується не комутація пакетів, а комутація осередків.

Варіанти завдань.

№ варіанту	К-сть компютерів	К-сть телефонів	Тип каналу	Тип трафіку
1.	5	2	*Station Line*	*LAN peer-to-peer traffic, E-Mail (SMTP)*
2.	8	4	*Trunk*	*File server's client, Small Office*
3.	9	6	*Trunk*	*Small office peer-to-peer, SQL server's client*
4.	4	8	*Station Line*	*E-mail (POP), Database*
5.	7	3	*Trunk*	*Small interLAN traffic, Database*
6.	10	5	*Station Line*	*E-mail (POP), HTTP*
7.	12	9	*Trunk*	*HTTP, E-Mail (SMTP)*

Як вихідні дані пропонується структура мережі з використанням технології *ISDN*, зображена на малюнку 1.



Малюнок 1 - Структура досліджуваної мережі *ISDN*

Практична частина:

Запустіть програму *NetCracker* і створіть проект у вигляді фрагмента мережі *ISDN*, представленого на малюнку 1. У цій моделі використовується три АТС *Meridian* 1, які сполучені між собою за допомогою хмари *ISDN*. До АТС підключені

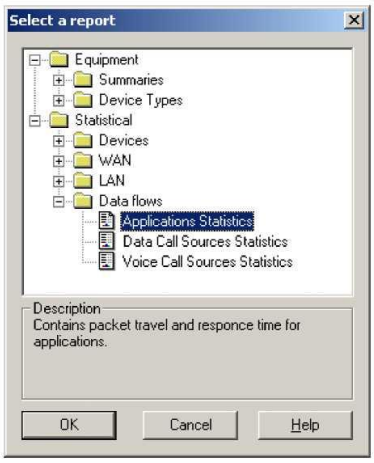
як телефони, так і комп'ютери, оснащені зовнішніми модемами *ISDN*. Модеми підключені до АТС по інтерфейсу *ISDN BRI* і забезпечують швидкість обміну інформацією 128 Кбіт/сек. АТС підключені до хмари *ISDN* також за допомогою інтерфейсу *BRI*. Зроблено це для того, щоб показати, як поводиться мережа *ISDN* при високій завантаженості каналу зв'язку.

Виконайте наступні завдання:

Завдання 1.

На першому етапі в моделі встановлений лише один потік даних - від *PC* до *SOHO Server*. Запустіть модель.

Для перегляду характеристик потоку даних (часу проходження пакету і часу відгуку) натискайте на клавіатурі *Ctrl+Shift+W*. З'явиться наступне діалогове вікно:



В ньому виберіть звіт *Data Flows/Application - Statistics* для перегляду статистики. З'явиться звіт наступного вигляду:

Applications Statistics
09.01.2004

Name	Source	Destination	Travel time	Response time
HTTP client	PC	SOHO Server	96.00 ms	313.00 ms

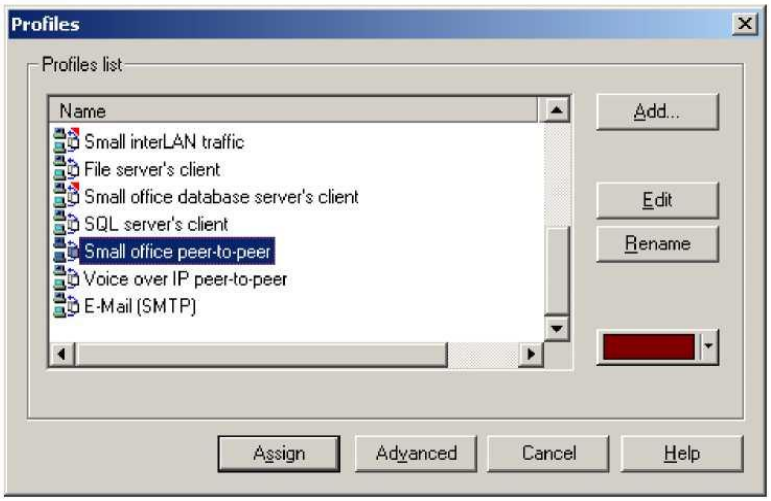
У звіті в першому стовпці вказується назва потоку даних, в другому і третьому - комп'ютер, що ініціює потік даних (відправник) і комп'ютер, що приймає потік (одержувач). Четверта колонка вказує час доставки пакету. П'ята - час відгуку (відповіді) мережі на запит. Випишіть ці значення в звіт про лабораторну роботу для подальшого аналізу.

В даному випадку мережа навантажена трохи, про що можна судити по індикаторах завантаження на моделі.

Для збільшення завантаженості мережі додамо потік даних від *PC*(2) до *PC*(4).

Для цього на панелі інструментів натискуйте кнопку додавання потоку:

Клацніть послідовно на комп'ютерах *PC*(2) і *PC*(4). З'явиться діалогове вікно вибору типу трафіку.



Виберіть тип трафіку *Small office peer-to-peer* і натисніть *Assign*.

Запустіть модель і знову прогляньте звіт про статистичне моделювання для першого потоку даних. Занесіть дані в звіт про роботу. Зверніть увагу, як змінилися час доставки пакету і час відгуку.

Аналогічно додайте зворотний потік даних від *PC*(4) до *PC*(2), але тип трафіку виберіть *Lan peer-to-peer*. Поглянете, як збільшується завантаження мережі і міняється час доставки пакетів і час відгуку мережі.

Занесіть дані в таблицю 1 звіту про роботу.

Таблиця 1 - Результати досліджень середнього навантаження і середнього часу доставки від інтенсивності комп'ютерного трафіку в мережі

№ дослідження	1	2
Середнє завантаження мережі, Мбайт/с		
Середній час доставки, с		
Час відгуку, с		

Зробіть висновки по дослідженню залежності характеристик мережі від завантаженості мережі.

Завдання 2.

Відомо, що основне призначення мереж *ISDN* - передача голосу. При цьому до мереж *ISDN* можуть бути підключені як звичайні телефонні апарати, так і цифрові.

Приведіть модель у початковий стан, відповідний випадку, коли в моделі встановлений лише один потік даних - від *PC* до *SOHO Server*. На моделі додайте голосові виклики між *ISDN* телефоном і звичайним телефоном. Для цього натискуйте кнопку додавання голосових викликів

Потім послідовно клацніть на *ISDN*-телефоні і на звичайному телефоні. З'явиться діалогове вікно, аналогічне вікну вибору типу трафіку. Виберіть тип *Voice call* і натискуйте кнопку *Assign*.

Запустіть модель і поглянете, як відбуваються голосові виклики.

Для перегляду характеристик потоку даних від *PC* до *SOHO Server* (часу проходження пакету і часу відгуку) натисніть на клавіатурі *Ctrl+Shift+W*. Проглянете в звіті про статистичне моделювання часу доставки пакету і часу відгуку в мережі між *PC* і *SOHO Server*.

Додайте голосові виклики між парою інших телефонів. Прогляньте в звіті інформацію про статистичне моделювання час доставки пакету і час відгуку в мережі між *PC* і *SOHO Server*.

Занесіть дані в таблицю 2 звіту про роботу і зробіть висновки.

Таблиця 2 - Результати досліджень середнього навантаження і середнього часу доставки від інтенсивності голосового трафіку в мережі

№ дослідження	1	2
Середнє завантаження мережі, Мбайт/с		
Середній час доставки, с		
Час відгуку, с		

Завдання 3.

Згідно з варіантом добавте робочі станції та телефони. Встановіть типи каналів та трафіку.

В першому досліді додайте трафік між комп'ютерами. В другому – між доданими телефонами. Прогляньте в звіті інформацію про статистичне моделювання час доставки пакету і час відгуку в мережі між *PC* і *SOHO Server*.

Результати моделювання занесіть в таблицю.

Лабораторна робота №3

Тема: Дослідження основних показників мережі *ATM*

Мета роботи:

1. Вивчити призначення і основні показники мереж *ATM*.
2. Отримати навики побудови моделі мережі *ATM* з використанням середовища моделювання *NetCracker*.
3. Дослідити характеристики мережі *ATM* за допомогою програми *NetCracker*.

Теоретичні відомості

Технологія *ATM* (*Asynchronous Transfer Mode* – асинхронний режим передачі) була створена на базі принципів роботи широкосмугових *ISDN*-мереж (*B-ISDN*), оскільки спочатку розглядалася як основний засіб швидкої передачі даних при організації комунікацій в мережах *B-ISDN*.

У міру розвитку ця мережева технологія зайняла свою нішу в локальних і глобальних мережах, не рахуючи мереж *B-ISDN*. Оператори телекомунікації і регіональні телефонні компанії пропонують *ATM* для реалізації глобальних комунікацій, і часто ці послуги йдуть в одному пакеті з можливостями *SONET, frame relay* і іншими послугами глобального зв'язку.

Технологія *ATM* має безліч переваг. Вона легко масштабується, тому швидкість передачі даних в локальних або глобальних мережах може збільшуватися у міру їх зростання або при переростанні локальної мережі в глобальну. З її допомогою можна вирішувати проблеми перевантаженості мережі, сегментувати мережі і навіть забезпечувати високошвидкісне підключення настільних систем. Крупні банки і університети використовують *ATM* для організації глобальних

комунікацій між віддаленими майданчиками, урядові організації застосовують *ATM* для зв'язку відділень в межах одного міста, а в кінематографічній промисловості технології *ATM* використовуються для передачі фільмів.

Основу технології *ATM* складають інтерфейс і протокол, за допомогою якого по звичайному комунікаційному каналу можна комутувати трафік, що має як постійну так і змінну швидкість. Також до складу АТМ входять програми і передавальне середовище, що відповідають стандартам протоколу *ATM*. *ATM* це інтегрований метод мережевого доступу, який багато виробників міжмережевого устаткування пропонують для реалізації в локальних мережах, а регіональні телефонні компанії – для організації глобальних мереж. При цьому досягаються високі швидкості передачі даних, а вартість послуг, що надаються, залежить від швидкості. На основі *ATM* реалізується масштабована магістральна інфраструктура, яка може взаємодіяти з мережами, що мають різні розміри швидкості і методи адресації.

Як і в деяких інших технологіях глобальних мереж (наприклад, *frame relay*), в *ATM*-мережах використовуються віртуальні ланцюги (*virtual circuit*), які називаються каналами (*channel*). Швидкість каналів *ATM* може складати 10 Гбіт/с, а на сьогоднішній день вже майже досягнута швидкість 40 Гбіт/с. Інформація передається у вигляді осередків, а не у вигляді пакетів. На відміну від пакетів, осередки (*cell*) мають корисне навантаження фіксованої довжини 53 байти, де 5 байт відводиться для заголовка і 48 байт — для даних користувача.

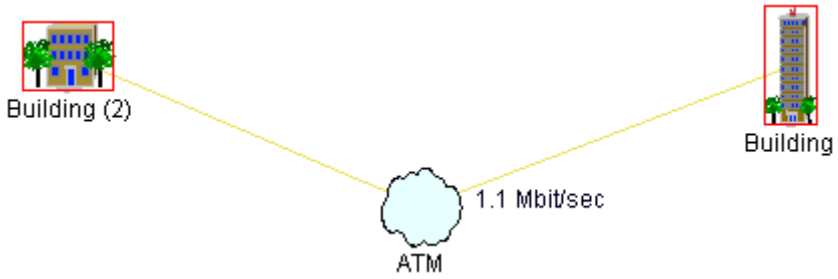
У *ATM* передбачено два типи мережевих інтерфейсів. Інтерфейс користувач-мережа (*User-Network Interface, UNI*) описує спосіб підключення користувачів до *ATM*-мережі. У свою чергу, міжмережевий інтерфейс (*Network-Network Interface, NNI*) визначає спосіб з'єднання двох мереж *ATM*. Оскільки приватні і загальні *ATM*-мережі повинні взаємодіяти між собою, специфікації *UNI* і *NNI* також були розділені на приватні та загальні.

Практична частина:

Завдання 1.

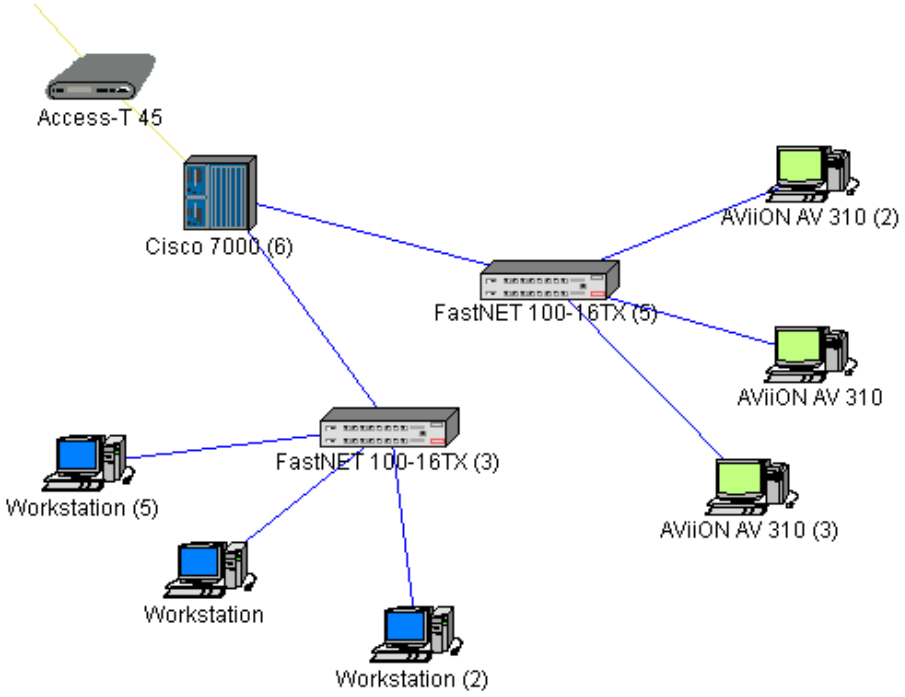
1. Запустіть програму *NetCracker* і створіть проект у вигляді фрагмента мережі *ATM*, представленого на малюнку 1.Всього вона складається з трьох частин,

дві з яких знаходяться усередині будівель (малюнок 2,3), а третя – зовнішня лінія, що сполучає мережі двох будівель на основі технології *ATM* з пропускною спроможністю каналу *T*3, – 44,738 Мбіт/с (див. мал. 1). Довжина з'єднання хмари *ATM* з кожною будівлею – 2 км.



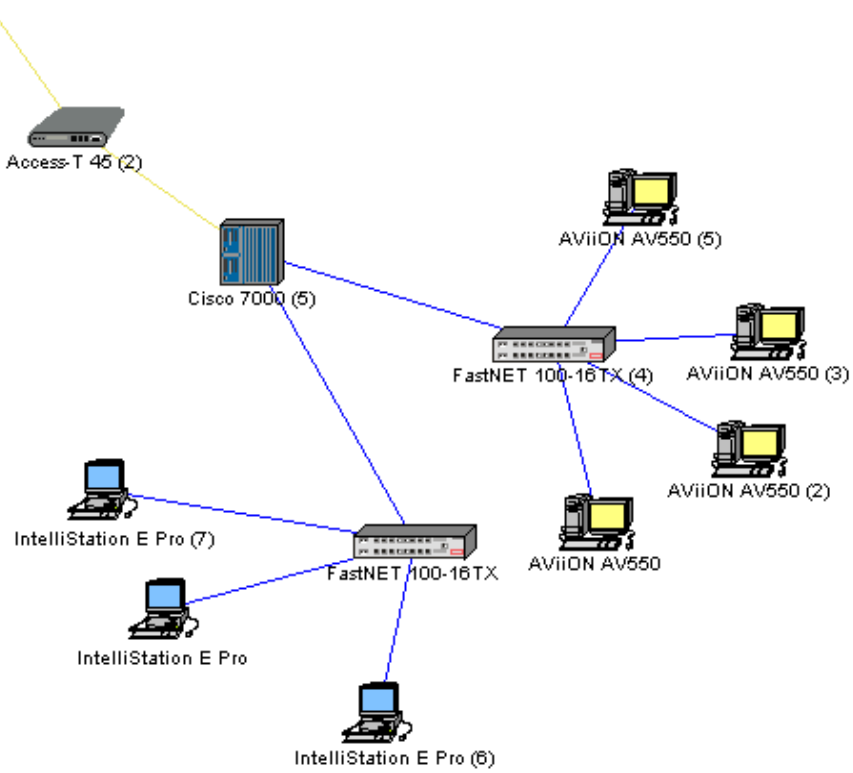
Малюнок 1 - Зовнішня лінія на основі технології ATM

2. Створіть інфраструктуру будівлі "*Building*". У ній містяться модем *DSU/CSU*, маршрутизатор *Cisco* 7000 зі встановленими модулями *HSSI* (для *ATM*) і *FastEthernet*, 2 комутатори і 6 робочих станцій.



Малюнок 2 – Перший фрагмент мережі *ATM* в будівлі "*Building*"

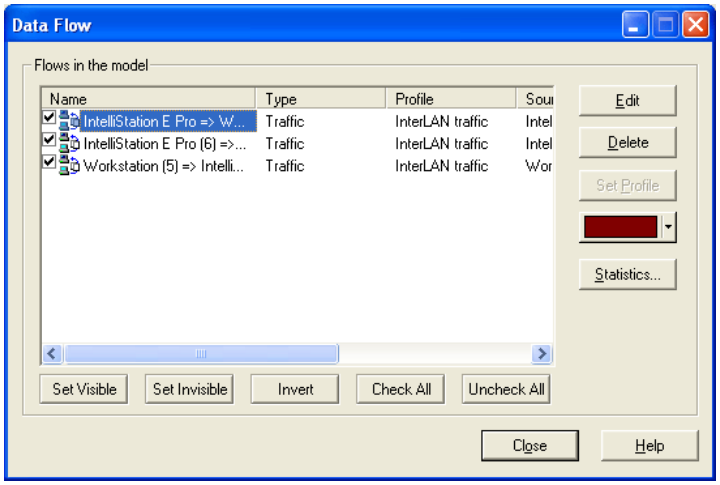
3. Аналогічно пункту 2 створіть "*Building* (2)". У ньому містяться модем *DSU/CSU*, маршрутизатор *Cisco* 7000 зі встановленими модулями *HSSI* (для *ATM*) і *FastEthernet*, 2 комутатори і 7 робочих станцій.



Малюнок 3 – Другий фрагмент мережі *ATM* в будівлі "*Building* (2)"

4. У меню *Global* програми виберіть пункт *Data Flow*. У вікні (мал. 4), що з'явилося, ви можете детально розглянути, як передається інформація в мережі.

Характеристика трафіку в даній мережі вибирається відповідно до таблиці, представленої на малюнку 4.



Малюнок 4 - Передавана по мережі інформація

Завдання 2. Дослідження моделі *ATM*

1. Запустіть модель кнопкою «*Start*» . Почекайте 15-20 секунд. Призупиніть моделювання кнопкою «*Pause/Resume*».

2. У меню програми «*Tools/Reports*» виберіть пункт «*Wizard*». У вікні, що з'явилося, виберіть тип звіту «*Statistical/Data Flows/Application Statistics*». Натисніть «*OK*».

У звіті, що з'явився, дивіться пункт «*Travel time*» - час доставки повідомлення між абонентами. Порахуйте середній час доставки повідомлення по мережі. Середнє завантаження мережі ви зможете знайти на верхньому рівні моделі біля хмари *ATM*.

3. Зупиніть роботу моделі.

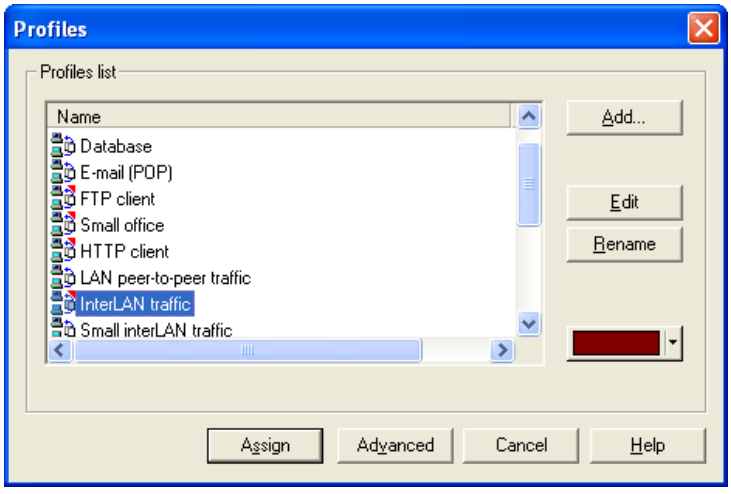
Дослідження залежності часу доставки повідомлення від міри завантаженості мережі.

1. Підготуйте таблицю наступного вигляду:

№	1	2	3
Середнє навантаження, Мбайт/с			
Середній час доставки, с			

Внесіть до 1-го стовпця дані, отримані раніше.

2. На панелі інструментів виберіть режим установки трафіку. У одній з будівель лівою кнопкою миші виберіть комп'ютер, що не взаємодіє з іншими (про відсутність з'єднань можна дізнатися, натиснувши на зображенні комп'ютера правою кнопкою миші і вибравши пункт меню «*Associated Data Flow*.»). Потім так само виділіть вільний комп'ютер з іншої будівлі. З'явиться таке вікно:



Малюнок 5 – Налаштування типів передаваної інформації

Виберіть типа “*INTERLAN traffic*”. Натисніть кнопку «*Assign*». Повторіть двічі для комп'ютерів, що залишилися.

3. Запустіть модель. Визначте середній час доставки, внесіть результати до 2-го стовпця таблиці.

4. Додайте ще 6 з'єднань між комп'ютерами різних будівель так, що б на один комп'ютер було не більше 2-х з'єднань. Запустіть модель. Результати внесіть до 3-го стовпця таблиці.

Заповніть підготовлену таблицю і зробіть висновки за результатами проведеного дослідження. Врахуйте, що пропускна спроможність каналу ATM дорівнює 44,738 Мбіт/с, порівняйте з середнім навантаженням на мережу.

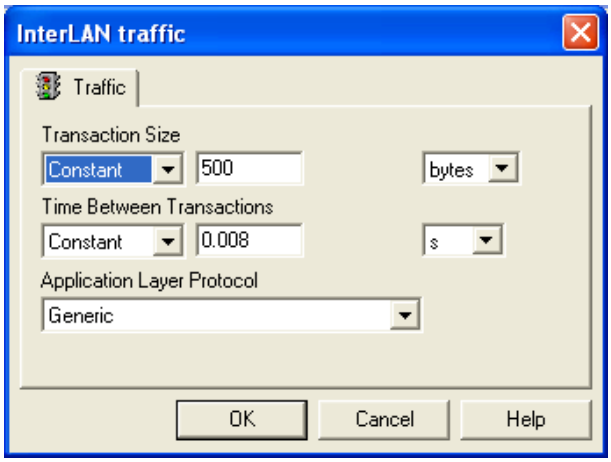
Дослідження залежності часу доставки повідомлення від розміру повідомлення.

1. Підготуйте таблицю наступного вигляду:

№	1	2	3	4
Розмір повідомлення, байт	4000	6000	8000	10000
Середній час доставки, с				

2. Відкрийте вікно “*Data Flow*”, виберіть яке-небудь із з'єднань і натисніть кнопку «*Edit*». З'явиться вікно, зображене на малюнку 5.

3. У ньому натискуйте кнопку «*Edit*». Відкриється редактор, зображений на мал. 6.



Малюнок 6 – Редактор трафіку

У пункті «*Transaction Size*» вкажіть новий розмір повідомлення 1000 байт. Натисніть «*OK*». З'явиться діалогове вікно з питанням – відповідайте «*Yes*». Потім натисніть кнопку «*Assign*» і закрийте вікно, що залишилося. Тепер розмір передаваного повідомлення між всіма машинами буде 1000 байт.

4. Запустіть модель. Порахуйте середній час доставки повідомлення. Впишіть в таблицю. Зверніть увагу на збільшення завантаженості мережі (середнє завантаження хмари *ATM*).

5. Заповніть таблицю повністю.

Побудуйте графік залежності часу доставки повідомлення від розміру повідомлення. Визначте типи залежності. Поясніть, чому збільшується час доставки повідомлення, посилаючись на особливості технології *ATM* . Зробіть висновки по даному пункту досліджень.

Дослідження залежності часу доставки повідомлення від розміру повідомлення при збільшенні кількості звернень до мережі.

У вашій поточній моделі знову відкрийте вікно налаштування якого-небудь з'єднання і, залишивши розмір повідомлення рівним 10000 байт, встановіть час між відправкою двох повідомлень рівним 0.004 з (у два рази менше поточного). Запустіть модель. Простежте, що відбувається. Якщо ви побачите на маршрутизаторах або комутаторах червоні спалахи, то вам вдалося навантажити мережу АТМ до межі, і тепер деякі пакети просто не проходять. Знайдіть середній час доставки повідомлення в мережі.

Для того, щоб в подробицях розглянути стан елементів мережі, проглянете звіт з меню програми «*Tools/Reports*» в пункті «*Wizard*» «*Advanced/Modules/Module Statistical Items*». У цьому звіті так само буде вказано кількість втрачених пакетів.

Зробіть висновки на основі отриманих результатів про властивості комбінованих мереж із використанням технології *ATM*. Порівняйте з результатами, отриманими в ході дослідження моделей мереж на основі інших технологій. Охарактеризуйте особливості технології *ATM* і ситуації в яких переважно саме її використання.

Лабораторна робота №4

Тема: Технології бездротових мереж.

Мета роботи : ознайомитися з протоколами і технологіями передачі даних у бездротових мережах на фізичному рівні, отримати навички вибору устаткування для побудови бездротової локальної обчислювальної мережі.

Теоретичні відомості

Сімейство стандартів IEEE 802.11

Стандарт *IEEE* 802.11, розробка якого була завершена в 1997 р., є базовим стандартом і визначає протоколи, необхідні для організації бездротових локальних мереж (*Wireless Local Area Network, WLAN*). Основні з них - протокол управління доступом до середовища *MAC* (*Medium Access Control* - нижній підрівень канального рівня) і протокол *PHY* передачі сигналів у фізичному середовищі. В якості останнього допускається використання радіохвиль і інфрачервоного випромінювання.

Стандартом *IEEE* 802.11 визначений єдиний підрівень *MAC*, що взаємодіє з трьома типами протоколів фізичного рівня, які відповідають різним технологіям передачі сигналів, - по радіоканалах в діапазоні 2,4 ГГц з широкосмуговою модуляцією з прямим розширенням спектру (*Direct Sequence Spread Spectrum, DSSS*) і перескоком частоти (*FHSS*), а також за допомогою інфрачервоного випромінювання. Специфікаціями стандарту передбачено два значення швидкості передачі даних - 1 і 2 Мбіт/с.

В порівнянні з кабельними локальними мережами *Ethernet* можливості підрівня *MAC* розширені за рахунок включення в нього ряду функцій, що зазвичай виконуються протоколами більш високого рівня, зокрема, процедур фрагментації і ретрансляції пакетів. Це викликано прагненням підвищити ефективну пропускну спроможність системи завдяки зниженню накладних витрат на повторну передачу пакетів.

Управління живленням

Для економії енергоресурсів мобільних робочих станцій, що використовуються у бездротових локальних мережах, стандартом *IEEE* 802.11

передбачений механізм перемикання станцій в так званий пасивний режим з мінімальним споживанням потужності.

Архітектура і компоненти мережі

В основу стандарту *IEEE* 802.11 покладена стільникова архітектура, причому мережа може складатися як з одного, так і декількох осередків. Кожен стільник управляється базовою станцією, що називається точкою доступу (*Access Point, AP*), яка разом з сусідніми, в межах радіусу дії, робочими станціями користувачів утворює базову зону обслуговування (*Basic Service Set, BSS*) Точки доступу багатостільникової мережі взаємодіють між собою через розподілену систему (*Distribution System, DS*), що є еквівалентом магістрального сегменту кабельних локальних мереж. Уся інфраструктура, що включає точки доступу і розподілену систему утворює розширену зону обслуговування (*Extended Service Set*).

Стандартом передбачений також одностільниковий варіант бездротової мережі, який може бути реалізований і без точки доступу, при цьому частина її функцій виконуються безпосередньо робочими станціями.

Забезпечення безпеки

Для захисту *WLAN* стандартом *IEEE* 802.11 передбачений цілий комплекс заходів безпеки передачі даних під загальною назвою *Wired Equivalent Privacy, WEP*. Він включає засоби протидії несанкціонованому доступу до мережі (механізми і процедури аутентифікації), а також запобігання перехопленню інформації (шифрування).

Стандарт IEEE 802.11a

Є найбільш "широкосмуговим" з сімейства стандартів *IEEE* 802.11, передбачаючи швидкість передачі даних до 54 Мбіт/с (редакцією стандарту, затвердженою в 1999 р., визначені три обов'язкові швидкості - 6, 12 і 24 Мбіт/с і п'ять необов'язкових - 9, 18, 36, 48 і 54 Мбіт/с).

На відміну від базового стандарту, орієнтованого на область частот 2,4 ГГц, специфікаціями *IEEE* 802.11a передбачена робота в діапазоні 5 ГГц. В якості методу модуляції сигналу вибрано ортогональне частотне мультиплексування (*OFDM*). Найбільш суттєва відмінність між цим методом і радіотехнологіями *DSSS* і *FHSS* полягає в тому, що *OFDM* припускає паралельну передачу корисного сигналу

одночасно по декількох частотах діапазону, тоді як технології розширення спектру передають сигнали послідовно. В результаті підвищується пропускна спроможність каналу і якість сигналу.

До недоліків *IEEE* 802.11a відносяться більш висока споживана потужність радіопередавачів для частот 5 ГГц, а так найменший радіус дії (устаткування для 2,4 ГГц може працювати на відстані до 300м, а для 5 ГГц - біля 100м).

Стандарт IEEE 802.11b

Завдяки високій швидкості передачі даних (до 11 Мбіт/с), практично еквівалентній пропускній спроможності звичайних дротяних локальних мереж *Ethernet*, а також орієнтації на "освоєний" діапазон 2,4 ГГц, цей стандарт завоював найбільшу популярність у виробників устаткування для бездротових мереж.

У остаточній редакції стандарт *IEEE* 802.11b, відомий також як *Wi-Fi* (*Wireless Fidelity*), був прийнятий в 1999р. В якості базової радіотехнології в ньому використовується метод *DSSS* з 8-розрядними послідовностями Уолша.

Оскільки устаткування, яке працює на максимальній швидкості 11 Мбіт/с має менший радіус дії, ніж на нижчих швидкостях, то стандартом 802.11b передбачено автоматичне пониження швидкості при погіршенні якості сигналу.

Як і у разі базового стандарту *IEEE* 802.11, чіткі механізми роумінгу специфікаціями IEEE 802.11b не визначені.

Специфікація IEEE 802.11g

Специфікації IEEE 802.11g є розвитком стандарту 802.11b і дозволяють підвищити швидкість передачі даних у безпровідних локальних мережах до 22 Мбіт/с (і вище) завдяки використанню ефективнішої модуляції сигналу. Одним з достоїнств стандарту є зворотна сумісність з IEEE 802.11b.

Існує декілька різних технологій розширення спектру в протоколах *IEEE* 802.11, використовується технологія розширення спектру методом прямої послідовності (*DSSS*).

Технологія DSSS

При потенційному кодуванні інформаційні біти - логічні нулі і одиниці - передаються прямокутними імпульсами напруги. Прямокутний імпульс тривалості *T*

має спектр, ширина якого обернено пропорційна до тривалості імпульсу. Тому чим менше тривалість інформаційного біта, тим більший спектр займає такий сигнал.

Для умисного розширення спектру спочатку вузькосмугового сигналу в технології *DSSS* в кожен інформаційний біт, який передається, (логічний 0 або 1) у буквальному розумінні вбудовується послідовність так званих чіпів. Якщо інформаційні біти - логічні нулі або одиниці - при потенційному кодуванні інформації можна представити у вигляді послідовності прямокутних імпульсів, то кожен окремий чіп - це теж прямокутний імпульс, але його тривалість у декілька разів менше тривалості інформаційного біта. Послідовність чіпів є послідовністю прямокутних імпульсів, тобто нулів і одиниць, проте ці нулі і одиниці не є інформаційними. Оскільки тривалість одного чіпа в *n* разів менша тривалості інформаційного біта, то і ширина спектру перетвореного сигналу буде в *n*-разів більше ширини спектру первинного сигналу. При цьому і амплітуда передаваного сигналу зменшиться в *n* разів.

Чіпові послідовності, що вбудовуються в інформаційні біти, називають шумоподібними кодами (*PN*-послідовності), що підкреслює ту обставину, що результуючий сигнал стає шумоподібним і його важко відрізнити від природного шуму.

Як розширити спектр сигналу і зробити його невідмітним від природного шуму, зрозуміло. Для цього, в принципі, можна скористатися довільною (випадковою) чіповою послідовністю. Проте, виникає питання: а як такий сигнал приймати? Адже якщо він стає шумоподібним, то виділити з нього корисний інформаційний сигнал не так просто, якщо взагалі можливо. Виявляється, можливо, але для цього треба відповідним чином підібрати чіпову послідовність. Використовувані для розширення спектру сигналу чіпові послідовності повинні задовольняти певним вимогам автокореляції. Під терміном автокореляції в математиці мають на увазі міру подібності функції самої собі в різні моменти часу. Якщо підібрати таку чіпову послідовність, для якої функція автокореляції матиме різко виражений пік лише для одного моменту часу, то такий інформаційний сигнал можливо буде виділити на рівні шуму. Для цього в приймачі отриманий сигнал множиться на ту ж чіпову послідовність, тобто обчислюється автокореляційна

функція сигналу. В результаті сигнал стає знову вузькосмуговим, тому його фільтрують у вузькій смузі частот і будь-яка перешкода, що потрапляє в смугу початкового широкосмугового сигналу, після множення на чіпову послідовність, навпаки, стає широкосмуговою і обрізується фільтрами, а у вузьку інформаційну смугу потрапляє лише частина перешкоди, по потужності значно менша, ніж перешкода, діюча на вході приймача.

Коди Баркера

Чіпових послідовностей, що відповідають вказаним вимогам автокореляції, існує досить багато, але особливий інтерес представляють так звані коди Баркера, оскільки саме вони використовуються в протоколі *IEEE* 802.11.

Коди Баркера мають найкращі серед відомих псевдовипадкових послідовностей властивості шумоподібності, що і зумовило їх широке застосування. У протоколах сімейства *IEEE* 802.11 використовується код Баркера завдовжки в 11 чіпів (11100010010). Для того, щоб передати сигнал логічна одиниця передається прямою послідовністю Баркера, а логічний нуль - інверсною послідовністю.

Швидкість 1 Мбіт/с

У стандарті *IEEE* 802.11 передбачено два швидкісні режими: 1 і 2 Мбіт/с. Для кодування даних на фізичному рівні використовується метод *DSSS* з 11-чіповими кодами Баркера. При інформаційній швидкості 1 Мбіт/с швидкість наслідування окремих чіпів послідовність Баркера складає 11×106 чіп/с, а ширина спектру такого сигналу складає 22 Мгц. Враховуючи, що ширина частотного діапазону складає 83,5 Мгц, отримуємо, що усього в цьому частотному діапазоні можна умістити 3 частоти каналу, що не перекриваються. Увесь частотний діапазон прийнято ділити на 11 частотних каналів, що перекриваються, по 22 Мгц, віддалених один від одного на 5 Мгц. Приміром, перший канал займає частотний діапазон від 2400 до 2423 Мгц і центрований відносно частоти 2412 Мгц. Другий канал центрований відносно частоти 2417 Мгц, а останній, 11 канал, центрований відносно частоти 2462 Мгц. При такому розгляді перший, шостий і 11 канал не перекриваються один з одним і мають 3 мегагерцевий проміжок один відносно одного. Саме ці три канали можуть використовуватися незалежно один від одного.

Для модуляції синусоїдального сигналу (процес, необхідний для інформаційного наповнення сигналу), використовується відносна двійкова фазова модуляція (*Differential Binary Phase Shift Key, DBPSK*). При цьому кодування інформації відбувається за рахунок зрушення фази синусоїдального сигналу по відношенню до попереднього стану сигналу. Двійкова фазова модуляція передбачає два можливі значення зрушення фази - 0 і π. Тоді логічний нуль може передаватися синфазним сигналом (зрушення по фазі дорівнює 0), а одиниця - сигналом, який зрушений по фазі на π.

Швидкість 2 Мбіт/с

Інформаційна швидкість 1 Мбіт/с є обов'язковою в стандарті *IEEE* 802.11 (*Basic Access Rate*), але опційно можлива і швидкість в 2 Мбіт/с (*Enhanced Access Rate*). Для передачі даних на такій швидкості використовується та ж технологія *DSSS* з 11-чіповими кодами Баркера, але для модуляції коливання застосовується відносна квадратурна фазова модуляція (*Differential Quadrature Phase Shiftey*). При відносній квадратурній фазовій модуляції зрушення фаз може набувати чотири різні значення: 0, π/2, π і 3π/2. Використовуючи чотири різні стани сигналу, можна в одному дискретному стані закодувати послідовність двох інформаційних біт (дібіт) і тим самим в два рази підвищити інформаційну швидкість передачі. Приміром, дібіту 00 може відповідати зрушення фази, рівне 0; дібіту 01 - зрушення фази, рівне π/2; дібіту 11 - зрушення фази, рівне π; дібіту 10 - зрушення фази, рівне 3π/2.

При інформаційній швидкості 2 Мбіт/с швидкість наслідування окремих чіпів послідовністю Баркера залишається колишньою, тобто 11×106 чіп/с, а отже, не міняється і ширина спектру сигналу.

Практична частина:

1. Використовуючи пакет *NetCracker*, вивчити склад і функціональні характеристики типового устаткування бездротових локальних мереж.

2. Відповідно до варіанту завдання побудувати бездротову мережу з використанням стандартів *IEEE* 802.11.

3. Для отриманої моделі мережі задати необхідні типи потоків даних між робочими станціями і серверами і зробити імітаційне моделювання роботи мережі.

4. Проаналізувати середнє завантаження мережевого устаткування, а також кількість втрачених пакетів. Зробити висновки.

Таблиця 1. Варіанти завдань

№ Варіанту	Технологія магістралі	Кількість *HTTP* серверів	Кількість *FTP* серверів	Кількість безпровідних станцій
1	*Ethernet*	1	2	6
2	*Token Ring*	2	3	7
3	*Ethernet*	3	2	5
4	*Token Ring*	4	1	4
5	*Ethernet*	1	3	5
6	*Token Ring*	2	4	4
7	*Ethernet*	3	3	5
8	*Token Ring*	4	2	6
9	*Ethernet*	1	4	3
10	*Token Ring*	2	1	7
11	*Ethernet*	3	4	5
12	*Token Ring*	4	2	3
13	*Ethernet*	1	1	7
14	*Token Ring*	2	2	4
15	*Ethernet*	3	3	2

Лабораторна робота №5

Тема: Розробка простого клієнт-серверного додатка

Мета роботи: навчитися розробляти клієнт-серверні додатки

Теоретичні відомості:

Алгоритм роботи з сокетними протоколами

При роботі з сокетом просто посилається іншому комп'ютеру послідовність символів. Отже цим методом можна посилати як прості повідомлення, так і цілі файли. Причому, контролювати правильність передачі не потрібно.

Розберемо схему роботи детальніше:

Визначення *Host* і *Port* - щоб успішно встановити з'єднання, потрібно призначити властивостям *Host* і *Port* компоненту *TClientSocket* необхідні значення. *Host* - це хост-ім'я (наприклад: *nitro.borland.com*) або *IP*-адреса (наприклад: 192.168.0.88) комп'ютера, з яким треба з'єднатися. *Port* - номер порту (від 1 до 65535) для встановлення з'єднання. Зазвичай номери портів беруться, починаючи з 1025 - оскільки номери менше 1024 можуть бути зайняті системними службами (наприклад, *POP* - 110);

Відкриття сокета - після того, як ви призначили властивостям *Host* і *Port* відповідні значення, можна приступити безпосередньо до відкриття сокета (сокет тут розглядається як черга, в якій містяться символи, що передаються від одного комп'ютера до іншого). Для цього можна викликати метод *Open* компоненти *TClientSocket*, або привласнити властивості *Active* значення *True*. Тут корисно ставити обробник виняткової ситуації на той випадок, якщо з'єднатися не вдалося;

Посилка/прийом даних - це, власне і є те, для чого відкривалося сокетне з'єднання. Протокол обміну даними також залежить від сервера;

Закриття сокета - після всіх виконаних операцій необхідно закрити сокет за допомогою методу *Close* компоненту *TClientSocket* (або привласнити властивості *Active* значення *False*).

Опис властивостей і методів компоненту *TclientSocket*

Властивості:

Active - показує, відкритий сокет чи ні. Ця властивість доступна для запису;

Host - рядок (Тип: *string*), який вказує на хост-ім'я комп'ютера, до якого слід підключитися;

Address - рядок (Тип: *string*), який вказує на IP-адресу комп'ютера, до якого слід підключитися. На відміну від *Host*, тут може міститися лише *IP*. Відмінність в тому, що якщо ви вкажете в *Host* символьне ім'я комп'ютера, то *IP* адреса, відповідна цьому імені буде запитана у *DNS*;

Port - номер порту (Тип: *Integer (Word)*), до якого слід підключитися. Допустимі значення - від 1 до 65535;

Service - рядок (Тип: *string*), що визначає службу (*ftp, http, pop*, і т.д.), до порту якої відбудеться підключення. Це своєрідний довідник відповідності номерів портів різним стандартним протоколам;

ClientType - тип з'єднання. *ctNonBlocking* - асинхронна передача даних, тобто посилати і приймати дані по сокету можна за допомогою *OnRead* і *OnWrite*. *ctBlocking* - синхронна (одночасна) передача даних. Події *OnRead* і *OnWrite* не працюють. Цей тип з'єднання корисний для організації обміну даними за допомогою потоків (тобто робота з сокетом як з файлом);

Методи:

Open - відкриття сокета (аналогічно задати значення *True* властивості *Active*);

Close - закриття сокета (аналогічно задати значення *False* властивості *Active*);

Події :

OnConnect -ця подія виникає при встановленні з'єднання. Тобто в обробнику цієї події вже можна починати авторизацію або прийом/передачу даних;

OnConnecting - виникає при встановленні з'єднання. Відмінність від *OnConnect* в тому, що з'єднання ще не встановлене. Зазвичай такі проміжні події використовуються для оновлення статусу;

OnDisconnect - виникає при закритті сокета. Причому, закриття як з вашої програми, так і із сторони віддаленого комп'ютера (або через збій);

OnError - Виникає при помилці в роботі сокета. Слід зазначити, що ця подія не допоможе Вам відловити помилку у момент відкриття сокета (*Open*). Для того, щоб уникнути видачі вікна повідомлення про помилку, треба вкласти відкриття сокета в блок *try..except* (обробка виняткових ситуацій);

OnLookup - виникає при спробі отримання від *DNS IP*-адреси вказаного хоста;

OnRead - виникає, коли видалений комп'ютер послав Вам які-небудь дані. При виникненні цієї події можлива обробка даних;

задати - виникає, коли Вам дозволений запис даних в сокет.

Приклад 1. Проста сокетна програма

Процедура з’єднання з сервером:

```
procedure Button1Click(Sender: TObject);
begin
  { Назначаємо властивостям Host (або Adress)і Port потрібні значення }
  ClientSocket1.Host := Edit1.Text;
  ClientSocket1.Port := StrToInt(Edit2.Text);
  ClientSocket1.Address := Edit4.Text;
  { Намагаємося відкрити сокет і встановити з'єднання }
  ClientSocket1.Open;
end;

procedure ClientSocket1Connect(Sender: TObject; Socket: TCustomWinSocket);
begin
  { Як тільки відбулося з'єднання - закриваємо сокет і перериваємо зв'язок }
  ClientSocket1.Close;
end;
```

Далі слідує інший приклад, в якому по сокету передаються і приймаються текстові повідомлення:

Приклад 2. Посилка/прийом текстових повідомлень по сокетах

Процедура з’єднання з сервером:

```
procedure Button1Click(Sender: TObject);
```

```
begin
  { Якщо з'єднання вже встановлене - перериваємо його.}
  if ClientSocket1.Active then
  begin
    ClientSocket1.Close;
    Exit; {...і виходимо з обробника }
  end;
  { Назначаємо властивостям Host і Port потрібні значення}
  ClientSocket1.Host := Edit1.Text;
  ClientSocket1.Port := StrToInt(Edit2.Text);
  { Намагаємося відкрити сокет і встановити з'єднання }
  ClientSocket1.Open;
end;

procedure ClientSocket1Connect(Sender: TObject; Socket: TCustomWinSocket);
begin
  { Як тільки відбулося з'єднання - посилаємо вітання }
  Socket.SendText('Hello!');
  ListBox1.Items.Add('< Hello!');
end;

procedure Button2Click(Sender: TObject);
begin
  ClientSocket1.Socket.SendText(Edit3.Text);
  ListBox1.Items.Add('< ' + Edit3.Text);
end;
procedure ServerSocket1ClientRead(Sender: TObject; Socket: TCustomWinSocket);
begin
 { Якщо прийшло повідомлення - додаємо його в ListBox }
 ListBox1.Items.Add('> ' +Socket.ReceiveText);
end;
```

Практична частина:

Розробити клієнт-серверний додаток, який дасть змогу обмінюватись службовою інформацією між клієнтом та сервером. Алгоритм роботи такий: клієнт робить запит за допомогою зарезервованого набору символів, а сервер відповідає результатом виконання функції, що зарезервована під цим набором.

Службовою інформацією має бути: дата та час на сервері, номер вінчестера сервера, назва локального хоста, кількість підключених клієнтів і ще 2 ви маєте придумати самі.

В даному додатку повинні бути враховані критичні випадки: невірний ввід, або введено не зарезервований набір символів, неможливість підключення до серверу і т.д. Також має бути повний запис дій як на сервері, так і на клієнті.

Хід роботи:

1. Створити 2 проекти, один для сервера і один для клієнта. Розробити зовнішній інтерфейс клієнта та сервера.
2. Запрограмувати події для компонентів, які використовувались при створенні додатка.
3. Запрограмувати обробку нестандартних ситуацій, а також запис дій клієнта та сервера
4. Протестувати роботу програми.
5. Подумати над вдосконаленням додатку, додаванням нових функцій та можливостей.

Лабораторна робота №6

Тема: Створення простого «браузера» для відображення *HTML* по вибраному посиланню

Мета роботи:

Створити простий веб-переглядач для відображення простого *HTML* по вибраному посиланню.

Теоретичні відомості

Web-браузери - це програмні засоби для роботи з гіпертекстовими документами *World Wide Web*. Також, за їх допомогою можна завантажувати довільні файли з мережі. В деякі браузери вже вбудовані поштові програми та редактори гіпертекстів.

ВИМОГИ ДО *WEB*-БРАУЗЕРІВ

- **від кінцевих користувачів:**
  - перегляд різноманітної інформації та "активного вмісту";
  - персоналізація роботи і налаштування представлення інформації;
  - комунікації з іншими користувачами за допомогою засобів електронної та мовної пошти;
- **від адміністраторів**, що керують локальними мережами з використанням технологій Інтранет:
  - простий та недорогий перевід настільних систем на клієнтське програмне забезпечення для роботи в інтрамережі;
  - скорочення вартості підтримки настільних персональних комп'ютерів, що підключені до мережі;
  - підвищення продуктивності роботи кінцевих користувачів локальних мереж;
- **від *Web*-дизайнерів та авторів документів Інтернет**, які хочуть отримати відкриту платформу, з використанням прийнятих стандартів, для створення активного "наповнення" *Web* і розробки *Web*-сторінок наступного покоління:

- широкого спектру мов сценаріїв і програмування для створення вмісту *Web* та їх підтримку;
- різних видів активних об'єктів *Java*, елементів керування *ActiveX* і розширень *HTML*, мультимедіа і інтегрованих модулів (*plug-in*);
- відкритої і розширеної архітектури, яка дозволяє додавання і інтеграцію нових технологій і можливостей перегляду інформації;

Найпопулярнішими *Web*-браузерами в ОС *Windows* є *Internet Explorer*, *Opera* та *Mozilla Firefox*.

Практична частина:

Розробити додаток, що дає можливість відображення простого *HTML* по вибраному посиланню. Має можливість навігації по Інтернет сторінкам. Передбачає обробку нестандартних ситуацій у додатку. Реалізувати декілька додаткових функцій програми «браузер» таких як можливість збереження історії переглянутих сторінок, створення швидкого набору сайтів, можливість створення закладок для улюблених сайтів.

Хід роботи.

1. Створити проект програмного додатку «браузера». Розробити зовнішній інтерфейс.
2. Запрограмувати обробку нестандартних ситуацій.
3. Протестувати роботу програми.
4. Подумати над вдосконаленням додатку, додаванням нових функцій та можливостей.

Створіть новий проект ("*File*" -> "*New Application*") та змініть заголовок та іконку. Встановлюємо на форму компонент "*WebBrowser*" та "*CoolBar*", що знаходиться на вкладці "*Win32*". Виділяємо *WebBrowser*1 і переходимо в *ObjectInspector*. виберіть властивість "*Align*" і поставте "a*lClient*". Тепер на *CoolBar*1 додаємо панель "*ToolBar*" із вкладки "*Win32*" и "*ComboBox*" із вкладки "*Standart*" палітри компонентів. Виділяємо *CoolBar*1 переходимо в *ObjectInspector* змінюємо "*AutoSize*" на "*true*" (за замовчуванням "*false*"). Виділяємо *ComboBox*1 переходимо на закладку "*Events*" активізуємо метод "*OnKeyDown*" і записуємо наступне:

```
procedure TForm1.ComboBox1KeyDown(Sender: TObject;
  var Key: Word; Shift: TShiftState);
```

```
begin
  if Key = VK_RETURN then
    WebBrowser1.Navigate(ComboBox1.Text);
end;
```

*Я*кщо кнопка с кодом *VK_RETURN*(*Enter*) натиснута, то компонент *WebBrowser*1, завантажує адрес записаний в поле *ComboBox*1.

Змінюємо властивості *ToolBar*1 "*AutoSize*", "*ShowCaption*" із "*Flat*" на "*true*". Тепер клікаєм правою клавішою по *ToolBar*1 і вибираємо пункт "*New Button*". На *ToolBar*1 повинна з'явитись нова кнопка з іменем "*ToolButton*1". Замінимо властивість *Caption* на "Відкрити". Створюємо ще декілька кнопок з назвами: Назад, Вперед, Стоп, Оновити і Друк. Встановлюємо на форму "*OpenDialog*" із вкладки "*Dialogs*" палітри компонентів. Тепер створюємо метод *Click* для кнопки "Відкрити":

```
procedure TForm1.ToolButton1Click(Sender: TObject);
begin
  if OpenDialog1.Execute then
  begin
    WebBrowser1.Navigate(OpenDialog1.FileName);
    ComboBox1.Text := OpenDialog1.FileName;
  end;
end;
```

Для інших кнопок також створюємо по методу:

```
procedure TForm1.ToolButton2Click(Sender: TObject);
begin
  WebBrowser1.GoBack;
end;

procedure TForm1.ToolButton3Click(Sender: TObject);
begin
  WebBrowser1.GoForward;
end;

procedure TForm1.ToolButton4Click(Sender: TObject);
begin
  WebBrowser1.Stop;
end;

procedure TForm1.ToolButton5Click(Sender: TObject);
begin
  WebBrowser1.Refresh;
```

```
end;

procedure TForm1.ToolButton6Click(Sender: TObject);
var
  PostData, Headers: OLEvariant;
begin
  WebBrowser1.ExecWB(OLECMDID_PRINT, OLECMDEXECOPT_DODEFAULT,
PostData,
    Headers);
end;
```

Встановлюємо *StatusBar* із закладки "*Win32*" і змінюємо властивість "*SimplePanel*" в *true* . Для *WebBrowser*1 створюємо метод "*OnStatusTextChange*":

```
procedure TForm1.WebBrowser1StatusTextChange(Sender: TObject;
  const Text: WideString);
begin
  StatusBar1.SimpleText := Text;
end;
```

Встановлюємо на форму *ProgressBar* із закладки "*Win32*". Змінюємо властивість "*Align*" на "*alBottom*". Для *WebBrowser*1 створюємо метод "*OnProgressChange*":

```
procedure TForm1.WebBrowser1ProgressChange(Sender: TObject; Progress,
  ProgressMax: Integer);
begin
  ProgressBar1.Max := ProgressMax;
  ProgressBar1.Position := Progress;
end;
```

Встановлюємо форму *ImageList* та додаємо в нього 6 зображення розміром 16x16. Змінюємо властивість *Images* на "*ImageList*1" елемента *ToolBar*1.

Лабораторна робота №7

Тема: Створення простої програми для роботи з протоколом *SMTP*.

Мета роботи:

Розробити додаток, що дає можливість отримання та відправки пошти по протоколу *SMTP*.

Теоретичні відомості

Simple Mail Transfer Protocol (Простий Протокол Пересилання Пошти) — це протокол, який використовується для пересилання електронної пошти до поштового сервера або з клієнта-комп'ютера, або між поштовими серверами. В *IANA* для *SMTP* зареєстрований порт 25. Формально SMTP визначений в *RFC* 821 (*STD* 10) та покращений *RFC* 1123 (*STD* 3) розділ 5. Протокол який використовується зараз також відомий як *ESMTP* і визначений в *RFC* 2821.

SMTP - порівняно простий, текстовий протокол, в якому з'єднання відбувається завжди за ініціативи відправника. *SMTP* - синхронний протокол і складається із серії команд, що посилаються клієнтом та відповідей сервера. Відправником зазвичай є поштовий клієнт кінцевого користувача або поштовий сервер. *SMTP* було розроблено як протокол транспортування і доставки, тому системи, що використовують *SMTP*, завжди повинні бути у робочому стані. Протокол часто використовується для передачі повідомлень клієнтами електронної пошти, які, проте, не мають можливості діяти як сервер.

Для роботи з мережею, в *Delphi* використовується *Indy* компоненти, які можна знайти на декількох вкладках середовища розробки. Знайдіть на вкладці *Indy Clients* - компонент *IdPOP3*, на вкладці *Indy Misc* - компонент *IdMessage*. За допомогою компонента *IdPOP3* можна здійснити підключення до поштового сервера по протоколу *POP3* й одержати необхідну кількість повідомлень. Компонент *IdMessage* буде використовуватись, як буфер для одержуваного листа. Ще буде потрібно *memo* для відображення тексту листа, і кнопка для завантаження чергового повідомлення.

Отримання одного листа:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
POP3.Host:=’mail.58r.ru’;
POP3.Port:=110;
POP3.Username:=’test+58r.ru’;
POP3.Password:=’12345666′;
IdMessage.Clear;
POP3.Connect;
Memo1.Clear;
POP3.Retrieve(1,IdMessage);
Memo1.Lines.AddStrings(IdMessage.Body);
POP3.Delete(1);
POP3.Disconnect;
end;
```

Отримання 10 листів:

```
procedure TForm1.Button2Click(Sender: TObject);
label f;
var
mailcicl:integer;
addr:string;
begin
POP3.Host:=’mail.58r.ru’;
POP3.Port:=110;
POP3.Username:=’test+58r.ru’;
POP3.Password:=’12345666′;
addr:=ExtractFilePath(Application.ExeName);
POP3.Connect;
for mailcicl:=1 to 10 do
begin
if POP3.CheckMessages<1 then goto f ;
```

```
IdMessage.Clear;
Memo1.Clear;
POP3.Retrieve(1,IdMessage);
Memo1.Lines.AddStrings(IdMessage.Body);
Memo1.Lines.SaveToFile(addr+inttostr(mailcicl)+'.txt');
POP3.Delete(mailcicl);
end;
f:
POP3.Disconnect;
```

Кожне повідомлення має атрибути - це адреса відправника, тема повідомлення, важливість й інші елементи.

Отримання листа з атрибутом:

```
procedure TForm1.Button3Click(Sender: TObject);
var
i,numPosts: Integer;
begin
POP3.Host:='mail.58r.ru';
POP3.Port:=110;
POP3.Username:='test+58r.ru';
POP3.Password:='12345666′;
IdMessage.Clear;
POP3.Connect;
Memo1.Clear;
POP3.Retrieve(1,IdMessage);
Memo1.Lines.AddStrings(IdMessage.Body);

Label1.Caption := IdMessage.From.Text;
Label2.Caption := IdMessage.Recipients.EmailAddresses;
Label3.Caption := IdMessage.CCList.EMailAddresses;
Label4.Caption := IdMessage.Subject;
Label5.Caption := FormatDateTime('dd mmm yyyy hh:mm:ss', IdMessage.Date);
```

```
Label6.Caption := IdMessage.ReceiptRecipient.Text;
Label7.Caption := IdMessage.Organization;
POP3.Delete(1);
POP3.Disconnect;
end;
```

Відправлення пошти здійснюється по протоколі *SMTP*, компонент *IdSMTP*, ви зможете знайти його на вкладці *Indy Clients*. У даному прикладі відправлення пошти буде здійснюватися після авторизації на сервері, лист відправляється з адреси test@58r.ua, на майл test@58r.ua .

Відправка одного листа:

```
procedure TForm1.Button4Click(Sender: TObject);
begin
SMTP.Host:=’mail.58r.ua’;
SMTP.Port:=25;
SMTP.Username:=’test+58r.ua’;
SMTP.Password:=’12345666′;
SMTP.AuthenticationType:=atLogin;
with IdMessage do
begin
Body.Assign(Memo1.Lines);
From.Text := ‘test@58r.ua’;
Recipients.EMailAddresses := ‘test@58r.ua’;
Subject := ‘Programmersclub.ua’;
end;
SMTP.Connect;
try
showmessage(’Произошло подключение к серверу’);
SMTP.Send(IdMessage);
finally
SMTP.Disconnect;
end; end;
```

Практична частина:

Розробити додаток, що дає можливість отримання та відправки пошти по протоколу *SMTP*. Має можливість відправки та отримання декількох повідомлень. Передбачає обробку нестандартних ситуацій у додатку та отримання пошти з атрибутами.

Хід роботи.

1. Створити проект програмного додатку для роботи з протоколом *SMTP*. Розробити зовнішній інтерфейс.
2. Запрограмувати обробку нестандартних ситуацій.
3. Протестувати роботу програми.
4. Подумати над вдосконаленням додатку, додаванням нових функцій та можливостей.

ЗМІСТ