МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

МЕТОДИЧНІ ВКАЗІВКИ

до лабораторних робіт
з дисципліни
«КОРПОРАТИВНІ КОМП'ЮТЕРНІ МЕРЕЖІ»

для студентів спеціальності 123 «Комп’ютерна інженерія»
за освітньою програмою «Комп’ютерні інтелектуальні технології»,
галузі знань 12 «Інформаційні технології»

Електронне видання

ЗАТВЕРДЖЕНО
кафедрою КІТС
Протокол № 1від 29.08.2022

ХАРКІВ 2022

Методичні вказівки до лабораторних робіт з дисципліни «КОРПОРАТИВНІ КОМП'ЮТЕРНІ МЕРЕЖІ» для студентів спеціальності 123 «Комп’ютерна інженерія» за освітньою програмою «Комп’ютерні інтелектуальні технології», галузі знань 12 «Інформаційні технології» [Електронне видання] / Розр.: Н.М.Сердюк – Харків: ХНУРЕ, 2022. – 33с.

Розробник: к.т.н., доцент кафедри комп’ютерних інтелектуальних технологій та систем Сердюк Н.М.

ЗМІСТ

1. Дослідження взаємодії клієнта та сервера на основі протоколу TCP/IP. Реалізація паралельного з'єднання з використанням багато поточності.

1.1. Мета роботи

Навчитися використовувати протокол TCP/IP задля взаємодії клієнта та сервера при реалізації паралельного зєднання з використанням багатопоточності

1.2. Завдання: Здійснити взаємодії клієнта та сервера на основі протоколу TCP/IP, здійснити паралельне зєднання з використанням багатопоточності. Функціонування клієнта та сервера реалізувати наступним чином: клієнт вводить строку символів з клавіатури та відправляє серверу. Ознака закінчення вводу - натискання клавіши ***Enter***. Функціональні можливості сервера реалізовані наступним чином: сервер отримує строку, повинен визначити довжину введеної строки, і, якщо довжина є більшою за 15 символів, то з неї вилучаються всі цифри. Клієнт отримує перероблену строку та кількість вилучених цифр.

Опис алгоритму:

Клієнт відправляє строку на сервер.

Сервер обробляє інформацію та перевіряє довжину строки.

Якщо вона більша за 15, то посимвольно порівнюється кожний символ на належність до цифри та записує її до буферу.

Буфер відправляється клієнту.

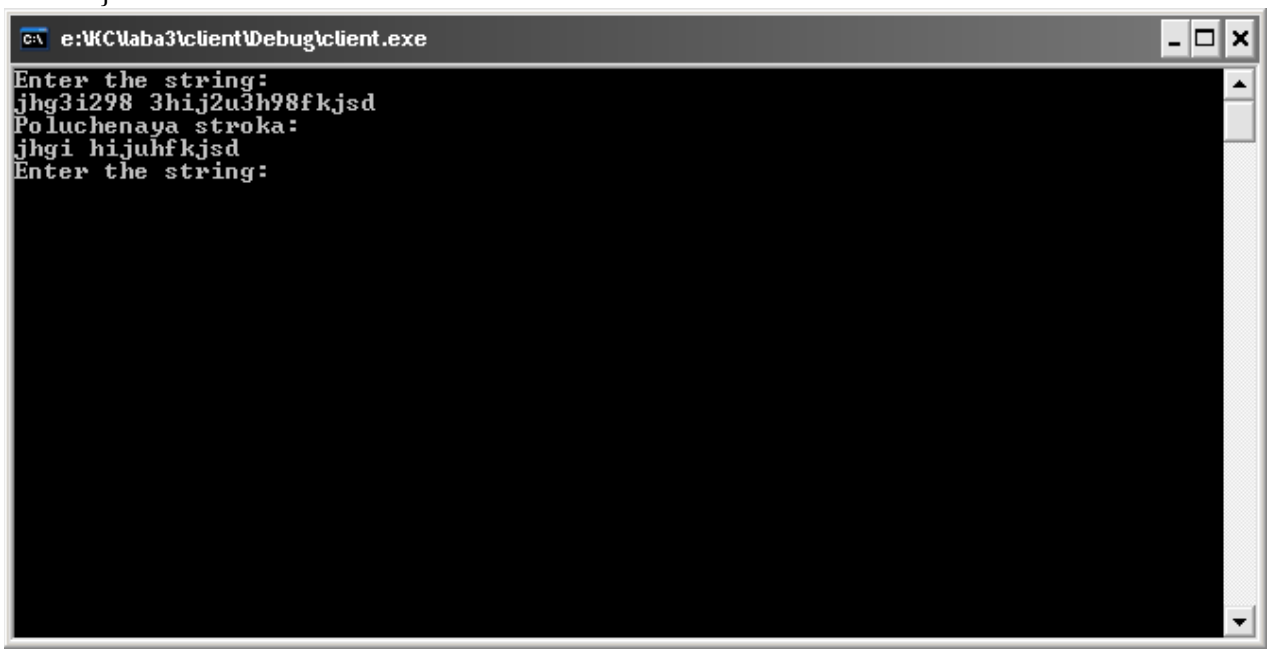
1.3. Порядок виконання роботи

Лістиіг:

Клієнт

```
#include<stdio.h>
#include<iostream>
#include<winsock2.h>

using namespace std;

void main()
{
	WORD wVersionRequested;
	WSADATA wsaData;
	int err;
	wVersionRequested = MAKEWORD( 2, 2 );
	err=WSAStartup( wVersionRequested, &wsaData );
	if ( err != 0 )
	{
		return;
```

```
	}

	while (true)
	{
		SOCKET s=socket(AF_INET,SOCK_STREAM,0);
		// Зазначення адреси та порту сервера
		sockaddr_in dest_addr;
		dest_addr.sin_family=AF_INET;
		dest_addr.sin_port=htons(1280);
		dest_addr.sin_addr.s_addr=inet_addr("127.0.0.1");
		connect(s,(sockaddr *)&dest_addr,sizeof(dest_addr));

		char buf[100];

		cout<<"Enter the string:"<<endl;
		fgets(buf,sizeof(buf),stdin);
		send(s,buf,100,0);

		if (recv(s,buf,sizeof(buf),0)!=0)
		{
			cout<<"Poluchenaya stroka:"<<endl<<buf<<endl;
		}
		closesocket(s);
	}
	WSACleanup();
}
```

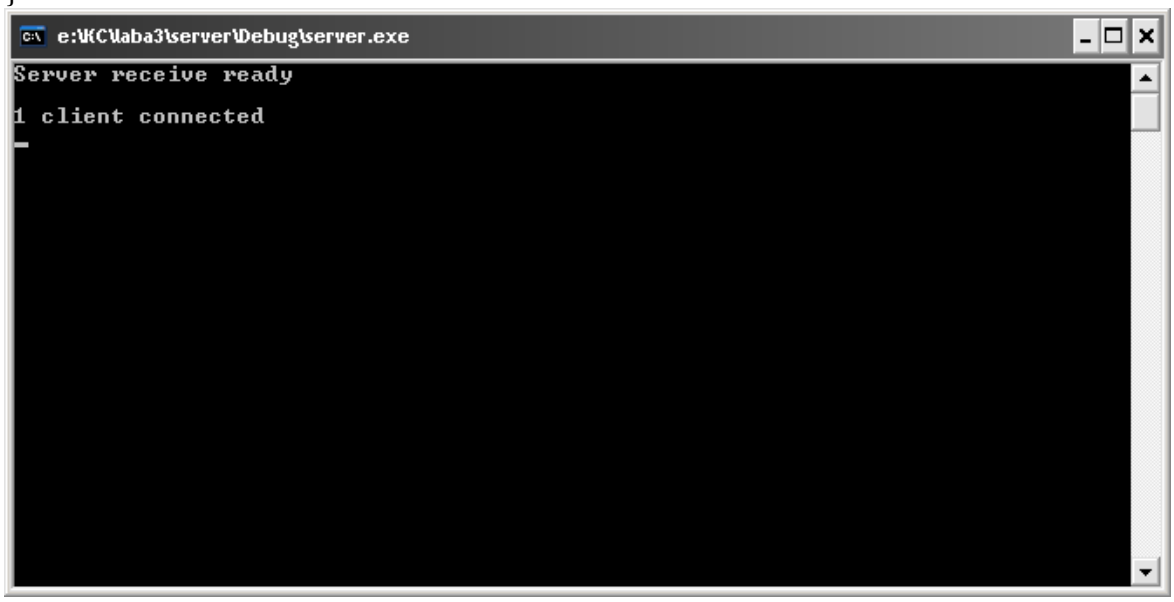


Сервер

```
#include<stdio.h>
#include<iostream>
#include<winsock2.h>

using namespace std;

DWORD WINAPI ThreadFunc(LPVOID client_socket)
{
```

```
	SOCKET s2=((SOCKET *) client_socket)[0];
	char buf[100];
	char buf1[100] = {0};
	char tmp[2] = {0};
// send(s2,"Welcome new client!\n",sizeof("Welcome new client!\n"),0);
	while(recv(s2,buf,sizeof(buf),0))
	{
		int k, j=0;
		int i;
		k=strlen(buf)-1;
//перевірка довжини строки
		if(k >= 15)
		{
//цикл перевірки, є символ числом або ні
			for(i = 0; i < k; i++)
			{
				if(!(buf[i] >= 48 && buf[i] <= 57))
				{
					tmp[0] = buf[i];
					tmp[1] = '\0';
					strcat(buf1, tmp);
					j++;
				}
			}

			buf1[k - (k - j)]='\0';
			strcpy(buf,buf1);
		}
		cout<<buf<<endl;
		send(s2,buf,100,0);
	}
	closesocket(s2);
return 0;
}

int numcl=0;

void print()
{
	if (numcl)
		printf("%d client connected\n",numcl);
	else
		printf("No clients connected\n");
}

void main()
{
	WORD wVersionRequested;
	WSADATA wsaData;
	int err;
	wVersionRequested = MAKEWORD( 2, 2 );
	err = WSAStartup( wVersionRequested, &wsaData );
	if ( err != 0 )
	{
		return;
	}
	SOCKET s=socket(AF_INET,SOCK_STREAM,0);
```

```
        sockaddr_in local_addr;
        local_addr.sin_family=AF_INET;
        local_addr.sin_port=htons(1280);
        local_addr.sin_addr.s_addr=0;
        bind(s,(sockaddr *) &local_addr,sizeof(local_addr));
        int c=listen(s,5);
        cout<<"Server receive ready"<<endl;
        cout<<endl;
// витягуємо повідомлення з черги
        SOCKET client_socket; // сокет для клієнта
        sockaddr_in client_addr; // адрес клієнта (заповнюється системою)
        int client_addr_size=sizeof(client_addr);
// Цикл отримання запитів на підключення з черги
        while((client_socket=accept(s,(sockaddr *)&client_addr, &client_addr_size)))
        {
                numcl++;
                print();
                // Виклик нового потоку для обслуговування клієнтів

                CreateThread(NULL,NULL,ThreadFunc,&client_socket,NULL,NULL);
        }
}
```



1.4. Зміст звіту

Звіт повинен містити:

1.Титульний лист із найменуванням лабораторної роботи, датою виконання і даними виконавців;

2. Мету роботи;

3. Результати роботи.

4. Висновки

1.5. Контрольні питання для захисту звіту

1.

2. Дослідження реалізації обмеження частоти відправки повідомлень з одного IP адреса за часом

2.1. Мета роботи

Навчитися використовувати протокол TCP/IP задля взаємодії клієнта та сервера при реалізації паралельного зєднання з використанням багатопоточності

2.2. Завдання: У протоколі відправлення SMS-повідомлень здійснити обмеження частоти відправки повідомлень з одного IP адреса за часом. Додати команду ***sms*** (без параметрів), яка б виводила правилв використання послугою

Опис алгоритму:

Реалізована команда ***sms*** в результаті чого клієнту видається інформація про заборону відправки більш 10 повідомлень за хвилину з одного IP адреса.

Клієнт відправляє строку на сервер у форматі:

sms <номер телефону> <текст повідомлення>

При підключенні клієнта до сервера порівнюється IPадреса клієнта, та записується до масиву.

Якщо така IP адреса вже існує, то лічильник відправки SMS-повідомлень буде додаватися с вже підключеним клієнтом з такою ж IP адресою.

При перевірці строки повідомлень записується час відправки повідомлень, а також працює лічильник відправки повідомлень.

Якщо кількість повідомлень перевищує 10 та пройшло більше хвилини, т о видається повідомлення про невдалу відправку повідомлення.

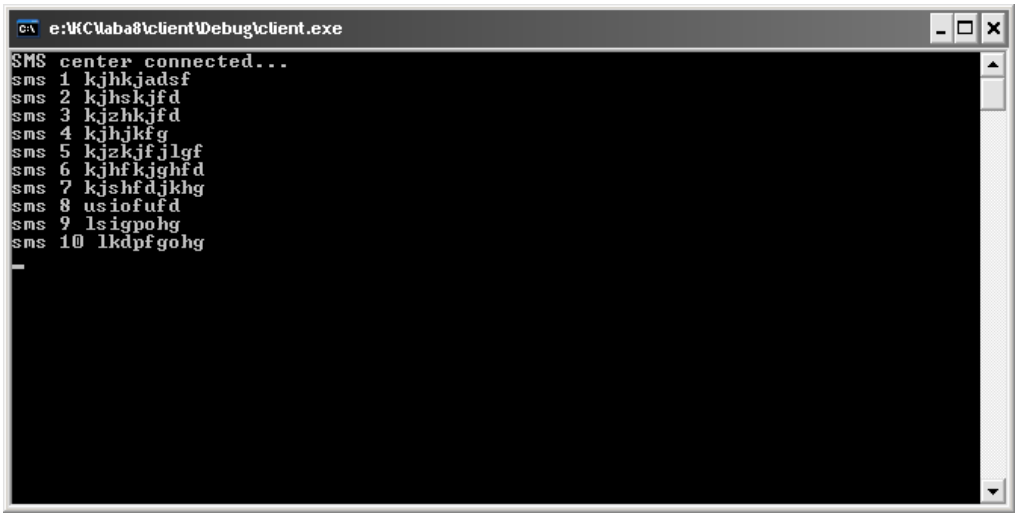
2.3. Порядок виконання роботи

Лістінг:

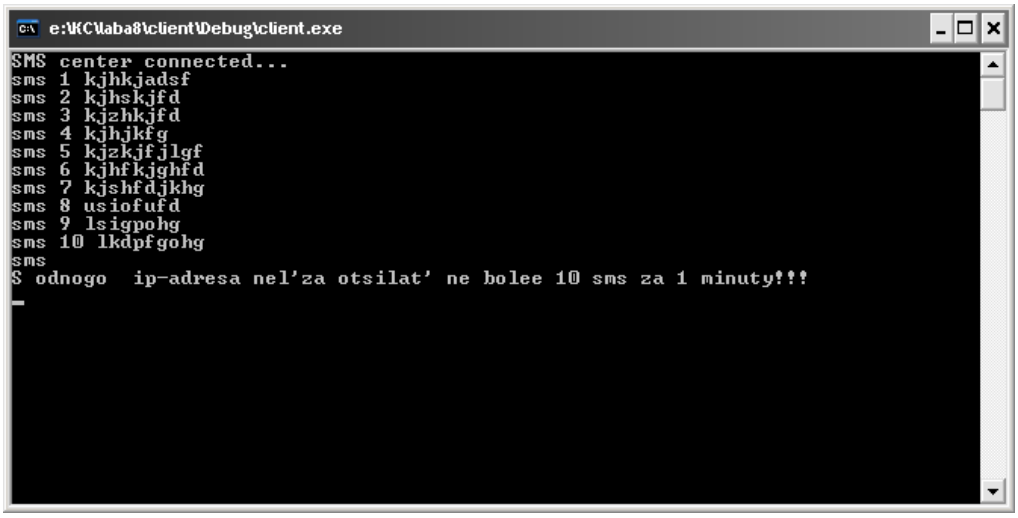
Клієнт

```
#include <winsock2.h>
#include <iostream>
using namespace std;
int main()
{
	WORD wVersionRequested;
	WSADATA wsaData;
	int err;
	wVersionRequested = MAKEWORD(2,2);
	err = WSAStartup(wVersionRequested, &wsaData);
	if(err != 0)
		return -1;
```

```
	struct sockaddr_in peer;
	peer.sin_family=AF_INET;
	peer.sin_port=htons(1280);
	peer.sin_addr.s_addr=inet_addr("127.0.0.1");

	SOCKET s=socket(AF_INET,SOCK_STREAM,0);
	connect(s,(struct sockaddr*) &peer,sizeof(peer));

	char b[200] = {0};
	char buf[200] = {0};

	recv(s,b,sizeof(b),0);
	cout<<b;
//	b[0]='\0';
	while (strcmp(b,"quit"))
	{
		cin.getline(b,200,'\n');
		send(s,b,sizeof(b),0);

		if(!strcmp(b,"sms"))
		{
			recv(s,b,sizeof(b),0);
			cout<<b;
		}
	}
	WSACleanup();
	return 0;
}
```



Демонстрація команди ***sms***.



Сервер

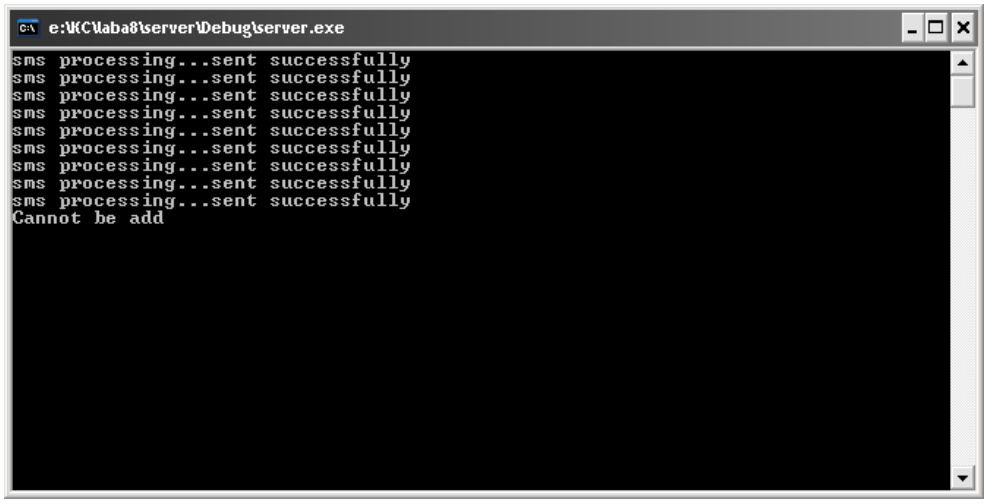
```
#include <iostream>
#include "afx.h"
#include <winsock2.h>
#include <process.h> /* _beginthread, _endthread */
#include <string.h>
#include <time.h>

using namespace std;

CFile f;
CFileException ex;
clock_t start, finish;
//структура зберігає кількість IP-адрес і номер адреси, з яким працює потік і т. д.
struct interthreads
{
	SOCKET newS;
	sockaddr_in local;
	int ip_kol;
	int number_node;
};
//структура для порівняння адреси і лічильника кількості текстових повідомлень, відправлених та часу відправки
struct valid
{
	int b1, b2, b3, b4;
	int sms_kol;
	int start, finish;
};

valid node_valid[100];

// видалити повідомлення з номером
void del(char* p,int n)
{
	char tel[200] = {0};
	int j=0;
	for (int i=n;p[i]!=' ';i++)
	{
```

```
                tel[j]=p[i];
                j++;
        }
//      tel[j]='\0';

        char fName[200];
        fName[0]='\0';
        strcat(fName,tel);
        DeleteFile((LPCWSTR)fName);
        cout<<"SMS is removed"<<endl;
}

//відіслати повідомлення до SMS-центру для номеру
void sms(char* p,int n)
{
        char tel[200];
        char str[200]; int j=0;
        for (int i=n;p[i]!=' ';i++)
        {
                tel[j]=p[i];
                j++;
        }
        tel[j]='\0';
        n=j+1;j=0;
        for(int i=n;p[i];i++)
        {
                str[j]=p[i];
                j++;
        }

        str[j]='\0';
        char fName[200] = {0};
//      fName[0]='\0';
        strcat(fName,tel);
        cout<<"sms processing...";

        if (!f.Open((LPCTSTR)fName,CFile::modeWrite | CFile::modeCreate,&ex))
        {
                cerr<<"SMS storage error. Try again\n";
                exit(EXIT_FAILURE);
        }
        f.Write(str,strlen(str));
        f.Close();
        cout<<"sent successfully"<<endl;
}

void SMSworking(void* newS)
{
        int c;
        char p[200] = {0};
        char com[200] = {0};
        interthreads* data;
        data = (interthreads*) newS;
//      com[0]='\0';p[0]='\0';
        strcat(p,"SMS center connected...\n");

        send((SOCKET)data->newS,p,sizeof(p),0);
```

```
		while((c=recv((SOCKET)data->newS,p,sizeof(p),0)!=0))
		{
	//команда, яка повертає інформацію про роботу SMS-сервісу клієнту
			if(!strcmp(p,"sms"))
			{
				strcpy(p,"S odnogo  ip-adresa nel'za otsilat' ne bolee 10 sms za 1 minuty!!!\n");
				send((SOCKET)data->newS,p,sizeof(p),0);
				continue;
			}

			int i=0;
			while (p[i]!=' ')
			{
				com[i]=p[i];
				i++;
			};
			com[i]='\0';i++;
			if (!strcmp(com,"sms"))
			{
				node_valid[data->number_node].sms_kol++;
				if(node_valid[data->number_node].sms_kol == 1)
					node_valid[data->number_node].start = clock();
				node_valid[data->number_node].finish = clock();
				// Перевірка кількості повідомлень, надісланих з цією IP-адресою, і часу, що минув
				if((double)(node_valid[data->number_node].finish - node_valid[data->number_node].start) / CLOCKS_PER_SEC < 60 && node_valid[data->number_node].sms_kol >= 10)
				{
					cout<<"Cannot be add"<<endl;
					node_valid[data->number_node].start = clock();
					node_valid[data->number_node].finish = clock();
					node_valid[data->number_node].sms_kol = 0;
				}
				else
					sms(p,i);

				com[0]='\0';
			}
			if (!strcmp(com,"del"))
			{
				finish = clock();
				// Якщо  було більше хвилини після відправки повідомлення
				if((double)(finish - start) / CLOCKS_PER_SEC > 60 )
					cout<<"Cannot be canceled"<<endl;
				else
					del(p,i);
				com[0]='\0';
			}
			if (!strcmp(com,"quit"))
			{
				closesocket((SOCKET)data->newS);
				exit(EXIT_SUCCESS);
				com[0]='\0';
			}
		}
	}
```

```
int main()
{
	WORD wVersionRequested;
	WSADATA wsaData;
	int err;

	wVersionRequested = MAKEWORD(2,2);
	err = WSAStartup(wVersionRequested, &wsaData);
	if(err != 0)
		return -1;

	sockaddr_in local;
	local.sin_family = AF_INET;
	local.sin_port = htons(1280);
	local.sin_addr.s_addr = htonl(INADDR_ANY);
	SOCKET s = socket(AF_INET, SOCK_STREAM,0);

	int c=bind(s,(struct sockaddr*)&local,sizeof(local));
	int r=listen(s,5);
	int b1, b2, b3, b4;
	interthreads params;
	params.ip_kol = 0;
	params.number_node = 0;

	while(true)
	{
		sockaddr_in remote;

		int j = sizeof(remote);
		int k = 0;
		SOCKET newS = accept(s, (struct sockaddr*) &remote, &j);
//копирование ип адреса в  буфер
		b1 = remote.sin_addr.S_un.S_un_b.s_b1;
		b2 = remote.sin_addr.S_un.S_un_b.s_b2;
		b3 = remote.sin_addr.S_un.S_un_b.s_b3;
		b4 = remote.sin_addr.S_un.S_un_b.s_b4;
	//проверка ип адреса
		if(params.ip_kol == 0)
		{
			params.number_node = params.ip_kol;
			node_valid[params.ip_kol].b1 = b1;
			node_valid[params.ip_kol].b2 = b2;
			node_valid[params.ip_kol].b3 = b3;
			node_valid[params.ip_kol].b4 = b4;
			params.ip_kol++;
			k = 1;
		}
		else
		{
			for(int i = 0; i < params.ip_kol; i++)
			{
				if(node_valid[i].b1 == b1 && node_valid[i].b2 == b2 &&
node_valid[i].b3 == b3 && node_valid[i].b4 == b4)
				{
					params.number_node = i;
					k = 1;
```

```
                                        break;
                                }
                        }
                }
                if(k == 0)
                {
                        params.number_node = params.ip_kol;
                        node_valid[params.ip_kol].b1 = b1;
                        node_valid[params.ip_kol].b2 = b2;
                        node_valid[params.ip_kol].b3 = b3;
                        node_valid[params.ip_kol].b4 = b4;
                        params.ip_kol++;
                }

                params.newS = newS;
                params.local = remote;

                _beginthread(SMSworking,0,&params);
        }
        WSACleanup();
        return 0;
}
```



2.4. Зміст звіту

Звіт повинен містити:

1.Титульний лист із найменуванням лабораторної роботи, датою виконання і даними виконавців;

2. Мету роботи;

3. Результати роботи.

4. Висновки

2.5. Контрольні питання для захисту звіту

3. Дослідження статичної та динамічної маршрутизації.

3.1. Мета роботи

Моделювання компонент та команд, що використовуються маршрутизаторами та комутаторами у невеликих мережах. Налаштування маршрутизаторів на базовому рівні. Конфігурування та усунення несправностей маршрутизаторів та комутаторів, розв'язування загальні проблеми протоколів RIP v2, та протоколів статичної маршрутизації.

3.2. Теоретична частина

Статична маршрутизація - вид маршрутизації, при якому інформація про маршрути заноситься в таблиці маршрутизації кожного маршрутизатора уручну адміністратором мережі. Статична маршрутизація має такі особливості:

• забезпечує підтримку ТМ для невеликих мереж, які не передбачено суттєво розширювати;

• забезпечує маршрутизацію для кінцевої (тупикової) мережі;

• задає єдиний маршрут за замовчуванням до будь-якої мережі, якщо ТМ не містить більш специфічного шляху.

Отже, аналізуючи вищесказане, можно навести переваги та недоліки статичної і динамічної маршрутизації.

Переваги статичної маршрутизації: мінімальне використання процесора; легша для розуміння адміністратора; легша для конфігурування.

Недоліки статичної М: конфігурування та обслуговування потребує багато часу; під час конфігурування можливі помилки (особливо у великих мережах); для підтримки заміни маршрутної інформації потрібне втручання адміністратора; зі зростанням мережі погано масштабується; для належного виконання потребує повного знання усієї мережі.

Переваги динамічної маршрутизації: потребує меншого втручання адміністратора, при доданні або вилученні мереж; протоколи автоматично реагують на зміни топології; конфігурація менш схильна до помилок; більш масштабована, нарощування мережі зазвичай не породжує проблем.

Недоліки динамічної маршрутизації: використовуються ресурси маршрутизатора; потребує більших знань адміністратора для конфігурування, перевірки та усунення несправностей.

Таблиця 3.1 – Порівняння статичної та динамічної маршрутизації

Критерій порівняння	Статична маршрутизація	Динамічна маршрутизація
Складність конфігурування	Ускладнюється зі зростанням складності мережі	В загальному плані не залежить від складності мережі
Вимоги до знань адміністратора	Потрібен невисокий рівень знань	Потрібен більший рівень знань
Зміни Топології	Потрібне адміністративне втручання	Автоматично адаптується під зміни
Масштабування	Підходить лише для простих топологій	Підходить і для складних і для простих топологій
Ступінь безпеки	Більш безпечна, ніж динамічна маршрутизація	Не гарантує безпеки
Ступінь застосовування ресурсів	Не потребує додаткових ресурсів	Застосовує процесор, оперативну пам'ять, смугу пропускання
Передбачуваність	Маршут завжди постійний	Маршрут залежить від поточної топології

Протокол маршрутної інформації (Routing Information Protocol – RIP) початково був визначений в документі RFC 1058 в 1988 році. Найбільш суттєвими є такі його характеристики:

• RIP є дистанційно-векторним протоколом маршрутизації;

• як метрики при виборі маршруту використовується кількість переходів (або хопів);

• якщо кількість переходів більше ніж 15, пакет відкидається;

• стандартно оновлення маршрутизації розсилаються широкомовним способом кожних 30 секунд.

Протокол RIP значно еволюціонував: від основаного на класах протоколу першої версії (RIPv1) до безкласового протоколу другої версії (RIPv2). Вдосконалення останнього такі:

• можливість переносити додаткову інформацію про маршрутизацію пакетів;

• механізм аутентифікації для забезпечення безпечного онов-лення ТМ;

• підтримка масок змінної довжини.

Протокол RIP призначений для невеликих мереж, в яких застосування простого протоколу дозволяє спростити проектування і скоротити тривалість настройки конфігурації. По суті, якщо мережа є достатньо невеликою для того, щоб в ній можна було застосовувати протокол RIP, то краще зупинитися на протоколі RIP.

3.3. Порядок виконання роботи

Після задання адміністратором статичного маршруту маршрутизатор запам'ятовує його у своїй таблиці маршрутизації (ТМ) і використовує для пересилання пакетів. Команда задання статичного маршруту має такий синтаксис:

Rt(config)#ip route prefix mask {ip | int-type int-num}[dist],

де prefix, mask – IP-адреса та маска пункту призначення, відповідно; ip, int-type, int-num – IP-адреса порту наступного транзитного переходу (хопа), тип та номер локального інтерфейсу на які слід надіслати пакет, котрий повинен дістатися вищевказаного пункту призначення; dist – адміністративна

відстань. Адміністративна відстань (АВ) – це необов’язковий параметр, який характеризує надійність маршруту. Чим менша АВ, тим надійнішим є маршрут. Маршрут з меншою адміністративною відстанню буде занесений у ТМ.

Конфігурування статичних маршрутів

Для конфігурування статичних маршрутів слід виконати такі кроки.

1. Визначити усі мережі-отримувачі, їх маски та шлюзи (як адресу шлюзу можна вказати або локальний інтерфейс маршрутизатора або адресу наступного хопа, на шляху до потрібного пункту призначення).
2. Увійти в режим глобального конфігурування.
3. Ввести команду ip route з відповідними параметрами як показано вище.
4. Повторити третій крок для усіх мереж-отримувачів, до яких слід задати статичний маршрут.
5. Вийти з режиму глобального конфігурування.
6. Виконати команду ***copy running-config startup-config***.

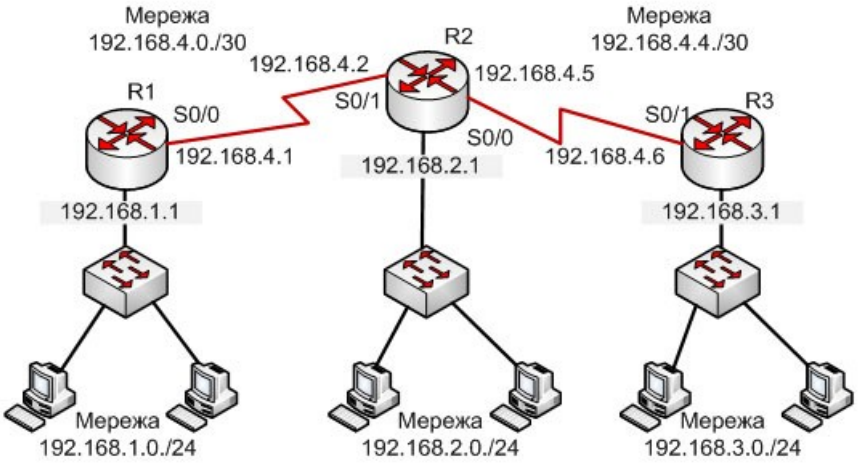


Рис.1 – Комп'ютерна мережа

Наприклад, для мережі наведеної на рис. 1 команди задання статичних маршрутів на прикладі маршрутизатора R2 будуть

R2(config)#ip route 192.168.1.0 255.255.255.0 192.168.4.1,

R2(config)#ip route 192.168.3.0 255.255.255.0 192.168.4.6,

(статичні маршрути до усіх інших мереж 192.168.2.0/24, 192.168.4.0/30, 192.168.4.4/30 маршрутизатор R2 знає, оскільки вони безпосередньо під’єднані до нього). У вищенаведених командах вказано IP-адреси наступних хопів на шляху до отримувачів. Якщо вказувати вихідні інтерфейси, команди задання статичних шляхів набудуть вигляду

R2(config)#ip route 192.168.1.0 255.255.255.0 s0/1,

R2(config)#ip route 192.168.3.0 255.255.255.0 s0/0.

Зауважимо, що дані та дві попередні команди для даного випадку еквівалентні. Єдина відмінність між ними полягає в тому, що будуть різні значення АВ. Стандартно, при застосуванні адреси наступного переходу АВ = 1, а вихідного інтерфейсу – АВ = 0.

Взагалі значення АВ – цілі числа в діапазоні від 0 до 255. Якщо треба ввести нестандартну адміністративну відстань (наприклад, яка дорівнює 140) то слід задати команду

R3(config)#ip route 192.168.1.0 255.255.255.0 192.168.4.5 140

Якщо маршрутизатор з деяких причин не може використовувати вихідний інтерфейс, заданий у маршруті – то цей маршрут не буде використовуватись, тобто не буде занесено до ТМ.

Іноді статичні маршрути використовують як резервні, котрі будуть використовуватись лише у випадку, якщо не вдається надіслати дані за динамічним маршрутом. В такому випадку АВ повинна бути більшою, ніж у маршруту, отриманого протоколом динамічної маршрутизації.

Зазначимо, що на маршрутизаторі R1 аналогічно слід прописати шляхи до мереж 192.168.2.0/24, 192.168.3.0/24 та 192.168.4.4/30, а на R3 – до мереж 192.168.1.0/24, 192.168.2.0/24 та 192.168.4.0/30. Проте замість того, щоб прописувати ці три шляхи, на маршрутизаторах R1 та R3 можна вказати лише по одному маршруту за замовчуванням.

Задання маршруту за замовчуванням

Маршрути за замовчуванням (або стандартні маршрути) використовуються маршрутизаторами у випадку, коли адреса мережі-отримувача пакета не збігається з жодним маршрутом ТМ. Стандартні маршрути, як правило, конфігуруються для передавання потоків даних через мережу Internet, оскільки нераціонально і немає необхідності підтримувати усі маршрути до всіх мереж Internet. Таким чином, стандартні маршрути дозволяють скоротити число записів у ТМ і зменшити час їх оброблення.

Задати стандартний маршрут можна за допомогою команди

ip route 0.0.0.0 0.0.0.0 {ip | int-type int-num}.

Операція логічного „І” над маскою 0.0.0.0 та IP-адресою пакета завжди дає результатом мережу 0.0.0.0. Якщо для пакета в ТМ не знаходиться відповідності мережі-отримувача – він надсилається у ме-режу 0.0.0.0.

Перевірка і усунення помилок у конфігурації статичних маршрутів

Після того, як статичні маршрути сконфігуровані – слід впевнитись, що вони є у ТМ і пересилка пакетів за ними виконується правильно. Для цього можна використати команди show running-config та show ip route. Перша команда дозволяє проглянути статичні маршрути у файлі робочого конфігурування маршрутизатора, а друга – у його ТМ. При цьому, якщо деякий маршрут введено неправильно – його слід вилучити, а замість ввести правильний.

Для пошуку та усунення помилок в конфігуруванні статичних маршрутів пропонується виконати такі кроки.

1. Впевнитись, що канал, який буде використовуватись як шлюз є доступним.

2. Виконати команду show interfaces і впевнитись у активнос-ті інтерфейсу і канального протоколу.

3. Перевірити правильність IP-адреси на інтерфейсі.

4. Виконати команду ping для IP-адреси віддаленого маршрутизатора, безпосередньо під'єднаного до шлюзу маршруту. Якщо результат цієї команди буде негативний – то проблема не пов'язана з маршрутизацією.

5. Якщо команда не спрацьовує на дальньому маршрутизаторі – слід виконати команду ***traceroute*** – для визначення вузла, де губиться пакет.

6. Під'єднатись до маршрутизатора, на якому не спрацювало трасування і виконати дії, описані у першому кроці.

7. Якщо команда ping спрацювала на дальньому кінці маршруту – тест можна вважати успішним і завершеним.

Конфігурування динамічних маршрутів протоколу RIP

Для конфігурування протоколу RIP слід увійти в режим конфігурування протоколу RIP за допомогою команди ***router rip*** (зупинити роботу протоколу RIP зі стиранням його конфігурування можна за допомогою команди no router rip). Далі за допомогою команди ***Rt(config-router)#network ip-directly-connected-classful-net*** слід вказати класові мережі, які безпосередньо під'єднані до даного маршрутизатора і повинні ним анонсуватись. Вона дозволяє надсилати та отримувати RIP-оновлення для інтерфейсів, що належать до цих мереж, а також дані мережі у RIP-повідомленнях**.**

Зауважимо, що у випадку, коли до маршрутизатора безпосередньо під'єднано кілька підмереж одного класу, то достатньо вказати лише одну цю класову мережу. Якщо ж вказати підмережу, то IOS автоматично конвертує її у повнокласову адресу (наприклад, якщо за-дати команду network 192.168.1.64 – маршрутизатор сприйме її як network 192.168.1.0)

Приклади таких настроювань для маршрутизаторів R1 – R3 мережі:

```
R1(config)# router rip
R1(config-router)# network 192.168.1.0
R1(config-router)# network 192.168.4.0

R2(config)# router rip
R2(config-router)# network 192.168.2.0
R2(config-router)# network 192.168.4.0

R3(config)# router rip
R3(config-router)# network 192.168.3.0
R3(config-router)# network 192.168.4.0
```

Зауважимо, що команди ***router rip*** та *network* є обов'язковими для настроювання протоколу.

За замовчуванням програмне забезпечення отримує пакети RIPv1 та RIPv2, а надсилає лише пакети RIPv1. Як відомо, протокол RIPv1 не підтримує технологію VLSM, і якщо у мережі використовуються маски змінної довжини цей протокол буде працювати некоректно, отже в такому випадку слід використовувати RIPv2.

Для того, щоб сконфігурувати маршрутизатор на відправлення та отримання пакетів лише однієї версії протоколу RIP слід використовувати такі команди: ***Router(config-router)# version {1 | 2}*** – вказує IOS на необхідність відправлення лише пакетів версії RIPv1 або RIPv2.

Router(config-router)# ip rip send version XX – конфігурує інтерфейс для відправлення пакетів протоколу RIP певної версії (XX може приймати значення "1" або "2" або "1 2", в останньому випадку приймаються пакети версії 1 або 2). Конфігурування інтерфейсу для отримання пакетів протоколу RIP певної версії виконується аналогічно. Відповідна команда конфігурування має синтаксис ***R1(config-router)# ip rip receive version XX***.

Після виконання цих команд можна проглянути ТМ на маршрутизаторах за допомогою команди show ip route. Для маршрутизатора R1 ТМ має вигляд:

C 192.168.1.0/24 is directly connected, FastEthernet0/0
R 192.168.2.0/24 [120/1] via 192.168.4.2, 00:00:21, Se-rial0/0
R 192.168.3.0/24 [120/2] via 192.168.4.2, 00:00:21, Se-rial0/0
192.168.4.0/30 is subnetted, 2 subnets
C 192.168.4.0 is directly connected, Serial0/0
R 192.168.4.4 [120/1] via 192.168.4.2, 00:00:21, Se-rial0/0

Для маршрутизатора R2 ТМ така

R 192.168.1.0/24 [120/1] via 192.168.4.1, 00:00:05, Se-rial0/1
C 192.168.2.0/24 is directly connected, FastEthernet0/0
R 192.168.3.0/24 [120/1] via 192.168.4.6, 00:00:00, Se-rial0/0
192.168.4.0/30 is subnetted, 2 subnets
C 192.168.4.0 is directly connected, Serial0/1
C 192.168.4.4 is directly connected, Serial0/0

Для маршрутизатора R3 ТМ така:

R 192.168.1.0/24 [120/2] via 192.168.4.5, 00:00:27, Serial0/1
R 192.168.2.0/24 [120/1] via 192.168.4.5, 00:00:27, Serial0/1
C 192.168.3.0/24 is directly connected, FastEthernet0/0
192.168.4.0/30 is subnetted, 2 subnets
R 192.168.4.0 [120/1] via 192.168.4.5, 00:00:27, Serial0/1
C 192.168.4.4 is directly connected, Serial0/1

В першому стовпці ТМ є символи, які вказують на джерело отримання даного маршруту. С – вказує на те, що це безпосередньо під’єднана до даного маршрутизатора мережа (даний запис з’являється в ТМ в результаті настроювання певного порту маршрутизатора), а R – що даний маршрут отриманий від протоколу RIP.

Розглянемо складові маршрутів у ТМ, наприклад, третього рядка таблиці маршрутизації маршрутизатора R1

R 192.168.3.0/24 [120/2] via 192.168.4.2, 00:00:21, Se-rial0/0

Тут 192.168.3.0/24 – адреса мережі призначення з її маскою;

[120/2]– адміністративна відстань і після слешу метрика маршруту;

192.168.4.2 – IP-адреса порту сусіднього пристрою через який отримано даний рядок;

00:00:21 – час, що минув з моменту отримання даного маршруту (пройшло 21 секунда, наступне поновлення повинно відбутись через 9 секунд);

Serial0/0 – тип та номер локального порту маршрутизатора, на який слід надіслати пакет, щоб він дістався вищевказаного пункту призначення.

Автосумаризація маршрутів

Розглянемо випадок, коли вищенаведені команди конфігурування протоколу RIP не приведуть до коректної роботи мережі. Так, якщо у складеній мережі існують підмережі, що належать одній класовій мережі, але під’єднані до різних маршрутизаторів, таблиці маршрутизації до цих підмереж будуть некоректними. Наприклад, якщо у мережі, наведеній на рис. 1 мережу 192.168.1.0/24 розбити на дві підмережі 192.168.1.0/25 і 192.168.1.128/25, перша з яких буде під’єднана до порту Fa0/0 маршрутизатора R1, а друга – Fa0/0 маршрутизатора R3 виникне така проблема. Маршрутизатори R1 і R3 будуть виконувати автоматичну сумаризацію підмереж у класову мережу, до якої входять дані підмережі (сумаризація виконується на граничному для цих підмереж маршрутизаторі, тобто маршрутизаторі, який має одну або більше підмереж, що входять до мережі певного класу і з’єднується з іншою частиною мережі через мережу, що не належить вищевказаній класовій мережі) і повідомляти маршрутизатор R2 про те, що мають безпосередній зв’язок з мережею 192.168.1.0/24. При цьому маршрутизатор R2 вважатиме, що існує два оптимальних шляхи до мережі 192.168.1.0/24. ТМ маршрутизатор R2 буде мати вигляд:

R 192.168.1.0/24 [120/1] via 192.168.4.1, 00:00:11, Serial0/1
[120/1] via 192.168.4.6, 00:00:23, Serial0/0
C 192.168.2.0/24 is directly connected, FastEthernet0/0
192.168.4.0/30 is subnetted, 2 subnets
C 192.168.4.0 is directly connected, Serial0/1
C 192.168.4.4 is directly connected, Serial0/0

ТМ для R1 буде такою

192.168.1.0/25 is subnetted, 1 subnets
C 192.168.1.0 is directly connected, FastEthernet0/0
R 192.168.2.0/24 [120/1] via 192.168.4.2, 00:00:25, Serial0/0
192.168.4.0/30 is subnetted, 2 subnets

C 192.168.4.0 is directly connected, Serial0/0
R 192.168.4.4 [120/1] via 192.168.4.2, 00:00:25, Serial0/0

ТМ для R3 буде такою

192.168.1.0/25 is subnetted, 1 subnets
C 192.168.1.128 is directly connected, FastEthernet0/0
R 192.168.2.0/24 [120/1] via 192.168.4.5, 00:00:24, Serial0/1
192.168.4.0/30 is subnetted, 2 subnets
R 192.168.4.0 [120/1] via 192.168.4.5, 00:00:24, Serial0/1
C 192.168.4.4 is directly connected, Serial0/1

В результаті маршрутизація в такій мережі буде неправильною. Наприклад, якщо з маршрутизатора R2 пропінгувати будь-який вузол мережі 192.168.1.0/25 або 192.168.1.128/25 пакети, внаслідок балансування навантаження будуть циклічно (почергово) передаватись різними шляхами через інтерфейси S0/0 та S0/1. Крім того, маршрутизатор R1 не містить маршруту до мережі 192.168.1.128/25, а R3 – мережі 192.168.1.0/25. Очевидно, що така ситуація недопустима.

Для вирішення даної ситуації на кожному маршрутизаторі слід відмінити автоматичну сумаризацію маршрутів за допомогою команди no auto-summary в режимі конфігурування протоколу RIP. Після цього ТМ маршрутизаторів набувають вигляду для R1:

192.168.1.0/25 is subnetted, 2 subnets
C 192.168.1.0 is directly connected, FastEthernet0/0
R 192.168.1.128 [120/2] via 192.168.4.2, 00:00:02, Serial0/0
R 192.168.2.0/24 [120/1] via 192.168.4.2, 00:00:02, Serial0/0
192.168.4.0/30 is subnetted, 2 subnets
C 192.168.4.0 is directly connected, Serial0/0
R 192.168.4.4 [120/1] via 192.168.4.2, 00:00:02, Serial0/0;

для R2:

192.168.1.0/25 is subnetted, 2 subnets
R 192.168.1.0 [120/1] via 192.168.4.1, 00:00:06, Serial0/1
R 192.168.1.128 [120/1] via 192.168.4.6, 00:00:01, Serial0/0
C 192.168.2.0/24 is directly connected, FastEthernet0/0
192.168.4.0/30 is subnetted, 2 subnets
C 192.168.4.0 is directly connected, Serial0/1
C 192.168.4.4 is directly connected, Serial0/0;

для R3:

192.168.1.0/25 is subnetted, 2 subnets
R 192.168.1.0 [120/1] via 192.168.4.1, 00:00:06, Serial0/1
R 192.168.1.128 [120/1] via 192.168.4.6, 00:00:01, Serial0/0
C 192.168.2.0/24 is directly connected, FastEthernet0/0
192.168.4.0/30 is subnetted, 2 subnets
C 192.168.4.0 is directly connected, Serial0/1
C 192.168.4.4 is directly connected, Serial0/0.

Вимкнення маршрутних оновлень

Ще одна проблема – небажане надсилання оновлень з деяких інтерфейсів. Справа в тому, що під час застосування команди network протокол RIP надсилає інформацію про маршрут до вказаної в цій команді мережі зі всіх інтерфейсів у діапазоні адрес цієї мережі. Для вимкнення відправлень (але не отримувань) таких оновлень з окремих інтерфейсів можна скористатися командою passive-interface. Так, повертаючись до нашої мережі (див рис. 1) бачимо, що надсилати RIP-оновлення недоцільно у порти fa0/0 маршрутизаторів R1 – R3, оскільки до відповідних мереж не під'єднані інші маршрутизатори, а лише робочі станції. Крім недоцільності такі оновлення ще будуть породжувати зайвий службовий трафік, який знижує пропускну спроможність мережі і надає можливість зловмисникам аналізувати ці оновлення. Отже, на маршрутизаторах R1 – R3 слід набрати команду ***Router(config-router)# passive-interface fa0/0***.

Застосування команди ip classless

Іноді маршрутизатор отримує пакети, призначені для невідомої підмережі деякої мережі, яка входить у безпосередньо під'єднані мережі пристрою. Для пересилання цих пакетів за найкращим шляхом використовується команда глобального конфігурування **ip classless**. Коли цю функцію вимкнено і пакет надсилається у підмережу мережі, до якої немає стандартного маршруту – пакет відкидається маршрутизатором.

Команда ip classless не впливає на ТМ, а лише на операцію пересилання пакета. Якщо маршрутизатор отримує пакет з невідомою адресою отримувача, що знаходиться в невідомій підмережі під'єднаної мережі, то передбачається, що такої підмережі не існує. Тому маршрутизатор відкидає пакет навіть якщо існує стандартний маршрут. Виконання команди ip classless – вирішує цю проблему за рахунок вказання маршрутизатору заснованої на класах межі мереж в його ТМ і просто вибирати стандартний маршрут.

Нагадаємо, що під час отримання інформації про мережу RIP-маршрутизатори покладаються на сусідні маршрутизатори. Протоколам RIP, як будь-якому ДВП властиві проблеми, що спричинюють повільну конвергенцію. Для уникнення петель маршрутизації і зациклювання пакетів протокол RIP використовує: розщеплення горизонту (**Split horizon**); вилучення маршрутів у зворотному напрямку (**Poison reverse**); таймери утримання інформації (**Holddown counters**); миттєві (тригерні) оновлення (**Triggered updates**). Деякі з цих методів потребують додаткового конфігурування, деякі – ні. В деяких випадках потрібно вимкнути механізм розщеплення горизонту. Таке вимкнення виконується за допомогою комади ***Router(config-if)#no ip split-horizon***.

Встановлення значень таймерів

Ще один механізм, який може потребувати змін – це застосування таймера утримання інформації. Такий таймер дозволяє попередити

зациклювання пакетів, проте збільшує час конвергенції мережі. Стандартно **Holddown counters** складає 180 секунд. Протягом цього часу не дозволяється оновлення внутрішніх маршрутів, однак і дійсні альтернативні маршрути також не будуть встановлюватись. Для прискорення конвергенції час **Holddown** counters може бути зменшено. В ідеальному випадку – це встановлення цього періоду утримання трошки більшим за максимальний час оновлення маршрутів у даній об'єднаній мережі.

Для змінення періоду таймера утримання інформації використовується команда ***Rt(config-router)#timers basic update invalid holddown flush [sleeptime]***, де **update** – таймер оновлення (стандартно 30 секунд) – задає період розсилання оновлень про маршрути; **invalid** – таймер дійсності маршруту (стандартно 180 секунд) – задає час, протягом якого маршрутизатор при відсутності анонсів про оновлення деякого маршруту чекає, перед тим, як об'явити цей маршрут недійсним. Маршрут зберігатиметься у ТМ поки не спливе час таймера скидання маршрутів **flush; holddown** – таймер утримання (стандартно 180 секунд) – задає час, протягом якого нові повідомлення про оновлення маршрутизації ігноруються; **flush** – таймер скидання маршрутів – задає час, який проходить до того, як маршрут буде вилучений з ТМ (стандартно 240 секунд).

Додатковим параметром, що впливає на час конвергенції і підлягає конфігуруванню є інтервал розсилання повідомлень оновлень маршрутів (за замовчуванням кожні 30 секунд). Цей час можна збільшити (для економії смуги пропускання) або зменшити (для скорочення часу конвергенції) за допомогою команди ***Router(config-router)#update-timer seconds***.

Під час конфігурування протоколу RIP можна встановити кількість паралельних маршрутів. Стандартно для більшості протоколів динамічної маршрутизації в ТМ встановлюється до чотирьох таких маршрутів (для статичних маршрутів їх може бути шість). Для змінення стандартної кількості паралельних маршрутів можна використати команду ***Router(config-router)#maximum-paths [number].***

Анонсування статичних та стандартних маршрутів

Статичні маршрути, вказані на деякому інтерфейсі за замовчуванням не анонсуються протоколом RIP. Якщо статичний маршрут призначений інтерфейсу, який не вказаний у команді network – то жоден протокол маршрутизації не анонсує такий маршрут. Дозволити анонсування можна за допомогою команди ***redistribute static***.

Стандартні маршрути (маршрути за замовчуванням), протоколом RIP за замовчуванням також не анонсуються. Якщо треба, щоб такі маршрути включалися до RIP-анонсів, на маршрутизаторі, де вказано маршрут за замовчуванням, слід виконати команду ***default-information originate***.

3.4. **Зміст звіту**

Звіт повинен містити:

1.Титульний лист із найменуванням лабораторної роботи, датою виконання і даними виконавців;

2. Мету роботи;

3. Результати роботи.

4. Висновки

3.5. **Контрольні питання для захисту звіту**

1. Наведіть команди настроювання статичного маршруту та маршруту за замовчуванням.
2. Поясніть, яку роль відіграє маршрут за замовчуванням. В яких випадках він використовується?
3. Покажіть на власному прикладі застосування команд статичної маршрутизації.
4. Наведіть команди пошуку та усунення помилок у настроюванні статичних маршрутів.
5. Охарактеризуйте протокол маршрутизації RIP. Наведіть його обмеження, переваги та недоліки.
6. Наведіть загальний алгоритм функціонування протоколу RIP.
7. Поясніть, як відбувається адаптація RIP-маршрутизаторів до змін станів мережі.
8. Наведіть причини виникнення петель маршрутизації.
9. Наведіть основні методи боротьби з фальшивими маршрутуми в протоколі RIP. Поясніть сутність цих методів.
10. Наведіть основні команди конфігурування протоколу RIP.
11. Поясніть що таке автосумаризація маршрутів. Які переваги та недоліки використання автосумаризації Ви знаєте?
12. Поясніть, з якою метою в протоколі RIP вимикають маршрутні оновлення через певні інтерфейси. Наведіть відповідну команду.
13. Поясніть застосування команди ***ip classless*** для протоколу RIP.
14. Які таймери використовуються у протоколі RIP та з якою метою? Наведіть команду змінення значень цих таймерів.

15. За допомогою якої команди у протоколі RIP можна встановити кількість паралельних маршрутів.

16. Поясніть, як налаштувати протокол RIP, щоб він анонсував статичні маршрути та маршрут за замовчуванням. Наведіть відповідні команди.

17. Наведіть кілька основних команд тестування та усунення помилок у роботі протоколу RIP та поясніть їх функції.

4. Налаштування маршрутизації та розсилки даних в мережі за допомогою Cisco Packet Tracer

4.1. Мета роботи

Моделювання мережі та мережевих систем, дослідження візуалізації, розробки та навчання складних технологічних принципів, розробка навичок виправлення неполадок з багатофункціональної програми мережевих моделей Cisco Packet Tracer.

4.2. Теоретична частина

Статична маршрутизація - вид маршрутизації, при якому інформація про маршрути заноситься в таблиці маршрутизації кожного маршрутизатора уручну адміністратором мережі. Як можна зрозуміти - з принципу організації даного вигляду відразу ж витікає ряд недоліків. Перш за все, це дуже погана масштабованість, оскільки при додаванні нової підмережі у велику мережу, де вже є N підмереж, потрібно буде зробити $2(N+1)$ записів про маршрути, причому при достатньо великої сегментації мережі (кількість підмереж понад 4…6) таблиця маршрутизації на кожному з маршрутизаторів сильно відрізнятиметься від таблиць на інших пристроях. Іншим недоліком є неможливість відстежити проблеми, що виникають внаслідок помилок на устаткуванні канального рівня, коли передача даних неможлива, а порт маршрутизатора як і раніше знаходиться в активному стані (стан up).

Необхідність уручну задавати маршрути також спричиняє за собою необхідність документування цих маршрутів. Всі ці проблеми вирішуються у протоколах **динамічної маршрутизації** за допомогою передачі службової широкомовної інформації в мережу.

Проте статична маршрутизація продовжує успішно використовуватися при організації роботи комп’ютерних мереж невеликого розміру (1-2 маршрутизатора), через легкість конфігурації та відсутності додаткового навантаження на мережу у вигляді широкомовного службового трафіку, характерного для динамічних протоколів маршрутизації. Також статична маршрутизація використовується на комп’ютерах усередині мережі. У такому разі зазвичай задається маршрут шлюзу за замовчуванням. При використанні статичних записів процесору маршрутизатора не потрібно проводити ніяких розрахунків, пов’язаних з визначенням маршрутів.

Статична маршрутизація в ОС Linux налаштовується за допомогою консольних ***утиліт route*** та ***ip route***. Дані команди дозволяють переглядати таблицю маршрутизації хоста, а так само видаляти і додавати записи в неї.

4.3. Порядок виконання роботи.

Завдання: побудувати сегмент корпоративної мережі, що складається з ***n*** комутаторів, а також ***а*** комп'ютерів і ***в*** серверів. Комутатор підключено до

маршрутизатора, до якого підключений сервер.
Встановити статичні IP-адреси для мережевих інтерфейсів маршрутизаторів, локальних комп'ютерів і серверів.
Налаштуйте маршрутизацію відповідно до протоколу RIP. Для забезпечення того, щоб дані протоколу ICMP могли бути переадресовані між усіма об'єктами в мережі.
Налаштуйте статичну маршрутизацію за варіантом завдання.

№ варіанту	***n***	***a***	***в***
1	2	8	2
2	1	4	1
3	2	12	1
4	1	5	2
5	3	6	2

1. Помістіть необхідні вузли на робоче поле за допомогою браузера в нижній частині вікна (рис. 1). З'єднайте вузли за завданням витою парою. Сервер маршрутизатора з'єднується з кросовером. З'єднання, позначені зеленим кольором, вказують на те, що вони активні, помаранчеві - що вони знаходяться на стадії з'єднання, червоний - не працює.

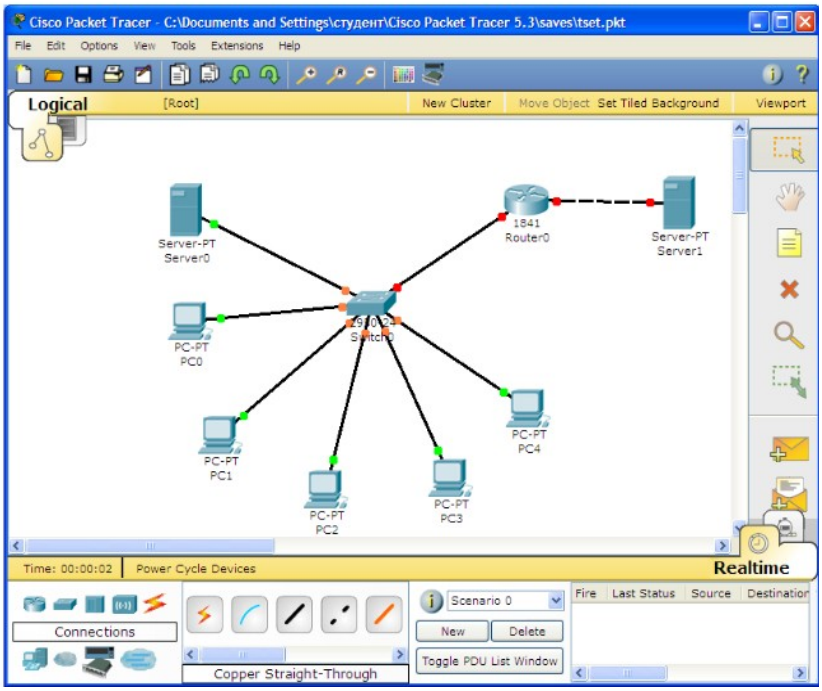


Рис. 1.Робоче поле

2. Задамо IP-адресу вузлам сегменту в діапазоні 192.168.0.x, а серверу, підключеному до маршрутизатору – 192.168.1.1. Маска підмережі – 255.255.255.0. (Рис. 2)

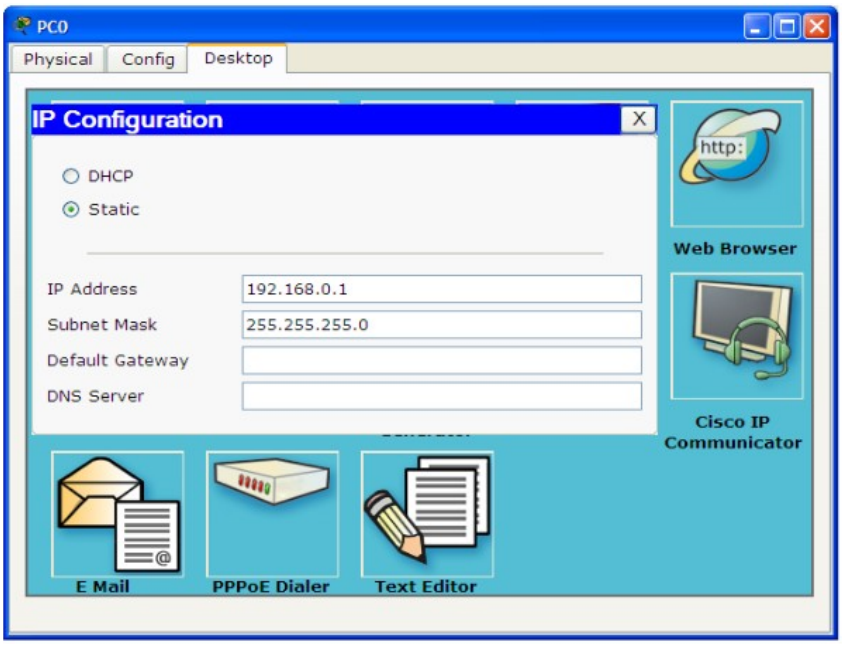


Рис. 2. IP конфігурація робочої станції.

3. Задамо відповідні IP адреси на інтерфейсах маршрутизатору та включимо ці порти. (Рис. 3).

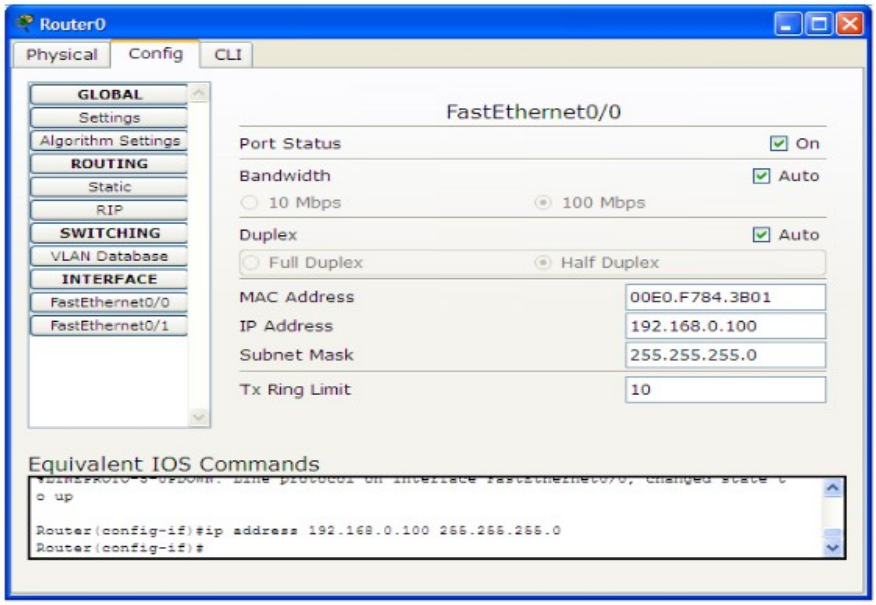


Рис.3 IP конфігурація маршрутизатору.

4. Зайдемо до **Command Line Interface** маршрутизатора та з допомогою команди ***enable secret*** задамо пароль доступу до привилеєрованого режиму та збережемо конфігурацію. (Рис.4).

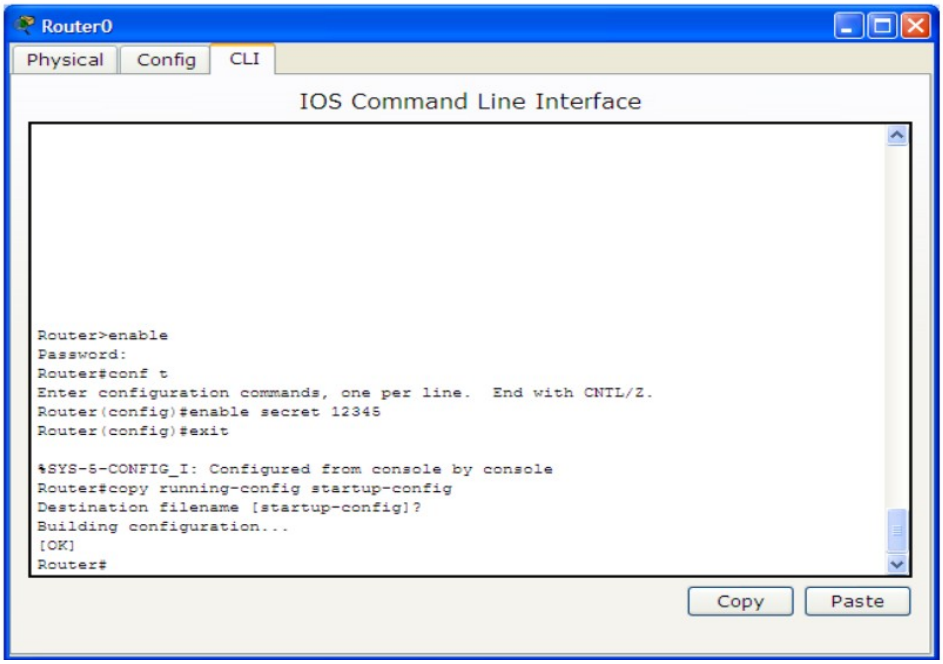


Рис. 4 Робота у **Command Line Interface**.

5. Для налаштування маршрутизації за протоколом RIP відкриємо вкладку **Config** у вікні властивостей маршрутизатора та оберемо пункт RIP. Задамо там адресу всіх підмереж, яким дозволено спілкування. (Рис. 5).

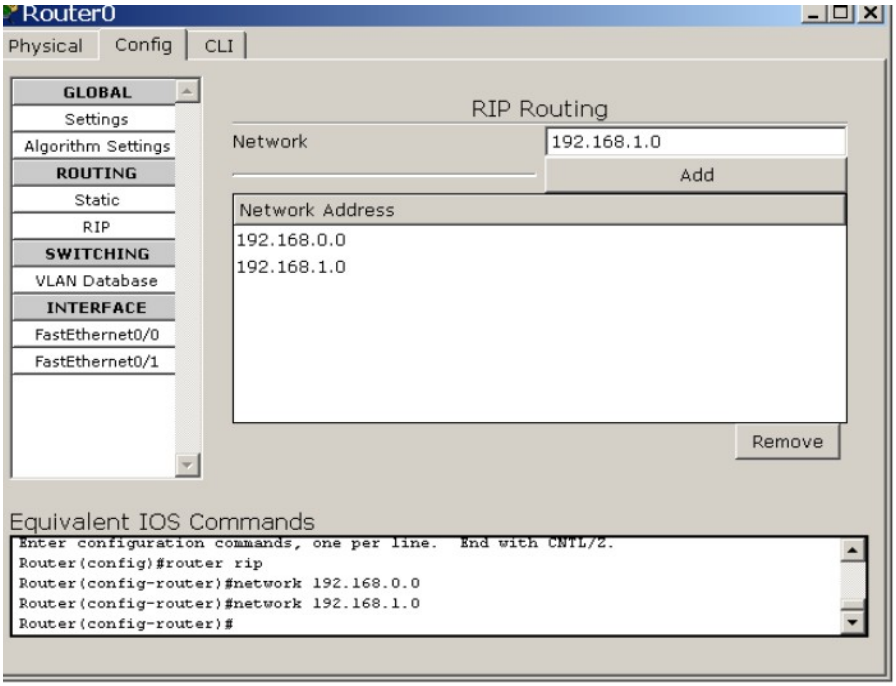


Рис 5 Налаштування маршрутизації за протоколом RIP

Перевіряємо доступність робочих станцій друг для друга. Для цього у правому стовбці обираємо інструмент **Add simple PDU** та обираємо станцію-відправника та станцію-отримувача. Переконаємося, що передача закінчена

вдало. (Рис. 6)

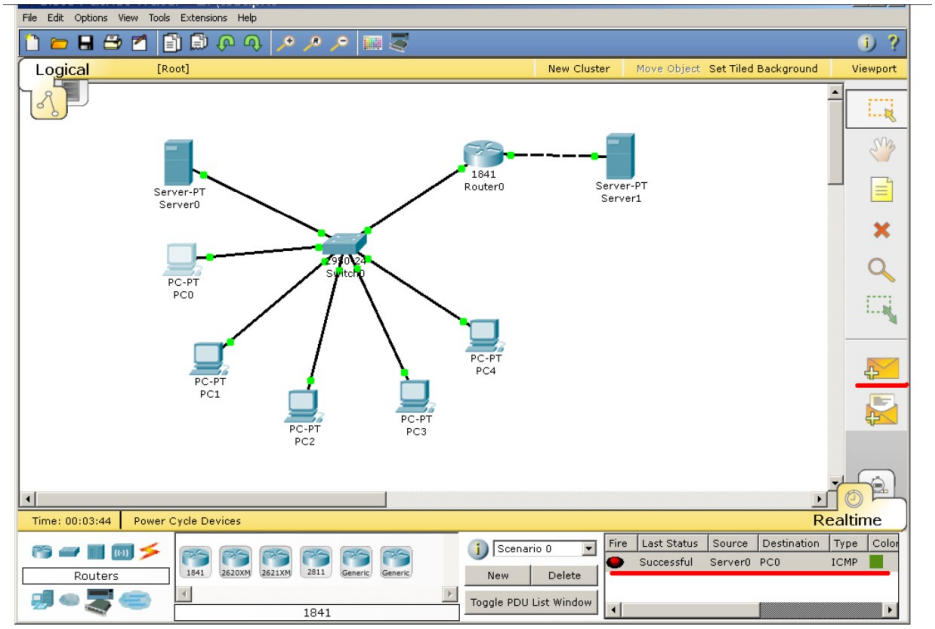


Рис. 6. Перевірка доступності вузлів у мережі.

Налаштувати для кожного з роутерів статичну маршрутизацію (зауваження: IP-адреси комп'ютерам назначати згідно схеми v.1.1.n, де v – номер варіанту, n – номер комп'ютера).

Вивести таблицю маршрутизації для кожного роутера (команда show ip route).

Перевірити правильність налаштування за допомогою команд ping і tracert в консолі кожного комп'ютера.

4.4. Зміст звіту

Звіт повинен містити:

1.Титульний лист із найменуванням лабораторної роботи, датою виконання і даними виконавців;

2. Мету роботи; варіант завдання;

3. Результати роботи програми Cisco Packet Tracer (скріншоти):

1. Модель мережі для налаштування статичної маршрутизації з призначеними всім елементам мережі відповідних IP-адрес.
2. Таблиці маршрутизації для кожного роутера.
3. Результати команд ***ping*** і ***tracert*** для всіх пар комп'ютерів.

4. Аналіз і пояснення до рисунків.

6. Висновки

4.5. Контрольні питання для захисту звіту

1. Для чого використовується програмний пакет Cisco Packet Tracer?

2. Яким чином можна задати IP-адресу вузлам мережі? Підмережі?
3. Які ви знаєте протоколи мережі. Для чого вони призначені?
3. Що таке RIP-протокол? Для чого використовується.
5. Роль RIP-протоколу у відкритій моделі OSI.
6. Як налаштувати маршрутизацію за протоколом RIP?
7. Яким чином можна перевірити доступність вузлів у мережі?