**Упражнение 1 – Въведение в TCP/IP и Сокети**

Internet cable network view  
  
 • <https://bbmaps.itu.int/bbmaps/>  
 • <https://www.submarinecablemap.com/>

Първоначални настройки:

Install IDE : IntelliJ IDEA Community Edition:

<https://www.jetbrains.com/idea/download/other.html>

Github:

<https://github.com/tvuchova/internet_programming>

1.IP address  
  
 • На този сайт може да видите IP адреса си:  
 <https://www.whatismyip.com/>  
  
 • команда : **ipconfig**

Показва: • Ethernet adapter or wireless adapter name  
 • IP address  
 • Subnet mask  
 • Default gateway  
 • DHCP server  
 • DNS servers

Пример:

Ethernet adapter Ethernet  
IP address: 192.168.1.100  
Subnet mask: 255.255.255.0  
Default gateway: 192.168.1.1  
Wireless LAN adapter WiFi  
  
IP address: 192.168.2.100  
Subnet mask: 255.255.255.0  
Default gateway: 192.168.2.1  
  
  
 2. DNS   
 - може да видите DNS информацията  
<https://www.nslookup.io/>

3. Ping – проверявате дали даден уебсайт е достъпен - Internet Control Message Protocol (ICMP) "echo request"

**Част 1: Запознаване с основните класове**

1.1. InetAddress и IP адреси

**Задача 1:** Напишете по-горните команди на java използвайки **InetAddress**:

* 1. Получаване и отпечатване на IP адреса на локалния хост (текущия компютър)
  2. Получаване и отпечатване на loopback адреса (обикновено 127.0.0.1)
  3. Пример Reverse DNS Lookup - получаване на хостнейм от IP адрес (Google DNS: 8.8.8.8)
  4. Получаване на IP адрес за дадено домейн име (пример: [www.yahoo.com](http://www.yahoo.com))
  5. Симулиране на nslookup за домейна google.com - връщане на всички свързани IP адреси(hostname, ipadress)
  6. Симулиране на "ping" - проверка дали хост с даден IP адрес е достъпен. Проверка дали хостът е достъпен в рамките на 5000 милисекунди (5 секунди)

използвайте класа:

**public class InetAddress**extends **Object** implements **Serializable:**

Тук може да видите документация и методите му:

<https://docs.oracle.com/javase/8/docs/api/java/net/InetAddress.html>

методите, които могат да помогнат при реализация на задачата ви:

|  |  |
| --- | --- |
| public static InetAddress getLocalHost() *throws UnknownHostException* | instance на InetAddress съдържаща local hostname and address. |
| public static InetAddress getByName( String host ) *throws UnknownHostException* | instance на InetAddress съдържаща IP and Host name на хоста. |
| public static InetAddress[] getAllByName( String hostName ) *throws UnknownHostException* | array от инстанции InetAddress class който съдържат IP addresses. |
| public static InetAddress getByAddress( byte IPAddress[] ) *throws UnknownHostException* | InetAddress обекта създаден от raw IP address. |
| public static InetAddress getByAddress( String hostName, byte IPAddress[] ) *throws UnknownHostException* | Създава и връща InetAddress обекта създаден от съответния hostname and IP address. |
| InetAddress.getLoopbackAddress() | The loopback address е специален IP address което се използва да тества network connections на локалната machine без да изпраща пакети по реална мрежа  refers to the IP address 127.0.0.1 in IPv4 or ::1 in IPv6.  • It is used to refer to "localhost", meaning the local machine itself.  • Using the loopback address allows you to send and receive network traffic from your own computer without requiring an external network connection. |

**Example output:**

IP of Local Host : R59RM22RCV/192.168.43.72

Loopback InetAddress - Local Host : localhost/127.0.0.1

Hostname for IP 8.8.8.8: dns.google

IP of www.yahoo.com is: 212.82.116.204

NSLookup for google.com:

Host Name: google.com

IP Address: 142.250.187.142

Host Name: google.com

IP Address: 2a00:1450:4017:80e:0:0:0:200e

Sending Ping Request to 133.192.31.42

Sorry ! We can't reach to this host

**1.2 Class NetworkInterface –** този класни позволява програмно да вземем информация за нашите мрежови интерфейси

Методи:

<https://docs.oracle.com/javase/8/docs/api/java/net/NetworkInterface.html>

**Задача 2.** Вие сте системен администратор и искате да разберете какви мрежови интерфейси са налични на сървъра, който поддържате. Освен това, искате да получите повече информация за тях, като IP адреси, MAC адреси, статус на връзките и други. Вашата задача е да създадете програма, която:

1. Изброява всички налични мрежови интерфейси на машината.
2. За всеки интерфейс показва:
   * Името на интерфейса
   * MAC адрес
   * Всички IP адреси (IPv4 и IPv6)
   * Дали интерфейсът е активен
   * Дали поддържа multicast
   * Скорост на интерфейса (ако е налична)

Искат от вас да използвате java и клас NetworkInterface

Полезни методи:

NetworkInterface.*getNetworkInterfaces*()

networkInterface.getHardwareAddress()

networkInterface.getInetAddresses();

networkInterface.isUp()

networkInterface.supportsMulticast()

networkInterface.isLoopback()

Примерен изход:

Интерфейс: eth0

Описание: Ethernet интерфейс

MAC адрес: 00-14-22-01-23-45

IP адрес: 192.168.1.10

Интерфейсът е активен: true

Поддържа multicast: true

Loopback интерфейс: false

---------------------------------------------------

Интерфейс: lo

Описание: Loopback интерфейс

MAC адрес: Няма информация.

IP адрес: 127.0.0.1

Интерфейсът е активен: true

Поддържа multicast: false

Loopback интерфейс: true

**Част 2: Въвеждане в потоково предаване на данни (Streams)**

**2.1.** Работа с входни и изходни потоци (I/O)

Основната комуникация в сокетите често използва потоци за четене и запис**.**

**Задача 3.** Управление на оценки на студенти чрез файлова система (I/O операции)

**Описание:**

Напишете програма, която управлява база данни с оценки на студенти, като използва файлове за съхранение на информацията. Програмата трябва да може да чете, записва и обработва информация за студенти и техните оценки, използвайки I/O операции (входно-изходни потоци).

Функционалности:

1. Добавяне на студент и оценка:

Програмата трябва да позволява добавяне на студент с неговото име и оценка.

Информацията трябва да бъде записана във файл (например "students.txt").

1. Преглед на оценки:

Програмата трябва да чете данни от файла и да показва списък с всички студенти и техните оценки.

Изчисляване на средна оценка:

Програмата трябва да може да изчислява и показва средната оценка на всички студенти, използвайки данните от файла.

Запис във файл:

Всички промени в информацията за студенти трябва да се запазват във файла.

Изисквания:

Файлът "students.txt" трябва да съхранява данните в следния формат: всяка линия представлява студент и оценката му, разделени със запетая (например: Иван Иванов,5.50).

Забележка:

Използвайте try-with-resources за работа с файлови потоци, за да осигурите автоматичното им затваряне.

**Част 3: SOCKET – CLIENT- SERVER**

A diagram of a computer program

Description automatically generated

**Socket class**

java.net.Socket позволява да създадем сокет свързан към порт и адрес:

*public Socket(InetAddress address, int port)*

Четене и записване на данни:

|  |
| --- |
| **Method** |
| * 1. public InputStream getInputStream() - InputStream атачнат към този сокет |
| * 1. public OutputStream getOutputStream() - връща OutputStream атачнат към тоя сокет |
| * 1. public synchronized void close() - затваря сокета |

**ServerSocket class**

The ServerSocket class може да използваме за да създадем server socket. Този обект се използва за да осъществи комуникация с клиента.

|  |
| --- |
| **Method** |
| 1. public Socket accept() – връща сокет и установява връзка между клиент и сървър |
| 1. public synchronized void close() – затваря сокета |

**Задача 4**.Напишете ехо програма: Клиента прочита текст от клавиатурата праща го към сървъра и сървъра му връща ехо същия текст

Спецификация:

<https://docs.oracle.com/javase/tutorial/networking/sockets/readingWriting.html>

<https://docs.oracle.com/en/java/javase/21/docs/api/java.base/java/net/Socket.html>

Примерен изход:

**Сървър:**

Hello World.I am Server!

Connection established

**Client:**

Hello World.I am Client!

Enter string

Hello from a client

Server echo says: Hello from a client

**Задача 5:**

Напишете клиент-сървър приложение, което използва **TCP сокети** за комуникация. Клиентът ще изпраща съобщение до сървъра, а сървърът ще отговаря с IP адреса на клиента и информация за мрежовия интерфейс на сървъра.

**Изисквания:**

1. **Сървърът**:
   * Сървърът трябва да приема връзки от клиента, като използва **ServerSocket**.
   * Когато клиент се свърже, сървърът трябва да изпрати IP адреса на клиента и информация за мрежовия интерфейс на сървъра.
2. **Клиентът**:
   * Клиентът трябва да изпрати кратко съобщение до сървъра.
   * Клиентът трябва да изчака и да получи отговор от сървъра, след което да го отпечата.

**Стъпки:**

1. Създайте сървър, който слуша за връзки от клиент и използва **InetAddress**, за да получи IP адреса на клиента.
2. Сървърът трябва да изпрати информация за мрежовия интерфейс към клиента, като използва **NetworkInterface**.
3. Клиентът трябва да изпрати съобщение към сървъра и да получи информацията за IP адреса и мрежовия интерфейс.

Примерен изход:

Server is started on port 9238...

IP address of client: 127.0.0.1

NetworkAddressOfServer:

Interface: utun3

Address: fe80:0:0:0:ce81:b1c:bd2c:69e%utun3

Interface: utun2

Address: fe80:0:0:0:7f9e:e87e:b2e0:6ad0%utun2

Interface: utun1

Address: fe80:0:0:0:807f:3647:86af:d1a5%utun1

Interface: utun0

Address: fe80:0:0:0:9b15:9fdb:1112:3ae0%utun0

Interface: llw0

Address: fe80:0:0:0:c44d:c9ff:fe15:9742%llw0

Interface: awdl0

Address: fe80:0:0:0:c44d:c9ff:fe15:9742%awdl0

Interface: ap1

Address: fe80:0:0:0:bcd0:74ff:fe73:3817%ap1

Interface: utun4

Address: fc00:35d5:1ba1:45fb:5b60:8195:67c3:e6b8%utun4

Address: fe80:0:0:0:bed0:74ff:fe73:3817%utun4

Address: 10.147.43.24

Interface: bridge100

Address: fd32:8233:d26c:df62:4e9:469d:1e64:18ff%bridge100

Address: fe80:0:0:0:bcd0:74ff:fe37:9464%bridge100

Address: 192.168.205.1

……

**Задача 6. UDP сървър**

Вие сте студент от първи курс на техническия университет, и искате да създадете прост чат чрез използване на **UDP сокети**. Задачата ви е да създадете приложение, което да позволява на два или повече потребителя да комуникират помежду си през мрежата. За разлика от TCP, UDP не гарантира доставянето на съобщенията, което прави задачата още по-интересна, тъй като ще научите как да изпращате и получавате съобщения в една "ненадеждна" среда.

**Задачи:**

1. Създайте **UDP сървър**, който да може да приема съобщения от клиенти и да изпраща обратно потвърждения.
2. Създайте **UDP клиент**, който да може да изпраща съобщения до сървъра и да получава отговори.
3. Разширете приложението така, че всеки клиент да може да изпраща съобщения и към други клиенти чрез сървъра.

**Как работи програмата:**

1. **UDP Сървър:**
   * Сървърът слуша на порт 9876 за входящи UDP пакети.
   * Всеки път, когато получи съобщение от клиент, сървърът го принтира на конзолата и връща потвърждение на същия клиент.
2. **UDP Клиент:**
   * Клиентът изпраща съобщение до сървъра на локалния адрес (localhost) на порт 9876.
   * След като съобщението бъде изпратено, клиентът чака отговор от сървъра и го извежда на екрана.

**Упътване:**

Използвайте DatagramSocket и DatagramPacket

serverSocket.receive(packet)

// Create a Datagram Socket  
 DatagramSocket clientSocket = new DatagramSocket(порт);

<https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/net/DatagramSocket.html>

<https://docs.oracle.com/en/java/javase/21/docs/api/java.base/java/net/DatagramPacket.html>

Изходен код:

UDP чат сървър стартиран на порт 9876

Получено съобщение: hello от /127.0.0.1:60331

Получено съобщение: test от /127.0.0.1:60331

Получено съобщение: test от /127.0.0.1:60879

Получено съобщение: reas от /127.0.0.1:60879

Получено съобщение: quit от /127.0.0.1:60879

Enter message: hello

Response: Сървър: получено съобщение 'hello'

Enter message: test

Response: Сървър: получено съобщение 'test'

Enter message: quit

Response: Сървър: получено съобщение 'quit'

Process finished with exit code 0

**Задача 7.** Задачата от част 2 да се направи със сокети:

Клиентът ще изпраща заявки към сървъра за добавяне на студент, преглед на всички оценки или изчисляване на средната оценка. Сървърът ще обработва тези заявки, ще чете/записва във файл и ще връща резултата на клиента.

**Сървър:**

Сървърът приема заявки от клиентите чрез TCP сокети. Управлява файл, в който се съхраняват данните за студентите и техните оценки.

Обработва заявки като добавяне на студент, преглед на списъка с оценки и изчисляване на средна оценка.

Клиент:

Клиентът изпраща заявки към сървъра (добавяне на студент, преглед на оценки или изчисляване на средна оценка) и получава резултати обратно от сървъра.

Изходен код:

--- Меню ---

1. Добавяне на студент

2. Преглед на всички студенти

3. Търсене на студент по име

4. Средна оценка на всички студенти

5. Изход

Изберете опция (1-5): 1

Въведете името на студента: test

Въведете оценката на студента: 5

Студентът е добавен успешно.

Sept 27, 2024 10:52:08 AM sockets\_part3.finalized.grades.StudentProcessor addStudent

INFO: Успешно добавихте студента: test с оценка: 5

Бонус:

Wireshark  
Wireshark е широко използван инструмент за улавяне и анализ на мрежовия трафик, който се използва широко както за образователни цели, така и за отстраняване на компютърни или мрежови проблеми. Wireshark работи с почти всички протоколи на модела OSI, има интерфейс, който е разбираем за обикновения потребител и удобна система за филтриране на данни. Освен всичко това, програмата е кросплатформена и поддържа следните операционни системи: Windows, Linux, Mac OS X, Solaris, FreeBSD, NetBSD, OpenBSD.  
  
  
 6.1 инсталиране:

https://www.wireshark.org/#downloadLink  
  
  
  
Wireshark allows you to filter the log before the capture starts or during analysis, so you can narrow down and zero in on   
what you’re looking for in the network trace. For example, you can set a filter to see TCP traffic between two IP addresses, or you can set it only to show you the packets sent from one computer.   
The filters in Wireshark are one of the primary reasons it has become the standard tool for packet analysis.

### Wireshark помага:

### • Мрежовите администратори отстраняват проблеми в мрежата

### • Инженерите по сигурността изследват проблемите със сигурността в мрежата

### • QA инженери проверяват приложенията

### • Разработчиците отстраняват грешки в реализациите на протокола

### • Мрежовите потребители научават за конкретен протокол

### След като вече имаме инсталиран Wireshark, нека да разгледаме как да активираме снифера на пакети Wireshark и след това да анализираме мрежовия трафик.

### Capturing data packets on Wireshark

Когато отворите Wireshark, виждате екран, който ви показва списък с всички мрежови връзки, които можете да наблюдавате. Имате и поле за филтър за улавяне, за да уловите само мрежовия трафик, който искате да видите.A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Може да изберете един или повече интерфейси.

Щракнете върху първия бутон от лентата с инструменти, озаглавен “Start capturing packets.”

A screenshot of a computer

Description automatically generated

По време на улавянето Wireshark ще ви покаже уловените пакети в реално време.

Анализиране на пакети данни в Wireshark

Wireshark ви показва три различни прозореца за проверка на пакетни данни. Списъкът с пакети, горният панел, изброява всички пакети в улавянето. Когато щракнете върху пакет, другите два панела се променят, за да ви покажат подробности за избрания пакет. Можете също така да разберете дали пакетът е част от разговор. Ето подробности за всяка колона в горния панел:

**No.**: Това е редът на номерата на заснетия пакет. Скобата показва, че този пакет е част от разговор.

**Time**: Тази колона показва колко време след като сте започнали улавянето, този конкретен пакет е бил уловен. Можете да промените тази стойност в менюто Настройки, за да покажете различна опция.

**Source**: Това е адресът на системата, която е изпратила пакета.

**Destination**: Това е адресът на дестинацията на пакета.

**Protocol**: Това е типът пакет. Например: TCP, DNS, DHCPv6 или ARP.

**Length**: Тази колона ви показва дължината на пакета, измерена в байтове.

**Info**: Тази колона ви показва повече информация за съдържанието на пакета, което ще варира в зависимост от типа на пакета.

A screenshot of a computer

Description automatically generated

## **Когато разглеждате пакет, който е част от разговор, можете да щракнете с десния бутон върху пакета и да изберете Следване, за да видите само пакетите, които са част от този разговор.**

Филтри на Wireshark

Някои от най-добрите характеристики на Wireshark са филтрите за улавяне и филтрите за показване. Филтрите ви позволяват да видите заснемането по начина, по който трябва да го видите, за да отстраните проблемите. По-долу има няколко филтъра, за да започнете.

Филтри за улавяне на Wireshark(Capture filters)

Филтрите за улавяне ограничават уловените пакети от избрания филтър. Ако пакетите не отговарят на филтъра, Wireshark няма да ги запази. Примерите за филтри за улавяне включват:

host IP-address: Този филтър ограничава уловения трафик към и от IP адреса

net 192.168.0.0/24: Този филтър улавя целия трафик в подмрежата

dst host IP-address: Улавяне на пакети, изпратени до посочения хост

порт 53: Улавяне на трафик само на порт 53

порт не 53 и не arp: Улавяне на целия трафик с изключение на DNS и ARP трафик

Филтри за показване на Wireshark(display filters)

Филтрите за показване на Wireshark променят изгледа на заснемането по време на анализ. След като сте спрели улавянето на пакети, използвайте филтри за показване, за да стесните пакетите в списъка с пакети, за да отстраните проблема си.

Един от най-полезните филтри за показване е:

ip.src==IP-address and ip.dst==IP-address

Този филтър показва пакети, изпратени от един компютър (ip.src) към друг (ip.dst). Можете също да използвате ip.addr, за да показвате пакети към и от този IP. Други филтри включват:

tcp.port eq 25: Този филтър ще ви покаже целия трафик на порт 25, който обикновено е SMTP трафик

icmp: Този филтър ще ви покаже само ICMP трафик в улавянето, най-вероятно това са ping

ip.addr != IP\_address: Този филтър ви показва целия трафик с изключение на трафика към или от посочения компютър

Анализаторите дори създават филтри за откриване на конкретни атаки, като този филтър, използван за откриване на червея Sasser:

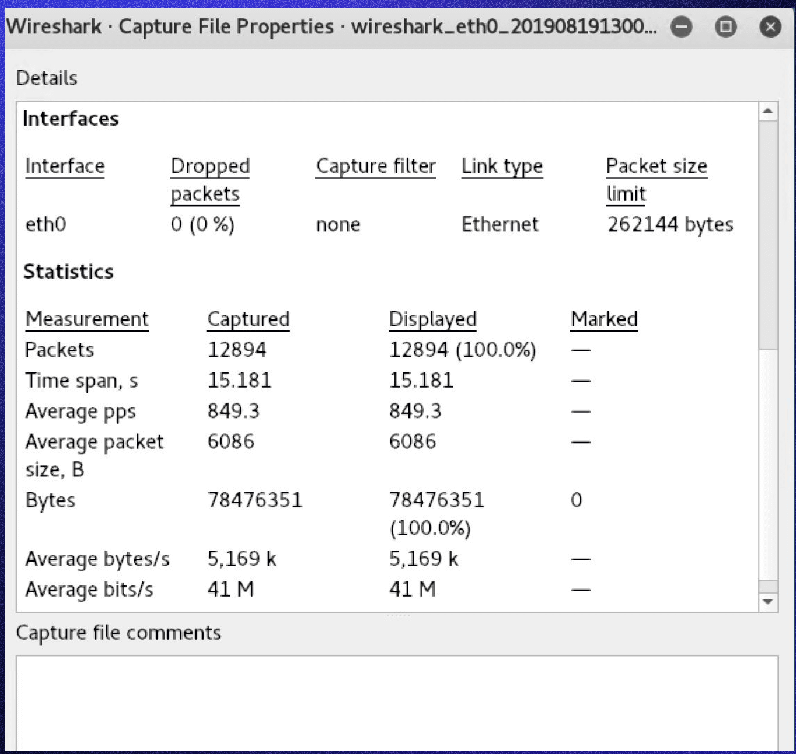
https://en.wikipedia.org/wiki/Sasser\_(computer\_worm)

### **Metrics and statistics**

Може да правите различни статистики

A screenshot of a computer

Description automatically generated



A graph on a computer screen

Description automatically generated

Още задачи:

1. **Проста игра "Камък, ножица, хартия" с UDP**

**Описание**: Направете игра "Камък, ножица, хартия" между два клиента, които комуникират чрез UDP.

**Изисквания**:

* Два клиента изпращат своите ходове към сървър.
* Сървърът определя победителя и изпраща резултата обратно към клиентите.
* Клиентите могат да играят няколко рунда.

1. **Преброяване на думи (TCP)**

**Описание**: Направете клиент-сървър приложение, при което клиентът изпраща текст към сървъра, а сървърът връща броя на думите в текста.

**Изисквания**:

* Клиентът изпраща произволен текст към сървъра.
* Сървърът преброява думите в текста и връща броя на думите на клиента.
* Клиентът отпечатва броя на думите.

1. **Търсене в база данни на студенти (TCP)**

**Описание**: Създайте приложение, което съхранява база данни с информация за студенти и позволява на клиентите да извършват заявки за търсене по име.

**Изисквания**:

* Сървърът съхранява база данни с имена и оценки на студенти.
* Клиентът изпраща име на студент към сървъра.
* Сървърът връща оценките на студента, ако съществува, или съобщение, че такъв студент не е намерен.

1. **Пренос на криптирани съобщения (TCP)**

**Описание**: Направете приложение, което шифрира съобщенията преди изпращането и ги дешифрира при получаване.

**Изисквания**:

* Клиентът въвежда съобщение, което се шифрира (например с проста Цезарова шифровка) и изпраща към сървъра.
* Сървърът дешифрира съобщението и го отпечатва на екрана.
* Добавете опция сървърът също да връща криптирани отговори.

1. **Прогноза за времето (TCP/UDP)**

**Описание**: Създайте сървър, който предоставя информация за прогноза за времето за различни градове.

**Изисквания**:

* Клиентът изпраща заявка за прогноза, като посочва име на град.
* Сървърът връща текущата прогноза за посочения град (може да бъде фиктивна информация, ако не се използва реална API услуга).
* Клиентът отпечатва прогнозата на екрана.