

ЛАБОРАТОРНА РОБОТА №6

Тема: Робота з мережею за допомогою бібліотеки WinSock.

Мета роботи: Вивчення основ роботи з сокетами у середовищі Windows за допомогою бібліотеки WinSock.

Завдання: Написати програму “сніфер”.

Лістинг програми:

```
#define SIO_RCVALL _WSAIOW(IOC_VENDOR,1)
#define _WINSOCK_DEPRECATED_NO_WARNINGS

#include <winsock2.h>
#include <ws2tcpip.h>
#include <iostream>
#include <chrono>
#include <sstream>
#include <iomanip>
#include <cstdint>
#include <ostream>
#include <string>

#pragma comment(lib, "Ws2_32.lib")

#define MAX_PACKET_SIZE 65536

std::string get_current_time()
{
    using namespace std::chrono;

    auto now = system_clock::now();
    auto now_time_t = system_clock::to_time_t(now);
    auto now_us = duration_cast<microseconds>(now.time_since_epoch()) %
1000000;
```

					ДУ«Житомирська політехніка».25.121.27.000 – Лр6			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Семенчук О.А.			Звіт з лабораторної роботи		Літ.	Арк.
Перевір.		Власенко О.В.						1
Керівник								8
Н. контр.							ФІКТ Гр. ІПЗ-23-1[2]	
Зав. каф.								

```

std::tm local_tm;
localtime_s(&local_tm, &now_time_t);

char buffer[64];
std::strftime(buffer, sizeof(buffer), "%Y-%m-%d %H:%M:%S", &local_tm);
std::ostringstream oss;
oss << buffer << "." << std::setw(6) << std::setfill('0') <<
now_us.count();
return oss.str();
}

std::string get_ip(unsigned int ip)
{
return std::to_string((ip)&0xFF) + "."
    + std::to_string((ip >> 8)&0xFF) + "."
    + std::to_string((ip >> 16)&0xFF) + "."
    + std::to_string((ip >> 24)&0xFF);
}

void parse_packet(const char* buffer, int length)
{
    struct IPHeader
    {
        unsigned char  ihl_ver;
        unsigned char  tos;
        unsigned short total_len;
        unsigned short id;
        unsigned short flags_offset;
        unsigned char  ttl;
        unsigned char  protocol;
        unsigned int   src_addr;
        unsigned int   dest_addr;
    };

    if (length < sizeof(IPHeader))
    {
        std::cout << "Packet too small for IP header." << std::endl;
        return;
    }

```

		Семенчук О.А.			ДУ «Житомирська політехніка».25.121.27.000 – Лр6	Арк.
		Власенко О.В				
Змн.	Арк.	№ докум.	Підпис	Дата		2

```

}

const IPHeader* ip = (IPHeader*)buffer;

int ip_header_length = (ip->ihl_ver & 0x0F) * 4;

std::cout << "Length: " << length << std::endl;
std::cout << "Time: " << get_current_time() << std::endl;
std::cout << "Time to live: " << static_cast<int>(ip->ttl) <<
std::endl;

std::cout << "IP: " << get_ip(ip->src_addr) << " > " << get_ip(ip-
>dest_addr) << std::endl;

switch (ip->protocol)
{
    case IPPROTO_UDP:
    {
        struct UDPHeader
        {
            uint16_t src_port;
            uint16_t dest_port;
            uint16_t length;
            uint16_t checksum;
        };

        const UDPHeader* udp = (UDPHeader*)(buffer + ip_header_length);

        std::cout << "Protocol: UDP" << std::endl;;
        std::cout << "UDP Length: " << ntohs(udp->length) << std::endl;
        std::cout << "UDP Port: " << ntohs(udp->src_port) << " > " <<
            ntohs(udp->dest_port) << std::endl;

        break;
    }

    case IPPROTO_TCP:
    {
        struct TCPHeader

```

		Семенчук О.А.			ДУ «Житомирська політехніка».25.121.27.000 – Лр6	Арк.
		Власенко О.В				
Змн.	Арк.	№ докум.	Підпис	Дата		3

```

        {
            uint16_t src_port;
            uint16_t dest_port;
            uint32_t seq_num;
            uint32_t ack_num;
            uint8_t  flags;
        };

        TCPHeader* tcp = (TCPHeader*)(buffer + ip_header_length);

        std::cout << "TCP Port: " << ntohs(tcp->src_port) << " > " <<
            ntohs(tcp->dest_port) << std::endl;

        std::cout << "Sequence number: " << ntohs(tcp->seq_num) <<
std::endl;
        std::cout << "Acknowledgment number: " << ntohs(tcp->ack_num)
<< std::endl;
        std::cout << "Flags: " << int(tcp->flags) << std::endl;

        break;
    }

    default:
        std::cout << "Protocol: Unknown (" << (int)ip->protocol <<
")\n";
        break;
    }

    std::cout << std::endl;
}

int main()
{
    WSADATA wsa_data;

    if (WSAStartup((WORD)2.2, &wsa_data) != 0)
    {
        std::cerr << "WSAStartup failed!" << std::endl;
    }
}

```

		Семенчук О.А.			ДУ «Житомирська політехніка».25.121.27.000 – Лр6	Арк.
		Власенко О.В				
Змн.	Арк.	№ докум.	Підпис	Дата		4

```

        return 1;
    }

    SOCKET sniffer = socket(AF_INET, SOCK_RAW, IPPROTO_IP);

    if (sniffer == INVALID_SOCKET)
    {
        std::cerr << "Socket creation failed! Error: " << WSAGetLastError()
<< std::endl;
        WSACleanup();
        return 1;
    }

    char hostName[256];

    if (gethostname(hostName, sizeof(hostName)) == SOCKET_ERROR)
    {
        std::cerr << "Failed to get hostname! Error: " << WSAGetLastError()
<< std::endl;
        closesocket(sniffer);
        WSACleanup();
        return 1;
    }

    struct hostent* localHost = gethostbyname(hostName);

    if (!localHost)
    {
        std::cerr << "Failed to get local host! Error: " << WSAGetLastError()
<< std::endl;
        closesocket(sniffer);
        WSACleanup();
        return 1;
    }

    sockaddr_in socketAddr = {};

    socketAddr.sin_family = AF_INET;

```

		Семенчук О.А.			ДУ «Житомирська політехніка».25.121.27.000 – Лр6	Арк.
		Власенко О.В				
Змн.	Арк.	№ докум.	Підпис	Дата		5

```

socketAddr.sin_port = htons(0);
socketAddr.sin_addr.s_addr = inet_addr("192.168.0.105");

if (bind(sniffer, (sockaddr*)&socketAddr, sizeof(socketAddr)) == SOCKET-
ET_ERROR)
{
    std::cerr << "Bind failed! Error: " << WSAGetLastError() <<
std::endl;
    closesocket(sniffer);
    WSACleanup();
    return 1;
}

DWORD flag = 1;
if (ioctlsocket(sniffer, SIO_RCVALL, &flag) == SOCKET_ERROR)
{
    std::cerr << "Failed to set promiscuous mode! Error: " << WSAGet-
LastError() << std::endl;
    closesocket(sniffer);
    WSACleanup();
    return 1;
}

char buffer[MAX_PACKET_SIZE];

while (true)
{
    int bytes_received = recv(sniffer, buffer, MAX_PACKET_SIZE, 0);

    if (bytes_received > 0)
    {
        parse_packet(buffer, bytes_received);

        std::cout << "-----" <<
std::endl;
    }

    else if (bytes_received == SOCKET_ERROR)

```

		Семенчук О.А.			ДУ «Житомирська політехніка».25.121.27.000 – Лр6	Арк.
		Власенко О.В				
Змн.	Арк.	№ докум.	Підпис	Дата		6

```

        {
            std::cerr << "Receive error! Error: " << WSAGetLastError() <<
std::endl;
            break;
        }
    }
    closesocket(sniffer);
    WSACleanup();
    return 0;
}

```

Результат:

```

Length: 68
Time: 2025-05-22 13:53:29.365181
Time to live: 1
IP: 192.168.0.102 > 224.0.0.251
Protocol: UDP
UDP Length: 48
UDP Port: 5353 > 5353
-----
Length: 68
Time: 2025-05-22 13:53:29.366392
Time to live: 1
IP: 192.168.0.102 > 224.0.0.251
Protocol: UDP
UDP Length: 48
UDP Port: 5353 > 5353
-----
Length: 68
Time: 2025-05-22 13:53:29.367469
Time to live: 1
IP: 192.168.0.102 > 224.0.0.251
Protocol: UDP
UDP Length: 48
UDP Port: 5353 > 5353

```

Рис. 1 Результат виконання програми

Перелік основних функцій:

- **WSAStartup** - Ініціалізує бібліотеку Winsock. Потрібно викликати перед будь-яким використанням сокетів. У вашому коді використовується для запуску мережевих функцій.
- **WSACleanup** - Завершує роботу з Winsock та звільняє пов'язані ресурси. Викликається в кінці, після завершення роботи з сокетом.
- **socket** - Створює новий сокет. У коді створюється сирий сокет (SOCK_RAW) для перехоплення IP-пакетів (IPPROTO_IP).

		Семенчук О.А.			ДУ «Житомирська політехніка».25.121.27.000 – Лр6	Арк.
		Власенко О.В				7
Змн.	Арк.	№ докум.	Підпис	Дата		

- `closesocket` - Закриває відкритий сокет і звільняє ресурси. У коді викликається при завершенні роботи або при помилці.
- `bind` - Прив'язує сокет до локальної IP-адреси та порту. Необхідно для прийому даних. У коді сокет прив'язується до IP-адреси комп'ютера.
- `recv` - Приймає дані з сокету. У цьому випадку — перехоплює пакети з мережі.
- `ioctlsocket` - Керує параметрами сокету. У коді використовується з `SIO_RCVALL`, щоб увімкнути проміжний режим (`promiscuous mode`), коли сокет отримує всі пакети незалежно від адресата.

Висновок: Програма успішно використовує функції Winsock для створення сирого сокета, перехоплення та аналізу IP-пакетів у мережі. Отримані дані дозволяють детально дослідити структуру мережевих протоколів і можуть бути використані для подальшого розвитку інструментів мережевого моніторингу.

		Семенчук О.А.			ДУ «Житомирська політехніка».25.121.27.000 – Лр6	Арк.
		Власенко О.В				
Змн.	Арк.	№ докум.	Підпис	Дата		8